Differentially Private Link Prediction with Protected Connections

Abir De, Soumen Chakrabarti
Indian Institute of Technology Bombay
{abir, soumen}@cse.iitb.ac.in

Abstract

Link prediction (LP) algorithms propose to each node a ranked list of nodes that are currently non-neighbors, as the most likely candidates for future linkage. Owing to increasing concerns about privacy, users (nodes) may prefer to keep some of their connections protected or private. Motivated by this observation, our goal is to design a differentially private LP algorithm, which trades off between privacy of the protected node-pairs and the link prediction accuracy. More specifically, we first propose a form of differential privacy on graphs, which models the privacy loss only of those node-pairs which are marked as protected. Next, we develop DPLP, a learning to rank algorithm, which applies a monotone transform to base scores from a non-private LP system, and then adds noise. DPLP is trained with a privacy induced ranking loss, which optimizes the ranking utility for a given maximum allowed level of privacy leakage of the protected node-pairs. Under a recently-introduced latent node embedding model, we present a formal trade-off between privacy and LP utility. Extensive experiments with several real-life graphs and several LP heuristics show that DPLP can trade off between privacy and predictive performance more effectively than several alternatives.

1 Introduction

Link prediction (LP) (Liben-Nowell and Kleinberg 2007; Lü and Zhou 2011) is the task of predicting future edges that are likely to emerge between nodes in a social network, given historical views of it. Predicting new research collaborators, new Facebook friends and new LinkedIn connections are examples of LP tasks. LP algorithms usually presents to each node u a ranking of the most promising non-neighbors v of u at the time of prediction.

Owing to the growing concerns about privacy (Tucker 2014; Machanavajjhala, Korolova, and Sarma 2011; Waniek et al. 2019), social media users often prefer to mark their sensitive contacts, *e.g.*, dating partners, prospective employers, doctors, etc. as 'protected' from other users and the Internet at large. However, the social media hosting platform itself continues to enjoy full access to the network, and may exploit them for link prediction. In fact, many popular LP algorithms often leak neighbor information (Waniek et al. 2019), at least in a probabilistic sense, because they take advantage of rampant *triad completion*: when node u links to v, very often there exists a node w such that edges (u, w) and (w, v) already exist. Therefore, recommending v to u allows u to learn about the edge (w, v), which may result in breach of privacy of v and w. Motivated by these observations, a recent line of work (Machanavajjhala, Korolova, and Sarma 2011; Xu et al. 2018) proposes privacy preserving LP methods. While they introduce sufficient noise to ensure a specified level of privacy, they do not directly optimize for the LP ranking utility. Consequently, they show poor predictive performance.

1.1 Present Work

In this work, we design a learning to rank algorithm which aims to optimize the LP utility under a given amount of privacy leakage of the node-pairs (edges or non-edges) which are marked as protected. Specifically, we make the following contributions.

Modeling differential privacy of protected node pairs. Differential privacy (DP) (Dwork and Roth 2014) is concerned with privacy loss of a *single entry* in a database upon (noisy) datum disclosure. As emphasized by Dwork and Roth (2014, Section 2.3.3), the meaning of *single entry* varies across applications, especially in graph databases. DP algorithms over graphs predominantly consider two notions of granularity, *e.g.*, node and edge DP, where one entry is represented by a node and an edge respectively. While node DP provides a stronger privacy guarantee than edge DP, it is often stronger than what is needed for applications. As a result, it can result in unacceptably low utility (Song et al. 2018; Dwork and Roth 2014). To address this challenge, we adopt a variant of differential privacy, called protected-pair differential privacy, which accounts for the loss of privacy of only those node-pairs (edges or non edges), which are marked as private. Such a privacy model is well suited for LP in social networks, where users may hide only some sensitive contacts and leave the remaining relations public (Dwork and Roth 2014). It is much stronger than edge DP, but weaker than node DP. However, it serves the purpose in the context of our current problem.

Maximizing LP accuracy subject to privacy constraints. Having defined an appropriate granularity of privacy, we develop DPLP, a perturbation wrapper around a base LP algo-

rithm, that renders it private in the sense of node-pair protection described above. Unlike DP disclosure of *quantities*, the numeric scores from the base LP algorithm are not significant in themselves, but induce a ranking. This demands a fresh approach to designing DPLP, which works in two steps. First, it transforms the base LP scores using a monotone deep network (Wehenkel and Louppe 2019), which ensures that the ranks remain unaffected in absence of privacy. Noise is then added to these transformed scores, and the noised scores are used to sample a privacy-protecting ranking over candidate neighbors. DPLP is trained using a ranking loss (AUC) surrogate. Effectively, DPLP obtains a sampling distribution that optimizes ranking utility, while maintaining privacy for protected pairs. Extensive experiments show that DPLP outperforms state-of-the-art baselines (Machanavajjhala, Korolova, and Sarma 2011; McSherry and Talwar 2007; Geng and Viswanath 2015) in terms of predictive accuracy, under identical privacy constraints[1].

Bounding ranking utility subject to privacy. Finally, we theoretically analyze the ranking utility provided by three LP algorithms (Liben-Nowell and Kleinberg 2007), viz., Adamic-Adar, Common Neighbors and Jaccard Coefficient, in presence of privacy constraints. Even without privacy constraints, these and other LP algorithms are heuristic in nature and can produce imperfect rankings. Therefore, unlike Machanavajjhala, Korolova, and Sarma (2011), we model utility not with reference to a non-DP LP algorithm, but a generative process that creates the graph. Specifically, we consider the generative model proposed by Sarkar, Chakrabarti, and Moore (2011), which establishes predictive power of popular (non-DP) LP heuristics. They model latent node embeddings in a geometric space (Hoff, Raftery, and Handcock 2002) as causative forces that drive edge creation but without any consideration for privacy. In contrast, we bound the *additional loss* in ranking utility when privacy constraints are enforced and therefore, our results change the setting and require new techniques.

1.2 Related Work

The quest for correct privacy protection granularity has driven much work in the DP community. Kearns et al. (2015) propose a model in which a fraction of nodes are completely protected, whereas the others do not get any privacy assurance. This fits certain applications *e.g.*, epidemic or terrorist tracing, but not LP, where a node/user may choose to conceal only a few sensitive contacts (*e.g.*, doctor or partner) and leave others public. Edge/non-edge protection constraints can be encoded in the formalism of group (Dwork 2011), Pufferfish (Song, Wang, and Chaudhuri 2017; Kifer and Machanavajjhala 2014; Song and Chaudhuri 2017) and Blowfish (He, Machanavajjhala, and Ding 2014) privacy. However, these do not focus on LP tasks on graphs. Chierichetti et al. (2015) partition a graph into private and public subgraphs, but for designing efficient sketching and sampling algorithms, not directly related to differentially private LP objectives. In particular, they do not use any information from private graphs; however, a differentially private algorithm does use private information after adding noise into it. Moreover, their algorithms do not include any learning to rank method for link prediction. Abadi et al. (2016) design a DP gradient descent algorithm for deep learning, which however, does not aim to optimize the associated utility. Moreover, they work with pointwise loss function, whereas our objective is a pairwise loss. Other work (Geng and Viswanath 2015; Geng et al. 2018; Ghosh, Roughgarden, and Sundararajan 2012) aims to find optimal noise distribution for queries seeking real values. However, they predominantly use variants of noise power in their objective, whereas we use ranking loss, designed specifically for LP tasks.

