Privacy-Preserving Link Prediction in Decentralized Online Social Networks

Yao Zheng(✉), Bing Wang, Wenjing Lou, and Y. Thomas Hou

Virginia Polytechnic Institute and State University, Blacksburg, USA
{zhengyao,bingwang,wjlou,thou}@vt.edu

Abstract. We consider the privacy-preserving link prediction problem in decentralized online social network (OSNs). We formulate the problem as a sparse logistic regression problem and solve it with a novel decentralized two-tier method using alternating direction method of multipliers (ADMM). This method enables end users to collaborate with their online service providers without jeopardizing their data privacy. The method also grants end users fine-grained privacy control to their personal data by supporting arbitrary public/private data split. Using real-world data, we show that our method enjoys various advantages including high prediction accuracy, balanced workload, and limited communication overhead. Additionally, we demonstrate that our method copes well with link reconstruction attack.

Keywords: Distributed algorithms · ADMM · Mobile computing · Privacy · Social networks

1 Introduction

The last decade has witnessed the rise of online social network (OSNs). Starting from the late 2000s, OSNs have seen a rapid growth in their popularity. In 2014, two most profitable OSNs, Facebook ($140 billion) and Twitter ($35 billion) [1], jointly hold 1.3 billion active users worldwide [2]. These people conduct their personal lives and house their personal data via OSNs. They sync valuable information such as profiles, microblogs and photos with OSN websites every day. This situation raises serious privacy concerns among general public. The privacy control mechanisms provided by OSNs are cumbersome and ineffective [3]. It does not stop unauthorized parties from peeking into users' private data. More important, the OSNs privacy agreements state that the OSNs own the content that users upload. This allows OSNs to monetize users' personal information for commercial purposes such as advertising [4]. Such invasive act exacerbates the public distrust.

In order to address such concerns, a decentralized architecture for OSNs was recently proposed [5–7]. Instead of storing users' data in the OSNs' centralized database, the new architecture advocates decentralized data storage to avoid personal data monetization. In a decentralized OSN, users' data exists as a

G. Pernul et al. (Eds.): ESORICS 2015, Part II, LNCS 9327, pp. 61–80, 2015.
DOI: 10.1007/978-3-319-24177-7_4

collection of private files stored on their personal cloud storage service. Any action upon these files must be directed to the private repositories and consented by the users. This way, users retain full control of their personal data.

However, decentralized data storing precludes useful functionalities commonly seen in centralized OSNs. For instance, link prediction [8, chap. 1] is a common OSN analysis problem that helps to discover entities with whom a user might wish to connect. It operates by mining users' friendship and affiliation preferences from their personal data. The mining is usually done on powerful OSN servers. In decentralized OSNs, the mining functionality is difficult to provide due to the users' dilemma between privacy and usability. On the one hand, they wish to limit the personal data exposure. On the other hand, they lack the computing resources to analyze their personal data locally.

In this work, we study the link prediction problem in decentralized OSNs. We assume users' personal data can be split into two parts, private and public. The public part can be accessed directly whereas the private part must remain secret. For instance, consider a user who owns two twitter accounts, one is open for public and the other one has access restriction. The user wishes to determine how likely he will follow another user's tweets by correlating the target user's tweets with the textual materials in both of his accounts. Due to limited computing resources, the user can only process the materials in his private account and authorizes his online service provider to process the materials reside in the public account. Such split pose a challenge for training. On the one hand, the prediction accuracy will be poor if the user and his online service provider train their prediction models separately and try to merge the result together by voting. On the other hand, naive collaborative trainings reveal private information to online service providers.

We propose a novel privacy-preserving training method to solve the dilemma. The method allows users and their online service providers to collaboratively train link prediction models without revealing users' private data. We grant users fine-grained privacy control by supporting arbitrary public/private data split. We prove that the workload is properly balanced between users and their online service providers according to their computation capabilities. We apply our method to a real-world social network dataset to prove its validity. Additionally, we study the security risk of our method. We evaluate the possibility of the link reconstruction attack when adversaries can access users' public data.

2 Related Work

Our work is targeted on decentralized OSNs that allow users to maintain their data on their personal cloud server [9]. A typical decentralized OSN consists of independent servers that communicate with each other. Users can either register on an existing server or create their own. In the later case, users stay in control of their data because they are the administrators of their servers. The personal data initially resides on users' own servers. If friends from other servers request this information, it will be transfered to their servers through a server-to-server protocol. Ideally, a decentralized OSN can completely eliminate personal data monetization if different servers reside on different cloud platforms. In practice, a decentralized OSN may rely on additional cryptographic mechanisms to protect users' data since multiple servers may belong to the same cloud service provider.

There has been a substantial amount of work to enhance decentralized OSNs. The early researches use advanced cryptographic mechanisms to protect users' privacy. Two examples are Persona [10] and Safebook [11]. Persona combines attribute-based encryption (ABE) with traditional public key cryptography to offer flexible and fine-grained access control to data. Safebook combines a peer-to-peer architecture with a certification distribution system based on distributed hash table to avoid unauthorized access to users' data. These methods usually work at the cost of limiting OSN functionalities. Later works shift focus in adding new services to decentralized OSNs. Musubi [5], for instance, provides a toolkit for decentralized OSNs to support application development for multi-party interaction. Omlet [7] is a commercial decentralized OSN based on Musubi. Despite of their efforts, most functionalities that rely on data analysis are still not supported in decentralized OSNs.

Link prediction is a common OSN analysis problem that forms the basis of numerous OSN functional features. In [8, chap. 1], Aggarwal gives a comprehensive survey on the methods used for link prediction. These methods can be divided into two categories, *i.e.*, structured-based prediction methods and attributes-based prediction methods. The former is applicable to large scale networks consist millions of nodes [12]. The latter analyzes median, personal networks with detailed node descriptions [13]. Most of these methods must be redesigned to fit into the decentralized architecture. In this work, we mainly focus on users' personal social networks, a method commonly known as egocentric [14]. We use an attributes-based prediction method due to the nature of decentralization.

The method we use falls into the categories of privacy-preserving machine learning and distributed optimization. The most noteworthy idea for privacy-preserving machine learning is the ϵ-differential privacy proposed by Dwork [15], in which carefully calibrated noise is injected into the dataset to achieve indistinguishability. However, ϵ-differential privacy is unnecessary when data is owned by a single user. Perhaps the closest to our work is done by Yu *et al.* [16]. They use a distributed algorithm to train a support vector machine such that it preserves the privacy of different data blocks. But their method cannot protect the feature privacy within the same data block, which we address in Sect. 4.

3 System Model and Privacy Goals

Here we describe the link prediction problem. The set-up for this problem is a variation of Guha's framework [17]: We consider a decentralized OSN involving both positive and negative links. The positive links are formed due to friendship, support, or approval whereas the negative links are formed due to disapproval, disagreement, or distrust. We consider a privacy conscious user, Alice, who is unwilling to reveal part of her personal data. We are interested in predicting the link of Alice's personal social network. We consider an honest-but-curious system, Sara, that can only access Alice's public data. We allow Alice and Sara to jointly learn a prediction model. Alice's privacy is violated if Sara learns her private data or part of the prediction model that is associated with the private data.

3.1 Network Abstraction

Here we show how we model Alice's social network. We take an egocentric approach [14] and examine only Alice's immediate neighbors and associated interconnections that are commonly stored as part of Alice's personal data. We interpret this network as a directed graph $G = (V, E)$. The ith link is associated with two node vectors $t_i \in \mathbf{R}^n$ and $h_i \in \mathbf{R}^n$ that characterize the tail and head node of the link. These features are extracted from the materials shared through the group sharing services supported by decentralized OSNs [5]. The ith link is also associated with a label $q_i \in \{-1, 1\}$. We define the sign of q_i to be positive or negative depending on whether the tail node expresses a positive or negative attitude toward the head node.

