DPNE: Differentially Private Network Embedding

Nhat Hai Phan
New Jersey Institute of Technology

Network embedding encompasses various methods for unsupervised, and sometimes supervised, learning of feature representations of nodes and links in a network.

DPNE: Differentially Private Network Embedding

Depeng Xu[1], Shuhan Yuan[1], Xintao Wu[1], and HaiNhat Phan[2]

[1] University of Arkansas, Fayetteville, AR, USA,
{depengxu, sy005, xintaowu}@uark.edu
[2] New Jersey Institute of Technology, Newark, NJ, USA
phan@njit.edu

Abstract. Learning the low-dimensional representations of the vertices in a network can help users understand the network structure and perform other data mining tasks efficiently. Various network embedding approaches such as DeepWalk and LINE have been developed recently. However, how to protect the individual privacy in network embedding has not been exploited. It is challenging to achieve high utility as the sensitivity of stochastic gradients in random walks and that of edge sampling are very high, thus incurring high utility loss when applying Laplace mechanism and exponential mechanism to achieve differential privacy. In this paper, we develop a differentially private network embedding method (DPNE). In this method, we leverage the recent theoretical findings that network embedding methods such as DeepWalk and LINE are equivalent to factorization of some matrices derived from the adjacency matrix of the original network and apply objective perturbation on the objective function of matrix factorization. We evaluate the learned representations by our DPNE from three different real world datasets on two data mining tasks: vertex classification and link prediction. Experiment results show the effectiveness of DPNE. To our best knowledge, this is the first work on how to preserve differential privacy in network embedding.

1 Introduction

Network embedding learns the lower dimensional representations of the vertices in a high-dimensional social network [1]. The latent representations encode the social relations in a continuous vector space, which can be used to conduct a variety of applications such as vertex classification and link prediction. The first network embedding model, DeepWalk [2], uses the sequences of vertices generated by random walks to learn the vertex representations. It adopts SkipGram [3], which was previously used to learn word representations in natural language processing. Several models based on the neural language model have been proposed, such as node2vec [4], Discriminative Deep Random Walk (DDRW) [5], Large-scale Information Network Embedding (LINE) [6] and Signed Network Embedding (SNE) [7].

However, releasing the representations of vertices in a social network gives malicious attackers a potential way to infer the sensitive information of individuals. For example, the widely-used network embedding methods like DeepWalk [2] and LINE [6] train the vertex representations based on the linkage information between vertices. Hence, the released vertex representations may potentially breach link privacy of social network users. Currently, it is under-exploited how to preserve differential privacy in network embedding.

Differential privacy is a formal standard for protecting individual privacy in data analysis [8]. Differential privacy ensures that the inclusion or exclusion of a single record from a dataset makes no statistical difference when we perform a data analysis task on the dataset. The mechanisms to achieve differential privacy mainly include the classic approach of adding Laplacian noise [8], the exponential mechanism [9], the objective perturbation approach [10], the functional perturbation approach [11] and the sample and aggregate framework [12]. There have been many studies on the application of differential privacy in some particular analysis tasks, e.g., data collection [13,14], stochastic gradient descents [15], regression [16], spectral graph analysis [17], causal graph discovery [18] and deep learning models[19,20]. In this work, we aim to ensure no "privacy loss" in case of the inclusion or exclusion of an edge between two vertices from a network, a.k.a. link privacy.

In DeepWalk, the inputs of the two embedding models are generated by random walks and edge sampling. The objective function derived from random walks has an uncertain and complex mapping from edges. Edge sampling directly discloses the presence or absence of an edge between a vertex pair, which leads to a high sensitivity for privacy protection. Thus, it is difficult to directly incorporate well-studied differential privacy mechanisms [15,21] onto DeepWalk. Meanwhile, the matrix factorization based method, which is proven to be equivalent to DeepWalk, learns representations through factorizing a matrix with the pointwise mutual information of the vertices pairs in a network. It is convenient to achieve differential privacy in network embedding via small perturbation on matrix factorization.

In this work, we focus on developing a differential privacy preserving method of network embedding based on the equivalent matrix factorization method. We propose a differentially private network embedding method (DPNE). In this method, we leverage the findings that network embedding methods such as DeepWalk and LINE are equivalent to factorization of some matrices derived from the adjacency matrix of the original network and apply objective perturbation on the objective function of matrix factorization. We show that with only adding a small amount of noise onto the objective function, the learned low-dimensional representations satisfy differential privacy. Experimental results show that the embedded representations learned by DPNE achieve good utility with a small privacy budget on both vertex classification and link prediction tasks. To our best knowledge, this is the first work on how to preserve differential privacy in network embedding.

