Privacy Preserving Semi-supervised Learning for Labeled Graphs

Hiromi Arai[1] and Jun Sakuma[1,2]

[1] Department of Computer Science, University of Tsukuba,
1–1–1 Tenoudai, Tsukuba, Japan
arai.hiromi.ga@u.tsukuba.ac.jp, jun@cs.tsukuba.ac.jp
[2] Japan Science and Technology Agency, 5–3, Yonban-cho, Chiyoda-ku, Tokyo, Japan

Abstract. We propose a novel privacy preserving learning algorithm that achieves semi-supervised learning in graphs. In real world networks, such as disease infection over individuals, links (contact) and labels (infection) are often highly sensitive information. Although traditional semi-supervised learning methods play an important role in network data analysis, they fail to protect such sensitive information. Our solutions enable to predict labels of partially labeled graphs without disclosure of labels and links, by incorporating cryptographic techniques into the label propagation algorithm. Even when labels included in the graph are kept private, the accuracy of our PPLP is equivalent to that of label propagation which is allowed to observe all labels in the graph. Empirical analysis showed that our solution is scalable compared with existing privacy preserving methods. The results with human contact networks showed that our protocol takes only about 10 seconds for computation and no sensitive information is disclosed through the protocol execution.

Keywords: privacy preserving data mining, semi-supervised learning.

1 Introduction

Label prediction of partially labeled graphs is one of the major machine learning problems. Graph-based semi-supervised learning is useful when link information is obtainable with lower cost than label information. Prediction of protein functions is one familiar example [12]. In this problem, the functions and similarities of proteins correspond to the labels and node similarities, respectively. Amassing information about the protein functions requires expensive experimental analyses, while protein similarities are often obtained computationally, with a lower cost. Taking advantage of this gap, the semi-supervised approach successfully achieves better classification accuracy, even when only a limited number of labeled examples is obtainable.

The key observation of this study is that the difficulty of preparing label information is not only its cost, but also its privacy. Even when a large number of labels have already been collected, the labels might not be observed or may require extreme caution in handling. In this paper, we consider label prediction

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part I, LNAI 6911, pp. 124–139, 2011.

in a situation where the entire graph cannot be observed by any entities, due to privacy reasons. Such a situation is often found in networks among social entities, such as individuals or enterprises, in the real world. The following scenarios pose intuitive examples where privacy preservation is required in label prediction.

Consider a physical contact network of individuals, in which individuals and their contacts correspond to nodes and links, respectively. Suppose an infectious disease is transmitted by contact. Some of the individuals have tested their infection states. Regarding infection states as node labels, semi-supervised learning is expected to predict the infection states of untested individuals, by exploiting the existing test results and the contact network. However, the contact information between individuals and their infection states can be too sensitive to disclose.

In this scenario, both the labels and the links must be kept private. In order to formulate such situations, we consider three types of typical privacy models. The first model, referred to as the *public model*, assumes that each node discloses its label and links to all other nodes. One example of this model is social network services such as facebook or LinkedIn, when the privacy policy is set as "everyone can see everything". Users (nodes) disclose their friends (links) and status (node labels), such as their education or occupations, to every user.

The second model, referred to as the *label-aware model*, assumes each node discloses its label and links only to the nodes that it is linked to. Let the node labels correspond to the flu infection states in the contact network. We can naturally assume that each individual (node) is aware of the individuals with whom he/she had contact before (links) and whether or not they had the flu (labels of the nodes it is linking to), but he/she would never know the contact information or the infection states of individuals he/she has never met.

The third model, referred to as the *label-unaware model*, assumes that each node does not disclose any links or labels to others. Consider the contact network again. Let links and labels correspond to sexual relationships and sexually transmitted infections, respectively. In such a case, no one would disclose their links and label to others; the label may not be disclosed even to individuals with whom he/she had a relationship.

In addition, asymmetries of relationships need to be considered. For example, in a business network whose links correspond to the ratio of the stock holdings, a directed graph should be used to represent the scope of observation. Thus, the privacy model to be employed depends on the nature and the sensitivity of the information in the graphs.

Related Works. Existing label prediction methods are designed with the implicit assumption that a supervisor exists who can view anything in the graph (the public model). If we could introduce a trusted third party (TTP)[1] as a supervisor, then any label prediction algorithms, such as TSVM [8] or Cluster kernel [12], would immediately work in the label-(un)aware model; however, facilitating such a party is unrealistic in general (Table 1, the first line).

[1] TTP is a party which never deviates from the specified protocol and does not reveal any auxiliary information.

Table 1. Comparison of label prediction methods. Δ: the maximum number of links of a node, T: the number of iterations of each algorithm.

method	privacy model	comp. cost
TSVM , Cluster kernel	public	—
k-NN	label-aware	$O(\Delta \log \Delta)$
LP	label-aware	$O(\Delta T)$
LP/SFE	label-unaware	$O(\text{poly})$
PPLP (proposal)	label-unaware	$O(\Delta T)$

The k-nearest neighbor (kNN) method predicts labels from the labels of the k-nearest nodes; kNN works in the label-aware model (Table 1, the second line). Label propagation (LP) [15] achieves label prediction in the label-aware model, when algorithms are appropriately decentralized (Table 1, the third line, see Sect. 3.2 for details). Note that even if it is decentralized, each node has to disclose its labels to the neighboring nodes. That is, kNN and LP do not work in the label-unaware model.

Secure function evaluation (SFE) [13], also referred to as Yao's garbled circuits, is a methodology for secure multiparty computation. Using SFE, any function, including label prediction methods, can be carried out without revealing any information except for the output value. That is, SFE allows execution of label prediction in the label-unaware model (Table 1, the fourth line). Although the computational cost of SFE is polynomially bounded, it can be too inefficient for practical use. We implemented label prediction on SFE (LP/SFE), and the efficiency of SFE is discussed with experiments in Sect. 6.