[1]Code: https://www.cse.iitb.ac.in/~abir/codes/dplp.zip

2 Preliminaries

In this section, we first set up the necessary notation and give a brief overview of a generic non-private LP algorithm on a social network. Then we introduce the notion of protected and non-protected node pairs.

2.1 Social Network And LP Algorithm

We consider a snapshot of the social network as an undirected graph $G = (V, E)$ with vertices V and edges E. Neighbors and non-neighbors of node u are denoted as the sets $\mathrm{nbr}(u)$ and $\overline{\mathrm{nbr}}(u)$, respectively. Given this snapshot, a non-private LP algorithm $\mathcal{A}$ first implements a scoring function $s_{\mathcal{A}}(u, \cdot) : \overline{\mathrm{nbr}}(u) \to \mathbb{R}^+$ and then sorts the scores $s_{\mathcal{A}}(u, v)$ of all the non-neighbors $v \in \overline{\mathrm{nbr}}(u)$ in decreasing order to obtain a ranking of the potential neighbors for each query node u. Thus, the LP algorithm $\mathcal{A}$ provides a map: $\pi_u^{\mathcal{A}} : \{1, 2, \ldots, |\overline{\mathrm{nbr}}(u)|\} \leftrightarrow_{1:1} \overline{\mathrm{nbr}}(u)$. Here, $\pi_u^{\mathcal{A}}(i)$ represents the node at rank i, recommended to node u by algorithm $\mathcal{A}$. For brevity, we sometimes denote $u_i^{\mathcal{A}} = \pi_u^{\mathcal{A}}(i)$. Note that, although $\pi_u^{\mathcal{A}}$ can provide ranking of all candidate nodes, most LP applications consider only top K candidates for some small value of K. When clear from context, we will use $\pi_u^{\mathcal{A}}$ to denote the top-K items.

2.2 Protected And Non-protected Node-pairs

We assume that each node $u \in V$ partitions all other nodes $V \backslash \{u\}$ into a protected node set $\mathrm{prot}(u)$ and a non-protected/public node set $\mathrm{pub}(u)$. We call (u, w) a protected node-pair if either w marks u as protected, *i.e.*, $u \in \mathrm{prot}(w)$ or vice-versa, *i.e.*, $w \in \mathrm{prot}(u)$. In general, any node u can access three sets of information: (i) its own connections with other nodes, *i.e.*, $\mathrm{nbr}(u)$ and $\overline{\mathrm{nbr}}(u)$; (ii) the possible connections between any node w (not necessarily a neighbor) with nodes $\mathrm{pub}(w)$ not protected by w; and, (iii) the ranked list of nodes recommended to itself by any LP algorithm. Note that a node-pair (u, w) is accessible to u even if $u \in \mathrm{prot}(w)$, since u knows its own edges (and non-edges). On the other hand, if a third node $v \in V \backslash \{u, w\}$ is marked as protected by w, *i.e.*, $v \in \mathrm{prot}(w)$, then u should not know whether $v \in \mathrm{nbr}(w)$ or $v \in \overline{\mathrm{nbr}}(w)$. However, when a non-private LP algorithm recommends nodes to u, u can exploit them to reverse-engineer such protected links. For example, if w is connected to both u and v, then a triad-completion based LP heuristic (Adamic-Adar or Common-Neighbor) is

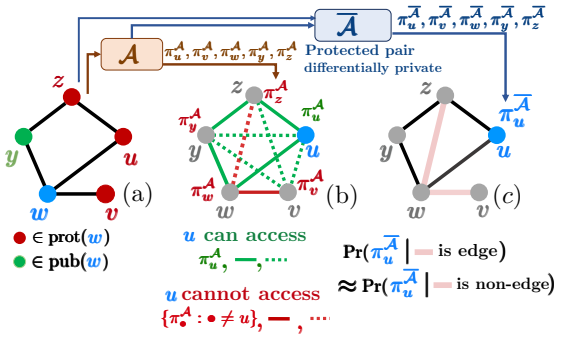

Figure 1: Problem setup. (a) Nodes in $\mathrm{prot}(w)$ colored red; nodes in $\mathrm{pub}(w)$ colored green. (b) u cannot see red edges (solid), red non-edges (dotted) and recommendations $(\pi^{\mathcal{A}}_{\bullet}, \bullet \neq u)$ to other nodes. u can see green edges (solid), green non-edges (dotted) and recommendation $(\pi^{\mathcal{A}}_{u})$ to itself. (c) Output of $\overline{\mathcal{A}}$ should be insensitive to protected pairs.

likely to recommend v to the node u, which in turn allows u to guess the presence of the edge (v, w). Figure 1 illustrates our problem setup.

3 Privacy Model For Protected Pairs

In this section, we first present our privacy model which assesses the privacy loss of protected node-pairs, given the predictions made by an LP algorithm. Then we introduce the sensitivity function, specifically designed for graphs with protected pairs.

3.1 Protected-pair Differential Privacy

In order to the prevent privacy leakage of the protected connections, we aim to design a privacy preserving LP algorithm $\overline{\mathcal{A}}$ based on $\mathcal{A}$, meaning that $\overline{\mathcal{A}}$ uses the scores $s_{\mathcal{A}}(u, \cdot)$ to present to each node u a ranking $\pi^{\overline{\mathcal{A}}}_{u}$, while maintaining the privacy of all the protected node pairs in the graph, where one of the two nodes in such a pair is not u itself. We expect that u has strong reasons to link to the top nodes in $\pi^{\overline{\mathcal{A}}}_{u}$, and that the same list will occur again with nearly equal probability, if one changes (adds or deletes) edges between any other node w and the protected nodes of w, other than u, *i.e.*, $\mathrm{prot}(w)\backslash\{u\}$. We formally define such changes in the graph in the following.

Definition 1 (**u-neighboring graphs**). *Given two graphs $G = (V, E)$ and $G' = (V, E')$ with the same node set V, and a node $u \in V$, we call G, G' as u-neighboring graphs, if there exists some node $w \neq u$ (not necessarily a neighbor), such that*

1. *protected nodes $\mathrm{prot}(w)$ are the same in G and G',*
2. *public nodes $\mathrm{pub}(w)$ are the same in G and G', and*
3. *$E'\backslash\{(w, v) \mid v \in \mathrm{prot}(w)\backslash u\} = E\backslash\{(w, v) \mid v \in \mathrm{prot}(w)\backslash u\}$.*

The set of all u-neighboring graph pairs is denoted as $\mathfrak{N}_u$.

In words, the last requirement is that, for the purpose of recommendation to u, the information not protected by w is the same in G and G'. u-neighboring graphs may differ only in the protected pairs incident to some other node w, except (w, u). Next, we propose our privacy model which characterizes the privacy leakage of protected node-pairs in the context of link prediction.