2 Preliminaries

2.1 Network Embedding

A social network is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of vertices and $\mathcal{E}$ is the set of edges. We use v_i to denote a vertex, and we use e_{ij} to denote an edge between a pair of vertices v_i and v_j. The goal of the network embedding is to learn the low-dimensional representations $\mathbf{X} \in \mathbb{R}^{|\mathcal{V}| \times k}$ for all vertices in $\mathcal{V}$, where $k \ll |\mathcal{V}|$. The i-th row of $\mathbf{X}$ (denoted as $\mathbf{x}_i$) is the k-dimensional latent representation of vertex v_i.

DeepWalk. DeepWalk adopted SkipGram [3], which was previously used to learn word representations, to learn vertex representations according to the network structure. DeepWalk first generates short random walks for each vertex. Then the model uses the sequences of vertices S generated by random walks as the input of SkipGram function to learn the vertex representations. In particular, for each target vertex $v_i \in \mathcal{V}$ and a context vertex v_j within t window size of v_i in a walk sequence ($v_j \in C_i = \{v_{i-t}, \dots, v_{i+t}\} \setminus \{v_i\}$), DeepWalk optimizes the co-occurrence probability between v_i and its context vertices within S:

$$\mathcal{L}(\mathcal{S}) = \frac{1}{|S|} \sum_{v_i \in S} \sum_{v_j \in C_i} \log \Pr(v_j | v_i), \tag{1}$$

where the probability function $\Pr(v_j|v_i)$ is defined by the softmax function:

$$\Pr(v_j|v_i) = \frac{\exp(\mathbf{x}_i \mathbf{y}_j^T)}{\sum_{v_{j'} \in \mathcal{V}_c} \exp(\mathbf{x}_i \mathbf{y}_{j'}^T)}, \tag{2}$$

where $\mathbf{x}_i$ and $\mathbf{y}_j$ are the k-dimensional representations of the target vertex v_i and the context vertex v_j, respectively; and $\mathcal{V}_c$ denotes the set of context vertices.

DeepWalk as Matrix Factorization. DeepWalk using SkipGram with Negative Sampling (SGNS) has been proven that it is equivalent to factorize a matrix $\mathbf{M}$ derived from $\mathcal{G}$ $\underset{|\mathcal{V}| \times |\mathcal{V}_c|}{\mathbf{M}} = \underset{|\mathcal{V}| \times k}{\mathbf{W}} \underset{k \times |\mathcal{V}_c|}{\mathbf{H}^T}$ [22]. The factorized matrices $\mathbf{W}, \mathbf{H}$ are equivalent to the vertex/context representations, as $\mathbf{w}_i = \mathbf{x}_i$ and $\mathbf{h}_j = \mathbf{y}_j$. Each value m_{ij} in $\mathbf{M}$ represents logarithm of the average probability that vertex v_i randomly walks to vertex v_j within fixed t steps. Formally, m_{ij} is defined as:

$$m_{ij} = \log \frac{[I_i(\mathbf{P} + \mathbf{P}^2 + \cdots + \mathbf{P}^t)]_j}{t}, \tag{3}$$

where $\mathbf{P}$ is the transition matrix of $\mathcal{G}$ with $p_{ij} = \frac{1}{d_i}$ if $e_{ij} \in \mathcal{E}$; d_i is the degree of v_i in $\mathcal{G}$; and I_i denotes an indicator vector, in which the i-th entry is 1 and the others are all 0.

Hence, we formalize the DeepWalk as matrix factorization $\mathbf{M} = \mathbf{W}\mathbf{H}^T$. Let Ω be the set of vertex pairs referenced by each entry m_{ij} of $\mathbf{M}$. The model aims to find matrices $\mathbf{W}$ and $\mathbf{H}$ to minimize the objective function as follows:

$$\mathcal{L}(\mathbf{w}_i, \mathbf{h}_j, \mathcal{G}) = \sum_{(i,j)\in\Omega} ||m_{ij} - \mathbf{w}_i\mathbf{h}_j^T||_2^2 + \lambda \sum_{i\in\mathcal{V}} ||\mathbf{w}_i||_2^2 + \mu \sum_{j\in\mathcal{V}_c} ||\mathbf{h}_j||_2^2, \tag{4}$$

where λ and μ are the weights of regularization terms.