To facilitate the problem formulation, we use $K-1$ scoring functions $f_k : \mathbf{R}^n \times \mathbf{R}^n \to \mathbf{R}$ to construct the link vector. Let $p_{i,k} = f_k(t_i, h_i)$ be the score between the tail node t_i and the head node h_i calculated by the kth scoring function. Let $p_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,K-1})$ be the link scores. We can represent each visible links in Alice's network with a vector (p_i, q_i). For reasons we will show later, we define the link vector to be $a_i = (q_i p_i, q_i)$. We use a matrix $A \in \mathbf{R}^{|E| \times K}$ to represent all the link vectors in Alice's network.

Alice can define her privacy preferences by veiling part of A. For instance, Alice may regard certain scores or the link sign as private, which corresponds to cloaking a particular column of A. Alice may also choose to hide certain link entirely, which corresponds to cloaking a particular row of A. Without loss of generality, we divide A into three parts[1]

$$A = \left(\begin{array}{cc} \overleftarrow{A} & \overrightarrow{A} \\ \multicolumn{2}{c}{A_{\downarrow}} \end{array} \right),$$

where the public features are within $\overleftarrow{A} \in \mathbf{R}^{|E|_{\uparrow} \times \overleftarrow{K}}$, the private features are within $\overrightarrow{A} \in \mathbf{R}^{|E|_{\uparrow} \times \overrightarrow{K}}$ and the private links are within $A_{\downarrow} \in \mathbf{R}^{|E|_{\downarrow} \times K}$. Note that $|E|_{\uparrow} + |E|_{\downarrow} = |E|$ and $\overleftarrow{K} + \overrightarrow{K} = K$. In practice, an implicit condition is $|E|_{\uparrow} \gg |E|_{\downarrow}$ and $\overleftarrow{K} \gg \overrightarrow{K}$, though our method can be applied to $\overleftarrow{A}$, $\overrightarrow{A}$ and $A_{\downarrow}$ with arbitrary sizes.

[1] We can rearrange the columns and rows of any A to separate the public features from the private features and the private links.

3.2 Training Goal

Here we describe our training goal. We consider two learning modules, $\mathcal{A}$ and $\mathcal{S}$, owned by Alice and Sara. Assume that $\mathcal{A}$ only processes $\overrightarrow{A}$ and $A_\downarrow$ due to limited resources whereas $\mathcal{S}$ is powerful but is only allowed to access $\overleftarrow{A}$. Using $\mathcal{A}$ and $\mathcal{S}$, Alice and Sara jointly fit a sparse logistic regression model

$$\text{minimize } \tfrac{1}{|E|}\sum_{i=1}^{|E|}\log\left(1+\exp\left(-q_i(p_i^T w+v)\right)\right)+\lambda\|w\|_1, \tag{1}$$

where $w \in \mathbf{R}^{K-1}$ is the weights for the link scores and $v \in \mathbf{R}$ is the intercept. Let $x \in \mathbf{R}^K$ equals (w, v). The problem is equivalent to

$$\text{minimize } \tfrac{1}{|E|}\sum_{i=1}^{|E|}\log\left(1+\exp\left(-Ax\right)\right)+\lambda r(x)\,, \tag{2}$$

where $r(x) = \|w\|_1$. Let x equals $(\overleftarrow{x}, \overrightarrow{x})$ where $\overleftarrow{x}$ is the weights of the public features and $\overrightarrow{x}$ is the weights of the private features. The method preserves Alice's privacy if Sara is oblivious of $\overrightarrow{A}$, $A_\downarrow$ and $\overrightarrow{x}$.

There exists a plethora of network link prediction models. The learning architectures range from shallow ones such as support vector machine [18] and statistical regressions [19] to deep ones such as graphical model [20] and deep neural networks [21]. The reason we choose a sparse logistic regression model are threefold: (1) The performances of all models are comparable given the appropriate feature set [8, chap. 1]. There is no clear and convincing evidence indicating that one model supersedes the others. (2) A sparse logistic regression model is representative of the types of shallow learning architectures that produce reliable and reproducible results [22, Sect. 4.4]. (3) More important, a sparse logistic regression model can be viewed as a regularized logistic neuron, which is the building block of deep learning architectures such as deep belief nets [23] and restricted Boltzmann machines [24]. Designing a privacy-preserving learning method for it opens the possibility of assembling more complicated privacy-preserving learning models.

3.3 Prediction Goal

Here we summarize our prediction goal. Once the model is jointly trained, we use it to predict the sign of any unknown link in which Alice is interested. Specifically, let $x^\star = (\overleftarrow{w}^\star, \overrightarrow{w}^\star, v^\star)$ where $\overleftarrow{w}^\star$ and $\overrightarrow{w}^\star$ are the optimal weights of the public and private link scores; $v^\star$ is the optimal intercept. Let $p_u = (\overleftarrow{p}_u, \overrightarrow{p}_u)$ be the link scores of the unknown link where $\overleftarrow{p}_u$ and $\overrightarrow{p}_u$ are the public and private scores. Let $\hat{q}_u$ be the predicted link sign. Alice and Sara should be able to assemble the logistic function

$$\mathbf{Prob}(\hat{q}_u = 1 \mid x^\star) = \frac{1}{1+\exp\left(-(\overleftarrow{p}_u^T\overleftarrow{w}^\star + \overrightarrow{p}_u^T\overrightarrow{w}^\star + v^\star)\right)}, \tag{3}$$

without Sara knowing $\overrightarrow{p}_u$, $\overrightarrow{x}^\star$ and $\hat{q}_u$. To constitute a good prediction model, we also require $\hat{q}_u$ to equal the true link sign q_u with high probability.

4 Methodology

We now present our method for the link prediction problem. We first give a short introduction of the core algorithm we use, *i.e.*, alternating direction method of multipliers (ADMM). Following that, we describe a two-tier training method. Specially, we show how we separate $\overrightarrow{A}$ and $A_{\downarrow}$ from $\overleftarrow{A}$ to protect Alice's privacy. We give a complexity analysis of the training method to show that the workload is properly divided base on the computation resources available for Alice and Sara. Finally, we show that our training method is capable of protecting Alice's prior knowledge about $x^{\star}$.

4.1 ADMM

ADMM, also known as the Douglas-Rachford splitting, is a *decomposition* procedure, in which the solutions to small local subproblems are coordinated to find a solution to a large global problem. It was first introduced in the mid-1970s by Glowinski and Marrocco [25] and Gabay and Mercier [26]. Originally, ADMM was designed to decouple the objective functionals to achieve better convergence. Later analyses [27] show that it is also well suited for large-scale distributed computing and massive optimization problems.