Privacy-preserving computations for graph mining are discussed in [4,10]. Both of them calculate HITS and PageRank from networks containing private information. They compute the principal eigenvector of the probability transition matrix of networks without learning the link structure of the network. Our solution for label prediction is similar to above link analyses in the sense that both of them consider computation over graphs containing private graph structures. However, the target of their protocols and ours are different; their protocol computes node ranking while our protocol aims at node label prediction.

Our Contribution. As discussed above, no label prediction methods that work efficiently in the label unaware model have been presented. We propose a novel solution for privacy preserving label prediction in the label-unaware model. Comparisons between our proposal and existing solutions are summarized in Table 1. First, we formulate typical privacy models for labeled graphs (Sect. 2). Then, a label propagation algorithm that preserves the privacy in the label-unaware model is presented (Sect. 5). Our proposal theoretically guarantees that (1) no information about links and labels is leaked through the entire process, (2) the predicted labels as the final output are disclosed only to the node itself, and (3) the final output is exactly equivalent to that of label propagation. Connections to our protocol and differential privacy are also discussed in Sect. 5.3.

The experiments show that the computational time of our proposal is more than 100 times shorter than that of LP/SFE. We also examined our proposal using the real social network data (Sect. 6).

2 Privacy in Labeled Graphs

In this section, we first formulate the label prediction problems with graphs. Then we define privacy models of graphs; the secure label prediction problem is formulated using these models.

2.1 Labeled Graph

In a network of social entities, the links and labels can be private. In order to formulate the privacy in graphs, we introduce *graph privacy models*, assuming that each node in the graph is an independent node whose computational power is polynomially bounded.

Let $G = (V, E)$ be a graph where $V = \{1, \ldots, n\}$ is a set of nodes and $E = \{e_{ij}\}$ is a set of links. Link e_{ij} has nonnegative weight w_{ij}; the weight matrix is $\mathbf{W} = (w_{ij})$. If $e_{ij} \notin E$, $w_{ij} = 0$. In directed graphs, we denote the nodes linked from node i as $N_{\text{out}}(i) = \{j | j \in V, e_{ij} \in E\}$; the nodes linking to node i are denoted as $N_{\text{in}}(i) = \{j | j \in V, e_{ji} \in E\}$. In undirected graphs, the nodes linking to node i and the nodes linked from node i are identical. So we denote $N(i) = N_{in}(i) = N_{out}(i)$.

In this study we consider wegited labeled graphs without disconnected singleton nodes. Typically, the node label is represented as an element of a finite set $\{1, \ldots, h\}$. To facilitate the computations introduced later, we introduce the *label matrix* $\mathbf{F} = (f_{ik}) \in \mathbb{R}^{n \times h}$ to represent the node labels. The i-th row of $\mathbf{F}$ represents the label of node i. The node is labeled as s, if $s = \arg\max_{1 \le k \le h} f_{ik}$.

Thus, links, link weights, and labels of graphs are represented as matrices. In order to define which part of a matrix is observable from a node and which is not, three different matrix partitioning models are introduced. Then, the graph privacy models are defined, based on the matrix partitioning models.

2.2 Matrix Partitioning Model

Let there be a set of n nodes V and a matrix $\mathbf{M} \in \mathbb{R}^{n \times r}$. Let the ith row of $\mathbf{M}$ by $\mathbf{m}_{i*}$; the ith column by $\mathbf{m}_{*i}$. Suppose $\mathbf{M}$ is partitioned into n parts, and each node in V is allowed to observe a different part of the matrix privately. We used the following typical patterns of partitioning [10].

Definition 1. *(row private) Let there be a matrix $\mathbf{M} \in \mathbb{R}^{n \times r}$ and n nodes. For all i, if the ith node knows the ith row vector $\mathbf{m}_{i*}$, but does not know other row vectors $\mathbf{m}_{p*}$ where $p \neq i$, then $\mathbf{M}$ is row private.*

Definition 2. *(symmetrically private) Let there be a square matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ and n nodes. For all i, if the ith node knows the i-th row vector $\mathbf{m}_{i*}$ and the i-th column vector $\mathbf{m}_{*i}$, but does not know other elements m_{pq} where p or $q \neq i$, then $\mathbf{M}$ is symmetrically private.*

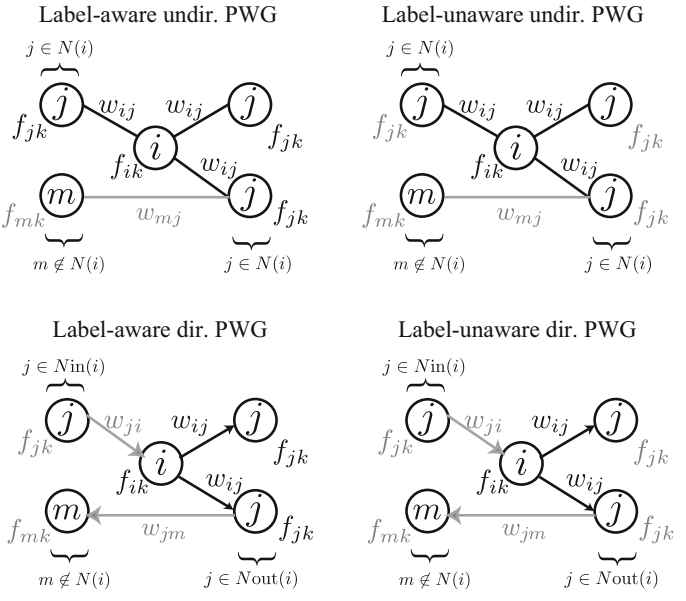

Fig. 1. Graph privacy models. Links, link weights w_{ij}, and label f_{ik} that are private from node i are depicted as gray.

We define an expansion of row private, in which node i is allowed to observe specified rows of $\mathbf{M}$.

Definition 3. *($U(i)$-row private) Let there be a matrix $\mathbf{M} \in \mathbb{R}^{n \times r}$ and n nodes. Let $U(i) \subseteq V$. For all i, if the ith node knows the jth row vector $\mathbf{m}_{j*}$ where $j \in U(i) \cup \{i\}$, but does not know other row vectors $\mathbf{m}_{p*}$ where $p \notin U(i) \cup \{i\}$, then $\mathbf{M}$ is $U(i)$-row private.*

2.3 Graph Privacy Model

Graph privacy models define the restriction of the observation of link weights $\mathbf{W}$ and labels $\mathbf{F}$ from each node, based on these matrix partitioning models.