We adopt the stochastic gradient descent (SGD) approach to minimize Equation 4. SGD iteratively learns $\mathbf{W}$ and $\mathbf{H}$. The partial derivatives of $\mathcal{L}(\mathbf{w}_i, \mathbf{h}_j, \mathcal{G})$ with respect to $\mathbf{w}_i$ and $\mathbf{h}_j$ are as follows:

$$\bigtriangledown_{\mathbf{w}_i} \mathcal{L}(\mathbf{w}_i, \mathbf{h}_j, \mathcal{G}) = -2 \sum_{j\in\mathcal{V}_c} \mathbf{h}_j(m_{ij} - \mathbf{w}_i\mathbf{h}_j^T), \tag{5}$$

$$\bigtriangledown_{\mathbf{h}_j} \mathcal{L}(\mathbf{w}_i, \mathbf{h}_j, \mathcal{G}) = -2 \sum_{i\in\mathcal{V}} \mathbf{w}_i(m_{ij} - \mathbf{w}_i\mathbf{h}_j^T). \tag{6}$$

2.2 Differential Privacy

In this work, we consider that differential privacy ensures the removal or addition of one edge in the graph $\mathcal{G}$ makes no statistical difference on the output.

Definition 1. ***Differential privacy*** *[8] A graph analysis mechanism $\mathcal{M}$ satisfies ϵ-differential privacy, if for all neighboring graphs $\mathcal{G}$ and $\mathcal{G}'$ and all subsets Z of $\mathcal{M}$'s range:*

$$\Pr(\mathcal{M}(\mathcal{G}) \in Z) \leq \exp(\epsilon) \cdot \Pr(\mathcal{M}(\mathcal{G}') \in Z), \tag{7}$$

where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$, $\mathcal{E}' = \mathcal{E} \bigcup \{e_{rs}\}$, e_{rs} is the differed edge between $\mathcal{G}$ and $\mathcal{G}'$.

The parameter ϵ denotes the privacy budget (smaller values indicates a stronger privacy guarantee).

Definition 2. ***Global sensitivity*** *[8] Given a function $f : \mathcal{G} \to \mathbb{R}^k$. The sensitivity $S_f(\mathcal{G})$ is defined as*

$$S_f(\mathcal{G}) = \max_{\mathcal{G},\mathcal{G}'} ||f(\mathcal{G}) - f(\mathcal{G}')||. \tag{8}$$

Definition 3. ***Laplace mechanism*** *[8] Given a graph $\mathcal{G}$ and a query f, a mechanism $\mathcal{M}(\mathcal{G}) = f(\mathcal{G}) + (Y_1, \cdots, Y_k)$ satisfies ϵ-differential privacy, where Y_i is drawn i.i.d. from $Lap(S_f(\mathcal{G})/\epsilon)$.*

Laplace mechanism ensures differential privacy for any function f by adding random noises generated from a Laplace distribution onto the true answer of $f(\mathcal{G})$. The global sensitivity of f controls the magnitude of the noise distribution.

For many data mining and machine learning algorithms, we usually optimize some objective functions (e.g., cross entropy) to derive coefficients of released models. Rather than adding noise to coefficients of the released model, Chaudhuri et al. [10] proposed an **objective perturbation** approach by perturbing the objective function $\mathcal{L}$ and then optimizing the perturbed objective function,

$$\mathcal{L}_{priv}(\boldsymbol{\omega}, \mathcal{G}) = \mathcal{L}(\boldsymbol{\omega}, \mathcal{G}) + \boldsymbol{\omega}\boldsymbol{\eta}^T, \tag{9}$$

where $\boldsymbol{\eta}$ is a random noise vector and its probability density is given by

$$\Pr(\boldsymbol{\eta}) \propto e^{-\beta||\boldsymbol{\eta}||}, \tag{10}$$

and the parameter β is a function of privacy budget ϵ and the scale of $||\bigtriangledown_{\boldsymbol{\omega}}\mathcal{L}(\boldsymbol{\omega}, e_{ij})||$. To implement this, we pick the norm of $\boldsymbol{\eta}$ from the $\Gamma(k, \beta)$ distribution and the direction of $\boldsymbol{\eta}$ uniformly at random. Then we compute the private output $\widehat{\boldsymbol{\omega}}$, where $\widehat{\boldsymbol{\omega}} = \arg\min_{\boldsymbol{\omega}} \mathcal{L}_{priv}(\boldsymbol{\omega}, \mathcal{G})$ satisfies ϵ-differential privacy.

3 Differentially Private Network Embedding

In this section, we describe the differentially private network embedding method (DPNE) based on the matrix factorization approach to achieve differential privacy in network embedding.