Definition 2 (**Protected-pair differential privacy**). *We are given a non-private base LP algorithm $\mathcal{A}$, and a randomized LP routine $\overline{\mathcal{A}}$ based on $\mathcal{A}$, which returns a ranking of K nodes. We say that "$\overline{\mathcal{A}}$ ensures ϵ_p-protected-pair differential privacy", if, for all nodes $u \in V$ and for all ranking R_K on the candidate nodes, truncated after position K, we have $\Pr(\pi^{\overline{\mathcal{A}}}_{u} = R_K \,|\, \mathcal{A}, G) \leq e^{\epsilon_p} \Pr(\pi^{\overline{\mathcal{A}}}_{u} = R_K \,|\, \mathcal{A}, G')$, whenever G and G' are u-neighboring graphs.*

In words, $\overline{\mathcal{A}}$ promises to recommend the same list to u, with nearly the same probability, even if we arbitrarily change the protected partners of any node w, except for u itself, as that information is already known to u.

3.2 Graph Sensitivity

Recall that $\mathcal{A}$ recommends v to u based on a score $s_{\mathcal{A}}(u, v)$. A key component of our strategy is to apply a (monotone) transform $f(s_{\mathcal{A}}(u, v))$ to balance it more effectively against samples from a noise generator. To that end, given base LP algorithm $\mathcal{A}$, we define the sensitivity of any function on the scores $f(s_{\mathcal{A}}(u, \cdot)) : V \to \mathbb{R}^+$, which will be subsequently used to design any protected-pair differentially private LP algorithm.

Definition 3 (**Sensitivity of $f \circ s_{\mathcal{A}}$**). *The sensitivity $\Delta_{f,\mathcal{A}}$ of a transformation f on the scoring function $s_{\mathcal{A}}$ is defined as*

$$\Delta_{f,\mathcal{A}} = \max_{u} \max_{(G,G') \in \mathfrak{N}_u} \max_{v} \big| f(s_{\mathcal{A}}(u, v|G)) - f(s_{\mathcal{A}}(u, v|G')) \big| \quad (1)$$

Recall $\mathfrak{N}_u$ represents all u-neighboring graph pairs.

4 Design of Privacy Protocols for LP

In this section, we first state our problem of designing a privacy preserving LP algorithm $\overline{\mathcal{A}}$— based on a non-private LP algorithm $\mathcal{A}$— which aims to provide an optimal trade-off between privacy leakage of the protected node pairs and LP accuracy. Next, we develop DPLP, our proposed algorithm which solves this problem,

4.1 Problem Statement

Privacy can always be trivially protected by ignoring private data and sampling each node uniformly, but such an LP 'algorithm' has no predictive value. Hence, it is important to design a privacy preserving LP algorithm which can also provide a reasonable LP accuracy. To this end, given a base LP algorithm $\mathcal{A}$, our goal is to design a protected-pair differentially private algorithm $\overline{\mathcal{A}}$ *based on* $\mathcal{A}$, which maximizes the link prediction accuracy. $\overline{\mathcal{A}}$ uses random noise, and induces a distribution $\pi^{\overline{\mathcal{A}}}_{u}$ over rankings presented to u. Specifically, our goal is solve the following optimization problem:

$$\text{maximize}_{\overline{\mathcal{A}}} \sum_{u \in V} \mathbb{E}_{R_K \sim \Pr(\pi^{\overline{\mathcal{A}}}_{u} | \mathcal{A}, G)} \mathrm{AUC}(R_K) \quad (2)$$

$$\text{s.t.} \quad \overline{\mathcal{A}} \quad \text{is } K\epsilon_p \text{ protected-pair differentially private.}$$

Here, AUC(R_K) measures the area under the ROC curve given by R_K— the ranked list of recommended nodes truncated at K. Hence, the above optimization problem maximizes the expected AUC over the rankings generated by the randomized protocol $\overline{\mathcal{A}}$, based on the scores produced by $\mathcal{A}$. The constraint indicates that predicting *a single link* using $\overline{\mathcal{A}}$ requires to satisfy ϵ_p-protected-pair differential privacy.

4.2 DPLP: A Protected-pair Differentially Private Link Predictor

A key question is whether well-known generic privacy protocols like Laplace or exponential noising are adequate solvers of the optimization problem in Eq. (2), or might a solution tailored to protected-pair differential privacy in LP do better. We answer in the affirmative by presenting DPLP, a novel DP link predictor, with five salient components.

1. Monotone transformation of the original scores from $\mathcal{A}$.
2. Design of a ranking distribution $\pi_u^{\overline{\mathcal{A}}}$, which suitably introduces noise to the original scores to make a sampled ranking insensitive to protected node pairs.
3. Re-parameterization of the ranking distribution to facilitate effective training.
4. Design of a tractable ranking objective that approximates AUC in Eq. (2).
5. Sampling current non-neighbors for recommendation.

Monotone transformation of base scores. We equip $\overline{\mathcal{A}}$ with a trainable monotone transformation $f : \mathbb{R}^+ \to \mathbb{R}^+$, which is applied to base scores $s_{\mathcal{A}}(u, \cdot)$. Transform f helps us more effectively balance signal from $\mathcal{A}$ with the privacy-assuring noise distribution we use. We view this transformation from two perspectives.

First, $f(s_{\mathcal{A}}(u, \cdot))$ can be seen as a new scoring function which aims to boost the ranking utility of the original scores $s_{\mathcal{A}}(u, \cdot)$, under a given privacy constraint. In the absence of privacy constraints, we can reasonably demand that the ranking quality of $f \circ s_{\mathcal{A}}$ be at least as good as $s_{\mathcal{A}}$. In other words, we expect that the transformed and the original scores provide the same ranking utility, i.e., the same list of recommended nodes, in the absence of privacy constraints. A monotone f ensures such a property.

Second, as we shall see, f can also be seen as a *probability distribution transformer*. Given a sufficiently expressive neural representation of f (as designed in Section 4.3), any arbitrary distribution on the original scores can be fitted with a tractable known distribution on the transformed scores. This helps us build a high-capacity but tractable ranking distribution $\Pr(\pi_u^{\overline{\mathcal{A}}} \mid \mathcal{A}, G)$, as described below.