Definition 4. *(label-aware undirected PWG) Let $G = (V, E)$ be an undirected graph with weight matrix $\mathbf{W}$ and label matrix $\mathbf{F}$. If $\mathbf{W}$ is symmetrically private and $\mathbf{F}$ is $N(i)$-row private, then G is a label-aware undirected private weighted graph (PWG).*

In this model, $\mathbf{W}$ is symmetrically private. It follows that node i is allowed to learn the nodes that node i is linking to/linked to from and the link weights thereof. Besides, label matrix $\mathbf{F}$ is row-$N(i)$ private. It follows that node i is allowed to learn the labels of the nodes that node i is linking to/linked to from. Fig 1 gives the schematic description of this graph privacy model.

Table 2. Matrix partitioning of graph privacy models

graph privacy model	**W**	**F**
label-aware undirected PWG	symmetry private	$N(i)$-row private
label-unaware undirected PWG	symmetry private	row private
label-aware directed PWG	row private	$N_{out}(i)$-row private
label-unaware directed PWG	row private	row private

Definition 5. *(label-unaware undirected PWG) In Definition 4, if* $\mathbf{F}$ *is row private, then* G *is a label-unaware undirected private weighted graph.*

The difference in the above two models is in the matrix partitioning model of label matrix $\mathbf{F}$. Node i is allowed to learn its neighbors' labels in label-aware undirected PWGs, while node i is allowed to learn only its own node label in label-unaware undirected PWGs. Recall the contact network whose nodes are labeled with infection states. If the labels correspond to the flu state, then the label-aware undirected PWG would be appropriate. On the other hand, if the labels correspond to disease state of sexually transmitted infection, then the label information should be handled in the label-unaware undirected PWG.

Table 2 summarizes the definitions of graph privacy models. In the following, we consider undirected graphs, unless specifically mentioned. The treatment of label-(un)aware directed PWGs is revisited in Sect. 5.4.

3 Our Approach

Our aim is to develop label prediction that securely works under given graph privacy models. First, we state the problem, based on graph privacy models and label propagation [15]. Then, we show that a decentralization of label propagation makes the computation secure in label-aware PWGs, but not secure in label-unaware PWGs. At the end of this section, we clarify the issues to be addressed to achieve secure label prediction in label-unaware PWGs.

3.1 Problem Statement

We consider the label prediction problem of node labeled graphs. Suppose some of the nodes are labeled (initial labels), while the remaining nodes are unlabeled. Then, the problem of label prediction is to predict the labels of the unlabeled nodes, using the initial labels and the weight matrix as input.

Let $\mathbf{P} = \mathbf{D}^{-1}\mathbf{W}$, which corresponds to the probability transition matrix of the random walk on graph G with weight matrix $\mathbf{W}$. $\mathbf{D}$ is the degree matrix, where $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n)$ and $d_i = \sum_{j \in V} w_{ij}$. Let $\mathbf{Y} = (y_{ij}) \in \mathbb{R}^{n \times c}$ be an initial label matrix, where $y_{ip} = 1$ if the label of node i is p; otherwise $y_{ip} = 0$.

Label propagation predicts the labels of unlabeled nodes by propagating the label information of labeled nodes through links following the transition probabilities. The label matrix is repeatedly updated by

$$\mathbf{F}^{(t+1)} = \alpha \mathbf{P}\mathbf{F}^{(t)} + (1 - \alpha)\mathbf{Y} \tag{1}$$

where α ($0 \leq \alpha < 1$). If α is closer to 1, then more label information is propagated from its neighbors (the first term). If α is closer to 0, then the prediction is affected more by its initial labels (the second term). With the condition $0 \leq \alpha < 1$, the following lemma shows the convergence property of label propagation [14].

Lemma 1. *Let* $\mathbf{F}^* = (1-\alpha)(\mathbf{I} - \alpha\mathbf{P})^{-1}\mathbf{Y}$. *When an arbitrary matrix* $\mathbf{F}^{(0)}$ *is updated iteratively by eq. 1,* $\lim_{t\to\infty} \mathbf{F}^{(t)} = \mathbf{F}^*$ *holds.*

The convergent matrix $\mathbf{F}^*$ gives the prediction for unlabeled nodes. $\mathbf{F}^*$ can be viewed as a certain type of quantity that refers to the probabilities of a node belonging to a label. This label propagation follows neither label-aware PWGs nor label-unaware PWGs because $\mathbf{W}$ and $\mathbf{Y}$ need to be observable from a node to execute eq. 1.

Suppose a partially labeled graph has to follow the label-(un)aware PWG, and the nodes in the graph wish to perform label prediction. In this study, we define the security of label prediction in the manner of secure multiparty computation [6], using the graph privacy models defined previously.

Statement 1. *Secure label prediction*: *Let there be a label-(un)aware PWG, where* $\mathbf{W}$ *and* $\mathbf{Y}$ *are private link weights and labels, respectively. After the execution of secure label prediction among the nodes, the final output* $\mathbf{F}^*$ *is correctly evaluated and distributed such that* $\mathbf{F}^*$ *is row private. Each node learns nothing but the number of iterations; the graph is still kept as a label-(un)aware PWG.*

For label prediction to be secure in the sense of Statement 1, the graph privacy models cannot be changed before and after label prediction.