3.1 Differentially Private Network Embedding (DPNE)

DPNE adopts the objective perturbation mechanism on matrix factorization to protect the individual's link privacy in the social network. Note that $\mathbf{M}$ represents logarithm of the average probability that one vertex randomly walks to another vertex within fixed steps. When an edge e_{ij} in $\mathcal{G}$ is added or removed, the entire entries in $\mathbf{M}$ are changed accordingly. Hence, although there are some works [23] to protect the privacy in terms of a value in a matrix, it is not straightforward to adapt the existing models to our scenario. We need to derive the scale of the objective function in terms of changing one edge in $\mathcal{G}$. Differential privacy

In DPNE, we define the perturbed objective function of matrix factorization in Equation 4 as follows:

$$\mathcal{L}_{priv}(\mathbf{w}_i, \mathcal{G}) = \sum_{(i,j)\in\Omega} ||m_{ij} - \mathbf{w}_i\mathbf{h}_j^T||_2^2 + \lambda \sum_{i\in\mathcal{V}} ||\mathbf{w}_i||_2^2 + \mu \sum_{j\in\mathcal{V}_c} ||\mathbf{h}_j||_2^2 + \sum_{i\in\mathcal{V}} \mathbf{w}_i\boldsymbol{\eta}_i^T, \tag{11}$$

where $\mathbf{N} = [\boldsymbol{\eta}_i]_{|\mathcal{V}|\times k}$ is a noise matrix with each row $\boldsymbol{\eta}_i$ of $\mathbf{N}$ as a k-dimensional noise vector. In practice, the context representation matrix $\mathbf{H}$ is kept confidential. We first minimize Equation 4 to update $\mathbf{H}$, then fix $\mathbf{H}$ and learn $\mathbf{W}$ by minimizing Equation 11. Hence, $\mathbf{W}$ is the only matrix variable in Equation 11.

Theorem 1. *Let $\mathbf{M}$ be a matrix where each of its entry m_{ij} is defined by Equation 3. Let $\boldsymbol{\eta}_i$ in Equation 11 be a k-dimensional noise vector that is independently and randomly picked for each vertex v_i from the density function* $\Pr(\boldsymbol{\eta}_i) \propto \exp(-\frac{\epsilon||\boldsymbol{\eta}_i||}{2\Delta})$*, where $\Delta = \max||\mathbf{M}' - \mathbf{M}||$. The derived* $\widehat{\mathbf{W}} = \arg\min_{\mathbf{W}} \mathcal{L}_{priv}(\mathbf{w}_i, \mathcal{G})$ *by minimizing Equation 11 satisfies ϵ-differential privacy.*

Proof. Let $\mathcal{G}$ and $\mathcal{G}'$ be two neighboring graphs differing by one edge, where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$, $\mathcal{E}' = \mathcal{E} \bigcup \{e_{rs}\}$. Let $\mathbf{M}$ and $\mathbf{M}'$ be two matrices derived from $\mathcal{G}$ and $\mathcal{G}'$ following Equation 3. Let $\mathbf{N}$ and $\mathbf{N}'$ be the noise matrices in Equation 11 when training with $\mathcal{G}$ and $\mathcal{G}'$. Meanwhile, $\mathcal{L}_{priv}(\mathbf{w}_i, \mathcal{G})$ is differentiable anywhere.

Let $\bar{\mathbf{W}} = \arg\min_{\mathbf{W}} \mathcal{L}_{priv}(\mathbf{w}_i, \mathcal{G}) = \arg\min_{\mathbf{W}} \mathcal{L}_{priv}(\mathbf{w}_i, \mathcal{G}')$, we have $\forall v_i \in \mathcal{V}$, $\bigtriangledown_{\mathbf{w}_i} \mathcal{L}_{priv}(\bar{\mathbf{w}}_i, \mathcal{G}) = \bigtriangledown_{\mathbf{w}_i} \mathcal{L}_{priv}(\bar{\mathbf{w}}_i, \mathcal{G}') = 0$. Thereby,

$$\boldsymbol{\eta}_i - 2\sum_{j\in\mathcal{V}_c} \mathbf{h}_j(m_{ij} - \bar{\mathbf{w}}_i\mathbf{h}_j^T) = \boldsymbol{\eta}'_i - 2\sum_{j\in\mathcal{V}_c} \mathbf{h}_j(m'_{ij} - \bar{\mathbf{w}}_i\mathbf{h}_j^T); \tag{12}$$

We can derive from Equation 12 that:

$$\boldsymbol{\eta}_i - \boldsymbol{\eta}'_i = 2\sum_{j\in\mathcal{V}_c} \mathbf{h}_j(m'_{ij} - \bar{\mathbf{w}}_i\mathbf{h}_j^T) - 2\sum_{j\in\mathcal{V}_c} \mathbf{h}_j(m_{ij} - \bar{\mathbf{w}}_i\mathbf{h}_j^T) = 2\sum_{j\in\mathcal{V}_c} \mathbf{h}_j(m_{ij} - m'_{ij});$$

$$\sum_{i\in\mathcal{V}}(\boldsymbol{\eta}_i - \boldsymbol{\eta}'_i) = 2\sum_{i\in\mathcal{V}}\sum_{j\in\mathcal{V}_c} \mathbf{h}_j(m_{ij} - m'_{ij});$$

