Group Sparsity Tensor Factorization for Re-identification of Open Mobility Traces

Takao Murakami, Atsunori Kanemura, and Hideitsu Hino Members, IEEE

Abstract—Re-identification attacks based on a Markov chain model have been widely studied to understand how anonymized traces are linked to users. This approach is known to enable users to be re-identified with high accuracy when an adversary trains a personalized transition matrix for each target user using a large amount of training data, and when all of the anonymized traces are from the target users. In reality, however, the amount of training data for each target user can be very small, since many users disclose only a small amount of their location information to the public. In addition, many of the anonymized traces are from “non-target” users, whose personalized transition matrices cannot be trained in advance.

This paper aims to quantify the risk of re-identification in the realistic situation explained above. We firstly utilize the fact that spatial data can form a group structure, and propose group sparsity tensor factorization to effectively train the personalized transition matrices from a small number of training traces. We secondly formulate a re-identification attack in an “open” scenario, where many of the anonymized traces are from non-target users. Specifically, we regard this type of attack as a biometric verification (or identification) task, and propose a framework and an algorithm for performing this task using a population transition matrix, which is computed from personalized transition matrices. Our experimental results using three real datasets show that a training method using tensor factorization significantly outperforms the ML (Maximum Likelihood) estimation method, and is further improved by incorporating group sparsity regularization.

Index Terms—location privacy, re-identification, biometrics, tensor factorization, group sparsity

I. INTRODUCTION

As a consequence of the increasing use of LBSs (Location-based Services) such as location check-in, route navigation, and POI (Point-of-Interest) search, a great number of mobility traces (time-series location trails) are collected from GPS-equipped devices (e.g., smart phones and in-car navigation systems) and accumulate in a data center. These data are termed “Spatial Big Data” [2], and are expected to provide useful information to users, industry, and society. For example, they can be provided to a third-party for analysis (e.g., mining fuel-efficient routes [2] or commonly frequented public areas [3]), or can be made public to provide traffic information to users [4].

The disclosure of mobility traces, however, can lead to a breach of users’ privacy (which is known as location privacy [5]–[7]). For example, it can expose users’ sensitive locations such as homes and hospitals, and other private information such as users’ demographic and employment details (e.g., age, work role) [8], [9], activities (e.g., sleeping, shopping) [8], [10], and social relationship(s) [11], [12]. There is also a risk that robbers [13] or stalkers [14] exploit location information of victims. Therefore, those who collect mobility traces (e.g., LBS providers) generally replace users’ identifiers (e.g., user names, addresses) with pseudonyms (or simply “nym”) before they make the mobility traces available to a third-party (or make them public). However, a number of studies [15]–[21] have shown that anonymized traces can be linked to users by using past mobility traces of the users as training data (i.e., anonymized traces can act as quasi-identifiers [22]). An attack based on this approach is known as a re-identification attack (or a de-anonymization attack), and is the focus of this paper.

Re-identification attacks based on a Markov chain model [18]–[21] have been widely studied and reported in the literature. Fig. 1 shows a framework for a re-identification attack based on this model. In this model, an adversary divides an area of interest into some regions (or POIs) $x_1, \ldots, x_M$ ($M$ regions in total), and creates fixed-interval time partitions (e.g., 30 minutes, 2 hours). The adversary then assumes the Markov property for the movement of target users $u_1, \ldots, u_N$ ($N$ users in total), and trains, for each target user $u_n$, a personalized transition matrix $P_n (M \times M)$ matrix) that comprises the probability $p_{nij}$ of moving from region $x_i$ to $x_j$. After obtaining anonymized traces (which may be obfuscated by, for example, adding noise or deleting some locations), the adversary attempts to link them to the target users using personalized transition matrices $P_1, \ldots, P_N$. Some studies [18]–[20] showed that this model is very effective when a large amount of training data are available to the adversary, and when all anonymized traces are from the target users (i.e., when the adversary performs an $N$-class classification task for each anonymized trace).

In reality, however, this model presents two issues that have to be addressed to enable it to be applied to a re-identification attack. First, a user generally discloses only a small amount of location information to the public (e.g., via geosocial networking applications) in his/her daily life (e.g., a few times per day), and therefore training data can be extremely sparse. Although previous studies [18]–[20] trained personalized transition matrices using the ML (Maximum Likelihood)
estimation method\textsuperscript{1}, training data can be extremely sparse in these cases. We explain this problem using the example shown in Fig. 1, where the ML estimation method is used to train personalized transition matrices. In this example, since a transition pattern from region $x_1$ or $x_5$ is not observed in the training traces of user $u_1$, the transition probabilities from region $x_1$ or $x_5$ cannot be estimated using the ML estimation method. We refer to the corresponding elements (marked with "?" in Fig. 1) as unobserved elements. Moreover, the remaining elements are overfitted to the training traces (e.g., the transition probabilities from $x_2$ are 0 or 1), which generally results in poor predictive performance \cite{24}. We refer to these elements as overfitted elements. The existence of many unobserved or overfitted elements would prevent an adversary from accurately re-identifying mobility traces.

Second, although all of the studies in \cite{18, 19, 20, 21} assumed that the set of target users (i.e., $u_1, \ldots, u_N$) coincides with the set of users who have provided anonymized traces, they do not coincide in general. We refer to the former and latter cases as a closed scenario and an open scenario, respectively (similarly, we refer to mobility traces of the former and latter types as closed mobility traces and open mobility traces, respectively). In reality (i.e., an open scenario), many of the anonymized traces can originate from “non-target” users (i.e., users other than $u_1, \ldots, u_N$), whose personalized transition matrices cannot be trained in advance. Furthermore, it is not always true that an adversary perfectly knows who, among the target users $u_1, \ldots, u_N$, have provided anonymized traces. For example, suppose that target users $u_1, \ldots, u_{10}$ used a certain LBS, and the LBS provider published their traces (along with traces from non-target users) after anonymization. Since user $u_i$ disclosed the fact that he/she used the above LBS via SNS, the adversary is confident that traces of user $u_i$ are included among the anonymized traces. However, the adversary is not sure about users $u_2, \ldots, u_{10}$, since they did not disclose the fact that they used the above LBS. As can be seen from this example, the number of target users who, to the adversary’s knowledge, have provided anonymized traces can be very small in reality (in the above example, the number of such users is only one).

\textbf{A. Our Aim and Approaches}

The aim of this paper is to quantify the risk of re-identification in the realistic situation explained above. To the best of our knowledge, this is the first study to address the two above-mentioned issues (i.e., the sparse data problem, an open scenario) in a re-identification attack based on the Markov chain model (see Section II, which describes related work, for more details). Regarding the first issue, we take two steps towards solving the sparse data problem: \textit{tensor factorization} \cite{25, 26, 27} and \textit{group sparsity regularization} (also known as \textit{group lasso}) \cite{28, 29}. In the following, we explain these steps using Fig. 2.

We first focus on a training method of personalized transition matrices using tensor factorization, which was proposed in \cite{23}. This training method assumes a set of personalized transition matrices as a third-order tensor that is composed of three modes: the “User,” “From Region,” and “To Region” modes. This tensor is termed a transition probability tensor. Then it uses tensor factorization \cite{25, 26, 27}, which decomposes the third-order tensor into low-rank matrices, to accurately train the transition probability tensor from a small number of training traces (see the left side of Fig. 2). Murakami \textit{et al.} \cite{23} applied this training method to a localization attack \cite{21}, which specifies the location of a target user at a certain time based on obfuscated locations disclosed by him/her, and showed that it outperformed the ML estimation method. However, since they did not apply it to the re-identification attack, it remains unclear whether this training method is also effective in this attack. In this paper, we apply this training method to the re-identification attack, and show that the method is indeed effective.

