DE GRUYTER OPEN

Proceedings on Privacy Enhancing Technologies ; 2017 (4):138–155

Takao Murakami*

Expectation-Maximization Tensor Factorization for Practical Location Privacy Attacks

Abstract: Location privacy attacks based on a Markov chain model have been widely studied to de-anonymize or de-obfuscate mobility traces. An adversary can perform various kinds of location privacy attacks using a personalized transition matrix, which is trained for each target user. However, the amount of training data available to the adversary can be very small, since many users do not disclose much location information in their daily lives. In addition, many locations can be missing from the training traces, since many users do not disclose their locations continuously but rather sporadically. In this paper, we show that the Markov chain model can be a threat even in this realistic situation. Specifically, we focus on a training phase (i.e. mobility profile building phase) and propose Expectation-Maximization Tensor Factorization (EMTF), which alternates between computing a distribution of missing locations (E-step) and computing personalized transition matrices via tensor factorization (M-step). Since the time complexity of EMTF is exponential in the number of missing locations, we propose two approximate learning methods, one of which uses the Viterbi algorithm while the other uses the Forward Filtering Backward Sampling (FFBS) algorithm. We apply our learning methods to a de-anonymization attack and a localization attack, and evaluate them using three real datasets. The results show that our learning methods significantly outperform a random guess, even when there is only one training trace composed of 10 locations per user, and each location is missing with probability 80% (i.e. even when users hardly disclose two temporally-continuous locations).

Keywords: Location privacy, Expectation-Maximization algorithm, Tensor factorization, Viterbi algorithm, FFBS algorithm

DOI 10.1515/popets-2017-0042

Received 2017-02-28; revised 2017-06-01; accepted 2017-06-02.

1 Introduction

The number of GPS-equipped devices such as smartphones, tablets, and in-car navigation systems has grown rapidly in recent years. As a consequence, many people are now using location-based services (LBS) such as Point-of-Interest (POI) search and route finding, and location-based social networks (LBSN) such as location check-in, tagging, and nearby friends [40]. In addition, a great number of mobility traces (time-series location trails), called Spatial Big Data [34], are stored in a data center of the LBS provider. These data can be provided to a third-party for analysis (e.g. finding culturally important places, commonly frequented public areas, or fuel-efficient routes [34, 47]), or can be made public to provide traffic information to users [16].

However, the disclosure of location information can lead to inference of users' sensitive locations such as homes and hospitals that are visited regularly, and other private information such as users' properties (e.g. age, occupation) and social relationship [9, 21]. There is also a risk that robbers [30] or stalkers [41] exploit location information of target users. This kind of privacy is called *location privacy*, and numerous studies have addressed this issue from various perspectives (e.g. attacks, defenses, and privacy metrics) [13, 19].

In this paper, we focus on location privacy attacks based on a Markov chain model, which have been widely studied in the literature [11, 22, 24–27, 35, 36, 44]. In this model, an adversary divides an area of interest into regions (or POIs) $x_1, \cdots, x_M$ (M regions in total), and partitions time at regular intervals (e.g. 30 minutes, one hour). The adversary then assumes a Markov property for the movement of target users $u_1, \cdots, u_N$ (N users in total), and trains, for each target user u_n ($1 \leq n \leq N$), an $M \times M$ personalized transition matrix P_n, which comprises the probability $p_{n,i,j}$ ($1 \leq i, j \leq M$) that user u_n moves from region x_i to x_j.

The adversary can perform various kinds of location privacy attacks using personalized transition matrices $P_1, \cdots, P_N$. For example, the adversary can perform a *de-anonymization attack* [11, 24, 26, 27, 35, 36], which identifies users from anonymized traces whose user IDs are replaced by pseudonyms. Another example is a *local-*

***Corresponding Author: Takao Murakami:** National Institute of Advanced Industrial Science and Technology (AIST), E-mail: takao-murakami@aist.go.jp

ization attack [25, 35, 36], which infers an actual location of a target user at a certain time based on his/her trace that is obfuscated (e.g. by adding noise, deleting some locations, merging regions). Many studies [11, 24, 36, 38] showed that the Markov chain model can successfully be used to de-anonymize traces or infer locations with very high accuracy when the amount of training data available to the adversary is very large (see Section 6.1 for more details).

However, the adversary has to address two major problems to use this model in practice. First, many users do not disclose much location information to the public via LBSN such as location check-in and tagging in their daily lives (unless they are heavy LBSN users and do not care much about their privacy). As a result, the amount of training data available to the adversary can be very small. Second, many users do not disclose their locations continuously but rather "sporadically" (Shokri *et al.* [36] defined sporadic locations as ones that are sparsely distributed over time; examples of sporadic applications include POI search, location check-in, and tagging). Therefore, many locations can be missing from the available training traces.

In this paper, we refer to the first problem as the *sparse data problem* (as also done in [25]), and the second problem as the *missing location problem*. For example, it is reported that more than half of LBS users are worried about their loss of privacy [43], and more than 72% LBSN users have less than one check-in per day [6]. Since such users would not disclose many locations nor two temporally-continuous locations, it is crucial for the adversary to address the two problem explained above. Despite the progress in the field of location privacy, no studies have satisfactorily addressed the two problems to our knowledge, as we describe in detail below.

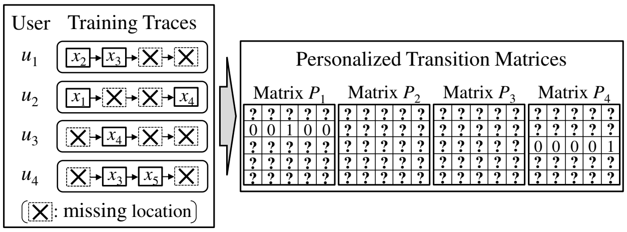


Fig. 1. Example of the sparse data problem and the missing location problem ($M = 5$). "?" is an unobserved element. All elements in matrices P_2 and P_3 are unobserved in this example.

1.1 Existing Learning Methods

Many studies [11, 24, 36, 44] computed transition matrices using the Maximum Likelihood (ML) estimation method. We illustrate that this learning method suffers from the sparse data problem and the missing location problem using the example shown in Fig. 1. In this example, there is only one training trace composed of four locations per user. Many locations are also missing, since the users disclose their locations sporadically. Focusing on user u_1, we can see that a transition pattern from region x_1, x_3, x_4, or x_5 is not observed in the training trace, and therefore transition probabilities from region x_1, x_3, x_4, or x_5 cannot be estimated using the ML estimation method. We refer to the corresponding elements, which are marked with "?" in Fig. 1, as *unobserved elements*. In addition, we can see that the remaining elements are obviously overfitted to the training data (i.e. transition probabilities of user u_1 from x_2 are either 0 or 1). We refer to these elements as *overfitted elements*. If there are many unobserved elements or overfitted elements, the adversary cannot perform location privacy attacks based on the ML estimation method with high accuracy.

Shokri *et al.* [35] proposed a learning method that considers a case where some locations are missing from training traces. Specifically, their learning method alternates between sampling a personalized transition matrix $P_n^{(l)}$ and sampling missing locations using the Gibbs sampling method until convergence ($l = 1, 2, \cdots$). Then it computes an average of $P_n^{(l)}$ over all samples l as an estimate of P_n. In this paper, we refer to this learning method simply as the *Gibbs sampling method*. The Gibbs sampling method outperforms the ML estimation method when the amount of training data is large and some locations are missing (as described in Section 5.2).