Let $f : \mathbf{R}^n \to \mathbf{R}$ and $g : \mathbf{R}^m \to \mathbf{R}$ be two functionals that are convex. The basic ADMM is an iterative method that solves problems in the form

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c, \end{array}$$

with variable $x \in \mathbf{R}^n$ and $z \in \mathbf{R}^m$, where $A \in \mathbf{R}^{p\times n}$, $B \in \mathbf{R}^{p\times m}$ and $c \in \mathbf{R}^p$. The augmented Lagrangian for the problem is

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2$$

where y is the dual variable or Lagrange Multiplier, ρ is the penalty parameter. Let $u = (1/\rho)y$ be the scaled dual variable. We can express each ADMM iteration as a full Gauss-Seidel iteration between x, z and u

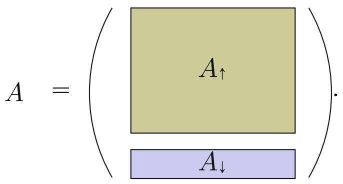
$$\begin{aligned} x^{k+1} &:= \underset{x}{\operatorname{argmin}} \left(f(x) + (\rho/2)\|Ax + Bz^k - c + u^k\|_2^2\right) \\ z^{k+1} &:= \underset{z}{\operatorname{argmin}} \left(g(z) + (\rho/2)\|Ax^{k+1} + Bz - c + u^k\|_2^2\right) \\ u^{k+1} &:= u^k + Ax^{k+1} + Bz^{k+1} - c. \end{aligned}$$

The algorithm fully splits the objective into two terms, *i.e.*, the x-update and z-update, which involve evaluating the proximal operators [28] with respect to f and g. If at least one of them is separable, we can run the algorithm in parallel fashion. Generally, evaluating such operators requires solving a convex optimization problem. But, depending on the nature of f and g, simpler or faster specialized methods usually exist. Due to the smoothing of the proximal operators, ADMM can deal with the case when f and g are not differentiable. For a more detailed discussion on ADMM, we refer the readers to Boyd's work [27].

4.2 Two-Tier Training

Here we describe the two-tier training method for the link prediction problem. To protect Alice's privacy, we formulate problems into two specific canonical forms, *i.e.*, *consensus* and *sharing*, and solve them using ADMM. At the first tier, we split $A_\downarrow$ from $\overleftarrow{A}$ and $\overrightarrow{A}$ to protect the private links. At the second tier, we split $\overrightarrow{A}$ from $\overleftarrow{A}$ to protect the private features.

Link Split. At the first tier, we split A by rows in order to protect the private links within $A_\downarrow$. Let $A_\uparrow \in \mathbf{R}^{|E|_\uparrow \times K}$ represents both $\overleftarrow{A}$ and $\overrightarrow{A}$. We first split A into $A_\uparrow$ and $A_\downarrow$



Define

$$l_\uparrow(A_\uparrow x_\uparrow) = \frac{1}{|E|_\uparrow} \sum_{i=1}^{|E|_\uparrow} \log(1 + \exp(-A_\uparrow x_\uparrow)),$$
$$l_\downarrow(A_\downarrow x_\downarrow) = \frac{1}{|E|_\downarrow} \sum_{i=1}^{|E|_\downarrow} \log(1 + \exp(-A_\downarrow x_\downarrow)).$$

We can explicitly convert the sparse logistic regression problem (Eq. 2) into consensus form [27]

$$\begin{aligned} &\text{minimize} \;\; l_\uparrow(A_\uparrow x_\uparrow) + l_\downarrow(A_\downarrow) + \lambda r(z_\updownarrow) \\ &\text{subject to } x_\uparrow - z_\updownarrow = x_\downarrow - z_\updownarrow = 0, \end{aligned}$$

with local variable $x_\uparrow, x_\downarrow \in \mathbf{R}^K$ and global variable $z_\updownarrow \in \mathbf{R}^K$.

The problem can be solved using the following ADMM algorithm

$$x_\uparrow^{k+1} := \underset{x_\uparrow}{\operatorname{argmin}} \left(l_\uparrow(A_\uparrow x_\uparrow) + (\rho/2)\|x_\uparrow - z_\updownarrow^k + u_\uparrow^k\|_2^2 \right) \tag{4}$$

$$x_\downarrow^{k+1} := \underset{x_\downarrow}{\operatorname{argmin}} \left(l_\downarrow(A_\downarrow x_\downarrow) + (\rho/2)\|x_\downarrow - z_\updownarrow^k + u_\downarrow^k\|_2^2 \right) \tag{5}$$

$$z_\updownarrow^{k+1} := \underset{z_\updownarrow}{\operatorname{argmin}} \left(r(z_\updownarrow) + (\rho/\lambda)\|z_\updownarrow - \overline{x}_\updownarrow^{k+1} - \overline{u}_\updownarrow^k\|_2^2 \right) \tag{6}$$

$$u_\uparrow^{k+1} := u_\uparrow^k + x_\uparrow^{k+1} - z_\updownarrow^{k+1} \tag{7}$$

$$u_\downarrow^{k+1} := u_\downarrow^k + x_\downarrow^{k+1} - z_\updownarrow^{k+1}, \tag{8}$$

where $u_\uparrow$ and $u_\downarrow$ are the scaled local dual variables correspond to $x_\uparrow$ and $x_\downarrow$; $\overline{x}_\updownarrow = (1/2)(x_\uparrow + x_\downarrow)$ and $\overline{u}_\updownarrow = (1/2)(u_\uparrow + u_\downarrow)$ are the averages of the local primal variables and scaled local dual variables. The termination criterion is that the primal and dual residuals must be small, *i.e.*,

$$\sqrt{\|x_\uparrow^k - \overline{x}_\updownarrow^k\|_2^2 + \|x_\downarrow^k - \overline{x}_\updownarrow^k\|_2^2} < \epsilon_\updownarrow^{\text{pri}}$$

and

$$2\rho\|\overline{x}_\updownarrow^k - \overline{x}_\updownarrow^{k-1}\|_2 < \epsilon_\updownarrow^{\text{dual}},$$

where $\epsilon_\updownarrow^{\text{pri}} > 0$ and $\epsilon_\updownarrow^{\text{dual}} > 0$ are feasibility tolerances for the primal and dual feasibility conditions [27].

The algorithm is very intuitive. The local primal variables, $x_\uparrow$ and $x_\downarrow$, and dual variables, $u_\uparrow$ and $u_\downarrow$, are separately updated through Eqs. 4, 5, 7 and 8. The local result are collected and brought into consensus through Eq. 6. When the algorithm terminates, $x_\uparrow$ and $x_\downarrow$ should both agree with $z_\updownarrow$.

Let $x_\uparrow$ equals $(\overleftarrow{x}_\uparrow, \vec{x}_\uparrow)$; $x_\downarrow$ equals $(\overleftarrow{x}_\downarrow, \vec{x}_\downarrow)$; $u_\uparrow$ equals $(\overleftarrow{u}_\uparrow, \vec{u}_\uparrow)$; $u_\downarrow$ equals $(\overleftarrow{u}_\downarrow, \vec{u}_\downarrow)$; $z_\updownarrow$ equals $(\overleftarrow{z}_\updownarrow, \vec{z}_\updownarrow)$. The variables that should be private to Alice are $\vec{x}_\uparrow$, $\vec{x}_\downarrow$, $\vec{u}_\uparrow$, $\vec{u}_\downarrow$ and $\vec{z}_\updownarrow$. To protect $A_\downarrow$, $\vec{x}_\downarrow$ and $\vec{u}_\downarrow$, we assign Eqs. 5 and 8 to $\mathcal{A}$ such that Alice can handle $A_\downarrow$, $x_\downarrow$ and $u_\downarrow$ exclusively. Equation 5 involves a ℓ_2 regularized logistic regression problem that can be efficiently solved by Quasi-Newton methods like L-BFGS [29]. To further reduce her efforts, Alice can mandate the maximum L-BFGS iterations to be small and rely on the second tier for accuracy.