We then utilize the fact that spatial data can form group structure. We explain this by using the diagram on the right-hand side of Fig. 2, in which regions are divided into three groups: $R_1$, $R_2$, and $R_3$ (an edge between two regions represents a well-traveled route, through which many people

\textsuperscript{1}Shokri \textit{et al.} \cite{21} proposed a training method that considers a case where some locations in training traces are missing. However, this method provides almost the same performance as the ML estimation method for training traces from which no locations are missing, as verified in \cite{23}.
pass within one time instant). For example, the transition probability between two close regions (i.e., within the same group) is generally higher than the transition probability between two distant regions (i.e., across two groups). Many people tend to visit urban areas (e.g., $R_1$, $R_2$), but Alice always stays in a rural area (e.g., $R_3$). We refer to a group structure such as this as a \textit{spatial group structure}. In the field of machine learning, Yuan and Lin [28] proposed group sparsity regularization to introduce a group structure as prior knowledge. In addition, Kim \textit{et al.} [29] proposed group sparsity regularization for matrix factorization. The above-mentioned spatial group structure can also be captured using group sparsity regularization (we discuss this in Section V-A in details). No studies, however, have incorporated group sparsity regularization into tensor factorization, to the best of our knowledge. Thus, we propose \textit{group sparsity tensor factorization}, which extends group sparsity matrix factorization proposed by Kim \textit{et al.} [29] to a third-order tensor, and incorporate it into the training method in [23].

Regarding the second issue, we formulate a re-identification attack in an open scenario. Specifically, let $u_1, \ldots, u_{N'}$ ($1 \leq N' \leq N$) be target users who, as the adversary knows with certainty, have provided anonymized traces (we can choose $u_1, \ldots, u_{N'}$ from $u_1, \ldots, u_N$ without loss of generality). The adversary determines, for each anonymized trace, whether it is generated from $u_1, \ldots, u_{N'}$, or from someone else (i.e., $N'+1$ class classification). $N'$ can be very small (e.g., $N' = 1$ or $2$) as explained above. In addition, a user is generally concerned about the risk that \textit{(not all traces but) his/her own traces are re-identified by the adversary.}

Thus, we first focus on the most basic case where $N' = 1$ (i.e., \textit{two-class classification}). We note here that a mobility trace could be regarded as reflecting the behavioral characteristics of human beings (examples of the behavioral biometric characteristics include gait, signature, and keystroke dynamics [30]). Taking this into account, we regard a re-identification attack in this case as a \textit{biometric verification task} [30], in which a system decides whether a user is genuine or not using his/her biometric information. We propose a framework and an algorithm for performing this task using a \textit{population transition matrix} (Section III).

1) We propose a training method that incorporates group sparsity regularization into the training method in [23] as follows:
   a) Divide regions into some groups from training traces using the Markov Cluster (MCL) algorithm [35] (Section V-B);
   b) Train personalized transition matrices using group sparsity tensor factorization. We note again that this is the first study to propose group sparsity tensor factorization, to our knowledge (Section V-C).

2) We regard a re-identification attack in an open scenario as a biometric verification (or identification) task, and propose a framework and an algorithm for this task using a population transition matrix (Section III).

3) We apply the ML estimation method [18]–[20] (denoted by \textit{ML}), \textit{TF}, and \textit{GSTF} to a re-identification attack in an open scenario, and conduct experiments using the three real datasets mentioned above. We first show that, although \textit{ML} does not pose a threat when the amount of training data is limited, \textit{TF} significantly outperforms the ML estimation method. We then show that \textit{GSTF} outperforms \textit{TF} (Sections VI and VII).

This paper is a significant extension of a previously published conference paper [1]. Note that, whereas the first contribution above is covered in [1], the second contribution is new (i.e., [1] considered only a closed scenario) and the third contribution is significantly extended.

With respect to the third contribution, [1] conducted experiments only in a closed scenario (which does not reflect the reality). In addition, [1] conducted experiments using only a single dataset (the Geolife dataset), which does not provide sufficient evidence for the effectiveness of \textit{GSTF}. Furthermore, [1] used the same number of traces per each user (10 traces per each user), and the same number of locations per each trace (10 locations per each trace). In this paper, we refer to this setting as an \textit{unbiased setting}, since the accuracy is not biased toward a specific user or trace in this case. In reality, however, the number of traces is different from user to user, and the number of locations is different from trace to trace. We refer to a setting such as this as a \textit{biased setting}. Experimental evaluation in a biased setting is also important, since it reflects the reality more closely. In addition, the size of actual traces is often very small (e.g., 2 to 5 locations, as shown in Section VII). Taking the above considerations into account, this paper provides the following extensions:

- We conduct experiments in an open scenario, and show the effectiveness of \textit{GSTF} in this scenario.
- We conduct experiments using three real datasets, and show the effectiveness of \textit{GSTF} in all the three datasets.
- We conduct experiments in both an unbiased setting and a biased setting, and show that \textit{GSTF} is a threat even when the size of anonymized traces is very small (e.g., 2 to 5 locations).
C. Paper Organization

The remainder of this paper is organized as follows. In Section I-D, we describe the basic notations used throughout this paper. In Section II, we describe related work in details. In Section III, we propose a framework and an algorithm for a re-identification attack in an open scenario. In Section IV, we explain the training method in [23]. In Section V, we propose a training method that incorporates group sparsity regularization into the training method in [23]. In Section VI, we present the experimental results in an unbiased setting. In Section VII, we present the experimental results in a biased setting. Finally, in Section VIII, we conclude the paper.

D. Notations

We denote the sets of real numbers and integers by \( \mathbb{R} \) and \( \mathbb{Z} \), respectively. Let \( [n] \) (\( n \): natural number) be the set of natural numbers less than or equal to \( n \) (i.e., \( [n] = \{1, 2, \ldots, n\} \)). Let \( X = \{u_i|n \in [N]\} \) be a set of target users, and \( \mathcal{X} = \{x_i|i \in [M]\} \) be a set of 4 robots (or POIs). We assume that time is discrete, and express time instants as integers (i.e., the set of time instants is \( \mathbb{Z} \)). Further, let \( P_n \) be a personalized transition matrix (\( M \times M \) matrix) of user \( u_n \), and \( p_{n,i,j} \) be its \((i, j)\)-th element (i.e., the probability that \( u_n \) moves from region \( x_i \) to \( x_j \)).

II. RELATED WORK

Research concerning location privacy has been widely reported in the literature (surveys can be found in [5]–[7]), and the re-identification of mobility traces [15]–[21] is one of the most popular topics. For example, Golle and Partridge [15] showed that the majority of the U.S. working population can be uniquely identified from their home/workplace location pairs. Subsequently, Freudiger et al. [16] showed that a person’s home/workplace pair can be identified at a rate of 65% to 75% using about 20 LBS queries from his/her home or workplace. Srivatsa and Hicks [17] showed that anonymized traces can be re-identified by using a pattern of meetings between users in the anonymized traces (contact graph) and their relationship in a social network (social network graph). However, people who care (even a little) about their privacy may neither use an LBS from their home/workplace, nor with their friends in a social network, even when they use it continuously. Thus, we consider a more general scenario, in which anonymized traces may neither contain home/workplace locations, nor places where users meet their friends.

The Markov chain model is suitable for such a general scenario. Some studies showed the effectiveness of this model using a large amount of training data. Mulder et al. [18] identified 100 users at a success rate of 77% to 88% using mobility traces collected during a period of one month as training data. Gambus et al. [19] identified 59 users in the Geolife dataset [32] at a success rate of 45% using half of the dataset (over two years) as training data. Shokri et al. [20] showed that an adversary who is familiar with personalized transition matrices \( \{P_n|n \in [N]\} \) can identify users with higher accuracy than an adversary who only knows the prior probabilities \( \{\pi_{n,x_i}|x_i \in \mathcal{X}\} \).

However, since all of the above studies [18]–[20] train personalized transition matrices \( \{P_n|n \in [N]\} \) using the ML estimation method, they are adversely affected by the sparse data problem (as shown in our experiments in Sections VI and VII). Shokri et al. [21] proposed a training method that considers the case in which some locations in training traces are missing. Specifically, they assumed a Dirichlet prior for a vector \( \mathbf{p}_{n,1} = (p_{n,1,1}, \ldots, p_{n,1,M}) \), and trained a distribution of \( P_n \) using the Gibbs sampling method. However, as this method trains each personalized transition matrix \( P_n \) independently, it is also affected by the sparse data problem. More specifically, when none of the locations are missing, the mode of the estimated distribution of \( P_n \) is equivalent to the ML estimate, and therefore this training method provides almost the same performance as the ML estimation method (as verified in [23]). Moreover, the studies in [18]–[21] (and the previous conference paper [1]) considered only a closed scenario.

Thus, in this paper, we first propose a framework and an algorithm for a re-identification attack in an open scenario (see Section III). We then propose a training method that uses group sparsity tensor factorization (i.e., GSTF) to address the sparse data problem (see Section V). In our experiments, we first show that the training method in [23] (i.e., TF) significantly outperforms the ML estimation method (note that it was previously unclear whether TF is effective in the task of re-identification). We then show that GSTF outperforms TF (see Sections VI and VII).