Design of a ranking distribution. Having transformed the scores $s_{\mathcal{A}}(u, .)$ into $f(s_{\mathcal{A}}(u, v))$, we draw nodes v using an iterative exponential mechanism on the transformed scores $f(s_{\mathcal{A}}(u, v))$. Specifically, we draw the first node v with probability proportional to $\exp(\epsilon_p f(s_{\mathcal{A}}(u, v))/2\Delta_{f,\mathcal{A}})$, then repeat for collecting K nodes in all. Thus, our ranking distribution in Eq. (2) is:

$$\Pr(\pi_u^{\overline{\mathcal{A}}} = R_K | \mathcal{A}, G) = \prod_{j=1}^{K} \frac{\exp\left(\frac{\epsilon_p f(s_{\mathcal{A}}(u, R_K(j)))}{2\Delta_{f,\mathcal{A}}}\right)}{\sum_{w \notin R_K(1,..,j-1)} \exp\left(\frac{\epsilon_p f(s_{\mathcal{A}}(u,w))}{2\Delta_{f,\mathcal{A}}}\right)}, \quad (3)$$

In the absence of privacy concerns ($\epsilon_p \to \infty$), the candidate node with the highest score $f(s_{\mathcal{A}}(u, .))$ gets selected in each step, and the algorithm reduces to the original non-private LP algorithm $\mathcal{A}$, thanks to the monotonicity of f. On the other hand, if $\epsilon_p \to 0$, then every node has an equal chance of getting selected, which preserves privacy but has low predictive utility.

Indeed, our ranking distribution in Eq. (3) follows an exponential mechanism on the new scores $f(s_{\mathcal{A}}(u, \cdot))$. However, in principle, such a mechanism can capture any arbitrary ranking distribution $\Pr(\pi_u^{\overline{\mathcal{A}}} | \mathcal{A}, G)$, given sufficient training and expressiveness of f.

Finally, we note that f can be designed to limit $\Delta_{f,\mathcal{A}}$. E.g., if we can ensure that $f(\cdot)$ is positive and bounded by B, then we can have $\Delta_{f,\mathcal{A}} \leq B$; if we can ensure that the derivative $f'(\cdot)$ is bounded by B', then, by the Lipschitz property, $\Delta_{f,\mathcal{A}} \leq B' \Delta_{\mathcal{A}}$.

Re-parameterization of the ranking distribution. In practice, the above-described procedure for sampling a top-K ranking results in high estimation variance during training (Zhao et al. 2011; Marbach and Tsitsiklis 2003; Peters and Schaal 2006; Sehnke et al. 2010). To overcome this problem, we make a slight modification to the softmax method. We first add i.i.d Gumbel noise $\eta_{u,v} \sim \text{GUM}(2\Delta_{f,\mathcal{A}}/\epsilon_p)$ to each score $f(s_{\mathcal{A}}(u, v))$ and then take top K nodes with highest noisy score (Durfee and Rogers 2019, Lemma 4.2). Such a distribution allows an easy re-parameterization trick — $\text{GUM}(2\Delta_{f,\mathcal{A}}/\epsilon_p) = (2\Delta_{f,\mathcal{A}}/\epsilon_p) \cdot \text{GUM}(1)$ — which reduces the parameter variance. With such a re-parameterization of the softmax ranking distribution in Eq. (3), we get the following alternative representation of our objective in Eq. (2):

$$\sum_{u \in V} \mathop{\mathbb{E}}_{\{\eta_{u,\bullet}\} \sim \text{GUM}(1)} \text{AUC}\left(R_K\left(\left\{f(s_{\mathcal{A}}(u, \bullet) + \frac{2\Delta_{f,\mathcal{A}}\eta_{u,\bullet}}{\epsilon_p}\right\}\right)\right),$$

where $R_K(.)$ gives a top-K ranking based on the noisy transformed scores $\{f(s_{\mathcal{A}}(u, \bullet) + 2\Delta_{f,\mathcal{A}}\eta_{u,\bullet}/\epsilon_p\}$ over the candidate nodes for recommendation to u.

Designing a tractable ranking objective. Despite the aforementioned re-parameterization trick, standard optimization tools face several challenges to maximize the above objective. First, optimizing AUC is an NP-hard problem. Second, truncating the list up to K nodes often pushes many neighbors out of the list at the initial stages of learning, which can lead to poor training (Liu 2009). To overcome these limitations, we replace AUC (Gao and Zhou 2015) with a surrogate pairwise hinge loss $\ell(\cdot)$ and consider a non-truncated list $R_{|V|-1}$ of all possible candidate nodes *during training* (Gao and Zhou 2015; Joachims 2005). Hence our

Algorithm 1 DPLP: Learns $\overline{\mathcal{A}}$ based on a non-private LP algorithm $\mathcal{A}$ and then uses it to recommend K nodes to u.

Input $G = (V, E)$; protected and non-protected node-sets $\text{prot}(\cdot), \text{pub}(\cdot)$; scores $s_{\mathcal{A}}(.,.)$ given by $\mathcal{A}$; privacy leakage ϵ_p, margin ϱ.
Output R_K : List of top-K recommended nodes to node u
Initialize $R_K(1 \dots K) \leftarrow \emptyset$
$f \leftarrow \text{TRAIN}(G, \{s_{\mathcal{A}}(w, v) \mid v \in \text{pub}(w), u \in V\}, \epsilon_p, \varrho)$
candidates $\leftarrow \overline{\text{nbr}}(u)$
for j in $1 \dots K$ **do**
  $w \sim \text{SOFTMAX}\left(\{\epsilon_p f(s_{\mathcal{A}}(u, v))/2\Delta_{f,\mathcal{A}} \mid v \in \text{candidates}\}\right)$
  candidates $\leftarrow$ candidates$\backslash\{w\}$
  $R_K(j) \leftarrow w$
Return R_K

optimization problem becomes:

$$\min_f \textstyle\sum_{u \in V} \mathbb{E}_{\{\eta_u, \bullet\} \sim \text{GUM}(1)} \ell\left(\{f(s_{\mathcal{A}}(u, \bullet)), \eta_{u\bullet}\}; G\right) \quad (4)$$

where $\ell\left(\{f(s_{\mathcal{A}}(u, \bullet)), \eta_{u\bullet}\}; G\right)$ is given by a pairwise ranking loss surrogate over the noisy transformed scores:

$$\sum_{\substack{g \in \text{nbr}(u) \cap \text{pub}(u) \\ b \in \overline{\text{nbr}}(u) \cap \text{pub}(u)}} \text{ReLU}\Bigg[\varrho + f(s_{\mathcal{A}}(u, b)) + \frac{2\Delta_{f,\mathcal{A}}\eta_{ub}}{\epsilon_p} - f(s_{\mathcal{A}}(u, g)) - \frac{2\Delta_{f,\mathcal{A}}\eta_{ug}}{\epsilon_p}\Bigg]. \quad (5)$$

The above loss function encourages that the (noisy) transformed score of a neighbor g exceeds the corresponding score of a non-neighbor b by at least (a tunable) margin ϱ. In absence of privacy concern, *i.e.*, $\epsilon_p \to \infty$, it is simply $\text{ReLU}\left[\varrho + f(s_{\mathcal{A}}(u, b)) - f(s_{\mathcal{A}}(u, g))\right]$, which would provide the same ranking as $\mathcal{A}$, due to the monotonicity of f.

We would like to point out that the objective in Eq 5 uses only non-protected pairs $\{(u, v) | v \in \text{pub}(u)\}$ to train f. This ensures no privacy is leaked during training.