To protect $\vec{z}_\updownarrow$, we split Eq. 6. Since $r(z_\updownarrow)$ is essentially the proximal operator of a ℓ_1 norm, we can calculate it using the *soft thresholding operator* [30]

$$S_\kappa(x) = (x - \kappa)_+ - (-x - \kappa)_-,$$

which is separable at the component level. We can split Eq. 6 into

$$\overleftarrow{z}_\updownarrow^{k+1} := (1/2)S_{\lambda/\rho}(\overleftarrow{x}_\uparrow^{k+1} + \overleftarrow{x}_\downarrow^{k+1} + \overleftarrow{u}_\uparrow^{k} + \overleftarrow{u}_\downarrow^{k}) \tag{9}$$

$$\vec{z}_\updownarrow^{k+1} := (1/2)S_{\lambda/\rho}(\vec{x}_\uparrow^{k+1} + \vec{x}_\downarrow^{k+1} + \vec{u}_\uparrow^{k} + \vec{u}_\downarrow^{k}), \tag{10}$$

We assign Eq. 9 to $\mathcal{S}$ but reserve and Eq. 10 to $\mathcal{A}$[2]. We allow Sara to send $\overleftarrow{z}_\updownarrow$ back to Alice since She need it to compute $x_\downarrow$ and $u_\downarrow$. To protect $\vec{u}_\uparrow$, we split Eq. 7 into

$$\overleftarrow{u}_\uparrow^{k+1} := \overleftarrow{u}_\uparrow^{k} + \overleftarrow{x}_\uparrow^{k+1} - \overleftarrow{z}_\updownarrow^{k+1} \tag{11}$$

$$\vec{u}_\uparrow^{k+1} := \vec{u}_\uparrow^{k} + \vec{x}_\uparrow^{k+1} - \vec{z}_\updownarrow^{k+1}. \tag{12}$$

We assign Eq. 11 to $\mathcal{S}$ but reserve and Eq. 12 to $\mathcal{A}$.

Finally, Eq. 4 contains data and variable that should be private to Alice, *i.e.*, $\vec{A}$ and $\vec{x}_\uparrow$, which we will handle at the second tier.

[2] Note that the intercept v is not regularized. Equations 9 and 10 can be modified to incorporate the intercept by dropping the soft thresholding operator on the corresponding element in $z_\updownarrow$.

Feature Split. At the second tier, we split $A_\uparrow$ by columns in order to protect the private features within $\overrightarrow{A}$ and the corresponding weight $\overrightarrow{x}_\uparrow$. Recall that

$$A_\uparrow = \begin{pmatrix} \overleftarrow{A} & \overrightarrow{A} \end{pmatrix}.$$

Define

$$\overleftarrow{r}(\overleftarrow{x}_\uparrow) = (\rho/2)\|\overleftarrow{x}_\uparrow - \overleftarrow{z}_\updownarrow^k + \overleftarrow{u}_\uparrow^k\|_2^2,$$
$$\overrightarrow{r}(\overrightarrow{x}_\uparrow) = (\rho/2)\|\overrightarrow{x}_\uparrow - \overrightarrow{z}_\updownarrow^k + \overrightarrow{u}_\uparrow^k\|_2^2.$$

we can explicitly convert Eq. 4 into sharing form [27]

$$\begin{aligned} &\text{minimize} \;\; l_\uparrow(\overset{\leftarrow}{\overleftrightarrow{z}} + \overset{\rightarrow}{\overleftrightarrow{z}}) + \overleftarrow{r}(\overleftarrow{x}_\uparrow) + \overrightarrow{r}(\overrightarrow{x}_\uparrow) \\ &\text{subject to} \; \overleftarrow{A}\,\overleftarrow{x}_\uparrow - \overset{\leftarrow}{\overleftrightarrow{z}} = \overrightarrow{A}\,\overrightarrow{x}_\uparrow - \overset{\rightarrow}{\overleftrightarrow{z}} = 0, \end{aligned}$$

with partial predictors $\overset{\leftarrow}{\overleftrightarrow{z}}, \overset{\rightarrow}{\overleftrightarrow{z}} \in \mathbf{R}_\uparrow^{|E|}$.

Let $\overleftrightarrow{u} \in \mathbf{R}_\uparrow^{|E|}$ be the single dual variable. The problem can be solved using the following ADMM algorithm

$$\overleftarrow{x}_\uparrow^{k'+1} := \underset{\overleftarrow{x}_\uparrow}{\text{argmin}} \left(\overleftarrow{r}(\overleftarrow{x}_\uparrow) + (\rho'/2)\| \overleftarrow{A}\,\overleftarrow{x}_\uparrow - \overleftarrow{A}\,\overleftarrow{x}_\uparrow^{k'} - \overline{\overleftrightarrow{z}}^{k'} + \overline{A_\uparrow x_\uparrow}^{k'} + \overleftrightarrow{u}^{k'} \|_2^2 \right) \tag{13}$$

$$\overrightarrow{x}_\uparrow^{k'+1} := \underset{\overrightarrow{x}_\uparrow}{\text{argmin}} \left(\overrightarrow{r}(\overrightarrow{x}_\uparrow) + (\rho'/2)\| \overrightarrow{A}\,\overrightarrow{x}_\uparrow - \overrightarrow{A}\,\overrightarrow{x}_\uparrow^{k'} - \overline{\overleftrightarrow{z}}^{k'} + \overline{A_\uparrow x_\uparrow}^{k'} + \overleftrightarrow{u}^{k'} \|_2^2 \right) \tag{14}$$

$$\overline{\overleftrightarrow{z}}^{k'+1} := \underset{\overline{\overleftrightarrow{z}}}{\text{argmin}} \left(l_\uparrow(2\overline{\overleftrightarrow{z}}) + \rho' \| \overline{\overleftrightarrow{z}} - \overline{A_\uparrow x_\uparrow}^{k'+1} - \overleftrightarrow{u}^{k'} \|_2^2 \right) \tag{15}$$

$$\overleftrightarrow{u}^{k'+1} := \overleftrightarrow{u}^{k'} + \overline{A_\uparrow x_\uparrow}^{k'+1} - \overline{\overleftrightarrow{z}}^{k'+1}, \tag{16}$$

where $\overline{A_\uparrow x_\uparrow} = (1/2)(\overleftarrow{A}\,\overleftarrow{x}_\uparrow + \overrightarrow{A}\,\overrightarrow{x}_\uparrow)$ and $\overline{\overleftrightarrow{z}} = (1/2)(\overset{\leftarrow}{\overleftrightarrow{z}} + \overset{\rightarrow}{\overleftrightarrow{z}})$ are the averages of the partial predictors. The termination criterion is that the primal and dual residuals must be small, *i.e.*,

$$2\|\overline{A_\uparrow x_\uparrow}^{k'} - \overline{\overleftrightarrow{z}}^{k'}\|_2 < \overleftrightarrow{\epsilon}^{\text{pri}}$$

and

$$\rho' \sqrt{\|\overleftarrow{A}^T(\overset{\leftarrow}{\overleftrightarrow{z}}^{k'} - \overset{\leftarrow}{\overleftrightarrow{z}}^{k'-1})\|_2^2 + \|\overrightarrow{A}^T(\overset{\rightarrow}{\overleftrightarrow{z}}^{k'} - \overset{\rightarrow}{\overleftrightarrow{z}}^{k'-1})\|_2^2} < \overleftrightarrow{\epsilon}^{\text{dual}}$$

where $\overleftrightarrow{\epsilon}^{\text{pri}} > 0$ and $\overleftrightarrow{\epsilon}^{\text{dual}} > 0$ are feasibility tolerances for the primal and dual feasibility conditions [27].

The algorithm is also intuitive. The local primal variables $\overleftarrow{x}_\uparrow$, and $\overrightarrow{x}_\uparrow$ and dual variable $\overleftrightarrow{u}$ are separately updated through Eqs. 13, 14 and 16. The partial predictors are collected, averaged then updated through a ℓ_2 regularized logistic regression problem (Eq. 15).