III. RE-IDENTIFICATION ATTACK IN AN OPEN SCENARIO

In an open scenario, an adversary, who knows that users \( u_1, \ldots, u_N \) (\( 1 \leq N' \leq N \)) provided anonymized traces, determines, for each anonymized trace, whether it is generated from \( u_1, \ldots, u_N \), or from someone else (i.e., \( N' \times 1 \) class classification). In this section, we propose a framework and an algorithm for a re-identification attack in such a scenario.

We first propose a framework and an algorithm for a re-identification attack in a case where \( N' = 1 \) (i.e., two-class classification) (Sections III-A and III-B). We then extend the proposed framework and algorithm to the general case when \( 1 \leq N' \leq N \) (Section III-C). We finally show the relationship between the accuracy of two-class classification and that of \( N' + 1 \) class classification (\( 1 \leq N' \leq N \)) (Section III-D).

A. Framework

Suppose that an adversary obtains side information that user \( u_1 \) provided anonymized traces (via \( u_1 \)’s SNS, for example). We denote users other than \( u_1 \) commonly by \( u_0 \) (i.e., \( u_0 \) is either \( u_2, u_3, \ldots, u_N \), or a non-target user; in many cases, \( u_0 \) is a non-target user). This enables us to formulate the re-identification attack in this case as a two-class classification task, in which the adversary determines, for each anonymized trace, whether it is generated from \( u_1 \) or \( u_0 \) (another person).

We regard this two-class classification task as a biometric verification task [30], and propose a framework for performing this task using a population transition matrix \( P_0 = (M \times M) \) matrix, which models an average behavior of \( u_0 \).
Fig. 3. Framework for a re-identification attack in an open scenario. An adversary trains a population transition matrix \( P_0 \) using personalized transition matrices \( \{ P_n \}_{n \in [N]} \). The adversary determines, for each anonymized trace, whether it is generated from user \( u_1 \) or \( u_0 \) (another person) using \( P_1 \) and \( P_0 \) (i.e., two-class classification).

Fig. 3 shows the proposed framework for a re-identification attack in an open scenario. We first assume that anyone who obtains anonymized traces could be a malicious adversary. The adversary trains, for each target user \( u_n \in \mathcal{U} \), personalized transition matrix \( P_n \) using training traces. The adversary then trains the population transition matrix \( P_0 \) using \( \{ P_n \}_{n \in [N]} \) (we describe the procedure for training \( P_0 \) in Section III-B in detail). Our procedure for training traces is based on the following general assumptions: target users may disclose their location information via geo-social networking applications (e.g., location check-in, tagging, nearby friends [36]), and the adversary may obtain training traces of target users by observing them in person. However, users generally disclose only a small amount of location information in their daily lives, as described in Section I. In addition, it is difficult for the adversary to observe target users in person, unless he/she is in a close relationship with them. Thus, in reality, the number of training traces would be very small.

After obtaining the anonymized traces (which may be obfuscated), the adversary performs a re-identification attack. Here we assume that multiple anonymized traces, each of which is assigned to a different pseudonym, can exist per user (e.g., in Fig. 3, the first trace of user \( u_1 \) is assigned to “58239” and the second trace of user \( u_1 \) is assigned to “69315”). This is because the same user can use an LBS from different devices or on different days. Apart from this, the LBS provider may divide a long trace of the same user into multiple traces with different pseudonyms before provision (or publication) for privacy reasons (we also show that the accuracy of a re-identification attack can be reduced by dividing a long trace into multiple traces in Sections VI and VII). The adversary determines, for each anonymized trace, whether it is generated from \( u_1 \) or \( u_0 \) using \( P_1 \) and \( P_0 \) (i.e., two-class classification).

We regard this two-class classification task as biometric verification. Fig. 4 shows the proposed framework for performing this task, where the adversary uses two modules: a matcher and a decision module. Fig. 4 shows the proposed framework for two-class classification in an open scenario, where the adversary uses two modules: a matcher and a decision module. The matcher computes, for an anonymized trace, a score \( s \in \mathbb{R} \) that measures the similarity between the anonymized trace and a personalized transition matrix \( P_1 \) (note that we can regard \( P_1 \) as a biometric template of user \( u_1 \); \( P_0 \) is used as a background model). Then a decision module compares the score \( s \) with a threshold \( A \in \mathbb{R} \), and decides that the anonymized trace is from \( u_1 \) if \( s \geq A \), and from \( u_0 \) otherwise.

**B. Algorithm**

We propose a likelihood ratio-based re-identification attack as an algorithm for a re-identification attack in an open scenario. This attack is based on a likelihood-ratio test, which maximizes the true positive probability (= 1 — type I error probability) for a given false positive probability (type I error probability) (i.e., the Neyman-Pearson lemma [37]).

Consider an anonymized trace from time \( t_1 \) to \( T \) (which may be obfuscated). Let \( X' \) be a set of possible obfuscated regions. For example, when the LBS provider uses a location deletion (or location hiding) method [21], [38], [39], which deletes some locations from a trace, \( X' \) can be expressed as \( X' = X \cup \{ \emptyset \} \), where \( \emptyset \) represents that a location is deleted. Let \( o_t \in X' \) \((1 \leq t \leq T)\) be an obfuscated region at time \( t \) in the anonymized trace, and \( H_1 \) (resp. \( H_0 \)) be a hypothesis that the anonymized trace is generated from user \( u_1 \) (resp. \( u_0 \)).

In the likelihood ratio-based re-identification attack, the adversary computes a likelihood-ratio, given the anonymized trace \( o_1, \ldots, o_T \in X' \), as a score:

\[
s = \frac{P(o_1, \ldots, o_T | H_1)}{P(o_1, \ldots, o_T | H_0)}. \tag{1}\n\]

The likelihood \( P(o_1, \ldots, o_T | H_1) \) in (1) is computed using \( P_1 \). Specifically, when the anonymized trace is not obfuscated \((X' = X)\), \( P(o_1, \ldots, o_T | H_1) \) is simply computed as follows:

\[
P(o_1, \ldots, o_T | H_1) = P(o_1 | H_1) \prod_{t=2}^{T} P(o_t | o_{t-1}, H_1) \tag{2}\n\]

\[
= \pi_{o_1} \prod_{t=2}^{T} \pi_{o_t | o_{t-1},\cdot}. \tag{3}\n\]

The matcher can compute (not a similarity but) a distance between the anonymized trace and \( P_1 \) as a score \( s \). In this case, the decision module decides that the anonymized trace is from \( u_1 \) if \( s \leq A \), and from \( u_0 \) otherwise. Hereinafter, we assume that the score is defined as a similarity without loss of generality.
where \( \pi_{t_1, o_1} \) is the prior probability that user \( u_1 \) is in region \( o_1 \). For example, we can use stationary probabilities, which are computed from \( P_1 \), as \( \{ \pi_{t_1, x_i} | x_i \in X \} \). The likelihood \( P(n_1, \cdots, n_T|H_0) \) in (1) is computed using \( P_0 \) in the same way as \( P(o_1, \cdots, o_T|H_1) \). When the anonymized trace is obfuscated, \( P(o_1, \cdots, o_T|H_1) \) and \( P(o_1, \cdots, o_T|H_0) \) are computed using the Forward algorithm [21] (for details, see [21]).

After computing the likelihood-ratio \( s \) in (1) using \( P_1 \) and \( P_0 \), the adversary compares \( s \) with a threshold \( A \), and decides that the anonymized trace is from \( u_1 \) if \( s \geq A \), and from \( u_0 \) otherwise. Thereby, the true positive probability (i.e., the probability of classifying \( u_1 \) as \( u_1 \)) is maximized for a given false positive probability (i.e., the probability of classifying \( u_0 \) as \( u_1 \)) if the likelihood-ratio \( s \) is accurately estimated.

For training purposes, we propose to simply compute the population transition matrix \( P_0 \) as the average of personalized transition matrices \( P_2, \cdots, P_N \):

\[
P_0 = \frac{1}{N-1} \left( \sum_{n=2}^{N} P_n \right). \tag{4}
\]

The reason for this is that it would be natural to estimate \( P_0 \), which models the average behavior of users other than \( u_1 \) (i.e., \( u_2, \cdots, u_N \), and non-target users), as the average behavior of \( u_2, \cdots, u_N \). In other words, we estimate the true mean as the empirical mean of the samples from \( u_2, \cdots, u_N \) (which converges to the true mean when each user appears in anonymized traces with equal probability, and the number of target users \( N \) is very large).