However, since the Gibbs sampling method in [35] estimates each personalized transition matrix P_n independently, it suffers from the sparse data problem. Specifically, when none of the locations are missing from training traces, the mode of the estimated distribution of P_n is equivalent to the ML estimate (see Appendix A). Thus, even if the adversary "perfectly" estimates missing locations, the Gibbs sampling method performs almost as well as the ML estimation method without missing locations. Consequently, it suffers from the sparse data problem in the same way as the ML estimation method (e.g. since there are only four locations per user in Fig. 1, there are many unobserved elements and overfitted elements in P_n). In addition, since missing locations are sampled based on $P_n^{(l)}$, the adversary cannot accurately estimate missing locations if $P_n^{(l)}$ is not accurate. Then, the adversary cannot accurately estimate $P_n^{(l+1)}$, since the missing locations are not accurate. Therefore, if $P_n^{(1)}$ is not accurate, the adversary cannot accurately estimate $P_n^{(2)}, P_n^{(3)}, \cdots$. In this paper, we show that this learning method performs worse than the ML estimation method when the amount of training data is very small and some locations are missing.

Murakami and Watanabe [25] proposed a learning method using tensor factorization [8, 32, 33]. This learning method regards a set of personalized transition matrices as a *third-order tensor*, and uses tensor factorization, which decomposes a third-order tensor into low-rank matrices, to accurately estimate the personalized transition matrices from a small amount of training data. Murakami *et al.* applied this learning method to the de-anonymization attack and the localization attack in [26, 27] and [25], respectively, and showed that it significantly outperforms the ML estimation method when the amount of training data is very small[1].

However, since their learning method does not estimate missing locations in training traces, it suffers from the missing location problem. For example, we can see from Fig. 1 that there are no temporally-continuous locations in the training traces of users u_2 and u_3, and therefore all elements in matrices P_2 and P_3 are unobserved. The adversary cannot de-anonymize traces of users u_2 and u_3 using matrices P_2 and P_3, unless he/she estimates missing locations (or more generally, a distribution of missing locations) in the training traces. In this paper, we show that this learning method also performs almost as well as the ML estimation method in the de-anonymization attack when many locations are missing from the training traces.

1.2 Our Contributions

The goal of this paper is to quantify the risk of location privacy attacks in a realistic situation where the amount of training data is very small and many locations are missing from the training traces (i.e. training locations are "sporadically" disclosed[2]). To achieve this goal, we focus on a training phase (i.e. mobility profile building phase) and extend a learning method in [25] to take account of missing locations. Specifically, we propose a learning method that incorporates tensor factorization into the Expectation-Maximization (EM) algorithm [5], which we call *Expectation-Maximization Tensor Factorization (EMTF)*.

Our contributions are summarized as follows:

- We propose EMTF, which alternates between computing a distribution of missing locations (E-step) and computing personalized transition matrices via tensor factorization (M-step) (Section 4.1). By doing so, we can accurately estimate personalized transition matrices from a small amount of training data that contain many missing locations. To our knowledge, this is the first study to incorporate tensor factorization into the EM algorithm to train a third-order tensor while estimating missing values in times-series data such as mobility traces (see Section 6.2 for more details).
- Since the time complexity of EMTF is exponential in the number of missing locations, it is intractable to perform EMTF in practice. Thus we propose two approximate learning methods, one of which uses the Viterbi algorithm [31] while the other uses the Forward Filtering Backward Sampling (FFBS) algorithm [28] (Sections 4.2 and 4.3).
- We apply our learning methods to two representative location privacy attacks: the de-anonymization attack and the localization attack, and evaluate them using three real datasets (Section 5). We first show that the performance of the existing learning methods is close to (or even worse than) that of a random guess in many cases. We then show that our learning methods significantly outperform a random guess in all of the three datasets. In particular, we show that *our learning methods significantly outperform a random guess, even when there is only one training trace composed of 10 locations per user, and each location is missing with probability 80% (all elements in a transition matrix are unobserved, as matrices P_2 and P_3 in Fig. 1, in more than 70% of the cases).* This means that our learning methods can be a threat even when users hardly disclose two temporally-continuous locations.

[1] In [26, 27], they also proposed a learning method that incorporates group sparsity regularization into tensor factorization to utilize the fact that spatial data can form group structure (e.g. Many people are likely to go to an urban area, but Alice likes a rural area). In this paper, we do not use group sparsity regularization for simplicity.

[2] Note the difference between this paper and [36]. Shokri *et al.* [36] considered the case when "testing" locations are sporadically disclosed. On the other hand, we propose a learning method in the case when "training" locations are sporadically disclosed.

1.3 Paper Organization

The rest of this paper is organized as follows. In Section 2, we review the de-anonymization attack and the localization attack. In Section 3, we review the learning method using tensor factorization [25]. In Section 4, we propose EMTF and two approximate learning methods. In Section 5, we show experimental results. In Section 6, we review the previous work closely related to this paper. Finally, in Section 7, we conclude this paper.

1.4 Notations

We describe basic notations used in this paper. We denote the set of integers and real numbers by $\mathbb{Z}$ and $\mathbb{R}$, respectively. We also denote a set of natural numbers less than or equal to n (n: natural number) by $[n]$ (i.e. $[n] = \{1, \cdots, n\}$). Let $\mathcal{U} = \{u_n | n \in [N]\}$ and $\mathcal{X} = \{x_m | m \in [M]\}$ be a set of target users and regions, respectively. We assume that time is discrete, and express time instants as integers (i.e. a set of time instants is $\mathbb{Z}$). Let P_n be an $M \times M$ personalized transition matrix of user u_n, and $p_{n,i,j}$ be its (i, j)-th element (i.e. probability that user u_n moves from region x_i to x_j). An adversary uses a set of personalized transition matrices $\{P_n | n \in [N]\}$ for his/her attack.

2 Location Privacy Attacks

In this paper, we evaluate the ability of our learning methods to de-anonymize traces and to de-obfuscate traces via the de-anonymization attack and the localization attack, respectively. Thus, we briefly review the de-anonymization attack and the localization attack based on the location-privacy framework introduced by Shokri *et al.* [35]. We review the de-anonymization attack in Section 2.1, and the localization attack in Section 2.2.

2.1 De-anonymization Attacks

Fig. 2 shows a framework for de-anonymization attacks. Consider a scenario where the LBS provider anonymizes mobility traces, and provides them to a third-party for analysis or makes them public. We assume that anyone who obtains anonymized traces (except for the LBS provider who has original traces) can be a malicious adversary. For example, if the LBS provider provides the anonymized traces to a third-party, the third-party can be an adversary. If the LBS provider makes the anonymized traces public, an unspecified number of people can be adversaries.

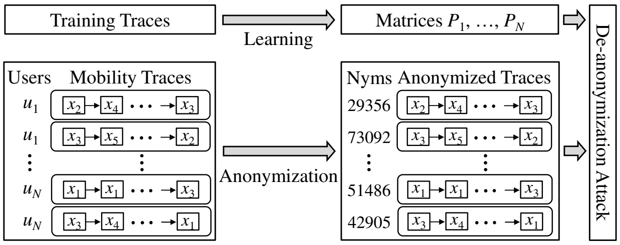


Fig. 2. Framework for de-anonymization attacks.