We normalize $||\mathbf{h}_j|| \leq 1$. Since $||\mathbf{M}' - \mathbf{M}|| \leq \Delta$, we have $||\mathbf{N} - \mathbf{N}'|| \leq 2\Delta$ regardless of $\mathbf{H}$. Then for $\mathcal{G}$ and $\mathcal{G}'$,

$$\begin{aligned} \frac{\Pr[W = \bar{W}|\mathcal{G}]}{\Pr[W = \bar{W}|\mathcal{G}']} &= \frac{\prod\limits_{i\in\mathcal{V}} \Pr(\boldsymbol{\eta}_i)}{\prod\limits_{i\in\mathcal{V}} \Pr(\boldsymbol{\eta}'_i)} = \exp(-\frac{\epsilon(\sum\limits_{i\in\mathcal{V}}||\boldsymbol{\eta}_i|| - \sum\limits_{i\in\mathcal{V}}||\boldsymbol{\eta}'_i||)}{2\Delta}) \\ &\leq \exp(-\frac{\epsilon(||\mathbf{N} - \mathbf{N}'||)}{2\Delta}) \leq \exp(\epsilon). \end{aligned} \tag{13}$$

The above proof also holds when we use $\left|\sum_{v_i\in\mathcal{V}} \mathbf{w}_i\boldsymbol{\eta}_i^T\right|$ as the noise term in the perturbed objective function. In our implementation, we use the absolute noise term to get a better performance on the optimization. Next, we show the upper bound of $\max||\mathbf{M}' - \mathbf{M}||$, which will be used for adding noise to the objective function.

Lemma 1. *The L_2-sensitivity of $\mathbf{M}$ is $\max||\mathbf{M}' - \mathbf{M}|| \leq \sqrt{2}$.*

Proof. When window size $t = 1$, $||\mathbf{M}' - \mathbf{M}||$ decreases as degrees of v_r and v_s (the vertices linked by the edge e_{rs} in $\mathcal{G}'$) increase. $||\mathbf{M}' - \mathbf{M}||$ takes the maximum value $\sqrt{2}$ when degrees of v_r and v_s are both 0 in $\mathcal{G}$ and 1 in $\mathcal{G}'$. When window size $t \geq 2$, $||\mathbf{P}'^t - \mathbf{P}^t|| \leq ||\mathbf{P}' - \mathbf{P}||$.

$$\underset{(t\geq 2)}{||\mathbf{M}' - \mathbf{M}||} = \left|\left|\frac{\mathbf{P}' + \ldots + \mathbf{P}'^t}{t} - \frac{\mathbf{P} + \ldots + \mathbf{P}^t}{t}\right|\right| \leq \left|\left|\frac{t(\mathbf{P}' - \mathbf{P})}{t}\right|\right| = \underset{(t=1)}{||\mathbf{M}' - \mathbf{M}||}.$$

3.2 DPNE vs. Other DP-preserving Embedding Approaches

A naive way to achieve differential privacy in network embedding is to get a differentially private matrix $\mathbf{M}$ (dpM) and then to apply matrix factorization, where $\mathbf{M}$ is calculated with the transition matrix $\mathbf{P}$ and its powers $\mathbf{P}^t$. To get a differentially private matrix $\mathbf{M}$, we can use the Laplace mechanism to add a noise matrix $\mathbf{N}$ on $\mathbf{M}$, where each entry n_{ij} of $\mathbf{N}$ is drawn i.i.d. from $Lap(S_f(\mathcal{G})/\epsilon)$, let $S_f(\mathcal{G})$ be the sensitivity of $\mathbf{M}$, $S_f(\mathcal{G}) = \max ||\mathbf{M}' - \mathbf{M}|| \leq \sqrt{2}$. The worst case for $||\mathbf{M}' - \mathbf{M}||$ is when adding an edge to two isolated vertices in $\mathcal{G}$. We will use dpM as a baseline in our empirical evaluation.

Another way to achieve differential privacy in network embedding is to enforce differential privacy in the process of network embedding models. Let D be a vertex-context set generated from the random walk sequences, where each member of D is a vertex-context pair (v_i, v_j). For a "walk step" from v_i to v_j in random walks, it is uncertain how many times in total the same "walk step" appears in D. For example, the number of times that an edge e_{ij} gets walked through by all the random walks sequences has the upper bound $\sum_{i=0}^{\min\{b,\lceil \log_a |\mathcal{V}| \rceil\}} a^i \times c \times (b-i)$, where a is the maximal degree of $\mathcal{G}$, b is the walk length, c is the number of walks starting at each vertex. Stochastic Gradient Descent (SGD) is used in SkipGram to learn the embedding vectors based on the objective function. The input for SGD is the vertex-context set D. Although there are existing works on how to apply objective perturbation [15] or exponential mechanism [21] on SGD to make private updates, the privacy of D and the privacy of $\mathcal{G}$ are not the same. For example, if we want to apply the functional mechanism [11] on DeepWalk with hierarchical softmax [2], in terms of the privacy of D, the hierarchical softmax function iterates one time over each vertex-context pair in D; the sensitivity of the hierarchical softmax function on D is about $\lceil \log_2 |\mathcal{V}| \rceil \left(k/2 + k^2/8\right)$. But in terms of the privacy of $\mathcal{G}$, for each edge in $\mathcal{G}$, the hierarchical softmax function iterates an unset number of times; thus, the sensitivity of the hierarchical softmax function on $\mathcal{G}$ is very large after multiplying the iteration times.