### C. Generalization to \( N' + 1 \) Class Classification

Consider a more general case where the adversary knows that target users \( u_1, \cdots, u_{N'} \) \((1 \leq N' \leq N)\) provided anonymized traces. We commonly denote users other than \( u_1, \cdots, u_{N'} \) by \( u_{N'} \) (i.e., \( u_{N'} \) is either \( u_{N'+1}, \cdots, u_N \), or a non-target user; in many cases, \( u_{N'} \) is a non-target user). The adversary attempts to determine, for each anonymized trace, whether it is generated from \( u_1, \cdots, u_{N'} \), or \( u_{N'} \) (i.e., \( N' + 1 \) class classification). We regard this type of attack as a biometric identification task [30], in which a system outputs candidates whose biometric templates are similar to a biometric sample in the authentication phase by performing the 1-to-\( N' \) matching. We extend the proposed framework and algorithm (which are described in Sections III-A and III-B, respectively) to this general task by performing the two-class classification task (i.e., biometric verification) for each of the \( N' \) target users (i.e., \( u_1, \cdots, u_{N'} \)) independently.

Let \( s_n \in \mathbb{R} \) \((1 \leq n \leq N')\) be a score between an anonymized trace and a personalized transition matrix \( P_n \), and let \( d_n \in \{u_0, u_1\} \) (in this case, \( u_0 \) is a user other than \( u_1 \)) be a decision result obtained by comparing \( s_n \) to a threshold \( A \) (if \( s_n \geq A \), \( d_n = u_1 \); otherwise, \( d_n = u_0 \)). Note the difference between \( u_1 \) and \( u_0 \). \( u_1 \) is a user other than \( u_1 \), \( u_{N'} \) (i.e., \( u_{N'} \) is either someone in \( \{u_i|N'+1 \leq i \leq N\} \) or a non-target user), whereas \( u_0 \) is a user other than \( u_1 \) (i.e., \( u_0 \) is either someone in \( \{u_i|1 \leq i \leq N', i \neq n\} \) or a non-target user). Thus, if the decision module outputs \( d_n = u_0 \), the adversary would believe that the anonymized trace is not generated from \( u_n \) with confidence. However, the adversary is still not sure whether the anonymized trace is from someone in \( \{u_i|1 \leq i \leq N', i \neq n\} \) or from \( u_{N'} \).

When we use the likelihood ratio-based re-identification attack (proposed in Section III-B), \( s_n \) can be expressed as follows:

\[
s_n = \frac{P(o_1, \cdots, o_T|H_n)}{P(o_1, \cdots, o_T|H_0)}. \tag{5}
\]

The adversary determines whether the anonymized trace is from \( u_1, \cdots, u_{N'}, \) or \( u_{N'} \), using a set of scores \( \{s_n|n \in [N']\} \) or a set of decision results \( \{d_n|n \in [N']\} \). One way to achieve this would be to compare the largest score \( s_{\text{max}} = \max\{s_1, \cdots, s_{N'}\} \) to the threshold \( A \), and decide that the anonymized trace is from the corresponding user (i.e., user who achieved \( s_{\text{max}} \)) if \( s_{\text{max}} \geq A \), and from \( u_{N'} \) otherwise. Another way would be to output all target users \( u_n \) such that \( d_n = u_n \) (i.e., \( \bigcup_{n=1}^{N'} \{u_n|d_n = u_n\} \)) as candidates (if there are no such users, output \( u_{N'} \)). We refer to the former rule as a max-score rule, and the latter rule as a threshold rule (since it outputs all target users whose scores are more than or equal to the threshold \( A \)). Note that both of the two rules output \( u_{N'} \) if all of the scores are below the threshold \( A \), since the adversary is confident that the anonymized trace is generated from a user other than \( u_1, \cdots, u_{N'} \). In this case, Fig. 5 shows an example of the threshold rule (\( N' = 5 \)).

### D. Accuracy of \( N' + 1 \) Class Classification

We show the relationship between the accuracy of two-class classification and that of \( N' + 1 \) class classification \((1 \leq N' \leq N)\). In the latter case, we use the threshold rule (shown in Fig. 5), since it is easy to analyze.

In two-class classification, an adversary determines whether an anonymized trace is generated from user \( u_1 \) or \( u_0 \). Let \( \text{TPP}_1 \) and \( \text{FPP}_1 \) be the true positive probability (i.e., the probability of classifying \( u_1 \) as \( u_1 \)) and the false positive probability (i.e., the probability of classifying \( u_0 \) as \( u_1 \)), respectively. Here we assume that these probabilities are independent of users. In other words, we assume that the probability that a score \( s \) between the same user (resp. different users) is more than or equal to the threshold \( A \) is given by \( \text{TPP}_1 \) (resp. \( \text{FPP}_1 \)), irrespective of the user (resp. a pair of users).
In $N' + 1$ class classification using the threshold rule, an adversary outputs some of $u_1, \cdots, u_{N'}$ as candidates, or outputs $u_{\perp}$. Suppose that user $u_1$ provided an anonymized trace (note that we can choose $u_1$ from $u_1, \cdots, u_{N'}$ without loss of generality). We define a true positive probability in this case as a probability that the adversary chooses (one or more) candidates that include $u_1$. Similarly, suppose that user $u_{\perp}$ provided an anonymized trace. We define the false positive probability in this case as the probability that the adversary does not choose $u_{\perp}$ (i.e., the adversary chooses one or more candidates). Let $TPP_{N'}$ and $FPP_{N'}$ be the true positive probability and the false positive probability in the $N' + 1$ class classification, respectively. They include $TPP_1$ and $FPP_1$ defined above as a special case when $N' = 1$.

Since personalized transition matrices $P_1, \cdots, P_{N'}$ are independent of each other, it is reasonable to assume that scores $s_1, \cdots, s_{N'}$ are also independent of each other (independence of scores is often assumed in biometric identification [40]–[42]). Then, since the probability that a score between the same user (resp. different users) is more than or equal to the threshold $A$ is given by $TPP_1$ (resp. $FPP_1$), we can relate the accuracy of $N' + 1$ class classification to that of two-class classification as follows:

$$TPP_{N'} = TPP_1$$

$$FPP_{N'} = 1 - (1 - FPP_1)^{N'}.$$  \hspace{1cm} (6)

$$FPP_{N'} \approx 1 - (1 - N' \cdot FPP_1)$$

$$= N' \cdot FPP_1.$$  \hspace{1cm} (7)

$$FPP_{N'} \approx 1 - (1 - N' \cdot FPP_1)$$

$$= N' \cdot FPP_1.$$  \hspace{1cm} (8)

We can also relate the number of candidates in $N' + 1$ class classification to the accuracy of two-class classification (i.e., $TPP_1$, $FPP_1$). Suppose that user $u_1$ provided an anonymized trace, and the adversary uses the threshold rule. Let $L_{N'}$ be a variable representing the number of candidates in this case. Then, since the probability that a score between the same user (resp. different users) is more than or equal to the threshold $A$ is given by $TPP_1$ (resp. $FPP_1$), the expectation of the number of candidates $E[L_{N'}]$ can be expressed as

$$E[L_{N'}] = TPP_1 + (N' - 1) \cdot FPP_1.$$  \hspace{1cm} (9)

In Section VI, we show that equations (6), (9), and (10) hold for real datasets.

It can be seen from (9) and (10) that when we use the threshold rule, $FPP_{N'}$ and $L_{N'}$ increase with an increase in $N'$. If we use the max-score rule explained above, the number of candidates $L_{N'}$ is at most one. However, in this case, $TPP_{N'}$ decreases with an increase in $N'$ (since the probability of classifying user $u_n$ as $u_m$ ($1 \leq m \leq N', m \neq n$), under the condition that $u_n$ provided an anonymized trace, increases). That is, there is a trade-off between $L_{N'}$ and $TPP_{N'}$.

### IV. Training Transition Matrices Using Tensor Factorization (Training Method in [23])

Since our training method incorporates group sparsity regularization into the training method in [23] (i.e., TF), we review this training method in this section. As this training method uses tensor factorization, we first describe matrix and tensor factorization (Section IV-A). We then describe the training method in [23] (Section IV-B).