The adversary first trains personalized transition matrices $\{P_n | n \in [N]\}$ using training traces. For training traces, we make a general assumption; target users can disclose their location information via LBSN; the adversary may obtain training traces of the target users by observing them in person. However, users generally disclose only a small amount of location information to the public in their daily lives (as described in Section 1). In addition, it is hard for the adversary to observe the target users in person, unless he/she is in a close relationship with them. Therefore, the amount of training data can be very small, and many locations can be missing from the training traces.

The adversary then obtains anonymized traces, whose user IDs are replaced with pseudonyms. Here we assume that all of the anonymized traces are from target users $u_1, \cdots, u_N$, as done in [11, 24, 27, 35, 36][3]. We also assume that multiple anonymized traces, each of which is assigned to a different pseudonym, can exist per user (e.g. "29356" for the first trace of u_1, and "73092" for the second trace of u_1 in Fig. 2), as done in [26, 27][4]. The adversary determines, for each anonymized trace, whether it is generated from user u_1, u_2, $\cdots$, or u_N (i.e. N-class classification) using the personalized transition matrices $\{P_n | \ n \in [N]\}$.

In this paper, we focus on the de-anonymization attack based on Bayesian decision theory [27], which minimizes the identification error probability in multi-class

[3] A recent study [26] proposed a de-anonymization attack in an *open scenario*, where many of the anonymized traces are from "non-target" users. In this paper, we do not consider such a scenario (i.e. we consider only a closed scenario) for simplicity.

[4] This is because a user can use LBS from different devices or on different days. The LBS provider may also divide a long trace of the same user into multiple traces with different pseudonyms for privacy reasons.

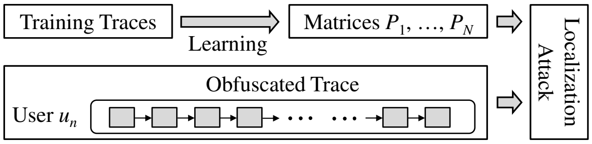


Fig. 3. Framework for localization attacks. Gray rectangles represent obfuscated regions.

classification[5]. Consider an anonymized trace from time 1 to T. Let $\mathbf{o} = (o_1, \cdots, o_T)$ ($o_t \in \mathcal{X}$ is a region at time t) be the anonymized trace, and H_n ($1 \leq n \leq N$) be the hypothesis that the anonymized trace $\mathbf{o}$ is generated from user u_n. The Bayesian de-anonymization attack computes a posterior probability that the hypothesis H_n is true, given $\mathbf{o}$:

$$\Pr(H_n|\mathbf{o}) \quad (1 \leq n \leq N) \tag{1}$$

($\Pr(H_n|\mathbf{o})$ can be computed using Bayes' theorem; see [27] for details), and chooses top N' ($1 \leq N' \leq N$) users whose $\Pr(H_n|\mathbf{o})$ are the largest as candidates. This attack minimizes the error probability that a correct answer is not included in N' candidate users, if $\Pr(H_n|\mathbf{o})$ is accurately estimated.

2.2 Localization Attacks

Fig. 3 shows a framework for localization attacks. For example, consider a scenario where a user discloses his/her locations via LBSN. To protect his/her privacy, we assume the user (or a trusted third party [12]) obfuscates his/her locations before these are disclosed. Examples of obfuscation include *perturbation* (adding noise to locations) [4, 37], *location generalization* (merging regions) [12, 35], and *location hiding* (deleting some locations) [15, 44].

The adversary trains personalized transition matrices $\{P_n | \ n \in [N]\}$ in the same way as in the de-anonymization attack. After obtaining an obfuscated trace of user u_n, the adversary infers an actual location of user u_n at a certain time using the personalized transition matrix P_n.

As in the case of the de-anonymization attack, we focus on the localization attack based on Bayesian decision theory [25, 35]. Consider an obfuscated trace of user u_n from time 1 to T. Let $\mathcal{X}'$ be a set of possible obfuscated regions. For example, we can use a power set of $\mathcal{X}$, which includes an obfuscated region produced by perturbation, location generalization, and location hiding as $\mathcal{X}'$ (i.e. $\mathcal{X}' = \mathfrak{P}(\mathcal{X})$). Let $\mathbf{o}_n = (o_{n,1}, \cdots, o_{n,T})$ ($o_{n,t} \in \mathcal{X}'$ is a region of user u_n at time t) be the obfuscated trace of user u_n, and $X_{n,t} \in \mathcal{X}$ be a random variable that represents a region of user u_n at time t. The Bayesian localization attack computes a posterior probability that the region user u_n is in at time t is $x_i \in \mathcal{X}$, given $\mathbf{o}_n$:

$$\Pr(X_{n,t} = x_i|\mathbf{o}_n) \quad (1 \leq i \leq M). \tag{2}$$

$\Pr(X_{n,t} = x_i|\mathbf{o}_n)$ can be computed using the Forward-Backward algorithm [31] (see [35] for details). Then it chooses top M' ($1 \leq M' \leq M$) regions whose $\Pr(X_{n,t} = x_i|\mathbf{o}_n)$ are the largest as candidates. This attack minimizes the error probability that a correct answer is not included in M' candidate regions, if $\Pr(X_{n,t} = x_i|\mathbf{o}_n)$ is accurately estimated.

It should be noted that the adversary can perform the localization attack using a *population transition matrix* P_*, which is common to all target users. By using a population matrix P_* instead of the personalized matrix P_n, the number of elements that need to be estimated is reduced from NM^2 to M^2. However, since the number of elements is still quadratic in M, the ML estimation method [11, 24, 36, 44] and the Gibbs sampling method [35] still suffer from the sparse data problem when M is large. It is reported in [25] that both P_n and P_* perform only as well as a random guess in many cases when the ML estimation method is used as a training method. On the other hand, it is shown in [25] that P_n can outperform P_* and a random guess when tensor factorization is used, since the sparse data problem is solved by tensor factorization and P_n further exploits unique features of each user's behavior (e.g. favorite places or routes). Thus, we use P_n in our experiments in Section 5.

3 Learning Transition Matrices Using Tensor Factorization (Learning Method in [25])

Murakami and Watanabe [25] proposed a learning method of personalized transition matrices using tensor factorization. A tensor is a multidimensional array that includes a vector as a first-order tensor and a matrix as

[5] Shokri *et al.* [35] formulated this type of attack in a case where there is one anonymized trace per user (they considered $N!$ possible permutations). However, since we assume that multiple traces can exist per user, we consider an attack that de-anonymizes each trace independently.

a second-order tensor, and tensor factorization is a generalization of matrix factorization [18] to a third-order tensor. We briefly review matrix factorization, tensor factorization, and the learning method in [25] in Sections 3.1, 3.2, and 3.3, respectively.

3.1 Matrix Factorization

Matrix factorization is a well-known technique for item recommendation [18]. It decomposes a large matrix into two low-rank matrices to approximate the original matrix, which can include *unobserved elements* (described in Section 1.1), from a small amount of training data.