Sampling nodes for recommendation. Once we train f by solving the optimization problem in Eq. (4), we draw top-K candidate nodes using the trained f. Specifically, we first sample a non-neighbor v with probability proportional to $\exp\left(\epsilon_p f(s_{\mathcal{A}}(u, v))/2\Delta_{f,\mathcal{A}}\right)$ and then repeat the same process on the remaining non-neighbors $\overline{\text{nbr}}(u)\backslash\{v\}$ and continue K times.

Overall algorithm. We summarize the overall algorithm in Algorithm 1, where the TRAIN($\bullet$) routine solves the optimization problem in Eq. (4). Since there is no privacy leakage during training, the entire process — from training to test — ensures $K\epsilon_p$ protected-pair differential privacy, which is formalized in the following proposition and proven in (De and Chakrabarti 2020).

Proposition 4. *Algorithm 1 implements $K\epsilon_p$-protected-pair differentially private LP.*

Incompatibility with node and edge DP. Note that, our proposal is incompatible with node or edge privacy (where each node or edge is protected). In those cases, objective in Eq. (5) cannot access any data in the training graph. However, our proposal is useful in many applications like online social networks, in which, a user usually protects a fraction of his sensitive connections, while leaving others public.

4.3 Structure Of The Monotonic Transformation

We approximate the required nonlinear function f with $f_{\boldsymbol{\theta}}$, implemented as a monotonic neural network with parameters $\boldsymbol{\theta}$. Initial experiments suggested that raising the raw score $s_{\mathcal{A}}(u, v)$ to a power $a > 0$ provided a flexible trade-off between privacy and utility over diverse graphs. We provide a basis set $\{a_i > 0\}$ of fixed powers and obtain a positive linear combination over raw scores $s_{\mathcal{A}}(u, \cdot)$ raised to these powers. More specifically, we compute:

$$\nu_{\boldsymbol{\beta}}(s_{\mathcal{A}}(u, v)) = \sum_{i=1}^{n_a} e^{\tau\beta_i} (s_{\mathcal{A}}(u, v))^{a_i}, \quad (6)$$

where $(a_i)_{i \in [n_a]} > 0$ are fixed a priori. $\boldsymbol{\beta} = (\beta_i)_{i \in [n_a]}$ are trainable parameters and τ is a temperature hyperparameter. This gave more stable gradient updates than letting a float as a variable.

While using $\nu_{\boldsymbol{\beta}}(s_{\mathcal{A}}(u, v))$ as $f(s_{\mathcal{A}}(u, v))$ is already an improvement upon raw score $s_{\mathcal{A}}(u, v)$, we propose a further refinement: we feed $\nu_{\boldsymbol{\beta}}(s_{\mathcal{A}}(u, v))$ to a Unconstrained Monotone Neural Network (UMNN) (Wehenkel and Louppe 2019) to finally obtain $f_{\boldsymbol{\theta}}(s_{\mathcal{A}}(u, v))$. UMNN is parameterized using the integral of a positive nonlinear function:

$$f_{\boldsymbol{\theta}}(s_{\mathcal{A}}(u, v)) = \int_0^{\nu_{\boldsymbol{\beta}}(s_{\mathcal{A}}(u,v))} g_{\boldsymbol{\phi}}(s) ds + b_0, \quad (7)$$

where $g_{\boldsymbol{\phi}}(.)$ is a positive neural network, parameterized by $\boldsymbol{\phi}$. UMNN offers an unconstrained, highly expressive parameterization of monotonic functions, since the monotonicity herein is achieved by only enforcing a positive integrand $g_{\boldsymbol{\phi}}(\cdot)$. In theory, with sufficient training and capacity, a 'universal' monotone transformation may absorb $\nu_\beta(\cdot)$ into its input stage, but, in practice, we found that omitting $\nu_\beta(\cdot)$ and passing $s_{\mathcal{A}}(u, v)$ directly to UMNN gave poorer results (De and Chakrabarti 2020). Therefore, we explicitly provide $\nu_{\boldsymbol{\beta}}(s_{\mathcal{A}}(u, v))$ as an input to it, instead of the score $s_{\mathcal{A}}(u, v)$ itself. Overall, we have $\boldsymbol{\theta} = \{\boldsymbol{\beta}, \boldsymbol{\phi}, b_0\}$ as the trainable parameters.

5 Privacy-accuracy Trade-off Of DPLP

It is desirable that a privacy preserving LP algorithm should be able to hide the protected node-pairs as well as give high predictive accuracy. However, in principle, LP accuracy is likely to deteriorate with higher privacy constraint (lower ϵ_p). For example, a uniform ranking distribution, *i.e.*, $\epsilon_p \to 0$ provides extreme privacy. However, it would lead to a poor prediction. On the other hand, in absence of privacy, *i.e.*, when $\epsilon_p \to \infty$, the algorithm enjoys an unrestricted use of the protected pairs, which is likely to boost its accuracy. In this section, we analyze this trade-off from two perspectives.

1. **Relative trade-off analysis.** Here, we assess the *relative* loss of prediction quality of $\overline{\mathcal{A}}$ with respect to the observed scores of the the base LP algorithm $\mathcal{A}$.
2. **Absolute trade-off analysis.** Here, we assess the loss of *absolute* prediction quality of $\overline{\mathcal{A}}$ with respect to the true utility provided by a latent graph generative process.

5.1 Relative Trade-off Analysis

Like other privacy preserving algorithms, DPLP also introduces some randomness to the base LP scores, and therefore, any DPLP algorithm $\overline{\mathcal{A}}$ may reduce the predictive quality of $\mathcal{A}$. To formally investigate the loss, we quantify the loss in scores suffered by $\overline{\mathcal{A}}$, compared to $\mathcal{A}$:

$$\gamma_u(\mathcal{A}, \epsilon_p) = \mathbb{E}_{\overline{\mathcal{A}}}\Big[\sum_{i\in[K]} s_{\mathcal{A}}(u, u_i^{\mathcal{A}}|G) - \sum_{i\in[K]} s_{\mathcal{A}}(u, u_i^{\overline{\mathcal{A}}}|G)\Big].$$

Recall that $u_i^{\mathcal{A}} = \pi_u^{\mathcal{A}}(i)$ and $u_i^{\overline{\mathcal{A}}} = \pi_u^{\overline{\mathcal{A}}}(i)$. We do not take absolute difference because the first term cannot be smaller than the second. The expectation is over randomness introduced by the algorithm $\overline{\mathcal{A}}$. The following theorem bounds $\gamma_u(\mathcal{A}, \epsilon_p)$ in two extreme cases (Proven in (De and Chakrabarti 2020)).

Proposition 5. *If ϵ_p is the privacy parameter and $\kappa_{u,\mathcal{A}} := \max_{i\in[|V|-1]}\{s_{\mathcal{A}}(u, u_i^{\mathcal{A}}) - s_{\mathcal{A}}(u, u_{i+1}^{\mathcal{A}})\}$, then we have:*

$$\lim_{\epsilon_p\to\infty} \gamma_u(\mathcal{A}, \epsilon_p) = 0 \text{ and, } \lim_{\epsilon_p\to 0} \gamma_u(\mathcal{A}, \epsilon_p) \le K\kappa_{u,\mathcal{A}}.$$

Here, $\kappa_{u,\mathcal{A}}$ gives maximum difference between the scores of two consecutive nodes in the ranking provided by $\mathcal{A}$ to u.