3.2 Decentralized Label Propagation

Before considering our approach, we first review a decentralization of eq. 1, in terms of privacy. Looking at an element of $\mathbf{F}$, the update of f_{ik} is described as:

$$f_{ik}^{(t)} \leftarrow \alpha\Big(\sum_{j\in N(i)} p_{ij} f_{jk}^{(t-1)}\Big) + (1-\alpha) y_{ik}. \tag{2}$$

In order for node i to update eq. 2, node i needs f_{jk} for $j \in N(i)$, y_{ik} for all k, and p_{ij} for all j. Noting that $p_{ij} = w_{ij} / \sum_{j\in N(i)} w_{ij}$, node i can have p_{ij} for all j, even if $\mathbf{W}$ is row private. y_{ik} is also obtained if $\mathbf{Y}$ is row private. On the other hand, $f_{jk}^{(t-1)}$ has to be obtained from the neighbor nodes $N(i)$; thus $\mathbf{F}$ has to be $N(i)$-row private. It follows that node i can securely update eq. 2 in label-aware PWGs. Consequently, decentralized label propagation is secure in the label-aware PWGs, while it is not secure in the label-unaware PWGs, in the sense of Statement 1.

In order to perform label prediction securely in the label-unaware PWG, node i needs to evaluate eq. 2 with $\mathbf{F}$ keeping row private. In other words, node i needs to compute the first term of eq. 2 without observing f_{jk}, where $j \in N(i)$. This is seemingly impossible, but it can be achieved by introducing a cryptographic tool, homomorphic encryption. This enables to evaluate addition of encrypted values without decryption. In next section, we review homomorphic encryption.

4 Cryptographic Tools

In a public key cryptosystem, encryption uses a *public key* that can be known to everyone, while decryption requires knowledge of the corresponding *private key*. Given a corresponding pair of $(\mathsf{sk}, \mathsf{pk})$ of private and public keys and a message m, then $c = \mathrm{Enc}_{\mathsf{pk}}(m; r)$ denotes the random encryption of m, and $m = \mathrm{Dec}_{\mathsf{sk}}(c)$ denotes the decryption. The encrypted value c uniformly distributes over $\mathbb{Z}_N = \{0, ..., N–1\}$, if r is taken from $\mathbb{Z}_N$ randomly. An *additive homomorphic cryptosystem* allows the addition of encrypted values, without knowledge of the private key. There is some operation $\cdot$ such that for any plaintexts m_1 and m_2,

$$\mathrm{Enc}_{\mathsf{pk}}(m_1 + m_2 \mod N; r) = \mathrm{Enc}_{\mathsf{pk}}(m_1; r_1) \cdot \mathrm{Enc}_{\mathsf{pk}}(m_2; r_2),$$

where r is uniformly random, provided that at least one of r_1 and r_2 is uniformly random. Based on this property, it also follows that given a constant k and the encryption $e_{\mathsf{pk}}(m_1; r)$, we can compute multiplications by k via repeated applications of $\cdot$, denoted as $\mathrm{Enc}_{\mathsf{pk}}(km \mod N; kr) = \mathrm{Enc}_{\mathsf{pk}}(m; r)^k$. The random encryption prevents inference of intermediate computation from messages passing in our solution. In non-probabilistic public cryptosystem, m corresponds to one-to-one with its cipher c. Therefore, when the size of message space is small, as in label prediction, regular public cryptosystem is not adequate for privacy preserving protocol including label prediction. In the probabilistic encryption, encryption of message m is randomized so that the encryption does not released information about m. This random number is not required for decryption or addition of the message, so in the following, we omit the random number r from our encryptions, for simplicity.

In an (n, θ)-*threshold cryptosystem*, n nodes share a common public key pk, while each node holds a private key $\mathsf{sk}^1, ..., \mathsf{sk}^n$. Each node can encrypt any message with the common public key. Decryption cannot be performed by fewer than θ nodes, and can be performed by any group of at least θ nodes, using a recovery algorithm based on the public key and their *decryption shares* $\mathrm{Dec}_{\mathsf{sk}^1}(c), ..., \mathrm{Dec}_{\mathsf{sk}^n}(c)$. Our solution makes use of (n, θ)-threshold additively homomorphic cryptosystem, such as the generalized Paillier cryptosystem [2].

5 The Main Protocol

As discussed in Sect. 3.2, the decentralized label prediction algorithm is not secure in the label-unaware undirected PWG. We show privacy preserving label propagation (PPLP) for label-unaware undirected PWGs, by incorporating homomorphic encryption into the decentralized label propagation.

The main body of our protocol is presented in Fig. 2. In principle, our protocol behaves almost equivalently to the decentralized label propagation, except that every message is encrypted so that it outputs the identical result to the regular label propagation. In what follows, we explain the details of our protocol and the security in label-unaware undirected PWGs.

5.1 Privacy Preserving Label Propagation

The protocol is presented in Fig. 2.

Setup. All nodes jointly prepare a key set, so that pk is commonly known to all nodes and sk^i is held only by the ith node. The key set is securely prepared by means of distributed key generation schemes. See [3] for examples. In this protocol, we assume all nodes can refer a global clock so that they can execute iteration synchronously.

Recall that $\mathbf{W}$ is symmetrically private, and $\mathbf{F}$ and $\mathbf{Y}$ are row private in the label-unaware undirected PWG. Private inputs of node i are $\mathbf{w_{i*}}$ and $\mathbf{y_{i*}}$. With these, node i can compute p_{ij} locally in Step 1 (a). αp_{ij} can be rational. Considering the homomorphic encryption takes only integers as inputs, a large integer L is multiplied so that $\tilde{p}_{ij} \in \mathbb{Z}_N$. $\alpha f_{ik}^{(0)}$ and $(1-\alpha)y_{ik}$ are similarly magnified so that $\tilde{f}_{ik}^{(0)}, \tilde{y}_{ik} \in \mathbb{Z}_N$ in Step 1 (b), and then encrypted in Step 1 (c).