To protect $\overrightarrow{A}$ and $\overrightarrow{x}_\uparrow$, we can assign all but Eq. 14 to $\mathcal{S}$. The reason is that only $\overleftarrow{\overleftrightarrow{z}} = \overleftarrow{A}\,\overleftarrow{x}_\uparrow$ and $\overrightarrow{\overleftrightarrow{z}} = \overrightarrow{A}\,\overrightarrow{x}_\uparrow$ are shared throughout the algorithm. From Eq. 1, we see that these partial predictors are in fact the *margin* of the training data, which is a monotonically decreasing function of the sub-optimality. Assuming the label q is within $\overrightarrow{A}$, sharing these partial predictors reveals neither $\overrightarrow{A}$ nor $\overrightarrow{x}_\uparrow$.

Using this two-tier training method, Alice, who processes $A_\downarrow$ and $\overrightarrow{A}$, learns the entire model coefficient $x^\star$. Sara, who processes $\overleftarrow{A}$, only learns $\overleftarrow{x}^\star$ while remains oblivious about $A_\downarrow$, $\overrightarrow{A}$ and $\overrightarrow{x}^\star$. When predicting a link with unknown sign, Alice can either assemble Eq. 3 by herself, or outsource $\overleftarrow{p}_u^T \overleftarrow{w}^\star$ to Sara without jeopardizing her privacy.

4.3 Complexity Analysis

Here we analyze the complexities of the tasks assigned to $\mathcal{A}$ and $\mathcal{S}$. We show that the workload is properly divided between $\mathcal{A}$ and $\mathcal{S}$ such that Sara handles a majority of work.

For each training iteration, the dominate tasks for $\mathcal{A}$ are Eqs. 5 and 14. Equation 5 is a ℓ_2 regularized logistic regression with a wide matrix $A_\downarrow$. Assuming we solve it with L-BFGS, the most expensive operations for each L-BFGS iteration are evaluating the function value, the gradient and approximating the Hessian matrix with a limited memory BFGS matrix [29]. The complexities for the first two are both $\mathcal{O}(|E|_\downarrow K)$. The complexity for the last one is $m^2|E|_\downarrow$, where m is the number of BGFS corrections[3] [29]. Equation 5 is a ℓ_2 regularized least squares problem with a tall matrix $\overrightarrow{A}$. The most expensive operations for that are forming and factoring the Gramian matrix $\overrightarrow{A}^T\overrightarrow{A}$. If we cache the factorization result, the total complexity for that is $\mathcal{O}(\overrightarrow{K}|E|_\uparrow)$ [31, Sect. 4.2].

For each training iteration, the dominate tasks for $\mathcal{S}$ are Eqs. 15 and 13. Equation 15 is essentially $|E|_\uparrow$s scalar ℓ_2 regularized logistic regressions, which can be solved using a lookup table for the approximate value, followed by one or two Newton steps [27]. The overall complexity for that is $\mathcal{O}(|E|_\uparrow)$. Equation 13 is a ℓ_2 least squares problem with a large matrix $\overleftarrow{A}$. If we cache the factorization result, the total complexity to solve that is $\mathcal{O}(\overleftarrow{K}|E|_\uparrow)$.

Assume the implicit condition that $|E|_\uparrow \gg |E|_\downarrow$ and $\overleftarrow{K} \gg \overrightarrow{K}$ holds, the workload ratio between $\mathcal{A}$ and $\mathcal{S}$ is approximately $c|E|_\downarrow/|E|_\uparrow$, where c is the maximums L-BFGS iterations controlled by Alice.

[3] In practice, we assume m to be small.

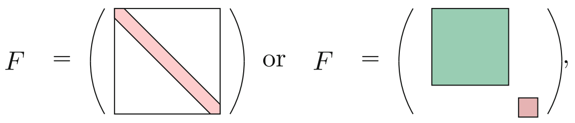
4.4 Protecting Prior Knowledge

Here we present two variations of the original learning model to incorporate Alice's prior knowledge. We assume that Alice knows *a priori* certain private features have higher influence on the link signs than the others within her network. To compensate that, we adjust the learning model by changing the regularization function. We show we can properly train the new model and protect Alice's prior knowledge using the same two-tier training method.

In practice, it is common to assume Alice vaguely knows the underlying reason of her link sign decisions. Although not wishing to reveal such reason, Alice would prefer a model that take her prior knowledge into account. Such preference can be expressed by swapping the ℓ_1 regularization function for a generalized ℓ_1 regularization function in Eq. 2,

$$\text{minimize } \frac{1}{|E|} \sum_{i=1}^{|E|} \log\left(1 + \exp\left(-Ax\right)\right) + \lambda \|Fx\|_1 \,, \tag{17}$$

where $F \in \mathbf{R}^{|E| \times |E|}$ is an arbitrary linear transformation matrix. Alice can define different regularization strengths for different feature combinations through F. If F is a diagonal matrix or a block diagonal matrix, *i.e.*,



Equations 6, 13 and 14 are separable. Therefore, we can still split the links and features through ADMM.

Another interesting variation is when Alice knows *a priori* that most private features affect her link sign decisions, *i.e.*, $\overleftarrow{w}$ is sparse but $\overrightarrow{w}$ is dense. Instead of ℓ_1 regularization, Alice can apply ℓ_2 regularization to $\overrightarrow{w}$. The problem becomes

$$\text{minimize } \frac{1}{|E|} \sum_{i=1}^{|E|} \log\left(1 + \exp\left(-Ax\right)\right) + \lambda \|\overrightarrow{w}\|_2^2 + \|\overleftarrow{w}\|_1 \,, \tag{18}$$

where the ℓ_2 regularization ensures that $\overrightarrow{w}$ is a dense vector. Since the regularizations are separable by $\overrightarrow{w}$ and $\overleftarrow{w}$, The ℓ_2 regularization is not revealed to Sara during training. Therefore, Alice's prior knowledge privacy is still protected.

5 Experimentation and Evaluation

Here we evaluate the performance of our method with real-word OSN data. Our experiments is conducted on the Wikipedia Request for Adminiship (RfA) dataset [19,32], which contains a directed, signed network with rich textual features. We use it to assess the prediction accuracy, the privacy-preserving property, and the efficiency of our method.

5.1 Wikipedia RfA Dataset

Leskovec *et al.* [19] created the Wikipedia RfA dataset by crawling and parsing the Wikipedia RfA process webpages from 2003 to 2013. The dataset contains votes casted by Wikipedia members for promoting individual editors to the role of administrator. To apply for adminship, a *request for adminship* must be submitted either by the candidate or another community member [19]. Any Wikipedia member can cast a supporting, neutral, or opposing vote along with a short comment for the RfA. The comment explains the reason of the vote. For instance, A comment for a supporting vote may read, '*I have seen him around, a trustworthy editor with a good knowledge of policy*', whereas a comment for an opposing vote may read, '*This candidate's lack of experience in the en:Wikipedia administrative arena*'.

This induces a directed, signed network in which nodes represent Wikipedia members and links represent votes. The vote comments provide rich textual features, which makes the dataset well-suited for our experiments. West *et al.* [32] post-processed the dataset to exclude all neutral votes. The current dataset contains 10,835 nodes, 159,388 links (76 % positive). The average length of vote comments is 34 characters.

5.2 Experimental Setup

We follow the same training and testing paradigm in [32]. We randomly select 10 focal nodes. For each focal node we carry out a breadth-first search (following both in-link and out-link) until we have visited 350 nodes. This gives us 10 subgraphs, each has 350 nodes. For each subgraph, we randomly select 10 % links and mark them as unknown. We use the rest 90 % links to train a sparse logistic model and test its performance using the unknown links. Just to make a fair comparison about prediction accuracy, we also train a model for each subgraph i and test it using subgraph $i + 1$ without any link masking, which follows the same setting in [32].