Let $\mathbf{A} \in \mathbb{R}^{N\times M}$ be the original matrix (N and M are large natural numbers). Matrix factorization decomposes $\mathbf{A}$ into two 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: $\hat{\mathbf{A}} = \mathbf{U}\mathbf{V}^T$, where $\hat{\mathbf{A}}$ is an approximation of $\mathbf{A}$. Let $a_{i,j}$ (resp. $\hat{a}_{i,j}$) $\in \mathbb{R}$ be the (i, j)-th element of $\mathbf{A}$ (resp. $\hat{\mathbf{A}}$), and $\mathbf{u}_i$ (resp. $\mathbf{v}_j$) $\in \mathbb{R}^K$ be the i-th row of $\mathbf{U}$ (resp. the j-th row of $\mathbf{V}$). Then $\hat{a}_{i,j}$ can be written as follows:

$$\hat{a}_{i,j} = \langle \mathbf{u}_i, \mathbf{v}_j \rangle = \textstyle\sum_{k=1}^{K} u_{i,k} v_{j,k}. \tag{3}$$

$\mathbf{U}$ and $\mathbf{V}$ are called *feature matrices*, $\mathbf{u}_i$ and $\mathbf{v}_j$ are called *feature vectors*, and $u_{i,k}$ and $u_{j,k}$ are called *model parameters*. By low-rank approximation, the number of parameters that need to be estimated is reduced from NM to $K(N + M)$, and therefore elements (including unobserved ones) in $\mathbf{A}$ can be effectively estimated from a small amount of training data.

3.2 Tensor Factorization

Tensor factorization is a generalization of matrix factorization to a third-order tensor [8]. Although there are various methods to decompose a third-order tensor (e.g. Tucker Decomposition, Canonical Decomposition [8]), we focus on PITF (Pairwise Interaction Tensor Factorization) [32, 33], which outperforms other methods, in the same way as [25].

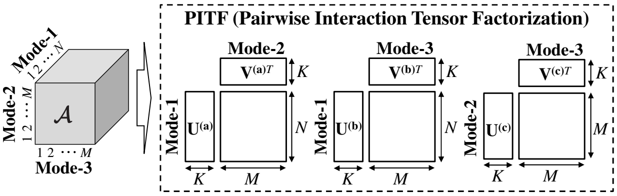


Fig. 4. PITF applied to a third-order tensor $\mathcal{A} \in \mathbb{R}^{N\times M\times M}$. It factorizes three matrices that represent the pairwise interactions between all the three modes.

We explain PITF using a third-order tensor $\mathcal{A} \in \mathbb{R}^{N\times M\times M}$ (which is of the same size as a set of personalized transition matrices) as an example. Let $a_{n,i,j} \in \mathbb{R}$ be the (n, i, j)-th element of $\mathcal{A}$. PITF approximates $a_{n,i,j}$ by modeling the pairwise interactions between all the three modes as follows:

$$\begin{aligned}\hat{a}_{n,i,j} =& \langle \mathbf{u}_n^{(a)}, \mathbf{v}_i^{(a)} \rangle + \langle \mathbf{u}_n^{(b)}, \mathbf{v}_j^{(b)} \rangle + \langle \mathbf{u}_i^{(c)}, \mathbf{v}_j^{(c)} \rangle \qquad (4)\\ =& \textstyle\sum_{k=1}^{K} u_{n,k}^{(a)} v_{i,k}^{(a)} + \sum_{k=1}^{K} u_{n,k}^{(b)} v_{j,k}^{(b)} + \sum_{k=1}^{K} u_{i,k}^{(c)} v_{j,k}^{(c)}, \qquad (5)\end{aligned}$$

where $\hat{a}_{n,i,j}$ is an approximation of $a_{n,i,j}$, and $\mathbf{u}_n^{(a)}$, $\mathbf{v}_i^{(a)}$, $\mathbf{u}_n^{(b)}$, $\mathbf{v}_j^{(b)}$, $\mathbf{u}_i^{(c)}$, $\mathbf{v}_j^{(c)} \in \mathbb{R}^K$ are *feature vectors*. We denote the corresponding *feature matrices* by $\mathbf{U}^{(a)} \in \mathbb{R}^{N\times K}$, $\mathbf{V}^{(a)} \in \mathbb{R}^{M\times K}$, $\mathbf{U}^{(b)} \in \mathbb{R}^{N\times K}$, $\mathbf{V}^{(b)} \in \mathbb{R}^{M\times K}$, $\mathbf{U}^{(c)} \in \mathbb{R}^{M\times K}$, and $\mathbf{V}^{(c)} \in \mathbb{R}^{M\times K}$, respectively (e.g. $\mathbf{u}_n^{(a)}$ is the n-th row of $\mathbf{U}^{(a)}$).

Then, we can see from (3) and (5) that PITF is a generalization of matrix factorization in that $\mathbf{U}^{(a)}\mathbf{V}^{(a)T}$ models the pairwise interaction between the first mode and the second mode, $\mathbf{U}^{(b)}\mathbf{V}^{(b)T}$ models the pairwise interaction between the first mode and the third mode, and $\mathbf{U}^{(c)}\mathbf{V}^{(c)T}$ models the pairwise interaction between the second mode and the third mode (see Fig. 4). By factorizing a tensor in this way, the number of parameters is reduced from NM^2 to $(2N+4M)K$, and elements (including unobserved ones) in $\mathcal{A}$ can be effectively estimated from a small amount of training data.

3.3 Learning Algorithm

Murakami and Watanabe [25] proposed a learning method of personalized transition matrices $\{P_n | n \in [N]\}$ using tensor factorization (PITF). This learning method regards a set of personalized transition matrices as a third-order tensor composed of the “User” mode, the “From Region” mode, and the “To Region” mode, which is called a *transition probability tensor* (see the upper right side of Fig. 5). It should be noted here that *the summation of transition probabilities over “To Region” is always* 1, and it is difficult to decompose the transition probability tensor under this constraint. Thus, the learning method in [25] decomposes a *transition count tensor*, whose (n, i, j)-th element is a transi-

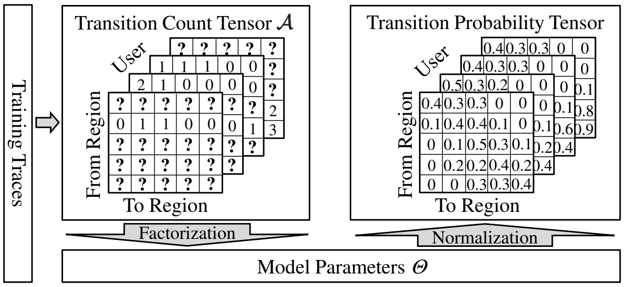

Training Traces
Transition Count Tensor $\mathcal{A}$
User
From Region
To Region
Transition Probability Tensor
User
From Region
To Region
Factorization
Normalization
Model Parameters Θ

Fig. 5. Learning method in [25]. After decomposing a transition count tensor, it normalizes counts to probabilities ("?" is an unobserved element).

tion count of user u_n from region x_i to x_j that is computed from training traces, and then normalizes transition counts to probabilities (so that the summation over "To Region" is 1). Fig. 5 shows an overview of the learning method in [25].