Iteration and Convergence. Step 2 updates the label vectors on the homomorphic encryption. In Step 2 (a), node i sends its encrypted label vector $c_{ik}^{(t-1)}$ to its neighboring nodes $N(i)$; in Step 2 (b), node i receives the encrypted label vectors $c_{jk}^{(t-1)}$ from the nodes it is linked to. Then, the label vectors are updated by the following equation:

$$\mathrm{Enc}_{\mathsf{pk}}(\tilde{f}_{ik}^{(t)}) \leftarrow \prod_{j \in N_{(i)}} \left((c_{jk}^{(t-1)})^{\tilde{p}_{ij}} \right) \cdot \mathrm{Enc}_{\mathsf{pk}} \left(L^{t-1} \tilde{y}_{ik} \right). \tag{3}$$

Let $\tilde{\mathbf{F}}^{(t)} = (\tilde{f}_{ij}^{(t)})$. The convergence of iterations of eq. 3 is shown as follows;

Lemma 2. *Let* $\mathbf{F}^* = (1-\alpha)(\mathbf{I} - \alpha\mathbf{P})^{-1}\mathbf{Y}$. *When an arbitrary matrix* $\tilde{\mathbf{F}}^{(0)}$ *is updated iteratively by eq. 3,* $\lim_{t\to\infty} \tilde{\mathbf{F}}^{(t)}/L^t = \mathbf{F}^*$ *holds.*

Proof. From the homomorphic property of cryptosystem, eq. 3 is:

$$\begin{aligned}
&\mathrm{Enc}_{\mathsf{pk}}(\tilde{f}_{ik}^{(t)}) \leftarrow \Big(\prod_{j \in N(i)} \mathrm{Enc}_{\mathsf{pk}}(\tilde{f}_{jk}^{(t-1)})^{\tilde{p}_{ij}} \Big) \cdot \mathrm{Enc}_{\mathsf{pk}} \left(L^{t-1} \tilde{y}_{ik} \right) \\
&= \Big(\prod_{j \in N(i)} \mathrm{Enc}_{\mathsf{pk}}(\tilde{p}_{ij} \tilde{f}_{jk}^{(t-1)}) \Big) \cdot \mathrm{Enc}_{\mathsf{pk}} \left(L^{t-1} \tilde{y}_{ik} \right) = \mathrm{Enc}_{\mathsf{pk}} \Big(\sum_{j \in N(i)} \tilde{p}_{ij} \tilde{f}_{jk}^{(t-1)} + L^{t-1} \tilde{y}_{ik} \Big)
\end{aligned} \tag{4}$$

Although eq. 4 cannot be decrypted by any nodes in the actual protocol, here we assume that a node could decrypt both terms of eq. 4. Then, we have

$$\tilde{f}_{ik}^{(t)} \leftarrow \sum_{j \in N(i)} \tilde{p}_{ij} \tilde{f}_{jk}^{(t-1)} + L^{t-1} \tilde{y}_{ik} = L\Big(\sum_{j \in N(i)} \alpha p_{ij} \tilde{f}_{jk}^{(t-1)} + L^{t-1}(1-\alpha) y_{ik} \Big). \tag{5}$$

We prove $\tilde{f}_{ik}^{(t)} = L^t f_{ik}^{(t)}$ holds by the inductive method. When $t = 1$, $\tilde{f}_{ik}^{(1)} = L f_{ik}^{(1)}$ obviously holds. Assuming $\tilde{f}_{ik}^{(u)} = L^u f_{ik}^{(u)}$ for any $u \in \mathbb{Z}$, $\tilde{f}_{ik}^{(u+1)} = L^{u+1} f_{ik}^{(u+1)}$ is readily derived using eq. 5 and the assumption. Thus, $\tilde{f}_{ik}^{(t)} = L^t f_{ik}^{(t)}$ holds. Consequently, $\tilde{\mathbf{F}}^{(t)}/L^t = \mathbf{F}^{(t)}$ holds and the lemma is proved by Lemma 1.

Privacy-preserving label propagation

- Public input: α and $L \in \mathbb{Z}_N$ s.t $\alpha L p_{ij} \in \mathbb{Z}_N$ and $(1-\alpha)Ly_{ik} \in \mathbb{Z}_N$ for all i, j, k.
- Private input of node i: link weights $\mathbf{w_{i*}}$ and label vector $\mathbf{y_{i*}}$ (W and Y are row private).
- Key setup: All nodes share public key pk; node i holds secret key sk^i for threshold decryption.

1. (Initialization) For all $j \in N(i)$, node i computes:
   (a) $p_{ij} \leftarrow w_{ij} / \sum_{j \in N(i)} w_{ij}$, $\tilde{p}_{ij} \leftarrow \alpha L p_{ij}$,
   (b) For $k = 1, ..., c$, $\tilde{f}_{ik}^{(0)} \leftarrow y_{ik}$, $\tilde{y}_{ik} \leftarrow (1-\alpha)Ly_{ik}$,
   (c) $c_{ik}^{(0)} \leftarrow \mathrm{Enc}_{\mathsf{pk}}(\tilde{f}_{ik}^{(0)})$ for all $k \in \{1, ..., h\}$ and $t \leftarrow 1$.
2. (Iteration) For all $k \in \{1, ..., h\}$:
   (a) Node i sends $c_{ik}^{(t-1)}$ to all $j \in N(i)$,
   (b) Node i receives $c_{jk}^{(t-1)}$ from all $j \in N(i)$ and updates $c_{ik}^{(t-1)}$ by eq. 3,
   (c) All nodes jointly perform the convergence detection and normalization using SFE if needed. If convergence is detected, go to step 3. Else, $t \leftarrow t+1$ and go to step 2 (a).
3. (Decryption) Node i and arbitrary $(\theta - 1)$ nodes jointly perform recovery scheme and output $\mathbf{f}_{i*}^* = (f_{i1}^*, \ldots, f_{ih}^*)$.

Fig. 2. Privacy-preserving label propagation

In the protocol, two computations that are not executable on homomorphic encryption are not mentioned. One is convergence detection [11]. From Lemma 2, the convergent value $\mathbf{F}^*$ is identical to that of label prediction. The convergence of eq. 3 has to be determined by having $\left|\tilde{f}_{ik}^{(t)}/L^t - \tilde{f}_{ik}^{(t-1)}/L^{t-1}\right| < \epsilon$ for all i and k in Step 2 (c). The other is normalization [11]. As shown in Lemma 2, $\tilde{f}_{ik}^{(t)}$ is multiplied by L at each iteration (although no node can observe this). Since the addition of homomorphic encryption is modulo N arithmetic, $\tilde{f}_{ik}^{(t)}$ has to be normalized so as not to exceed N after addition. The normalization does not have to be invoked so often since typically parameter is set to a big integer, such as $N = 2^{1024}$. To realize this normalization, we use *private division* to divide encryption without spilling the value of $\tilde{f}_{ik}^{(t)}$.