We use the term frequencies of the 10,000 most frequent words as link features, We excludes words that paraphrase link labels, *i.e.*, *support* or *oppose*, or words whose prefixes paraphrase link labels, *i.e.*, *support* or *oppos*. For feature split, we pre-train a regular sparse logistic model using a random sample of 10,000 comments without testing. We choose the 100 words that have highest weights and 900 random samples of the from the rest of the words as private features and the remaining 9000 words as public features. For link split, we choose half the opposing links as private links and the other half along with all supporting links as public links.

We train the model by solving a sparse logistic regression with different regularization parameters for $\overleftarrow{w}$ and $\overrightarrow{w}$, *i.e.*,

$$r(x) = \lambda_1 \|\overleftarrow{w}\|_1 + \lambda_2 \|\overrightarrow{w}\|_1,$$

where $\lambda_1 = 0.1$ and $\lambda_1 = 0.01$. We use a Python implementation to perform the two-tier training in parallel. The parallelism is provided by the IPython parallel engine. We use the `fmin_l_bfgs_b` in SciPy to update Eqs. 4, 5, and 15, which is essentially Nocedal's Fortran 77 implementation of L-BFGS [29]. We set the maximum L-BFGS iterations to 10 to limit Alice's effort. To verify the result, we also train the model without ADMM using a MATLAB implementation with CVX [33] and Gurobi [34]. All experiments are conducted on a Cray CS-300 cluster with 2.60 GHz octa-core Intel Sandy Bridge CPUs.

5.3 Evaluation Metrics

The metrics we use to evaluate the prediction accuracy are the areas under the curve (AUC) of the receiver operating characteristic (ROC) curves as well as the precision-recall (PR) curves. We only report the PR curve of the opposing links because it better describes the prediction accuracy [32]. The reason is because the class imbalance of 76 % supporting links. Even random guessing can achieve an AUC of 0.76 for supporting links comparing to an AUC of 0.24 for opposing links.

We show the privacy-preserving property by reporting the AUC/PR curves and the classification margins for the joint model, a model solely uses private features and a model solely uses public features. The last one represents the model Sara attains after the two-tier training. The differences between the three models signifies the information leakage due to exposing the public features.

The metrics we use to evaluate the algorithm efficiency and the communication overhead are the iteration versus suboptimality (IVS) curve and the cumulative runtime versus suboptimality (CRVS) curve. We consider one iteration as a complete cycle of both tiers. Due to parallelism, we report the cumulative runtime for both $\mathcal{A}$ and $\mathcal{S}$. The optimal value was verified using CVX [33] and Gurobi [34].

5.4 Results

We first verify that our training objective produces high quality prediction model. In Fig. 1, we compare our model with the sentiment model used in [32], which is trained through a ℓ_2 regularized logistic regression. We randomly mask a set of links and train both models using the comments of the remaining links. Even with data rearrangement and splitting, the performance of our model is comparable to the sentiment model in terms of AUC/ROC curve and AUC/PR curve. The sentiment model slightly edges ours at the fourth decimal point, due to the sparsity constrain in our objective. The two models agree on 95 % of the signs among the top 100 weights. Interestingly, the improvement through increasing the visible link ratio is not significant for both models. This suggests that the kurtosis of the feature weights distributions is high. Most weights have small correlation with the link sign.

Our second experiment examines the privacy-preserving property of the two-tier training method. We compare the prediction accuracies of three models, the joint model that uses both public and private features, a private model that solely uses private features and a public model that solely uses public features.

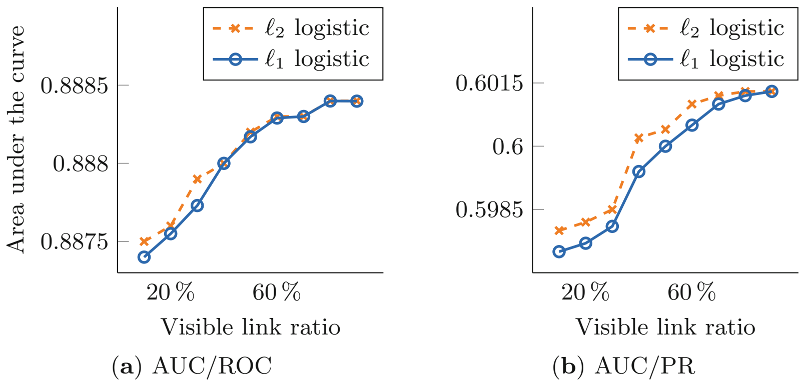


Fig. 1. Assess the model quality by comparing our model with the ℓ_2 regularized logistic regression model used in [32]. (**a**) The AUC/ROC curves are comparable between the two models. (**b**) The AUC/PR curves are comparable between the two models.

Figure 2**a** shows the AUC/ROC curves of the three. Consider a baseline model that predicts link signs through random guess. The AUC/ROC for the baseline model is exactly 0.5. The public model's performance is 10 % better than the baseline model whereas the other two are 76 % and 50 % better than the baseline model. Since Sara only learns $\overleftarrow{x}^{\star}$, her prediction accuracy is 86.8 % lower than Alice.

The public model does enjoy a slight performance bump when increasing the visible link ratio. That is because the corresponding increases of nonzero entities in $\overleftarrow{A}$, which enlarge the classification margin. But, such improvement is limited. Figure 2**b** shows the classification margin of the three models. We normalized the margins according to the largest one, *i.e.*, the margin of the joint model. The classification margin of the public model is the lowest among all three. It indicates that most predictions the public model makes are borderline cases with low confidence.

Finally, we report the training efficiency and workload for Alice and Sara. Using the experimental setup described earlier, the average number of links for each subgraph is 5000, among which 1000 are negative and 4000 are positive. This produces a matrix A of size 5000 by 10,000, which divides into three parts, a matrix $A_{\downarrow}$ of size 500 by 10,000, a matrix $\overleftarrow{A}$ of size 4500 by 9900 and a matrix $\vec{A}$ of size 4500 by 100.

We measure the training process using the objective suboptimality. Let $\tilde{o}^k$ be the objective value at the kth iteration

$$\tilde{o}^k = \frac{1}{|E|}\sum_{i=1}^{|E|} \log\left(1+\exp\left(-Az_{\updownarrow}^k\right)\right) + r(z_{\updownarrow}^k).$$

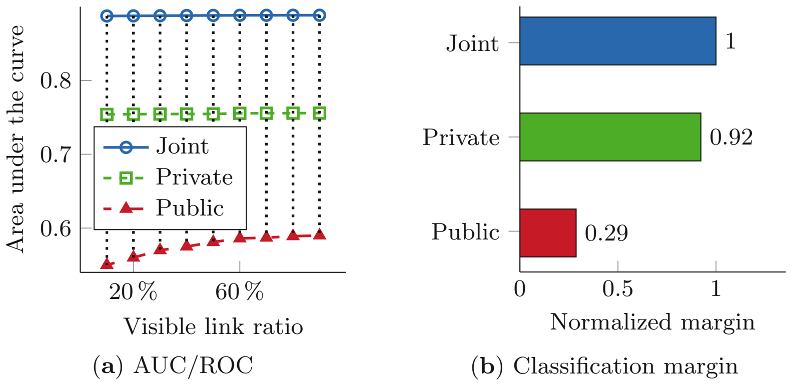


Fig. 2. Observe the private preserving property by comparing the prediction accuracies and the classification margins. (**a**) Using only public features, Sara's prediction accuracy is 86.8 % lower than Alice. (**b**) Using only public features, Sara's classification margins (prediction confident) is 71.0 % lower than Alice.