We now explain the learning method in [25] in more details. They applied a transition count tensor to $\mathcal{A} \in \mathbb{R}^{N \times M \times M}$ in Section 3.2, and considered an optimization problem that minimizes the regularized sum of squared errors:

$$\hat{\Theta} = \underset{\Theta \geq 0}{\arg\min} \sum_{A_o} (a_{n,i,j} - \hat{a}_{n,i,j})^2 + \lambda ||\Theta||_F^2, \qquad (6)$$

where $\Theta = \{\mathbf{U}^{(a)}, \mathbf{V}^{(a)}, \mathbf{U}^{(b)}, \mathbf{V}^{(b)}, \mathbf{U}^{(c)}, \mathbf{V}^{(c)}\}$ (i.e. a set of model parameters) and $A_o = \{(n,i,j) | n \in [N], i,j \in [M], \sum_{j'=1}^{M} a_{n,i,j'} \geq 1\}$ (i.e. a set of *observed elements*, which are *not* marked with "?" in Fig. 5). The first term in (6) is a summation of squared errors over A_o (note that they computed $\hat{\Theta}$ using only *observed elements*, and then computed $\hat{a}_{n,i,j}$ for *all elements including unobserved ones* using (5)). The second term in (6) is a *regularization term* introduced to avoid overfitting the observed elements. $|| \cdot ||_F^2$ is the square of the Frobenius norm (i.e. square sum of all elements), and λ (> 0) is a regularization parameter, which is usually determined by cross-validation [18]. The constraint $\Theta \geq 0$ means that all parameters in Θ are non-negative, which guarantees the non-negativity of estimated transition counts $\hat{a}_{n,i,j}$ (see (5)).

Although the optimization problem (6) is not convex, it is quadratic with regard to one parameter (and is solved optimally). Thus, an approximate solution can be found by using the Alternating Least Squares (ALS) algorithm [8], which solves (6) for one parameter θ while fixing the others $\Theta \setminus \{\theta\}$, and iterates it in a cyclic manner until convergence (for details of update formulae, see [25]).

The learning algorithm in [25] is summarized as follows:

Algorithm 1 (Learning Algorithm in [25]):

1. Compute a transition count tensor $\mathcal{A}$ (i.e. $\{a_{n,i,j} | n \in [N], i,j \in [M]\}$) from training traces.
2. Compute model parameters Θ from $\mathcal{A}$ (by solving the optimization problem (6) using ALS).
3. Compute $\{\hat{a}_{n,i,j} | n \in [N], i,j \in [M]\}$ from Θ using (5), and normalize them to probabilities (i.e. $\sum_j \hat{a}_{n,i,j} = 1$).

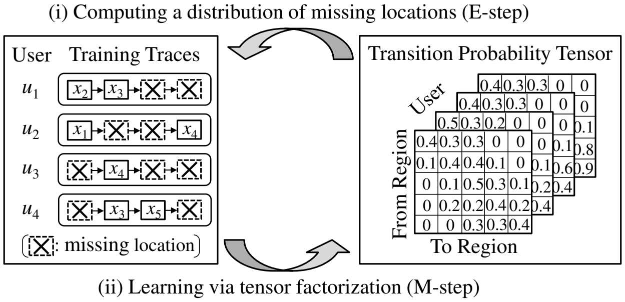
4 Expectation-Maximization Tensor Factorization

In location privacy attacks, the amount of training data can be very small and many locations can be missing from the training traces. To accurately estimate personalized transition matrices in this realistic situation, we propose Expectation-Maximization Tensor Factorization (EMTF), which incorporates tensor factorization into the EM algorithm. We first propose an algorithm for EMTF in Section 4.1. We then propose two approximate learning methods in Sections 4.2 and 4.3.

4.1 The EMTF Algorithm

We begin by describing an overview of EMTF, which is shown in Fig. 6. This learning method alternates between (i) computing a distribution of missing locations (E-step) and (ii) learning personalized transition matrices $\{P_n | \ n \in [N]\}$ (M-step) until convergence (or for a fixed number of times). It is important to note here that *minimizing the regularized sum of squared errors in (6) is equivalent to maximizing a log-posterior probability of model parameters Θ given training traces* (as will be described later in details). Based on this fact, we incorporate tensor factorization into the M-step of the EM algorithm.

We now describe EMTF in more details. Let $\mathbf{x}$ (resp. $\mathbf{z}$) be a vector that consists of observed locations (resp. missing locations) in training traces of all target users, and $\mathcal{Z}$ be a set of all possible values for $\mathbf{z}$. Let further L_{tot} be the total number of missing locations in training traces of all target users. In an example shown in Fig. 6, $\mathbf{x}$ can be expressed as $\mathbf{x} = (x_2, x_3, x_1, x_4, x_4, x_3, x_5)$, and there are $M^{L_{tot}} = 5^9$ possible values for $\mathbf{z}$ (i.e. $|\mathcal{Z}| = M^{L_{tot}} = 5^9$). We regard $\mathbf{z}$ as a *latent variable*



0.4	0.3	0.3	0	0
0.1	0.4	0.4	0.1	0
0	0.1	0.5	0.3	0.1
0	0.2	0.2	0.4	0.2
0	0	0.3	0.3	0.4

Fig. 6. Overview of EMTF ($M = 5$). It alternates between (i) computing a distribution of missing locations (E-step) and (ii) learning personalized transition matrices via tensor factorization (M-step).

(or *hidden variable*), and use the EM algorithm to learn a set of model parameters Θ from $\mathbf{x}$ (while computing a distribution of $\mathbf{z}$).

The EM algorithm is an iterative method to find a set of parameters Θ that maximizes a posterior probability $\Pr(\Theta|\mathbf{x})$ given observed data $\mathbf{x}$ [5][6]. To maximize $\Pr(\Theta|\mathbf{x})$, it repeatedly performs the following two steps until convergence (we assume that $\Theta \geq 0$ in the same way as in Section 3.3):

E-step: Compute a distribution of latent variables $\mathbf{z}$:

$$Q(\mathbf{z}) := \Pr(\mathbf{z}|\mathbf{x}, \Theta). \tag{7}$$

M-step: Compute $\hat{\Theta}$ given by

$$\hat{\Theta} = \operatorname*{arg\,max}_{\Theta \geq 0} \sum_{\mathbf{z}} Q(\mathbf{z}) \log \Pr(\Theta|\mathbf{x}, \mathbf{z}). \tag{8}$$

(Then, let $\Theta \leftarrow \hat{\Theta}$ and return to the E-step.)

Each EM cycle is guaranteed to increase $\Pr(\Theta|\mathbf{x})$ [5]. In the E-step, the distribution $Q(\mathbf{z})$ in (7) can be computed using the Forward-Backward algorithm [31] in the same way as the localization attack in Section 2.2. Thus, the remaining question is how to compute $\hat{\Theta}$ in (8), which maximizes the expectation of the log-posterior probability $\log \Pr(\Theta|\mathbf{x}, \mathbf{z})$, in the M-step.