5.2 Absolute Trade-off Analysis

Usually, $\mathcal{A}$ provides only imperfect predictions. Hence, the above trade-off analysis which probes into the relative cost of privacy with respect to $\mathcal{A}$, may not always reflect the real effect of privacy on the predictive quality. To fill this gap, we need to analyze the *absolute* prediction quality — utility in terms of the true latent generative process — which is harder to analyze. Even for a non-private LP algorithms $\mathcal{A}$, absolute quality is rarely analyzed, except for Sarkar, Chakrabarti, and Moore (2011), which we discuss below.

Latent space network model. We consider the latent space model proposed by Sarkar, Chakrabarti, and Moore (2011), which showed why some popular LP methods succeed. In their model, the nodes lie within a D-dimensional hypersphere having unit volume. Each node u has a co-ordinate $\boldsymbol{x}_u$ within the sphere, drawn uniformly at random. Given a parameter $r > 0$, the latent generative rule is that nodes u and v get connected if the distance $d_{uv} = ||\boldsymbol{x}_u - \boldsymbol{x}_v||_2 < r$.

Relating ranking loss to absolute prediction error. If an oracle link predictor could know the underlying generative process, then its ranking π_u^* would have sorted the nodes in the increasing order of distances $d_{u\bullet}$. An imperfect non-private LP algorithm $\mathcal{A}$ does not have access to these distances and therefore, it would suffer from some prediction error. $\overline{\mathcal{A}}$, based on $\mathcal{A}$, will incur an additional prediction error due to its privacy preserving randomization. We quantify this absolute loss by extending our loss function in Eq. (4):

$$\ell(\pi_u^{\overline{\mathcal{A}}}; \pi_u^*) := \textstyle\sum_{i<j\le K}[d_{uu_i^{\overline{\mathcal{A}}}} - d_{uu_j^{\overline{\mathcal{A}}}}]_+. \qquad (8)$$

Here, recall that $u_i^{\overline{\mathcal{A}}} = \pi_u^{\overline{\mathcal{A}}}(i)$. Analyzing the above loss for any general base LP method, especially deep embedding methods like GCN, is extremely challenging. Here we consider simple triad based models *e.g.*, Adamic Adar (AA), Jaccard coefficients (JC) and common neighbors (CN). For such base LP methods, we bound this loss in the following theorem (Proven in (De and Chakrabarti 2020)).

Theorem 6. *Given $\epsilon = \sqrt{\frac{2\log(2/\delta)}{|V|}} + \frac{7\log(2/\delta)}{3(|V|-1)}$. For $\mathcal{A} \in \{AA, CN, JC\}$, with probability $1 - 4K^2\delta$:*

$$\mathbb{E}_{\overline{\mathcal{A}}}\left[\ell(\pi_u^{\overline{\mathcal{A}}}; \pi_u^*)\right] = O\left(\left[2K\epsilon + \gamma_u(\mathcal{A}, \epsilon_p)/|V|\right]^{\frac{1}{KD}}\right), \quad (9)$$

where D is the dimension of the underlying hypersphere for the latent space random graph model.

The $2K\epsilon$ term captures the predictive loss due to imperfect LP algorithm $\mathcal{A}$ and $\gamma_u(\mathcal{A}, \epsilon_p)/|V|$ characterizes the excess loss from privacy constraints. Note that ϵ here is unrelated to privacy level ϵ_p; it characterizes the upper bounds of the expected losses due to $\mathcal{A}$, and it depends on δ that quantifies the underlying confidence interval. The expectation is taken over random choices made by differentially private, and not randomness in the process generating the data.

6 Experiments

We report on experiments with five real world datasets to show that DPLP can trade off privacy and the predictive accuracy more effectively than three standard DP protocols (Geng and Viswanath 2015; Machanavajjhala, Korolova, and Sarma 2011; McSherry and Talwar 2007). In the extended draft (De and Chakrabarti 2020), we provide additional details about the experiments and more experimental results.

6.1 Experimental Setup

Datasets. We use five networks: Facebook (Leskovec and Mcauley 2012), USAir (Batagelj and Mrvar 2006), Twitter (Leskovec and Mcauley 2012), Yeast (Von Mering et al. 2002), PB (Ackland et al. 2005).

Evaluation protocol and metrics. For each of these datasets, we first mark a portion (σ_{prot}) of all connections to be protected. Following previous work (Backstrom and Leskovec 2011; De, Ganguly, and Chakrabarti 2013), we sort nodes by decreasing number of triangles in which they occur, and allow the top 80% to be query nodes. Next, for each query node q in the disclosed graph, the set of nodes $V\backslash\{q\}$ is partitioned into neighbors $\text{nbr}(q)$ and non-neighbors $\overline{\text{nbr}}(q)$. Finally, we sample 80% of $|\text{nbr}(q)|$ neighbors and 80% of $\overline{\text{nbr}}(q)$ non-neighbors and present the resulting graph G_{sampled} to $\mathcal{A}$—the non-private base LP, and subsequently to $\overline{\mathcal{A}}$—the protected-pair differentially private LP which is built on $\mathcal{A}$. For each query node q, we ask $\overline{\mathcal{A}}$ to provide a top-K list of potential neighbors from the held-out graph, consisting of both public and private connections. We report the predictive performance of $\overline{\mathcal{A}}$ in terms of average AUC of the ranked list of nodes across all the queries Q. The exact choices of σ_{prot} and K vary across different experiments.

Candidates for base LP algorithms $\mathcal{A}$. We consider two classes of base LP algorithms $\mathcal{A}$: (i) algorithms based on the triad-completion principle (Liben-Nowell and Kleinberg

	AA						CN					
	$\mathcal{A}$	DPLP	DPLP-Lin	Staircase	Lapl.	Exp.	$\mathcal{A}$	DPLP	DPLP-Lin	Staircase	Lapl.	Exp.
Facebook	0.842	**0.788**	0.211	0.163	0.168	0.170	0.827	**0.768**	0.544	0.169	0.181	0.177
USAir	0.899	**0.825**	0.498	0.434	0.461	0.423	0.873	**0.819**	0.670	0.432	0.482	0.435
Twitter	0.726	**0.674**	0.517	0.504	0.515	0.488	0.718	**0.667**	0.630	0.505	0.523	0.493
Yeast	0.778	**0.696**	0.179	0.154	0.154	0.149	0.764	**0.667**	0.359	0.146	0.160	0.151
PB	0.627	**0.558**	0.287	0.254	0.263	0.255	0.593	**0.537**	0.389	0.277	0.272	0.257
	GCN						Node2Vec					
	$\mathcal{A}$	DPLP	DPLP-Lin	Staircase	Lapl.	Exp.	$\mathcal{A}$	DPLP	DPLP-Lin	Staircase	Lapl.	Exp.
Facebook	0.463	0.151	0.154	**0.159**	0.145	0.150	0.681	**0.668**	0.202	0.157	0.152	0.155
USAir	0.483	**0.446**	0.391	0.389	0.399	0.420	0.752	**0.681**	0.424	0.403	0.422	0.447
Twitter	0.504	0.496	0.498	**0.516**	0.509	0.505	0.608	**0.555**	0.511	0.485	0.512	0.505
Yeast	0.369	**0.163**	0.131	0.116	0.125	0.112	0.836	**0.803**	0.180	0.151	0.159	0.145
PB	0.503	0.287	**0.293**	0.262	0.286	0.261	0.525	**0.473**	0.304	0.266	0.282	0.255