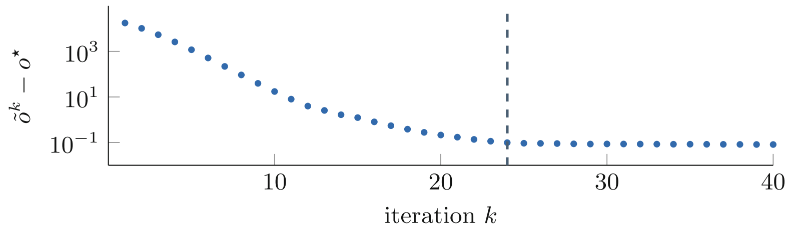
Let $o^\star$ be the optimal objective value

$$o^\star = \frac{1}{|E|}\sum_{i=1}^{|E|} \log\left(1 + \exp\left(-Ax^\star\right)\right) + r(x^\star).$$

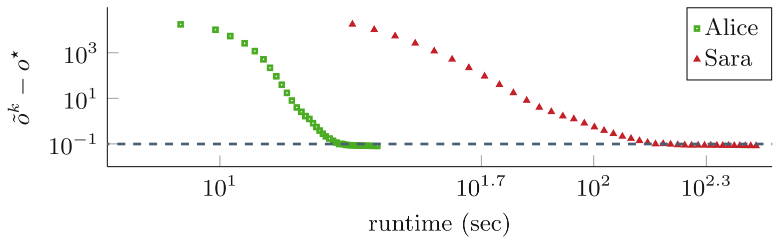
The objective suboptimality is the difference between $\tilde{o}^k$ and $o^\star$, *i.e.*, $\tilde{o}^k - o^\star$. The optimal value $o^\star = 0.9752 \times 10^5$ is verified by our MATLAB implementation. Figure 3**a** shows the training progress by iteration. The dashed line marks the iteration when the stopping criterion is satisfied. The algorithm only takes 24 iterations to reach the optimal, which greatly reduces the communication overhead between Alice and Sara.

Figure 3**b** shows the CRVS curves for Alice and Sara. The dashed line marks convergence. The main task for Alice is to compute Eq. 5 using L-BFGS. We use L-BFGS with warm starting to reduce Alice's workload. This approach is effective in later iteration when the iterates approach consensus. The main task for Sara is to compute Eqs. 13, 15 and various matrix-vector multiplications. We cache the matrix factorization to reduce Sara's workload. However, Sara still need to compute large back-solves to produce the result. For both Alice and Sara, the runtime of early iterations is significantly longer than the latter ones. Overall, Sara's workload is approximately 10 times larger than Alice's workload.

To summarize, the experiments show three points: (1) Our decentralized method achieves equally high predication accuracy as the sentiment model used in [32]. Using data splitting to protect private data does not affect the modal quality. (2) Sara is oblivious of Alice's private data and their corresponding weights. Without Alice's help, Sara's prediction accuracy is fairly poor. Alice, on the other hand, enjoy the full benefit of the collaboration and is able to acquire high prediction accuracy with minimal efforts. (3) The data splitting



(**a**) IVS



(**b**) CRVS

Fig. 3. Convergence analysis of the two-tier training method. The stopping criterion is satisfied at iteration 24, marked by the dashed lines. (**a**) X axis is the number of iterations, Y axis it the suboptimality measured by $\tilde{o}^k - o^\star$. (**b**) X axis is the cumulative runtime measured in seconds. The total runtime for Alice is 90.4 % lower than Sara.

assigns appropriate workload for Alice and Sara to fully utilize their computation resources.

6 Conclusion

In this paper, we studied the privacy-preserving link prediction problem in decentralized OSNs. We proposed a novel decentralized two-tier method that allows end users to collaborate with their online service providers without revealing their private data. Using a real-world social network dataset, we showed that our method produces high quality prediction model while eases users' computing burden. Additionally, we showed that our method can be secure against the link reconstruction attack. In the era of "Big Data", our method bridges the gap between the increasing volume of personal data and the insufficient analyzing resources of privacy conscious users in decentralized OSNs.

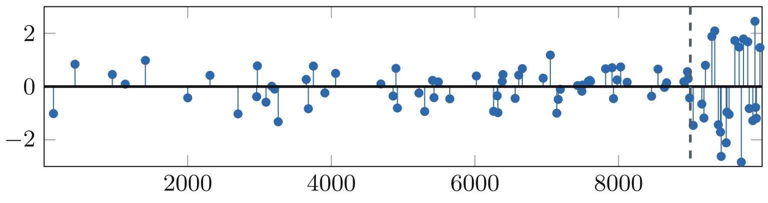
Acknowledgment. This work was supported by US National Science Foundation under grants CNS-1405747, CNS-1156318, CNS-1443889, and CSR-1217889.

A Appendix: Link Reconstruction Attack

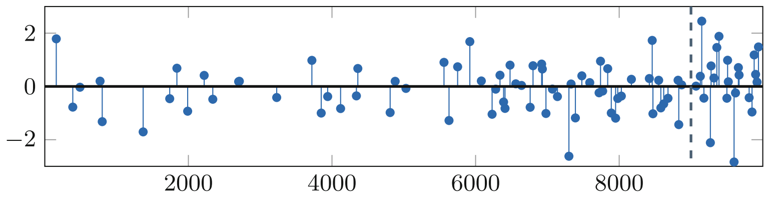
Here we assess the possibility of link reconstruction attack. In a link reconstruction attack, we consider a passive adversary, Eve, tries to predict the link signs of Alice's network. Just like Sara, Eve can access the public data matrix $\overleftarrow{A}$. But, unlike Sara, Eve cannot collaborate with Alice. We assume Eve knows the training method Alice and Sara use. The goal for Eve is to build a model that has the same prediction power as the joint model. We evaluate the attack possibility by measuring Eve's prediction accuracy.

A.1 Experimental Setup

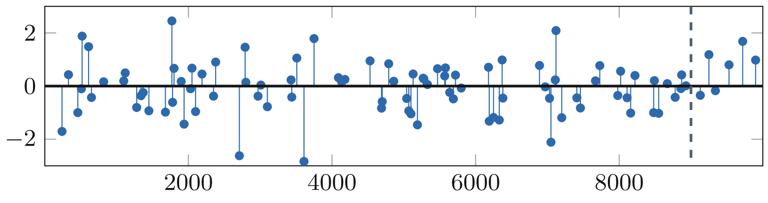
Our previous experiments assume that Alice makes rational decision and protects features that have high correlation with the link sign. Although unlikely, Alice can split the data in such a way that all private features are irrelevant to the link sign. To consider different public/private splits, we shuffle the features of the matrix A used in previous experiments.



(**a**) Initial weight vector



(**b**) Partial mixed



(**c**) Full mixed

Fig. 4. Weight vectors generated by random walking the nonzero entries in the initial weight vector. The dashed line marks the feature split point. (**a**) The initial weight vector is the same as the optimal weight vector $w^\star$ learned from the Wikipedia RfA dataset. (**b**) Private features and public features are partially mixed. (**c**) Private features and public features are uniformly distributed.

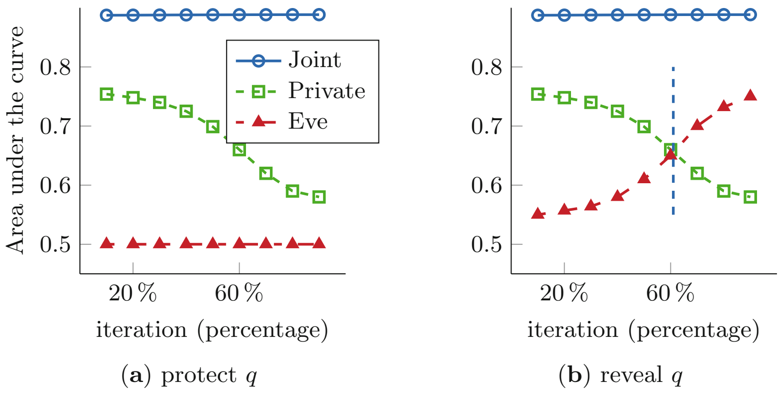


Fig. 5. AUC/ROCs of the public and private models versus shuffling iteration. The shuffling iterations is represented as the percentage of the maximum iterations. (**a**) Eve can not improve her prediction accuracy if Alice protect q. (**b**) Eve can improve her prediction accuracy if Alice reveal q. The dashed line marks the tipping point where Eve's model surpasses the private model.