Values exchanged in the protocol are all encrypted, while SFE requires unencrypted values as inputs. To securely evaluate normalization and convergence detection with encrypted values, we used the following scheme:

1. Encrypted values are randomly partitioned by means of the homomorphic property of encryption so that these form random shares if decrypted,
2. Shares are independently decrypted and are taken as inputs of SFE,
3. SFE for normalization or convergence detection is executed with recovering the values in execution of SFE.

Note that the computation time of SFE is usually large; however, both computations are not necessarily required at every update, the cost per iteration can be small, which will not create a bottleneck. We discuss this further in Sect. 6.

5.2 Security of the protocol

The security of privacy preserving label prediction is proved as follows:

Lemma 3. *Let there be a label-unaware undirected PWG whose nodes behave semi-honestly*[2]*. Assuming that more than θ nodes do not collude, node i learns the ith row of $\mathbf{F}^*$ and the number of iterations but nothing else, after the execution of privacy preserving label prediction. Furthermore, the graph is kept as the label-unaware undirected PWG.*

The proof should show the indistinguishability of the simulation view and nodes' views as in [6]. However, due to space limitations, we will explain the security of our protocol in an intuitive way. All of the messages exchanged among nodes throughout the execution of the PPLP protocol are:

1. Step 2(a): $c_{jk}^{(t)} = \mathrm{Enc}_{\mathsf{pk}}(\tilde{f}_{jk}^{(t)})$
2. Step 2(c): random shares of $\tilde{f}_{jk}^{(t)}$ for inputs of SFE
3. Step 2(c) and 3: decryption shares of $\mathrm{Enc}_{\mathsf{pk}}(\tilde{f}_{jk}^{*})$ for $j = 1, \ldots, n, k = 1, \ldots, h$.

If nothing can be learned from these messages, other than the output, we can conclude that the protocol is privacy preserving. The messages exchanged in Step 2(a) are all encrypted; these cannot be decrypted without the collusion of more than θ parties. Random shares of Step 2(c) do not leak anything obviously. In Step 3 or Step 2(c), node i receives $(\theta - 1)$ decryption shares of $\mathrm{Enc}_{\mathsf{pk}}(\tilde{f}_{jk}^{*})$. If $i = j$, then node i can recover $\mathrm{Enc}_{\mathsf{pk}}(\tilde{f}_{jk}^{*})$, but this is the final output of node i itself. If $i \neq j$, then $\mathrm{Enc}_{\mathsf{pk}}(\tilde{f}_{jk}^{*})$ cannot be decrypted by node i, unless more than θ nodes collude.

Based on the above discussion, node i can learn nothing but its final output and the number of iterations throughout the execution of the PPLP protocol. From Lemma 2 and Lemma 3, the following theorem is readily proved.

Theorem 1. *Assuming all nodes behave semi-honestly and there is no collusion of more than θ nodes, PPLP executes label propagation in label-unaware undirected PWG, in the sense of Statement 1.*

5.3 Output Privacy of Label Propagation

Our protocol enables computation of label prediction without sharing links and labels while the output of our protocol might leak information about links or

[2] This assumes that the nodes follow their protocol properly, but might also use their records of intermediate computations in order to attempt to learn other nodes' private information.

labels of neighboring nodes, especially when the neighboring nodes have only small a number of links. Differential privacy provides a theoretical privacy definition in terms of outputs. Output perturbation using Laplace mechanism can guarantee the differential privacy for a specified function under some conditions.

Each node can guess only a limited amount of information if such a mechanism is applied to the output, even when each node has strong background knowledge about the private information. The security of the computation and secure disclosure of outputs are mutually independent in label prediction. Therefore, output perturbation can be readily combined with our protocol while the design of perturbation for label propagation is not straightforward. We do not pursue this topic in this paper any further and this problem is remained for our future works.

5.4 Expansion to Directed Graphs

Let us expand PPLP protocol so that it can be applied to directed graphs. In the label-unaware directed PWGs, the labels are propagated only to the node it links to. Then, the following update is used, instead of eq. 3.

$$\mathrm{Enc}_{\mathsf{pk}}(\tilde{f}_{ik}^{(t)}) \leftarrow \prod_{j \in N_{\mathrm{out}}(i)} \left((c_{jk}^{(t-1)})^{\tilde{p}_{ij}} \right) \cdot \mathrm{Enc}_{\mathsf{pk}} \left(L^{t-1} \tilde{y}_{ik} \right).$$

In order to update the above, note that: (1) node i has to send $c_{ik}^{(t-1)}$ to $N_{\mathrm{out}}(i)$ in Step 2 (a), and (2) node i has to receive $c_{jk}^{(t-1)}$ from $j \in N_{in}(j)$ in Step 2 (b). In the directed graph, since $\mathbf{W}$ is row-private, node i can know to whom it is linking to ($N_{\mathrm{out}}(i)$), but cannot know from whom it is linked to ($N_{\mathrm{in}}(i)$). Each node in $N_{\mathrm{in}}(i)$ can send a request for connection to node i. However, the request itself violates the given privacy model. Thus, in directed graphs, the problem is not only in the secrecy of messages but also in the anonymity of connections. For this, we make use of the onion routing [7], which provides anonymous connection over a public network. By replacing every message passing that occurs in the protocol with the onion routing, we obtain the PPLP protocol for label-unaware directed PWGs (PPLP-D). Techniques to convert the protcool in the label-aware model to that in the label-unaware model can be found in [10], too.

6 Experimental Analysis

We compared the label prediction methods in terms of the accuracy, privacy loss, and computation cost.