#### A. Matrix and Tensor Factorization

Matrix (resp. tensor) factorization [25], [43] decomposes a large matrix (resp. third-order tensor) that contains unobserved elements into low-rank matrices to enable the original matrix (resp. third-order tensor) to be estimated from a small amount of training data.

We begin by describing matrix factorization. Let $\mathbf{A} \in \mathbb{R}^{N \times M}$ be a large matrix (i.e., $N$ and $M$ are large). Matrix factorization approximates $\mathbf{A}$ using low-rank matrices $\mathbf{U} \in \mathbb{R}^{N \times K}$ and $\mathbf{V} \in \mathbb{R}^{M \times K}$ ($K \ll N, M$) as follows:

$$\hat{\mathbf{A}} = \mathbf{U} \mathbf{V}^T,$$  \hspace{1cm} (11)

where $\hat{\mathbf{A}}$ is an approximation of $\mathbf{A}$ (see the left panel of Fig. 6). Let $u_{ni} \in \mathbb{R}^{K}$ ($1 \leq n \leq N$) be the $n$-th row of $\mathbf{U}$, $v_{ij} \in \mathbb{R}^{K}$ ($1 \leq i \leq M$) be the $i$-th row of $\mathbf{V}$, $u_{nk}, v_{ik} \in \mathbb{R}$ be the $k$-th element of $u_{ni}$, and $v_{ij} \in \mathbb{R}$ be the $k$-th element of $v_{ij}$. Further, let $a_{ni}$ (resp. $\hat{a}_{ni}$) be the $(n, i)$-th element of $\mathbf{A}$ (resp. $\hat{\mathbf{A}}$). Then, $\hat{a}_{ni}$ can be written as follows:

$$\hat{a}_{ni} = \langle u_{ni}, v_{ij} \rangle = \sum_{k=1}^{K} u_{nk}v_{ik}.$$  \hspace{1cm} (12)

$\mathbf{U}$ and $\mathbf{V}$ are termed feature matrices, $u_{ni}$ and $v_{ij}$ are termed feature vectors, and $u_{nk}$ and $v_{ik}$ are termed model parameters. A set of model parameters can be expressed as $\{\mathbf{U}, \mathbf{V}\} = \{u_{nk}, v_{ik} | n \in [N], i \in [M], k \in [K]\}$.

By approximating matrix $\mathbf{A}$ in this way, the number of parameters that need to be estimated is reduced from $NM$ to $K(N + M)$. As a consequence, matrix $\mathbf{A}$ (including unobserved elements) can be effectively estimated from a small amount of training data. A set of model parameters $\{\mathbf{U}, \mathbf{V}\}$ can be trained by using, for example, ANLS (Alternating Non-Negative Least Squares) [25], [44].

We then describe tensor factorization, which is the generalization of matrix factorization to a third-order tensor. There are various methods for factorizing a third-order tensor, of which TD (Tucker Decomposition) and CD (Canonical Decomposition) are popular methods [25]. However, we focus on PITF (Pairwise Interaction Tensor Factorization) [26], [27] because it provides high performance (it outperforms TD and CD in tag recommendation [26]). We explain PITF using a tensor composed of the “User,” “From Region,” and “To Region” modes.

---

Fig. 6. Matrix factorization and PITF applied to a tensor composed of the “User,” “From Region,” and “To Region” modes.
Let $\mathcal{A} \in \mathbb{R}^{N \times M \times M}$ be a large tensor composed of the “User,” “From Region,” and “To Region” modes (i.e., $N$ and $M$ are large). Further, let $a_{n,i,j} \in \mathbb{R}$ be the $(n, i, j)$-th element of $\mathcal{A}$. Then, PITF approximates $a_{n,i,j}$ as follows:

$$\hat{a}_{n,i,j} = (u^{(a)}_i, v^{(a)}_j) + (u^{(b)}_i, v^{(b)}_j) + (u^{(c)}_i, v^{(c)}_j)$$

$$= \sum_{k=1}^{K} u^{(a)}_{i,k} v^{(a)}_{j,k} + \sum_{k=1}^{K} u^{(b)}_{i,k} v^{(b)}_{j,k} + \sum_{k=1}^{K} u^{(c)}_{i,k} v^{(c)}_{j,k},$$

where $\hat{a}_{n,i,j}$ is an approximation of $a_{n,i,j}$, $u^{(a)}_i, v^{(a)}_j, u^{(b)}_i, v^{(b)}_j, u^{(c)}_i, v^{(c)}_j \in \mathbb{R}^K$ ($K \ll N, M$) are feature vectors, and $u^{(a)}_i, v^{(a)}_j, u^{(b)}_i, v^{(b)}_j, u^{(c)}_i, v^{(c)}_j \in \mathbb{R}^M$ are model parameters.

Let $U^{(a)} \in \mathbb{R}^{M \times M}$, $V^{(a)} \in \mathbb{R}^{M \times K}$, $U^{(b)} \in \mathbb{R}^{M \times N}$, $V^{(b)} \in \mathbb{R}^{M \times N}, U^{(c)} \in \mathbb{R}^{N \times K}$, and $V^{(c)} \in \mathbb{R}^{N \times K}$ be feature matrices corresponding to these feature vectors (e.g., $U^{(a)} = (u^{(a)}_1, \ldots, u^{(a)}_M)^T$). Then, it can be seen from (11), (12), and (13) that PITF generalizes matrix factorization by modeling the interaction between the “From Region” and “To Region” as $U^{(a)}V^{(a)T}$, the interaction between the “User” and “To Region” as $U^{(b)}V^{(b)T}$, and the interaction between the “User” and “From Region” as $U^{(c)}V^{(c)T}$ (see the panel on the right in Fig. 6). A set of model parameters can be expressed as $\Theta = \{U^{(a)}, V^{(a)}, U^{(b)}, V^{(b)}, U^{(c)}, V^{(c)}\}$.

By approximating tensor $\mathcal{A}$ in this way, the number of parameters that need to be estimated is reduced from $NM^2$ to $K(2N + 4M)$. As a consequence, tensor $\mathcal{A}$ (including unobserved elements) can be effectively estimated from a small amount of training data.

### B. Training Algorithm

The training method in [23] uses tensor factorization to estimate personalized transition matrices $\{P_n|n \in [N]\}$ (i.e., transition probability tensor) from a small number of training traces. It should be noted here that the summation of the transition probabilities over the “To Region” is always 1, and it is difficult to decompose $\{P_n|n \in [N]\}$ under this constraint. To avoid this problem, the training method in [23] first computes the transition count tensor, of which the $(n, i, j)$-th element is a transition count of user $u_n$ from region $x_i$ to $x_j$, from the training traces. Then it decomposes the transition count tensor using tensor factorization, and normalizes the transition counts to probabilities such that the summation over the “To Region” is 1. Fig. 7 shows an overview of the training method in [23].

We now describe the training algorithm in [23] to provide more details. The algorithm regards the transition count tensor as $\mathcal{A}$ in Section IV-A (i.e., $a_{n,i,j}$ is the transition count of user $u_n$ from region $x_i$ to $x_j$), and solves the following optimization problem:

$$\Theta = \arg \min_{\Theta \geq 0} \sum_{(n,i,j) \in \mathcal{D}} (a_{n,i,j} - \hat{a}_{n,i,j})^2 + \alpha ||\Theta||_F^2,$$

where $\hat{\Theta}$ is an estimation of $\Theta = \{U^{(a)}, V^{(a)}, U^{(b)}, V^{(b)}, U^{(c)}, V^{(c)}\}$, and $\mathcal{D} = \{(n,i,j) | \sum_{j'=1}^M \delta_{n,i,j'} \geq 1\}$ (i.e., a set of observed elements, which are not marked with “?” in Fig. 7). The first term in (15) is the summation of the squared errors over the observed elements, and the second term is a regularization term to avoid overfitting of the observed elements. $||\Theta||_F^2$ is the Frobenius norm of $\Theta$ (i.e., the square sum of all model parameters), and $\alpha$ is a regularization parameter that is often determined by cross-validation [43]. The constraint $\Theta \geq 0$ means that all model parameters in $\Theta$ are non-negative. It follows from (14) that the transition count $\hat{a}_{n,i,j}$ is guaranteed to be non-negative by this constraint.