Also, negative sampling is used for SkipGram function in DeepWalk or LINE. The processes of positive sampling and negative sampling already indicate link privacy. It is viable to make the sampling process private by applying the exponential mechanism [9] or the Laplace mechanism [8]. However, the sensitivity of edge sampling domain is also large. Hence, it is challenging to intuitively apply the existing differentially private approaches in DeepWalk to achieve privacy protection on $\mathcal{G}$.

The equivalent matrix factorization approach avoids random walks and negative sampling. The effect of window size in DeepWalk is expressed as the powers of the transition matrix. It considers the expected times that the vertex pairs appear rather than the randomly generated times. It is more convenient to apply differential privacy mechanisms on the matrix factorization based network embedding method. The only remaining challenge is to bridge the "gap" between the privacy of D and the privacy of $\mathcal{G}$. Theorem 1 addresses this problem.

4 Evaluation

In this section, we evaluate the performance of DPNE on two tasks: vertex classification and link prediction. For vertex classification, we predict the label of each vertex in the network by using vertex embeddings as inputs to build a classifier. For link prediction, we aim to use vertex embeddings to predict whether there is an edge between two vertices.

Baselines. We compare our differentially private network embedding method (**DPNE**) with the naive method (**dpM**) and the non-private network embedding as matrix factorization method (**non-priv**).

Datasets. We adopt the following three datasets to evaluate the proposed model. 1) **Wiki** contains 2,405 documents from 19 classes and 17,981 links between them. 2) **Cora** is a research paper set which contains 2,708 machine learning papers from 7 classes and 5,429 links between them. 3) **Citeseer** is another research paper set which contains 3,312 publications from 6 classes and 4,732 links between them.

Parameter Settings. In our experiments, we set the window size $t = 2$ and $\mathbf{M} = (\mathbf{P} + \mathbf{P}^2)/2$. For all three datasets, we choose a series of values for the embedding size $k = \{10, 20, 50, 100, 200, 500, 1000\}$ for vertex representations, and a series of values for the privacy budget $\epsilon = \{0.01, 0.1, 1, 10, 100, 1000\}$. We set the regularization coefficients $\lambda = \mu = 0.001$ and the learning rate $\gamma = 0.015$. The train ratios of SVM and logistic regression in the two tasks are both 10%. For each parameter setting, we report the average result over 10 different runs.

4.1 Vertex Classification Task

For the vertex classification task, we first get the vertex representation $\mathbf{W}$ based on the matrix factorization method. Then, we train a multi-class support vector machine (SVM) on a training dataset L based on a subset of vertex representations $\mathbf{W}_L$ and further predict the labels of a testing dataset U by the SVM classifier based on $\mathbf{W} \backslash \mathbf{W}_L$.

Table 1. Comparing the accuracy for vertex classification with different privacy budget ϵ [non-priv: (a) Wiki 0.555 (b) Cora 0.700 (c) Citeseer 0.505]

Dataset	(a) Wiki		(b) Cora		(c) Citeseer	
ϵ	dpM	DPNE	dpM	DPNE	dpM	DPNE
0.01	0.094	0.093	0.187	0.181	0.179	0.178
0.1	0.091	0.096	0.189	0.213	0.182	0.186
1	0.088	0.521	0.190	0.662	0.177	0.452
10	0.090	0.527	0.196	0.669	0.180	0.459
100	0.355	0.537	0.500	0.677	0.311	0.466
1000	0.545	0.552	0.679	0.700	0.497	0.490

Different Privacy Budgets. We evaluate the performance of two private algorithms on all three datasets with embedding size $k = 100$ and different privacy budgets ϵ. Table 1 shows the comparison results of each method for vertex classification on three datasets. We can observe that both DPNE and dpM have the similar trend on three datasets in terms of accuracy while we increase the privacy budget. The classification results of DPNE are close to the non-private method when $\epsilon \geq 1$. However, the performance of dpM method has a big improvement when $\epsilon \geq 100$. It indicates that, compared with dpM, DPNE can achieve the same performance with a much smaller privacy budget. Meanwhile, when the $\epsilon \leq 0.1$, both private methods have poor performance. It indicates that when the privacy budget is low, the matrix factorization method cannot be converged due to the large noisy injected to the objective function or matrix itself.