Datasets. Two labeled graphs were taken from the real-world examples. Romantic Network (ROMN) is a network in which students and their sexual contacts correspond to the nodes and links, respectively [1]. We used the largest component (288 nodes) of the original network. 5 randomly selected nodes and the

nodes within 5 steps of them were labeled as "infected" (total 80 nodes); other nodes were labeled as "not-infected" (208 nodes)[3]. ROMN is undirected; the weight matrix $w_{ij} = 1$ if there is a link between i and j, and $w_{ij} = 0$ otherwise.

MITN is a network in which the mobile phone users, their physical proximities, and their affiliations correspond to the nodes, links, and node labels, respectively. The proximity is measured by Bluetooth devices of the mobile phones, in MIT Reality Mining Project [5]. 43 nodes are labeled as "Media Lab" and 24 nodes are labeled as "Sloan" (total 67 nodes). MITN is a directed graph; the weight w_{ij} is set as the time length of user i detecting user j.

Settings. Three types of label propagation were implemented, decentralized label propagation (LP), label propagation implemented by SFE (LP/SFE), and our proposal, PPLP and PPLP-D. For comparison, kNN was also tested. In PPLP and PPLP-D, we set the parameters as $L = 10^3$ and $\alpha = 0.05$ and normalization was performed once per 100 updates of eq. 3. For kNN, we set $k = 1, 3, 5$. Results were averaged over 10 trials.

In Sect. 6.1, we evaluated trade-off between prediction accuracy and privacy loss. In Section 6.2, we evaluated computational efficiency of PPLP, PPLP-D, and LP/SFE with complexity analysis. The generalized Paillier cryptosystem [2] with 1024-bit keys was used in PPLP and PPLP-D. For SFE implementation, FairPlay [9] was used. Experiments were performed using Linux with 2.80GHz (CPU), 2GB (RAM).

We empirically compared computational cost of our solutions because complexity analysis of computations implemented by SFE is difficult. Furthermore, implementation details are as follows. In PPLP and PPLP-D, all computational procedure including normalization using SFE is implemented. In a computational cost of an update, one percent of normalization cost is accounted because normalization is executed once per 100 times. If LP/SFE is implemented in a naive manner, the computational time can be so large that computational analysis is unrealistic. Instead, we decomposed LP/SFE into single summation of the decentralized label propagation (eq. 2) and implemented each summation using SFE. This implementation allows the nodes to leak elements of the label matrix computed in the middle of label propagation, while the required computation time can be largely reduced. We regarded the computation time of this relaxed LP/SFE as the lower bound of the computation time of LP/SFE in experiments.

6.1 Privacy-Accuracy Trade-Off

As discussed, PPLP securely works in the label-unaware model. In other words, PPLP does not require observation of the labels of the neighbor nodes $N(i)$ for label prediction. On the other hand, both kNN and LP require the labels of neighbor nodes $N(i)$ for label prediction. Although it is impossible to run kNN and LP in label-unaware model, these can be executed if not all but some of the nodes in $N(i)$ dare to disclose their labels to node i. Considering this, we also

[3] Although this setting might not appropriate in terms of epidemiology, we employed this setting in order to examine the efficiency and practicality of our solution.

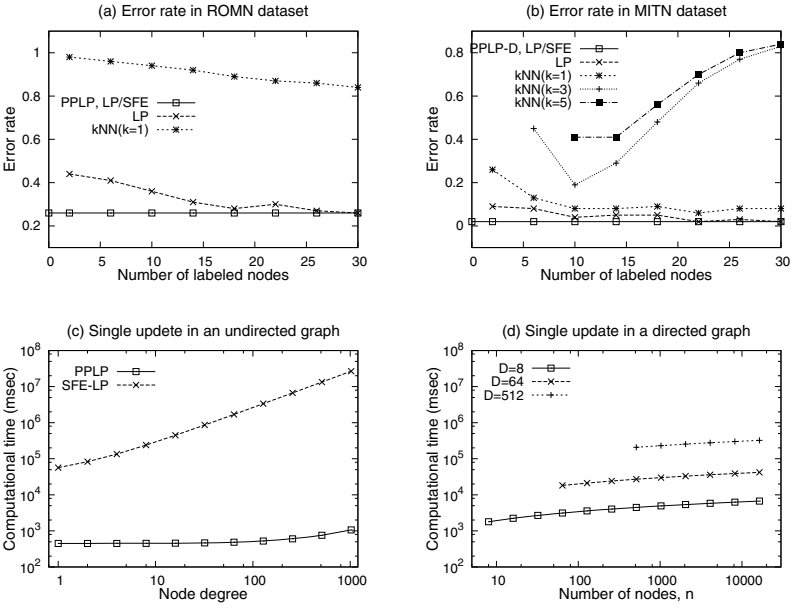

Fig. 3. Accuracies and computational costs of PPLP and PPLP-D vs. other methods. (a) and (b) are the accuracy changes with respect to the number of label disclosure ℓ. (c) shows the scalability with respect to the maximum number of links per node Δ. (d) is the scalability of PPLP-D with respect to Δ and network size n.

tested intermediate models between the label-aware and label-unaware models. Let ℓ_{ini} be the number of nodes initially labeled. In these intermediate models, ℓ ($0 \leq \ell \leq \ell_{\text{ini}}$) nodes disclose their labels to neighbor nodes. If $\ell = \ell_{\text{ini}}$, it corresponds to the label-aware model because all initially labeled nodes disclose their labels. If $\ell = 0$, it corresponds to the label-unaware model because all nodes keep there labels secret.

Fig. 3 (a) and (b) show the change of the prediction errors in ROMN and MITN with respect to ℓ, respectively. kNN and LP require larger ℓ for better prediction, and cannot be executed when ℓ=0. kNN shows the higher error rate as compared to the other methods for all k. kNN cannot predict labels for nodes that connected to nodes which disclose their labels. Since we treated this as misclassification, the error rate of kNN can be large. The results of 3NN and 5NN are not shown in Fig. 3 (a) because they cannot predict labels for most of nodes. Error rates of 3NN and 5NN in Fig. 3 (b) increase with ℓ when ℓ is large. The reason for these results is considered as follows. Large k accounts for more distant data points than small k. Therefore kNN of large k may not perform well with unbalanced data.