Although the optimization problem (15) is not convex for all model parameters in $\Theta$, it is quadratic with regard to one model parameter $\theta \in \Theta$. Thus, the training method in [23] uses ANLS [25], [44] to find an approximate solution to (15). ANLS alternately solves (15) for one model parameter $\theta \in \Theta$ while keeping the others constant, and iterates it until convergence. For more details of the update formulae, see [23].

In summary, the training algorithm in [23] is as follows:

**Algorithm 1 (Training Algorithm in [23]):**

1. Compute a transition count tensor $\mathcal{A}$ from training traces.
2. Compute a set of model parameters $\Theta$ from $\mathcal{A}$ by solving (15) using ANLS.
3. Compute $\{\hat{a}_{n,i,j}|n \in [N], i,j \in [M]\}$ from $\Theta$ using (14), and normalize them to probabilities ($\sum_{j} \hat{a}_{n,i,j} = 1$).

### V. INCORPORATING SPATIAL GROUP STRUCTURE INTO TENSOR FACTORIZATION

We now incorporate group sparsity regularization into the training method in [23] to introduce the spatial group structure, which is described in Section I-A, as prior knowledge. The proposed training method is composed of the following two processes: (a) dividing regions $\{x_i| i \in [M]\}$ into some groups from training traces using the Markov Cluster (MCL) algorithm [35]; (b) training personalized transition matrices $\{P_n|n \in [N]\}$ using group sparsity tensor factorization.

We first describe the notion of group sparsity, and discuss how this notion can formulate the spatial group structure (Section V-A). We then propose a method for dividing regions into some groups from training traces (Section V-B). We finally propose group sparsity tensor factorization (Section V-C).

**A. Spatial Group Structure via Group Sparsity**

In the field of machine learning, Yuan and Lin [28] proposed a group sparsity regularization method. This method promotes model parameters that belong to the same group to share the
same sparse pattern (i.e., group sparsity). We now provide an example of group sparsity, and discuss how this notion can formulate the spatial group structure.

Fig. 8 shows an example of group sparsity in model parameters of PITF (as shown in Section IV-A, $U^{(a)}V^{(a)T}$, $U^{(b)}V^{(b)T}$, and $U^{(c)}V^{(c)T}$ model “From Region $\times$ To Region”, “User $\times$ To Region”, and “User $\times$ From Region”, respectively). In this example, regions are divided into three groups: $R_1, R_2, R_3$ (as in Fig. 2). We denote submatrices of $U^{(a)}, V^{(a)}, V^{(b)},$ and $V^{(c)}$ corresponding to the $g$-th group by $U^{(a)}_g, V^{(a)}_g, V^{(b)}_g,$ and $V^{(c)}_g$, respectively ($1 \leq g \leq 3$). Dark (resp. light) gray areas represent elements with large (resp. small) positive values. White areas with “0” represent elements filled with zero values (i.e., group sparsity).

It can be seen from the left side of Fig. 8 that $U^{(a)}_g$ and $V^{(a)}_g$ ($1 \leq g \leq 3$) have similar sparse patterns, whereas $U^{(a)}_g$ and $V^{(h)}_g$ ($1 \leq g, h \leq 3, g \neq h$) have different sparse patterns (in Section VI, we show that $U^{(a)}_g$ and $V^{(h)}_g$ indeed have similar sparse patterns if $g = h$, and do not if $g \neq h$). Then, intra-group (resp. inter-group) elements in $U^{(a)}_gV^{(a)T}$ have large (resp. small) values. As a consequence, intra-group (resp. inter-group) transition probabilities for all users are increased (resp. decreased), since $U^{(a)}V^{(a)T}$ contributes to all elements in a tensor (see the first term of (13)). This captures the fact that the transition probability between two close (resp. distant) regions is generally high (resp. low).

Similarly, it can be seen from the right side of Fig. 8 that if the $n$-th user’s feature vector $u^{(a)}_n$ (resp. $u^{(c)}_n$) and the $g$-th group’s submatrix $V^{(b)}_g$ (resp. $V^{(c)}_g$) have similar density patterns, then the $n$-th row of $U^{(b)}_gV^{(b)T}_g$ (resp. $U^{(c)}_gV^{(c)T}_g$) have large values. As a consequence, the $n$-th user’s transition probability to the $g$-th group is increased. This captures a spatial group structure that is specific to each user (e.g., Alice always stays in $R_3$). In Section VI, we also show some examples of the user-specific spatial group structure by visualizing traces of users who have large values in $U^{(b)}V^{(b)T}$.

The above example shows that the spatial group structure can be formulated by promoting group sparsity in model parameters of PITF. In Sections V-B and V-C, we propose a method to promote such group sparsity from training traces.
column of $V_g$ (see the diagram in the left of Fig. 9). Then, the $L_{1,q}$-norm of $V_g$ is defined as follows:

$$\|V_g\|_{1,q} = \sum_{k=1}^{K} \|\tilde{V}_{g,k}\|_q,$$

where $\|\cdot\|_q$ is the $L_q$ norm ($1 < q < \infty$).

Kim et al. [29] proposed group sparsity matrix factorization. Specifically, their method solves the following optimization problem:

$$\min_{U \geq 0, V \geq 0} \|A - U V^T\|_F^2 + \alpha \|U\|_F^2 + \beta \sum_{g=1}^{G} \|V_g\|_{1,q},$$

where $\alpha (> 0)$ and $\beta (> 0)$ are regularization parameters (see the diagram in the middle of Fig. 9). We describe how the optimization problem (17) promotes group sparsity by using the same example as in [28]. Suppose that $V$ is composed of the following two groups: $V_1 = (v_1, v_2)^T \in \mathbb{R}^2$, and $V_2 = v_3 \in \mathbb{R}$ (i.e., $G = 2, M_1 = 2, M_2 = 1, K = 1$). Then, the sum of the $L_{1,2}$-norms (i.e., $\sum_{g=1}^{G} \|V_g\|_{1,2}$) has a contour in the shape of a “circular cone” (see the diagram in the right of Fig. 9). Since this contour tends to meet the contour of the square error function at the point where $v_1 = v_2 = 0$ (or $v_3 = 0$), which promotes group sparsity.

Based on this idea, we propose group sparsity tensor factorization. Specifically, we extend group sparsity matrix factorization in [29] to a transition count tensor $A$ as follows:

$$\min_{\Theta \geq 0} \sum_{(n,i,j) \in \mathcal{D}} (a_{n,i,j} - \tilde{a}_{n,i,j})^2 + \alpha \|U^{(b)}\|_F^2 + \beta \sum_{g=1}^{G} \|V_g\|_{1,q} + \sum_{g=1}^{G} \|V_g\|_{1,q},$$

where $\mathcal{D} = \{(i,j) | \sum_{j'=1}^{M} a_{n,i,j'} \geq 1\}$. Note that the optimization problems (21) and (23) are the same as the ones solved in [23]. Since they are quadratic, they are easily solved optimally. As for the optimization problems (19), (20), (22), and (24), they can be solved via the Fenchel duality [48] in the same way as [29].

The proposed training method alternately computes (19), (20), (21), (22), and (24), and iterates until convergence. This enables us to estimate a set of model parameters $\Theta$ that captures the spatial group structure. After computing $\Theta$, it normalizes the transition counts to probabilities so that the summation over the “To Region” is 1, in the same way as [23].

In summary, the proposed algorithm for training personalized transition matrices $\{P_{ij} | n \in [N]\}$ using group sparsity tensor factorization is as follows (note that the proposed training algorithm is composed of Algorithm 2 and Algorithm 3; after running Algorithm 2, we run Algorithm 3):

**Algorithm 3 (Group Sparsity Tensor Factorization):**

1) Compute a set of model parameters $\Theta$ from a transition count tensor $A$ by iteratively solving (19), (20), (21), (22), (23), and (24) until convergence.

2) Compute $\{\hat{a}_{n,i,j} | n \in [N], i, j \in [M]\}$ from $\Theta$ using (14), and normalize them to probabilities $\sum_j \hat{a}_{n,i,j} = 1$.

**VI. EXPERIMENTAL EVALUATION IN AN UNBIASED SETTING**

We evaluated the effectiveness of our training method in an open scenario by conducting experiments using three real datasets. It should be noted here that, as described in Section I-B, the accuracy is not biased toward a specific user or trace in an unbiased setting, whereas a biased setting reflects the reality more closely. Thus, we conducted experiments in both an unbiased setting and a biased setting.
In this section, we describe the experiments in an unbiased setting (we describe the experiments in a biased setting in Section VII). We first describe the experimental set-up (Section VI-A). We then present the experimental results (Section VI-B).