Table 1: Performance (AUC) for base non-private LP algorithm $\mathcal{A}$ and five different DP protocols $\overline{\mathcal{A}}$, viz., DPLP, DPLP-Lin, Staircase (Geng and Viswanath 2015), Laplace (Machanavajjhala, Korolova, and Sarma 2011) and Exponential (Machanavajjhala, Korolova, and Sarma 2011) on 20% held-out set. Four base LP algorithms $\mathcal{A}$ *i.e.*, AA, CN, GCN, Node2Vec, across all five datasets. We set the fraction of protected edges $\sigma_{\text{prot}} = 0.3$, the privacy leakage $\epsilon_p = 0.1$ and the number of recommended nodes $K = 30$. Baseline LP algorithms AA and CN are triad-based. Baselines GCN and Node2Vec are based on node embeddings. Bold numbers correspond to the best performing $\overline{\mathcal{A}}$, for the given base algorithm $\mathcal{A}$.

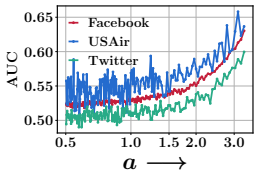

Figure 2: AUC vs. a when the base algorithm $\mathcal{A} = \text{CN}$.

2007) viz., Adamic Adar (AA) and Common Neighbors (CN); and, (ii) algorithms based on fitting node embeddings, viz., GCN (Kipf and Welling 2016) and Node2Vec (Grover and Leskovec 2016). Moreover, in (De and Chakrabarti 2020), we also present results on a wide variety of LP protocols. *e.g.*, Preferential Attachment, Jaccard coefficient, Struct2Vec (Ribeiro, Saverese, and Figueiredo 2017), DeepWalk (Perozzi, Al-Rfou, and Skiena 2014), LINE (Tang et al. 2015) and PRUNE (Lai et al. 2017).

DPLP and baselines ($\overline{\mathcal{A}}$). We compare DPLP against three state-of-the-art perturbation methods *i.e.*, Staircase (Geng and Viswanath 2015), Laplace (Machanavajjhala, Korolova, and Sarma 2011) and Exponential (Machanavajjhala, Korolova, and Sarma 2011; McSherry and Talwar 2007), which maintain differential privacy off-the-shelf. Moreover, as an ablation, apart from using $\nu_{\boldsymbol{\beta}}(.)$ — the linear aggregator of different powers of the score — as a signal in $f_{\boldsymbol{\theta}}(.)$ (Eqs. 6 and 7), we also use it as an independent baseline *i.e.*, $f_{\boldsymbol{\theta}}(s) = \nu_{\boldsymbol{\beta}}(s)$ with $\boldsymbol{\theta} = \{\boldsymbol{\beta}\}$. We refer such a linear "monotone transformation" as DPLP-Lin.

6.2 Results

We first present some empirical evidence which motivates our formulation of f, specifically, the choice of the component $\nu_{\boldsymbol{\beta}}(s_{\mathcal{A}}(\cdot,\cdot))$. We draw nodes using exponential mechanism with different powers of the scores. To recommend neighbors to node u, we draw nodes from the distribution in Eq. (3), with $f(s_{\mathcal{A}}(u,\cdot)) = (s_{\mathcal{A}}(u,\cdot))^a$ for different values of a. Figure 2 illustrates the results for three datasets with the fraction of protected connections $\sigma_{\text{prot}} = 0.3$, the privacy leakage $\epsilon_p = 0.1$ and $\mathcal{A} = \text{CN}$. It shows that AUC improves as a increases. This is because, as a increases, the underlying sampling distribution induced by the exponential mechanism with new score $f(s_{\text{CN}}) = s^a_{\text{CN}}$ becomes more and more skewed towards the ranking induced by the base scores $s_{\mathcal{A}}$, while maintaining privacy level ϵ_p. It turns out that use of UMNN in Eq. (7) further significantly boosts predictive performance of DPLP.

We compare DPLP against three state-of-the-art DP protocols (Staircase, Laplace and exponential), at a given privacy leakage $\epsilon_p = 0.1$ and a given fraction of protected edges $\sigma_{\text{prot}} = 0.3$, for top-$K$ (=30) predictions. Table 1 summarizes the results, which shows that DPLP outperforms competing DP protocols, including its linear variant DPLP-Lin, for $\mathcal{A} = \text{AA, CN}$ and Node2Vec. There is no consistent winner in case of GCN, which is probably because GCN turns out be not-so-good link predictor in our datasets.

7 Conclusion

We proposed protected-pair differential privacy, a practically motivated variant of differential privacy for social networks. Then we presented DPLP, a noising protocol designed around a base non-private LP algorithm. DPLP maximizes ranking accuracy, while maintaining a specified privacy level of protected edges. It transforms the base score using a trainable monotone neural network and then introduces noise into these transformed scores to perturb the ranking distribution. DPLP is trained using a pairwise ranking loss function. We also analyzed the loss of ranking quality in a latent distance graph generative framework. Extensive experiments show that DPLP trades off ranking accuracy and privacy better than several baselines.

Our work opens up several future works, *e.g.*, analyzing collusion between queries and graph steganography attacks (Backstrom, Dwork, and Kleinberg 2007), analyzing utility-privacy trade-off in other graph models, etc.

Broader Impact

Privacy concerns have increased with the proliferation of social media. Today, law enforcement authorities routinely demand social media handles of visa applicants[2]. Insurance providers can use social media content to set premium levels[3]. One may avoid such situations, by making individual features *e.g.*, demographic information, age, etc. private. However, homophily and other network effects may still leak user attributes, even if they are made explicitly private. Someone having skydiver or smoker friends may end up paying large insurance premiums, even if they do not subject themselves to those risks. A potential remedy is for users to mark some part of their connections as private as well (Waniek et al. 2019). In our work, we argue that it is possible to protect privacy of such protected connections, for a specific graph based application, *i.e.*, link prediction, while not sacrificing the quality of prediction. More broadly, our work suggests that, a suitably designed differentially private algorithm can still meet the user expectation in social network applications, without leaking user information.

Acknowledgements

Both the authors would like to acknowledge IBM Grants. Abir De would like to acknowledge Seed Grant provided by IIT Bombay.

References

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–318.

Ackland, R.; et al. 2005. Mapping the US political blogosphere: Are conservative bloggers more prominent? In *BlogTalk Downunder 2005 Conference, Sydney*. BlogTalk Downunder 2005 Conference, Sydney.