Table 2. Comparing the accuracy for vertex classification with different embedding size k and privacy budget ϵ (dataset=Wiki)

ϵ	non-priv	DPNE					
k		0.01	0.1	1	10	100	1000
10	0.324	0.080	0.108	0.255	0.265	0.272	0.350
20	0.466	0.072	0.095	0.420	0.428	0.429	0.457
50	0.537	0.080	0.087	**0.526**	0.520	0.515	0.539
100	**0.555**	0.093	0.096	0.521	**0.527**	**0.537**	**0.552**
200	0.530	0.096	0.101	0.470	0.511	0.515	0.539
500	0.474	0.105	0.110	0.285	0.437	0.424	0.482
1000	0.429	**0.109**	**0.112**	0.169	0.270	0.295	0.405

Different Embedding Sizes. Based on Equation 10, $\boldsymbol{\eta}_i$ increases with embedding size k. There is potentially a compromised performance of DPNE at a larger k. We evaluate the performance of the DPNE algorithm on the Wiki dataset with different embedding size k and privacy budget ϵ. For non-priv, as shown in the second column of Table 2, the highest accuracy 55.5% is achieved when $k = 100$. When increasing or decreasing embedding size k, the accuracy decreases. For DPNE, with relatively large privacy budget, $\epsilon = 10, 100, 1000$, high accuracy is also achieved when $k = 100$. When $\epsilon = 1$, high accuracy is achieved when $k = 50$. However, with very small privacy budget, $\epsilon = 0.01, 0.1$, DPNE has significantly lower accuracy comparing to non-priv no matter how we choose k.

4.2 Link Prediction Task

For the link prediction task, we first use vertex representations to compose edge representations. Given a pair of vertices v_i, v_j connected by an edge, we use an Hadamard operator to combine the vertex vectors $\mathbf{w}_i$ and $\mathbf{w}_j$ to compose the

Table 3. Comparing the accuracy for link prediction under different privacy budget ϵ [non-priv: (a) Wiki 0.734 (b) Cora 0.697 (c) Citeseer 0.699]

Dataset	(a) Wiki		(b) Cora		(c) Citeseer	
ϵ	dpM	DPNE	dpM	DPNE	dpM	DPNE
0.01	0.502	0.520	0.501	0.532	0.498	0.535
0.1	0.502	0.524	0.501	0.541	0.501	0.523
1	0.503	0.743	0.500	0.698	0.503	0.690
10	0.527	0.713	0.552	0.673	0.566	0.666
100	0.708	0.720	0.729	0.666	0.751	0.662
1000	0.725	0.725	0.706	0.703	0.695	0.696

Table 4. Comparing the accuracy for link prediction with different embedding size k and privacy budget ϵ (dataset=Wiki)

ϵ	non-priv	DPNE					
k		0.01	0.1	1	10	100	1000
10	0.607	0.499	0.552	0.614	0.612	0.609	0.605
20	0.648	0.500	0.538	0.661	0.650	0.654	0.649
50	0.717	0.510	0.525	0.718	0.694	0.695	0.718
100	0.734	0.520	0.524	0.743	0.713	0.720	0.725
200	**0.747**	0.532	0.541	**0.763**	0.734	**0.741**	**0.746**
500	0.670	0.550	0.566	0.733	0.778	0.721	0.695
1000	0.650	**0.568**	**0.595**	0.698	**0.784**	0.726	0.679

edge vector $\widehat{\mathbf{e}}_{ij} = \mathbf{w}_i * \mathbf{w}_j$ [24]. Then, we use the constructed edge vectors as inputs to train a logistic regression classifier and adopt the classifier to predict the presence or absence of an edge.

Different Privacy Budgets. We evaluate the performance of two private algorithms on three datasets for link prediction with embedding size $k = 100$ and different privacy budgets ϵ. Table 3 shows the link prediction accuracy of DPNE and dpM. We can observe that when $\epsilon \leq 0.1$, both private algorithms have only about 50% accuracy on three datasets. The accuracy of DPNE has a big improvement while ϵ increases to 1. However, dpM can achieve the comparable results when the $\epsilon \geq 100$. It indicates DPNE can achieve better performance with a small privacy budget.

Different Embedding Sizes. We also evaluate the performance of the DPNE algorithm on the Wiki dataset with different embedding size k and privacy budget ϵ. Table 4 shows our result. For non-priv, it achieves the highest accuracy 74.7% when $k = 200$. For DPNE, it often achieves the highest accuracy with $k = 200$ for large privacy budget values and with $k = 1000$ for small privacy budget values. More interestingly, DPNE even outperforms non-priv at large k for $\epsilon = 10$ or 100.