A. Experimental Set-up

We conducted experiments in an unbiased setting using three real datasets: the Geolife [32], Gowalla [33], and Foursquare [34] datasets. In the following, we describe details of these datasets and how we extracted traces from these datasets.

- **Geolife dataset:** The Geolife dataset [32] was collected by Microsoft Research Asia from April 2007 to August 2012. It contains mobility traces of 182 users, and records a variety of movements such as going home, going to work, shopping, and dining, mostly in Beijing. In our experiments, we used the mobility traces acquired in Beijing. We chose 80 users with long traces, and extracted, for each user, 10 traces each of which comprises 10 locations and has a time interval of more than 30 minutes (we eliminated the remaining 102 users, since we had insufficient data to extract 10 such traces for these users).

- **Gowalla dataset:** The Gowalla dataset [33] contains data relating to 6442890 check-ins by 196591 users from February 2009 to October 2010. Although the check-in data are scattered all over the world, we used traces acquired in New York and Philadelphia in our experiments. We chose 250 users with long traces, and extracted, for each user, 10 traces each of which comprises 10 locations and has a time interval of more than 30 minutes.

- **Foursquare dataset:** The Foursquare dataset in [34] contains data pertaining to 573703 check-ins in Tokyo from April 2012 to February 2013. We chose 400 users with long traces, and extracted, for each user, 10 traces each of which comprises 10 locations and has a time interval of more than 30 minutes.

We divided each of the above three areas into $16 \times 16$ regions with regular intervals ($M = 256$). We then randomly divided the users in each dataset into two equal groups, and regarded users in the first group (resp. second group) as target users (resp. non-target users). Thus, the number of target users is $N$ ways to randomly choose $N$ target users, and we conducted, for each case, the following “two-class classification” experiments. We assumed that the target user and $N$ non-target users provided anonymized traces (each user provided $10 \rightarrow Y$ traces; hence, there are $10 \rightarrow Y$ traces from the target user and $N(10 \rightarrow Y)$ traces from the non-target users). We performed, for each anonymized trace, the likelihood ratio-based re-identification attack (described in Section III-B) to decide whether the trace is associated with the target user. Then we computed the true positive rate and the false positive rate in this case. The number of anonymized traces from target users (resp. non-target users) is $N(10 \rightarrow Y)$ (resp. $N^2(10 \rightarrow Y)$) in total. We also evaluated the performance for the case in which each user’s $10 \rightarrow Y$ traces were combined into one trace (i.e., the same pseudonym was assigned to the $10 \rightarrow Y$ traces).

We subsequently evaluated the case in which the above anonymized traces are obfuscated. One of the most popular obfuscation methods is a location generalization (or region merging) method [7], [50], [51], which combines multiple regions into one group. However, this method can significantly reduce the data utility when it is applied to a long trace, since the size of obfuscated regions can be very large [7]. Moreover, obfuscated regions do not contain accurate location information (such as offices, shops, and landmarks), which is necessary for analysis or data mining. Thus, we used a location deletion (or location hiding) method [21], [38], [39] instead, which deletes some locations from a trace. This method enables a location that is not deleted to be provided to a third-party (or to be made public) as it is. In our experiments, we set the number of deleted locations $T'$ per trace to $T' \in \{0, 1, 2, 3, 4, 5\}$. To minimize the true positive rate for a given $T'$ (i.e., to optimize the privacy-utility trade-off), we deleted $T'$ locations from a target user’s trace such that the likelihood $P(o_1, \cdots, o_T|H_1)$ in (1) was minimized (as for a non-target user’s trace, we randomly deleted $T'$ locations). As a performance measure, we used the ROC (Receiver Operating Characteristic) curve, which is obtained by plotting the true positive rate against the false positive rate at various thresholds. We also used the AUC (Area Under the Curve),
Fig. 10. ROC curve in an unbiased setting ($Y = 1$, $T' = 0$): performance in the case when the 9 testing traces are combined into one; dashed line: random guess.

Table I

<table>
<thead>
<tr>
<th>AUC (Area Under the Curve) [%] in an Unbiased Setting ($Y = 1$, $T' = 0$). The Best Performance Is Highlighted in Boldface.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>(i) Geolife</td>
</tr>
<tr>
<td>(ii) Gowalla</td>
</tr>
<tr>
<td>(iii) Foursquare</td>
</tr>
</tbody>
</table>

which is the area under the ROC curve. When we computed the ROC curve and the AUC, we averaged the true positive rate and the false positive rate over all 10 ways to randomly choose $N$ target users. This allowed us to stabilize the performance.

B. Experimental Results

We firstly evaluated the performance for the case in which only one trace per user was used as training data and testing traces were not obfuscated ($Y = 1$, $T' = 0$). Fig. 10 and Table I show the ROC curve and the AUC, respectively. It can be seen that the performance of ML is poor. In particular, the performance of ML is the same as, or even worse than, the performance of a random guess when the true positive rate is high, and is not improved by combining the 9 testing traces into one. This is because many testing traces (and many combined traces) have transition patterns that do not exist in the training traces, and the likelihood $P(o_1, \cdots, o_T|H_1)$ in (1) becomes very small for these traces. In other words, ML is adversely affected by the sparse data problem.

On the other hand, TF and GSTF significantly outperform ML, and are improved by combining the 9 testing traces, which indicates that TF and GSTF solve the sparse data problem. It can also be seen that GSTF outperforms TF, and that GSTF* performs the most accurately for all three of the datasets, which confirms the effectiveness of group sparsity regularization. Later, we also show that GSTF indeed has the ability to capture the spatial group structure, as we discussed in Section V-A.

We secondly evaluated the AUC by changing the number of deleted locations $T'$ (per testing trace) from 0 to 5. Fig. 11 shows the results. It can be seen that ML performs much worse than a random guess (where AUC = 50%) when $T' = 5$. This is because we deleted $T'$ locations from the target user’s trace so that the likelihood $P(o_1, \cdots, o_T|H_1)$ in (1) was minimized. In other words, we consider a worst-case scenario for the adversary, and found that ML can perform much worse than a random guess in this scenario. On the other hand, GSTF outperforms a random guess even in this worst-case scenario, which indicates that GSTF can be a threat even when anonymized traces are obfuscated.

We thirdly evaluated the AUC of ML, TF, and GSTF by changing the number of training traces $Y$ (per user) from 1 to 9. Fig. 12 shows the results. It can be seen that the performance of ML increases rapidly as the number of training traces $Y$ increases. When $Y$ is large, ML outperforms GSTF on the Geolife and Gowalla datasets, which indicates that ML is not affected by the sparse data problem in this case. In reality, however, the number of training traces $Y$ can be very small (e.g., $Y = 1, 3$), since a user generally discloses only a small amount of location information in his/her daily life (as described in Section I). GSTF can re-identify a user much more accurately than a random guess even in this realistic case, as shown in Fig. 10 and 11.

The reason for GSTF outperforming ML and TF for small Y can be explained as follows. We analyzed model parameters $\Theta$ trained using GSTF by setting $Y = 1$ and $N = 250$ in the Gowalla dataset (we omit the analysis results obtained for the Geolife and Foursquare datasets due to the lack of space). We first examined whether $U_g^{(a)}$ and $V_h^{(a)}$ had similar sparse patterns if $g = h$, and whether they did not if $g \neq h$, as assumed in Section V-A. Specifically, we computed intra-group correlation ($g = h$) and inter-group correlation ($g \neq h$) as to whether $\hat{u}_{g,k}^{(a)}$ and $\hat{v}_{h,k}^{(a)}$ (trained using GSTF) are sparse. The intra-group (resp. inter-group) correlation coefficient was 0.68 (resp. 0.017). That is, there was high (resp. low) intra-group (resp. inter-group) correlation, which supports the assumption in Section V-A.