Backstrom, L.; Dwork, C.; and Kleinberg, J. 2007. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW*.

Backstrom, L.; and Leskovec, J. 2011. Supervised random walks: predicting and recommending links in social networks. In *WSDM Conference*, 635–644. URL http://cs.stanford.edu/people/jure/pubs/linkpred-**s**wsdm11.pdf.

Batagelj, V.; and Mrvar, A. 2006. USAir Dataset.

Chierichetti, F.; Epasto, A.; Kumar, R.; Lattanzi, S.; and Mirrokni, V. 2015. Efficient Algorithms for Public-Private Social Networks. In *SIGKDD*.

De, A.; and Chakrabarti, S. 2020. Differentially Private Link Prediction With Protected Connections URL https://arxiv.org/abs/1908.04849.

De, A.; Ganguly, N.; and Chakrabarti, S. 2013. Discriminative link prediction using local links, node features and community structure. In *ICDM*, 1009–1018. IEEE.

[2]https://bit.ly/2A5R3kS

[3]https://bit.ly/30oV0Mr

Durfee, D.; and Rogers, R. M. 2019. Practical Differentially Private Top-k Selection with Pay-what-you-get Composition. In *Advances in Neural Information Processing Systems*, 3527–3537.

Dwork, C. 2011. Differential privacy. *Encyclopedia of Cryptography and Security* 338–340.

Dwork, C.; and Roth, A. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science* 9(3–4): 211–407.

Gao, W.; and Zhou, Z.-H. 2015. On the consistency of AUC pairwise optimization. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Geng, Q.; Ding, W.; Guo, R.; and Kumar, S. 2018. Optimal noise-adding mechanism in additive differential privacy. *arXiv preprint arXiv:1809.10224* .

Geng, Q.; and Viswanath, P. 2015. The optimal noise-adding mechanism in differential privacy. *IEEE Transactions on Information Theory* 62(2): 925–951.

Ghosh, A.; Roughgarden, T.; and Sundararajan, M. 2012. Universally utility-maximizing privacy mechanisms. *SIAM Journal on Computing* 41(6): 1673–1693.

Grover, A.; and Leskovec, J. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*.

He, X.; Machanavajjhala, A.; and Ding, B. 2014. Blowfish privacy: Tuning privacy-utility trade-offs using policies. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, 1447–1458.

Hoff, P. D.; Raftery, A. E.; and Handcock, M. S. 2002. Latent space approaches to social network analysis. *Journal of the American Statistical Association* 97(460): 1090–1098.

Joachims, T. 2005. A support vector method for multivariate performance measures. In *ICML*.

Kearns, M.; Roth, A.; Wu, Z. S.; and Yaroslavtsev, G. 2015. Privacy for the protected (only). *arXiv preprint arXiv:1506.00242* .

Kifer, D.; and Machanavajjhala, A. 2014. Pufferfish: A framework for mathematical privacy definitions. *ACM Transactions on Database Systems (TODS)* 39(1): 1–36.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* .

Lai, Y.-A.; Hsu, C.-C.; Chen, W. H.; Yeh, M.-Y.; and Lin, S.-D. 2017. PRUNE: Preserving proximity and global ranking for network embedding. In *NeurIPS*, 5257–5266.

Leskovec, J.; and Mcauley, J. J. 2012. Learning to discover social circles in ego networks. In *NeuIPS*.

Liben-Nowell, D.; and Kleinberg, J. 2007. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology* 58(7): 1019–1031. ISSN 1532-2890. doi:10.1002/asi.20591. URL https://onlinelibrary.wiley.com/doi/full/10.1002/asi.20591.

Liu, T.-Y. 2009. Learning to Rank for Information Retrieval. In *Foundations and Trends in Information Retrieval*, volume 3, 225–331. Now Publishers. doi:http:/dx.doi.org/10.1561/1500000016. URL http://www.nowpublishers.com/product.aspx?product=INR&doi=1500000016.

Lü, L.; and Zhou, T. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390(6): 1150–1170. URL https://www.sciencedirect.com/science/article/pii/S037843711000991X.

Machanavajjhala, A.; Korolova, A.; and Sarma, A. D. 2011. Personalized Social Recommendations - Accurate or Private? In *VLDB*.

Marbach, P.; and Tsitsiklis, J. N. 2003. Approximate gradient methods in policy-space optimization of Markov reward processes. *Discrete Event Dynamic Systems* 13(1-2): 111–148.

McSherry, F.; and Talwar, K. 2007. Mechanism Design via Differential Privacy. In *FOCS*, 94–103.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*, 701–710.

Peters, J.; and Schaal, S. 2006. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2219–2225. IEEE.

Ribeiro, L. F.; Saverese, P. H.; and Figueiredo, D. R. 2017. struc2vec: Learning node representations from structural identity. In *SIGKDD Conference*, 385–394.

Sarkar, P.; Chakrabarti, D.; and Moore, A. W. 2011. Theoretical justification of popular link prediction heuristics. In *COLT*.

Sehnke, F.; Osendorfer, C.; Rückstieß, T.; Graves, A.; Peters, J.; and Schmidhuber, J. 2010. Parameter-exploring policy gradients. *Neural Networks* 23(4): 551–559.

Song, S.; and Chaudhuri, K. 2017. Composition properties of inferential privacy for time-series data. In *2017 55th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 814–821. IEEE.

Song, S.; Little, S.; Mehta, S.; Vinterbo, S.; and Chaudhuri, K. 2018. Differentially Private Continual Release of Graph Statistics. *arXiv preprint arXiv:1809.02575* .

Song, S.; Wang, Y.; and Chaudhuri, K. 2017. Pufferfish privacy mechanisms for correlated data. In *Proceedings of the 2017 ACM International Conference on Management of Data*, 1291–1306.

Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; and Mei, Q. 2015. LINE: Large-scale information network embedding. In *WWW Conference*, 1067–1077.

Tucker, C. E. 2014. Social Networks, Personalized Advertising, and Privacy Controls. *Journal of Marketing Research* 51(5): 546–562. doi:10.1509/jmr.10.0355.

Von Mering, C.; Krause, R.; Snel, B.; Cornell, M.; Oliver, S. G.; Fields, S.; and Bork, P. 2002. Comparative assessment of large-scale data sets of protein–protein interactions. *Nature* 417(6887): 399.

Waniek, M.; Zhou, K.; Vorobeychik, Y.; Moro, E.; Michalak, T. P.; and Rahwan, T. 2019. How to Hide one's Relationships from Link prediction Algorithms. *Scientific reports* 9(1): 1–10.

Wehenkel, A.; and Louppe, G. 2019. Unconstrained monotonic neural networks. In *Advances in Neural Information Processing Systems*, 1543–1553.

Xu, D.; Yuan, S.; Wu, X.; and Phan, H. 2018. DPNE: differentially private network embedding. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 235–246. Springer.

Zhao, T.; Hachiya, H.; Niu, G.; and Sugiyama, M. 2011. Analysis and improvement of policy gradient estimation. In *Advances in Neural Information Processing Systems*, 262–270.