Thus, if label information could be partially disclosed and its running time is crucial, LP could be a good choice. In contrast, when privacy preservation of

Table 3. The computation time until convergence (10 itr.) in ROMN ($\Delta = 9$) and disclosed information

Method	disclosed info.	comp. time
kNN	y_{jk} $(j \in N(i))$	7.4×10^{-5}(ms)
LP	$f_{jk}^{(t)}$ $(j \in N(i))$	3.0×10^{-4}(ms)
LP/SFE (relaxed)	$p_{ij} f_{jk}^{(t)}$ $(j \in N(i))$	88.4 (min.)
LP/SFE (strict)	none	>88.4 (min.)
PPLP	none	10.7 (sec.)

labels and links is essential, PPLP or LP/SFE could be a good choice, because the accuracies of PPLP and LP/SFE which observe no labels are always equivalent to that of LP which is allowed to observe all labels.

6.2 Computational Efficiency

Complexity and Scalability. We evaluated the computation times of LP/SFE, PPLP, and PPLP-D with artificially generated networks where the maximum number of links is Δ. In PPLP, the computational complexity of a single update of eq. 3 is $O(\Delta)$. In PPLP-D, the computational cost of onion routing $O(\log n)$ is additionally required (in total $O(\Delta + \log n)$).

Fig. 3 (c) shows the change of the computational times for a 2 class classification of artificially generated labeled graphs, with respect to Δ. We evaluated PPLP and PPLP-D including normalization and convergence detection. The results show that the computational cost of LP/SFE is large; at least a few hours are required for a single update when $\Delta = 2^{10}$. On the other hand, the computation time of PPLP is a few seconds even when $\Delta = 2^{10}$. Fig. 3 (d) shows the change of the computational times of PPLP-D with respect to network size n. The single update of PPLP-D needs a few minutes when $\Delta = 2^3$, but a few hours when the $\Delta = 2^9$. This fact indicates that PPLP-D can work efficiently even in large-scale networks if networks are sparse. Computational cost of LP/SFE in directed graphs become larger than that in undirected graphs because of onion routing. In addition, onion routing for LP/SFE have to be executed by SFE. The computational cost of this will become unrealistically large, so LP/SFE for directed graphs is not evaluated. We can see that computational cost of PPLP-D is even smaller than that of LP/SFE for undirected graph in Fig.3 (c) and (d).

Completion Time in the Real World Network Data. Table 3 summarizes the computational costs and the information disclosure until convergence of label prediction methods in ROMN. In the experiments, all labels of initially labeled 30 nodes are disclosed to other nodes for execution of LP and kNN. The results showed that PPLP achieved both efficient computations and privacy preservation, while the others did not. Recall that computational cost of PPLP does not increase drastically with the maximum number of links as shown in Fig 3 (a). In addition, the maximum number of links is not often very large, typically in the networks containing private information, such as sexual contacts. From above, we can conclude that PPLP is practical even for large-scale networks.

7 Conclusion

In this paper, we introduced novel privacy models for labeled graphs, and then stated secure label prediction problems with these models. We proposed solutions for secure label prediction, PPLP and PPLP-D, which allow us to execute label prediction without sharing private links and node labels. Our methods are scalable compared to existing privacy preserving methods. We experimentally showed that our protocol completed the label prediction of a graph with 288 nodes in about 10 seconds. In undirected graphs, the complexity is proportional to the maximum number of links, rather than the network size. We can conclude that our protocol achieves both privacy preservation and scalability, even in large-scale networks. Scalability in directed graphs is relatively larger than that in undirected graphs. Our future work will involve the implementation of our proposal in real social problems.

References

1. Bearman, P., Moody, J., Stovel, K.: Chains of affection: The structure of adolescent romantic and sexual networks. American J. of Sociology 110(1), 44–91 (2004)
2. Dåmgard, I., Jurik, M.: A Generalisation, a Simplification and Some Applications of Paillier's Probabilistic Public-Key System. In: Kim, K.-c. (ed.) PKC 2001. LNCS, vol. 1992, pp. 119–136. Springer, Heidelberg (2001)
3. Damgård, I.B., Koprowski, M.: Practical threshold RSA signatures without a trusted dealer. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 152–165. Springer, Heidelberg (2001)
4. Duan, Y., Wang, J., Kam, M., Canny, J.: Privacy preserving link analysis on dynamic weighted graph. Comp. & Math. Organization Theory 11(2), 141–159 (2005)
5. Eagle, N., Pentland, A., Lazer, D.: Inferring social network structure using mobile phone data. In: PNAS (2007)
6. Goldreich, O.: Foundations of cryptography: Basic applications. Cambridge University Press, Cambridge (2004)
7. Goldschlag, D., Reed, M., Syverson, P.: Onion routing. Communications of the ACM 42(2), 39–41 (1999)
8. Joachims, T.: Transductive inference for text classification using support vector machines. In: Proc. ICML (1999)
9. Malkhi, D., Nisan, N., Pinkas, B., Sella, Y.: Fairplay: secure two-party computation system. In: Proc. of the 13th USENIX Security Symposium, pp. 287–302 (2004)
10. Sakuma, J., Kobayashi, S.: Link analysis for private weighted graphs. In: Proceedings of the 32nd International ACM SIGIR, pp. 235–242. ACM, New York (2009)
11. Sakuma, J., Kobayashi, S., Wright, R.: Privacy-preserving reinforcement learning. In: Proceedings of the 25th International Conference on Machine Learning, pp. 864–871. ACM, New York (2008)
12. Weston, J., Leslie, C., Ie, E., Zhou, D., Elisseeff, A., Noble, W.: Semi-supervised protein classification using cluster kernels. Bioinformatics 21(15), 3241–3247 (2005)
13. Yao, A.: How to generate and exchange secrets. In: Proc. of the 27th IEEE Annual Symposium on Foundations of Computer Science, pp. 162–167 (1986)
14. Zhou, D., Bousquet, O., Lal, T., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference, pp. 595–602 (2004)
15. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: ICML (2003)