We then examined the training traces of top 5 users whose average values in $U^{(b)}V^{(b)^T}$ ($1 \leq g \leq 4$) are the largest when we used TF or GSTF as a training method on the Gowalla dataset. Fig. 13 shows the results (groups $R_1, \cdots, R_4$ are de-
The average number of candidates, we set the threshold such that $\text{TPR}_1 = 50\%$ and $75\%$, respectively. It can be seen that $\text{FPR}_{N^*}$ increases almost in proportion to $N^*$ (i.e., (6) and (9) hold) when $\text{FPR}_{N^*} \ll 1$. It can also be seen that the average number of candidates increases linearly as $N^*$ increases (i.e., (10) holds). These results show that it is difficult for the adversary to accurately classify anonymized traces into $N^* + 1$ categories (i.e., $u_1, \cdots, u_{N^*}$ or someone else) when $N^*$ is very large. However, even in this case, the adversary can decide whether the anonymized traces are from a specific user or not (i.e., two-class classification) with much higher accuracy than a random guess by using GSTF, as shown in Fig. 10 and 11. Also, $N^*$ can be very small, as described in Section I. We can say that GSTF is a threat in such cases.

In summary, we can conclude that TF and GSTF significantly outperform ML and a random guess in a realistic situation in which the amount of training data is small, and in which many anonymized traces are from non-target users (i.e., an open scenario). We can also conclude that GSTF captures the spatial group structure, and outperforms TF.

VII. EXPERIMENTAL EVALUATION IN A BIASED SETTING

In this section, we describe the experiments in a biased setting, which reflects the reality more closely. In this setting, the accuracy might be biased toward a specific user or trace. However, as we show in this section, the experimental results in a biased setting are similar to the experimental results in an unbiased setting, which means that our training method is effective in both of the two settings. We also evaluate the relationship between the size of the testing trace (i.e., the number of locations per testing trace) and the AUC, and show that our training method is a threat even when the size of the testing trace is very small (e.g., 2 to 5 locations).

We first describe the experimental set-up (Section VII-A). We then present the experimental results (Section VII-B).

A. Experimental Set-up

We conducted experiments in a biased setting using the same datasets (i.e., the Geolife [32], Gowalla [33], and
TABLE II
AUC (AREA UNDER THE CURVE) [%] IN A BIASED SETTING.

<table>
<thead>
<tr>
<th></th>
<th>ML</th>
<th>ML*</th>
<th>TF</th>
<th>TF*</th>
<th>GSTF</th>
<th>GSTF*</th>
</tr>
</thead>
<tbody>
<tr>
<td>(i) Geolife</td>
<td>57.1</td>
<td>56.2</td>
<td>62.3</td>
<td>57.5</td>
<td>65.4</td>
<td>64.6</td>
</tr>
<tr>
<td>(ii) Gowalla</td>
<td>57.7</td>
<td>57.4</td>
<td>75.9</td>
<td>81.8</td>
<td>77.3</td>
<td>83.9</td>
</tr>
<tr>
<td>(iii) Foursquare</td>
<td>52.7</td>
<td>48.0</td>
<td>76.7</td>
<td>87.4</td>
<td>77.2</td>
<td>89.1</td>
</tr>
</tbody>
</table>

We divided the area in each dataset into $16 \times 16$ with regular intervals ($M = 256$) in the same way as Section VI. We then randomly divided the user in each dataset into two equal groups (we attempted 10 ways to randomly divide users into groups), and regarded users in the first group (resp. second group) as target users (resp. non-target users). The number of target users is $N = 40, 125, 200$ in the Geolife, Gowalla, and Foursquare datasets, respectively.

We used, for each target user, a small fraction of traces as training data, and the remaining traces as testing data. Specifically, since the Geolife, Gowalla, and Foursquare datasets contain traces for 65, 21, and 11 months, respectively (see Section VI-A), we used, for each target user, the first 1.5% (= 1/65), 4.8% (= 1/21), and 9.1% (= 1/11) of the traces as training data, respectively. This corresponds to the situation in which the first one month is used as a training period. We trained personalized transition matrices $\{P_i|n \in [N]\}$ from the training data using ML, TF, or GSTF with the same parameters as in Section VI.

Using the testing data, we conducted, for each of the $N$ target users, “two-class classification” experiments in the same way as Section VI. Specifically, we assumed that the target user and $N$ non-target users provided anonymized traces (we did not obfuscate the anonymized traces for simplicity), and performed the likelihood ratio-based re-identification attack for each anonymized trace. Then we computed the true positive rate and the false positive rate. We averaged the true positive rate and the false positive rate over all 10 ways to randomly choose $N$ target users, and evaluated the ROC curve and the AUC as a performance measure.

B. Experimental Results

Fig. 16 and Table II show the ROC curve and the AUC, respectively, where “*” represents the performance in the case when each user’s testing traces are combined into one (i.e, the same pseudonym was assigned to these traces). It can be seen from Figs. 10 and 16 that the results in a biased setting are similar to the results in an unbiased setting (i.e., TF significantly outperforms ML and a random guess; GSTF...
outperforms TF). This means that our training method is effective in not only an unbiased setting but also a biased setting, which reflects the reality more closely.

We also evaluated the relationship between the size of the testing trace (i.e., the number of locations per testing trace) and the AUC. Specifically, we divided the testing traces into six groups according to the number of locations (2 to 5, 6 to 10, 11 to 15, 16 to 20, 21 to 25, or more than 25), and evaluated the AUC of ML, TF, and GSTF for each group. Fig. 17 shows the results. It can be seen that the AUC tends to increase with an increase in the number of locations (i.e., the size of the testing trace). However, it should be noted that GSTF significantly outperforms ML and a random guess even when the number of locations is 2 to 5 (GSTF achieves 64.4%, 76.8%, and 76.1% in the GeoLifc, Gowalla, and Foursquare datasets, respectively). In reality, the size of anonymized traces could be very small (e.g., 2 to 5 locations, as shown in Fig. 15). As a conclusion from our experiments, we can say that our training method is a threat even in such cases.

VIII. CONCLUSION

In this paper, we proposed a training method for personalized transition matrices \( \{ P_{n} \}_{n \in [N]} \) using group sparsity tensor factorization (GSTF). We also proposed a framework and an algorithm for a re-identification attack in an open scenario using a population transition matrix \( P_{0} \), which is computed from \( \{ P_{n} \}_{n \in [N]} \). We applied the ML estimation method (ML), the training method in [23] (TF), and GSTF to re-identification in an open scenario, and conducted experiments using three real datasets. We first showed that TF significantly outperforms ML (which was unclear in the task of re-identification), and then showed that GSTF outperforms TF. We also showed that GSTF is a threat even when the size of anonymized traces is very small (e.g., 2 to 5 locations). We believe that this work significantly contributes to understanding re-identification of mobility traces in a realistic situation in which the amount of training data is very small, and in which many anonymized traces are from non-target users (open scenario).

ACKNOWLEDGMENT

The authors would like to thank the reviewers and the associate editor for valuable comments on this paper. In addition, this study was supported in part by JSPS KAKENHI 26880030, 26120504, 26270103, and 15K12112.

REFERENCES


Takao Murakami is a researcher with the Information Technology Research Institute (ITRI), National Institute of Advanced Industrial Science and Technology (AIST). He received B.S., M.S., and Ph.D. degrees from the University of Tokyo in 2004, 2006, and 2014, respectively. He joined Hitachi, Ltd. in 2006, and AIST in 2014. His research interests include location privacy, biometrics, and information security. He received the IPSJ Yamashita SIG Research Award in 2011, and the IEEE TrustCom Best Paper Award for [1] in 2015. He also received the Dean’s Award of the Graduate School of Information Science and Technology, the University of Tokyo, for his Ph.D. dissertation in 2014. He is a member of IEEE, IEICE, and IPSJ.

Atsunori Kanemura received the M.Eng. degree from the Nara Institute of Science and Technology, Japan, in 2006 and the Ph.D. degree in Informatics from Kyoto University, Japan, in 2009. After holding positions as a Researcher at Kyoto University, the University of California, Santa Cruz, ATR Neural Information Analysis Labs., and Osaka University, he is currently a Research Scientist at the National Institute of Advanced Industrial Science and Technology (AIST), Japan. His research focuses on the intelligent analysis of human-related data. He received the JNNS Best Paper Award in 2010 and the IEEE TrustCom Best Paper Award for [1] in 2015. He is a member of IEEE, IEICE, and IPSJ.

Hidetsu Hino received his B. Eng. in 2003, and M. Informatics in Applied Mathematics and Physics in 2005 from Kyoto University, Japan. He was with Hitachi’s Systems Development Laboratory from April 2005 to August 2007. He earned his D. Eng. in 2010 from Waseda University, Japan. In April 2013, he joined the University of Tsukuba where he is presently an Associate Professor. His research interests include machine learning and applied statistics. He received the IEEE TrustCom Best Paper Award for [1] in 2015. He is a member of IEEE.