Let $\mathcal{A}^{\mathbf{z}} \in \mathbb{R}^{N \times M \times M}$ be a transition count tensor that can be obtained from "complete" training traces $\{\mathbf{x}, \mathbf{z}\}$. For example, Fig. 7 shows the transition count tensor $\mathcal{A}^{\mathbf{z}}$ in a case where $\mathbf{z} = (x_4, x_4, x_3, x_2, x_3, x_5, x_4, x_2, x_3)$. Similarly, let $a^{\mathbf{z}}_{n,i,j} \in \mathbb{R}$ be the (n, i, j)-th element of $\mathcal{A}^{\mathbf{z}}$, and $A^{\mathbf{z}}_o = \{(n, i, j) | n \in [N], i, j \in [M], \sum_{j'=1}^{M} a^{\mathbf{z}}_{n,i,j'} \geq 1\}$ be a set of *observed elements* in $\mathcal{A}^{\mathbf{z}}$ (which are *not* marked with "?" in Fig. 7). We assume that a transition count $a^{\mathbf{z}}_{n,i,j}$ is represented as an approximation $\hat{a}_{n,i,j}$ in (5) plus a Gaussian noise with zero mean and variance σ_1^2. We also use a half-normal distribution with variance σ_2^2 as a prior distribution over each parameter $\theta \in \Theta$ (we use a half-normal distribution, since $\theta \geq 0$).

Then, maximizing the log-posterior probability $\log \Pr(\Theta|\mathbf{x}, \mathbf{z})$ is equivalent to minimizing the regularized sum of squared errors in (6):

$$\operatorname*{arg\,max}_{\Theta \geq 0} \; \log \Pr(\Theta|\mathbf{x}, \mathbf{z}) \tag{9}$$

$$= \operatorname*{arg\,max}_{\Theta \geq 0} \; \log \Pr(\mathbf{x}, \mathbf{z}|\Theta) + \log \Pr(\Theta) \tag{10}$$

$$= \operatorname*{arg\,max}_{\Theta \geq 0} \left(-\frac{\sum_{A^{\mathbf{z}}_o} (a^{\mathbf{z}}_{n,i,j} - \hat{a}_{n,i,j})^2}{2\sigma_1^2} - \frac{||\Theta||_F^2}{2\sigma_2^2} \right) \tag{11}$$

$$= \operatorname*{arg\,min}_{\Theta \geq 0} \; \sum_{A^{\mathbf{z}}_o} (a^{\mathbf{z}}_{n,i,j} - \hat{a}_{n,i,j})^2 + \lambda ||\Theta||_F^2 \tag{12}$$

(λ is a constant greater than 0). Similarly, the optimization problem (8) can be expressed as follows:

$$\hat{\Theta} = \operatorname*{arg\,max}_{\Theta \geq 0} \sum_{\mathbf{z}} Q(\mathbf{z}) \log \Pr(\Theta|\mathbf{x}, \mathbf{z}) \tag{13}$$

$$= \operatorname*{arg\,min}_{\Theta \geq 0} \sum_{\mathbf{z}} Q(\mathbf{z}) \sum_{A^{\mathbf{z}}_o} (a^{\mathbf{z}}_{n,i,j} - \hat{a}_{n,i,j})^2 + \lambda ||\Theta||_F^2. \tag{14}$$

Since the optimization problem (14) is quadratic with regard to one parameter (in the same way as [25]), we use the ALS algorithm [8] to find an approximate solution to (14)[7]. For an initial value of Θ, we use a value obtained using the learning algorithm in [25] (i.e. **Algorithm 1**).

The EMTF algorithm can be summarized as follows:

Algorithm 2 (EMTF Algorithm):

1. Initialize model parameters Θ (using **Algorithm 1**).
2. **E-step:** Compute a distribution of missing locations

   $$Q(\mathbf{z}) \; (= \Pr(\mathbf{z}|\mathbf{x}, \Theta))$$

   using the Forward-Backward algorithm.

[6] The EM algorithm is normally used to maximize a likelihood $L(\Theta; \mathbf{x})$ $(= \Pr(\mathbf{x}|\Theta))$. However, it can also be used to maximize $\Pr(\Theta|\mathbf{x})$ by introducing a prior over Θ [5].

[7] A recent study [20] proposed a better parameter estimation method, which outperforms ALS. However, exploring the best parameter estimation method is beyond the scope of this paper. We use ALS, since it is simple to implement.

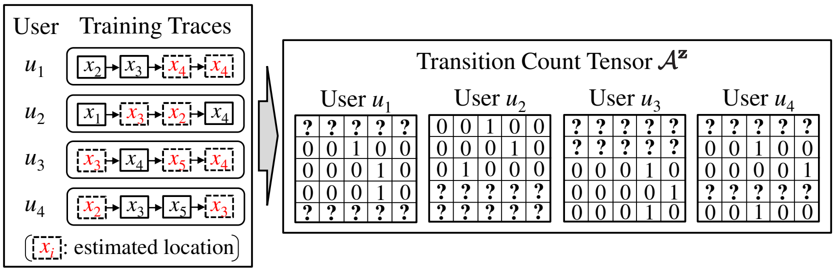


Fig. 7. Training traces with estimated locations ($\mathbf{z} = (x_4, x_4, x_3, x_2, x_3, x_5, x_4, x_2, x_3)$) and a transition count tensor $\mathcal{A}^{\mathbf{z}}$ ("?" is an unoberved element).

3. **M-step:** Compute $\hat{\Theta}$ given by

$$\hat{\Theta} = \underset{\Theta \geq 0}{\arg\min} \sum_{\mathbf{z}} Q(\mathbf{z}) \sum_{A_o^{\mathbf{z}}} (a_{n,i,j}^{\mathbf{z}} - \hat{a}_{n,i,j})^2 + \lambda ||\Theta||_F^2$$

using ALS. Then, let $\Theta \leftarrow \hat{\Theta}$ and return to step 2 until convergence (or for a fixed number of times).

Since each EM cycle is guaranteed to increase the posterior probability $\Pr(\Theta|\mathbf{x})$, the estimates of Θ and $Q(\mathbf{z})$ become more and more accurate as the E-step and the M-step are repeated. After running this algorithm, we can obtain personalized transition matrices $\{P_n | n \in [N]\}$ from Θ by normalizing transition counts to probabilities (in the same way as the learning method in [25]).

Note that the Gibbs sampling method [35] also alternates between sampling a personalized transition matrix P_n and sampling missing locations. However, since this learning method estimates each personalized transition matrix P_n independently, it suffers from the sparse data problem (as described in Section 1.1). Specifically, even if the adversary "perfectly" estimates missing locations, the Gibbs sampling method performs almost as well as the ML estimation method without missing locations (see Appendix A). In addition, since missing locations are sampled based on $P_n^{(l)}$, the adversary cannot accurately estimate missing locations if $P_n^{(l)}$ is not accurate. (Then, he/she cannot accurately estimate $P_n^{(l+1)}$, since the missing locations are not accurate.)

A way to solve this problem is to estimate P_n with the help of "other users" (instead of estimating P_n independently). Since EMTF trains transition matrices $\{P_n | n \in [N]\}$ so that each matrix P_n influences the others (via factorization), it can estimate P_n and missing locations of user u_n with the help of other users. This is the key to solving both the sparse data problem and the missing location problem simultaneously.