5 Conclusions and Future Work

In this work, we proposed a differentially private network embedding method (DPNE) based on DeepWalk as matrix factorization. We applied the objective perturbation approach on the objective function of matrix factorization. Our evaluation shows that on both vertex classification and link prediction tasks our DPNE achieves satisfactory performance. DPNE can be easily employed in other network embedding methods if there exists an equivalent matrix factorization of a certain matrix. For example, LINE is proven to be factorizing a similar matrix to **M** [25]. We would derive differential privacy preserving LINE similarly. One potential limitation of the matrix factorization based methods is that they are not scalable to large networks. Graph factorization [26] uses a streaming algorithm for graph partitioning to improve factorization based embedding methods. In future work, we will extend our DPNE to deal with large networks.

Acknowledgments

This work is supported in part by U.S. National Institute of Health (1R01GM103309) and National Science Foundation (1564250, 1523115 and 1502273).

References

1. Goyal, P., Ferrara, E.: Graph embedding techniques, applications, and performance: A survey. CoRR **abs/1705.02801** (2017)
2. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. (2014) 701–710
3. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. (2013) 3111–3119
4. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. (2016) 855–864
5. Li, J., Zhu, J., Zhang, B.: Discriminative deep random walk for network classification. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. (2016)
6. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: LINE: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web. (2015) 1067–1077
7. Yuan, S., Wu, X., Xiang, Y.: Sne: signed network embedding. In: Pacific-Asia conference on knowledge discovery and data mining, Springer (2017) 183–195
8. Dwork, C., McSherry, F., Nissim, K., Smith, A.D.: Calibrating noise to sensitivity in private data analysis. In: Theory of Cryptography Conference. (2006) 265–284
9. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: 48th Annual IEEE Symposium on Foundations of Computer Science. (2007) 94–103
10. Chaudhuri, K., Monteleoni, C., Sarwate, A.D.: Differentially private empirical risk minimization. Journal of Machine Learning Research **12** (2011) 1069–1109

11. Zhang, J., Zhang, Z., Xiao, X., Yang, Y., Winslett, M.: Functional mechanism: regression analysis under differential privacy. Proceedings of the VLDB Endowment **5**(11) (2012) 1364–1375
12. Nissim, K., Raskhodnikova, S., Smith, A.: Smooth sensitivity and sampling in private data analysis. In: Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing, ACM (2007) 75–84
13. Erlingsson, U., Pihur, V., Korolova, A.: Rappor: Randomized aggregatable privacy-preserving ordinal response. In: Proceedings of the 2014 ACM SIGSAC conference on computer and communications security. (2014) 1054–1067
14. Wang, Y., Wu, X., Hu, D.: Using randomized response for differential privacy preserving data collection. In: Proceedings of the 9th International Workshop on Privacy and Anonymity in the Information Society. (2016)
15. Song, S., Chaudhuri, K., Sarwate, A.D.: Stochastic gradient descent with differentially private updates. In: 2013 IEEE Global Conference on Signal and Information Processing. (2013) 245–248
16. Chaudhuri, K., Monteleoni, C.: Privacy-preserving logistic regression. In: Advances in Neural Information Processing Systems. (2008) 289–296
17. Wang, Y., Wu, X., Wu, L.: Differential privacy preserving spectral graph analysis. In: Pacific-Asia conference on knowledge discovery and data mining, Springer (2013) 329–340
18. Xu, D., Yuan, S., Wu, X.: Differential privacy preserving causal graph discovery. In: Privacy-Aware Computing (PAC), 2017 IEEE Symposium on. (2017) 60–71
19. Abadi, M., Chu, A., Goodfellow, I., McMahan, H.B., Mironov, I., Talwar, K., Zhang, L.: Deep learning with differential privacy. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. (2016) 308–318
20. Phan, N., Wang, Y., Wu, X., Dou, D.: Differential privacy preservation for deep auto-encoders: an application of human behavior prediction. In: AAAI. (2016) 1309–1316
21. Zhang, J., Xiao, X., Yang, Y., Zhang, Z., Winslett, M.: Privgene: Differentially private model fitting using genetic algorithms. In: Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. (2013) 665–676
22. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: IJCAI. (2015) 2111–2117
23. Hua, J., Xia, C., Zhong, S.: Differentially private matrix factorization. In: IJCAI. (2015) 1763–1770
24. Grover, A., Leskovec, J.: node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. (2016) 855–864
25. Qiu, J., Dong, Y., Ma, H., Li, J., Wang, K., Tang, J.: 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. (2018) 459–467
26. Ahmed, A., Shervashidze, N., Narayanamurthy, S.M., Josifovski, V., Smola, A.J.: Distributed large-scale natural graph factorization. In: Proceedings of the 22nd international conference on World Wide Web. (2013) 37–48