The shuffle is done by rearranging features that have nonzero weights. We use a Monte Carlo random walk algorithm to shuffle matrix A. For each iteration, the algorithm moves the features that have nonzero weights randomly to the left or to the right. The resulting 'true' weight vectors are shown in Fig. 4. The initial weight vector represents the split that assigns top weighted features to the private data matrix $\overrightarrow{A}$. The random walk mixes the features in $\overrightarrow{A}$ with the features in $\overleftarrow{A}$. As the walking time increases, the 'true' weights approaches a steady state uniform distribution.

We use the total variation mixing time [35], *i.e.*, $t = n \log n$, where n equals 10,000, as the maximum mixing iterations. For each $0.1t$, we record the shuffled matrix. This gives us 10 matrices whose 'true' weights gradually change from unevenly distributed to uniformly distributed. We split these matrices at a fixed index position. For each matrix, We train a prediction model for Eve solely using $\overleftarrow{A}$. We compare Eve's model with the private model and the joint model trained by Alice and Sara.

A.2 Results

Figure 5 shows the prediction accuracy of Eve's model versus shuffling iteration. The results are distinct for two different scenarios. If the link sign column q is within $\overrightarrow{A}$, shown in Fig. 5**a**, Eve is forced to train her model with random guesses. In that case, moving high weighted features into $\overleftarrow{A}$ does not improve Eve's model. Eve's prediction accuracy is exactly 0.5 regardless the features she uses.

If the link sign column q is within $\overleftarrow{A}$, shown in Fig. 5**b**, Eve can properly train her model using the same sparse logistic regression. In that case, Eve's model can be improved if more high weighted features are within $\overleftarrow{A}$. In our experiment, Eve is able to increase her prediction accuracy by over 50 % when the weights are uniformly distributed.

Although the security properties are different, the impacts on Sara's prediction accuracy are the same. When the high weighted features are moved from $\overrightarrow{A}$ to $\overleftarrow{A}$, the predication accuracy of the private model decreases; the prediction accuracy of the public model increases; the prediction accuracy of the joint model remains the same. Sara, who benefits from collaborating with Alice, is able to make more accurate prediction using the public model.

To summarize, the experiments show: (1) that Alice can prevent the link reconstruction attack by marking the link sign ground truth as private. (2) Although not violating Alice's privacy, inappropriate data split could accidentally increases Sara's prediction power.

References

1. Google finance. Accessed 19 June 2014. https://www.google.com/finance
2. Statistic brain. Accessed 20 June 2014. http://www.statisticbrain.com
3. Zheleva, E., Getoor, L.: To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In: Proceedings of the 18th International Conference on World Wide Web, pp. 531–540 (2009)
4. Facebook. Data use policy. Accessed 25 June 2014. https://www.facebook.com/about/privacy
5. Dodson, B., Vo, I., Purtell, T., Cannon, A., Lam, M.: Musubi: disintermediated interactive social feeds for mobile devices. In: Proceedings of the 21st International Conference On World Wide Web, pp. 211–220 (2012)
6. Diaspora Inc. Accessed 28 May 2014. http://diasporaproject.org
7. Omlet Inc. Accessed 28 May 2014. http://www.omlet.me
8. Aggarwal, C.C.: Social Network Data Analytics. Springer, US (2011)
9. Datta, A., Buchegger, S., Vu, L.-H., Strufe, T., Rzadca, K.: Decentralized online social networks. In: Furht, B. (ed.) Handbook of Social Network Technologies and Applications, pp. 349–378. Springer, US (2010)
10. Baden, R., Bender, A., Spring, N., Bhattacharjee, B., Starin, D.: Persona: an online social network with user-defined privacy. ACM SIGCOMM Comput. Commun. Rev. **39**(4), 135–146 (2009)
11. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: a privacy-preserving online social network leveraging on real-life trust. IEEE Commun. Mag. **47**(12), 94–101 (2009)
12. Backstrom, L., Leskovec, J.: Supervised random walks: predicting and recommending links in social networks. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, pp. 635–644 (2011)
13. Leskovec, J., Mcauley, J.J.: Learning to discover social circles in ego networks. Adv. Neural Inf. Process. Syst. **25**, 539–547 (2012)
14. Fisher, D.: Using egocentric networks to understand communication. IEEE Internet Comput. **9**(5), 20–28 (2005)
15. Dwork, C.: Differential privacy. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 1–12. Springer, Heidelberg (2006)
16. Yu, H., Jiang, X., Vaidya, J.: Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. In: Proceedings of the 2006 ACM Symposium on Applied Computing, pp. 603–610. ACM (2006)

17. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrust. In: Proceedings of the 13th International Conference on World Wide Web, pp. 403–412 (2004)
18. Al Hasan, M., Chaoji, V., Salem, S., Zaki, M.: Link prediction using supervised learning. In: Proceedings of SDM Workshop on Link Analysis, Counter-terrorism and Security (2006)
19. Leskovec, J., Huttenlocher, D., Kleinberg, J.: Predicting positive and negative links in online social networks. In: Proceedings of the 19th International Conference on World Wide Web, pp. 641–650 (2010)
20. Kim, M., Leskovec, J.: Latent multi-group membership graph model. In: Proceedings of the 29th International Conference on Machine Learning, pp. 1719–1726 (2012)
21. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted boltzmann machines for collaborative filtering. In: Proceedings of the 24th International Conference on Machine Learning, pp. 791–798 (2007)
22. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. Springer, Heidelberg (2009)
23. Hinton, G., Osindero, S., Teh, Y.-W.: A fast learning algorithm for deep belief nets. Neural Comput. **18**(7), 1527–1554 (2006)
24. Hinton, G.: A practical guide to training restricted boltzmann machines. Momentum **9**(1), 926 (2010)
25. Glowinski, R., Marroco, A.: Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. ESAIM: Math. Model. Numer. Anal. Modélisation Math. Anal. Numérique **9**(R2), 41–76 (1975)
26. Gabay, D., Mercier, B.: A dual algorithm for the solution of nonlinear variational problems via finite element approximation. Comput. Math. Appl. **2**(1), 17–40 (1976)
27. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learn. **3**(1), 1–122 (2011)
28. Parikh, N., Boyd, S.: Proximal algorithms. Found. Trends Optim. **1**(3), 123–231 (2013)
29. Byrd, R.H., Lu, P., Nocedal, J., Zhu, C.: A limited memory algorithm for bound constrained optimization. SIAM J. Sci. Comput. **16**(5), 1190–1208 (1995)
30. Donoho, D.L.: De-noising by soft-thresholding. IEEE Trans. Inf. Theory **41**(3), 613–627 (1995)
31. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore (2013)
32. West, R., Paskov, H.S., Leskovec, J., Potts, C.: Exploiting social network structure for person-to-person sentiment analysis. Trans. Assoc. Comput. Linguist. **2**, 297–310 (2014)
33. Grant, M., Boyd, S.: CVX: Matlab software for disciplined convex programming, version 2.1 (2014). http://cvxr.com/cvx
34. Gurobi Optimization Inc., Gurobi optimizer reference manual, version 5.6 (2014). http://www.gurobi.com/documentation
35. Levin, D.A., Peres, Y., Wilmer, E.L.: Markov Chains and Mixing Times. American Mathematical Society, Providence (2009)