4.2 Approximation Using the Viterbi Algorithm

A major drawback of EMTF is its time complexity. Since there are $M^{L_{tot}}$ possible values for $\mathbf{z}$ (i.e. $|\mathcal{Z}| = M^{L_{tot}}$), both the E-step and M-step have time complexity $O(M^{L_{tot}})$ if we do not assume independence of missing locations. If we assume that missing locations are independent from user to user (this assumption is reasonable because each user generally produces traces independently of the others), we can reduce the time complexity to $O(M^{L_{max}})$, where L_{max} is the maximum number of missing locations per user. However, since $M^{L_{max}}$ is very large (e.g. if $M = 256$ and $L_{max} = 8$, then $M^{L_{max}} = 2^{64}$), it is still infeasible to perform EMTF. Note that we cannot reduce the above time complexity by assuming further independence, since there are correlations between locations in the same user. To overcome this drawback, we propose two approximate learning methods.

Our first approximate learning method is based on the Viterbi algorithm [31]. This method approximates a distribution of missing locations $Q(\mathbf{z})$ (in both the E-step and M-step) by the most probable missing location vector $\mathbf{z}_*$:

$$\mathbf{z}_* = \underset{\mathbf{z} \in \mathcal{Z}}{\arg\max}\, Q(\mathbf{z}). \tag{15}$$

$\mathbf{z}_*$ can be computed by using the Viterbi algorithm [31] with time complexity $O(M^2 L_{tot})$ (we do not explain the computation of (15) using the Viterbi algorithm; for details, see [31]). We denote this approximate learning method by $\textbf{EMTF}_{\textbf{Viterbi}}$.

The algorithm for $\textbf{EMTF}_{\textbf{Viterbi}}$ is as follows:

Algorithm 3 ($\textbf{EMTF}_{\textbf{Viterbi}}$ Algorithm):

1. Initialize model parameters Θ (using **Algorithm 1**).

2. **E-step:** Compute $\mathbf{z}_*$ given by
$$\mathbf{z}_* = \operatorname*{arg\,max}_{\mathbf{z} \in \mathcal{Z}} Q(\mathbf{z})$$
using the Viterbi algorithm.
3. **M-step:** Compute $\hat{\Theta}$ given by
$$\hat{\Theta} = \operatorname*{arg\,min}_{\Theta \geq 0} \sum_{A_o^{\mathbf{z}_*}} (a_{n,i,j}^{\mathbf{z}_*} - \hat{a}_{n,i,j})^2 + \lambda ||\Theta||_F^2$$
using ALS. Then, let $\Theta \leftarrow \hat{\Theta}$, and return to step 2 until convergence (or for a fixed number of times).

Let η be the number of iterations in ALS. Then, the time complexity of the E-step and M-step in this algorithm can be expressed as $O(M^2 L_{tot})$ and $O(\eta K |A_o^{\mathbf{z}_*}|)$, respectively (the latter is the same as the time complexity of solving the optimization problem (6) using ALS [25]).

4.3 Approximation Using the FFBS Algorithm

Our second approximate learning method is based on the Forward Filtering Backward Sampling (FFBS) algorithm [28], which approximates a distribution of missing locations $Q(\mathbf{z})$ in a more accurate manner. Specifically, this method takes S (> 0) samples from $Q(\mathbf{z})$:

$$\mathbf{z}_s \sim Q(\mathbf{z}) \; (1 \leq s \leq S), \tag{16}$$

and approximates $Q(\mathbf{z})$ by these samples. The sampling of $\mathbf{z}_1 \cdots, \mathbf{z}_S$ can be performed by using the FFBS algorithm [28], which performs the Forward algorithm and then performs sampling from $Q(\mathbf{z})$ in a backward pass, with time complexity $O(SM^2 L_{tot})$ (for more details of the FFBS algorithm, see [28]). We denote this approximate learning method by $\mathbf{EMTF_{FFBS}}$.

The algorithm for $\mathbf{EMTF_{FFBS}}$ is as follows:

Algorithm 4 ($\mathbf{EMTF_{FFBS}}$ Algorithm):
1. Initialize model parameters Θ (using **Algorithm 1**).
2. **E-step:** Sample $\mathbf{z}_1 \cdots, \mathbf{z}_S$ from $Q(\mathbf{z})$ using FFBS:
$$\mathbf{z}_s \sim Q(\mathbf{z}) \; (1 \leq s \leq S).$$
3. **M-step:** Compute $\hat{\Theta}$ given by
$$\hat{\Theta} = \operatorname*{arg\,min}_{\Theta \geq 0} \frac{1}{S} \sum_{s=1}^{S} \sum_{A_o^{\mathbf{z}_s}} (a_{n,i,j}^{\mathbf{z}_s} - \hat{a}_{n,i,j})^2 + \lambda ||\Theta||_F^2$$
using ALS. Then, let $\Theta \leftarrow \hat{\Theta}$, and return to step 2 until convergence (or for a fixed number of times).

The time complexity of the E-step and M-step in this algorithm can be expressed as $O(SM^2 L_{tot})$ and $O(\eta K (\sum_{s=1}^{S} |A_o^{\mathbf{z}_s}|))$, respectively (the latter increases almost in proportion to S).

As the number of samples S becomes large, the empirical distribution of the samples $\mathbf{z}_1 \cdots, \mathbf{z}_S$ approaches $Q(x)$. Thus, $\mathbf{EMTF_{FFBS}}$ can provide a more accurate approximation than $\mathbf{EMTF_{Viterbi}}$. However, the time complexity of $\mathbf{EMTF_{FFBS}}$ increases almost in proportion to the number of samples S. In other words, there is a trade-off between the accuracy of approximation and the time complexity.

In addition, the imputation of missing locations by $\mathbf{z}_s$ might cause the overfitting problem, since a transition count is generally very small (as shown in Fig. 7) and $\mathbf{z}_s$ might be sampled from a low-probability part of $Q(\mathbf{z})$. One way to solve this problem is to sample $\mathbf{z}_s$ from only a high-probability part of $Q(\mathbf{z})$ (e.g. $Q(\mathbf{z}) \geq \alpha Q(\mathbf{z}_*)$ ($0 \leq \alpha \leq 1$; $\mathbf{z}_*$ is a maximizer of $Q(\mathbf{z})$)). Another solution is to delete an estimated location in $\mathbf{z}_s$ if it produces a rare transition, whose count in $A_o^{\mathbf{z}_s}$ is very small (e.g. one). In our experiments in Section 5, however, we do not adopt such solutions for simplicity, and show that $\mathbf{EMTF_{FFBS}}$ can still provide the highest accuracy.

5 Experimental Evaluation

5.1 Experimental Set-up

We performed experiments to evaluate our learning methods. To provide sufficient evidence for the effectiveness of our learning methods, we used three real datasets: the Geolife dataset [48], the Gowalla dataset [7], and the Foursquare dataset in [45]. The details of these datasets are as follows:

- **Geolife dataset:** The Geolife dataset [48] was collected by Microsoft Research Asia from April 2007 to August 2012. This dataset contains mobility traces of 182 users, and contains a variety of users' movements such as going home, going to work, shopping, dining, and cycling, mostly in Beijing. In our experiments, we used traces in Beijing. We chose 80 users who had long traces ($N = 80$), and extracted, for each user, 10 traces each of which comprises 10 locations and has a time interval of more than 30 minutes (locations in this dataset were sampled with this interval). We eliminated the remaining 102 users, because we had insufficient data to extract such 10 traces for these users.

- **Gowalla dataset:** The Gowalla dataset [7] was collected from February 2009 to October 2010. It contains 6442890 check-ins of 196591 users all over the world. In our experiments, we used traces in New York and Philadelphia. We chose 250 users who had long traces ($N = 250$), and extracted, for each user, 10 traces each of which comprises 10 locations and has a time interval of more than 30 minutes (in the same way as the Geolife dataset).
- **Foursquare dataset:** The Foursquare dataset in [45] was collected from April 2012 to February 2013. It contains 573703 check-ins in Tokyo. In our experiments, we chose 400 users who had long traces ($N = 400$), 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 × 16 regions ($M = 256$). In the de-anonymization attack, we determined boundaries at regular intervals. In the localization attack, we determined boundaries so that all of the traces were uniformly distributed along each axis (i.e. $1/16 = 6.25\%$ per each bin), in the same way as [25], to reduce the size of regions in a crowded area. The size of the smallest region (i.e. the region in the most crowded area) in the Geolife dataset, the Gowalla dataset, and the Foursquare dataset was as follows: [vertical width, horizontal width] = [190m, 205m], [793m, 304m], [275m, 163m], respectively. We used, for each user, one trace as a training trace, and the remaining 9 traces as testing traces. Here we attempted all the 10 ways to choose a training trace (all of the 10 training traces are distinct), and performed, for each case, the following experiments.

Using the training data, we trained personalized transition matrices $\{P_n | n \in [N]\}$. Here we randomly deleted each location (i.e. created a missing location) in the training data with probability (referred to as *missing probability*) $\psi = 0.4$ or 0.8. The latter case (i.e. $\psi = 0.8$) models a scenario where targets users hardly disclose two temporally-continuous locations. After deleting locations, we compared the following learning methods:

ML: The ML estimation method [11, 24, 36, 44].

GS: The Gibbs sampling method [35].

TF: The learning method using tensor factorization [25] (see Section 3.3).

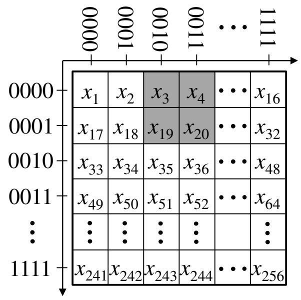

0000
0001
0010
0011
⋯
1111
0000
0001
0010
0011
⋮
1111
x_1 x_2 x_3 x_4 ⋯ x_{16}
x_{17} x_{18} x_{19} x_{20} ⋯ x_{32}
x_{33} x_{34} x_{35} x_{36} ⋯ x_{48}
x_{49} x_{50} x_{51} x_{52} ⋯ x_{64}
x_{241} x_{242} x_{243} x_{244} ⋯ x_{256}

Fig. 8. Example of a merged region ($M = 256$). **LGLH**(1, 0.8), on input region x_{19}, outputs $\{x_3, x_4, x_{19}, x_{20}\}$ (gray area) with probability 0.8 and deletes x_{19} with probability 0.2.

EMTF$_{\textbf{Viterbi}}$: The approximate EMTF method using the Viterbi algorithm (see Section 4.2).

EMTF$_{\textbf{FFBS}}$: The approximate EMTF method using the FFBS algorithm (see Section 4.3).

In **ML** and **GS**, we set transition probabilities for unobserved elements to be uniform (i.e. $p_{n,i,j} = 1/M$), and assigned a very small positive value ($= 10^{-8}$) to an element whose value was 0 in the same way as [24]. In **TF**, **EMTF$_{\textbf{Viterbi}}$**, and **EMTF$_{\textbf{FFBS}}$**, we set the dimensionality K of feature vectors to $K = 16$ (in the same way as [25]), and determined the regularization parameter λ using 10-fold cross-validation. In **TF**, we initialized a set of model parameters Θ using the random initialization method [2], which initializes each parameter as a random value between 0 and 1. In **EMTF$_{\textbf{Viterbi}}$** and **EMTF$_{\textbf{FFBS}}$**, we iterated the E-step and the M-step for three times (we confirmed the performance converged after the three-time iteration). In **EMTF$_{\textbf{FFBS}}$**, we set the number of samples S to $S = 10$ (we increased S from 1 to 20, and confirmed that the performance converged around $S = 10$).

Using the testing data, we evaluated the performance of the Bayesian de-anonymization attack (see Section 2.1) and the Bayesian localization attack (see Section 2.2). In the de-anonymization attack, we evaluated the performance in a case where the adversary de-anonymized each testing trace ($9N$ traces in total) by choosing N' ($1 \leq N' \leq N$) users as candidates. In the localization attack, we obfuscated each testing trace using the *location generalization and location hiding method* [35] with parameters b and ϕ (denoted by **LGLH**(b, ϕ)). This method merges (generalizes) a region by dropping lower b bit(s) for each of the x-coordinate and y-coordinate represented as binary sequences, and deletes (hides) a region with probability

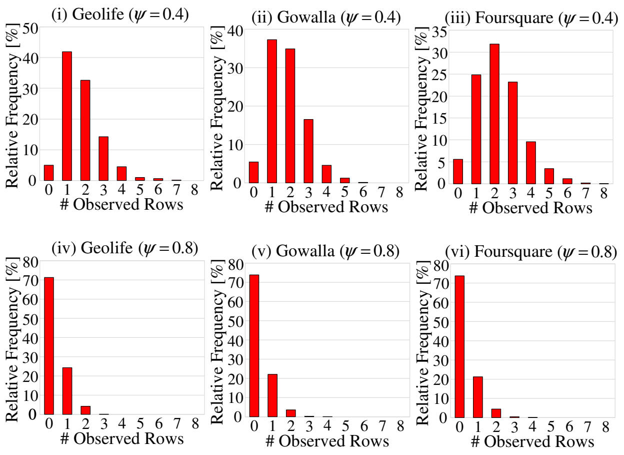


Fig. 9. Frequency distribution of the number of observed rows (whose elements are not marked with "?") per transition matrix.

ϕ. For example, **LGLH**(1, 0.8), on input x_{19}, outputs $\{x_3, x_4, x_{19}, x_{20}\}$ with probability 0.8 and deletes x_{19} with probability 0.2 (see Fig. 8). We set the parameters b and ϕ to $(b, \phi) = (0, 0.5)$ or $(2, 0.5)$ (**LGLH**(0, 0.5) is a location hiding method). After obfuscating the testing traces, we evaluated the performance in a case where the adversary de-obfuscated each region by choosing M' ($1 \leq M' \leq M$) regions as candidates.

As a performance measure, we evaluated an *attack success rate*, which is the ratio of the number of successful attacks (i.e. attacks in which a correct answer is included in the candidates) divided by the total number of attacks ($9N$ in the de-anonymization attack and $90N$ in the localization attack). We averaged the attack success rate over all the 10 ways to choose a training trace to obtain stable performance.

5.2 Experimental Results

We firstly investigated a frequency distribution of the number of *observed rows*, whose elements are *not* marked with "?" (as the 2nd row of matrix P_1 in Fig. 1), per transition matrix. Fig. 9 shows the results (recall that we deleted each location in the training traces with probability $\psi = 0.4$ or 0.8). It can be seen that when the missing probability is $\psi = 0.8$, there are *no* observed rows (i.e. all elements are *unobserved*, as matrices P_2 and P_3 in Fig. 1) in more than 70% of the cases. We emphasize again that the adversary cannot de-anonymize traces of the corresponding users, unless he/she estimates missing locations in the training traces.

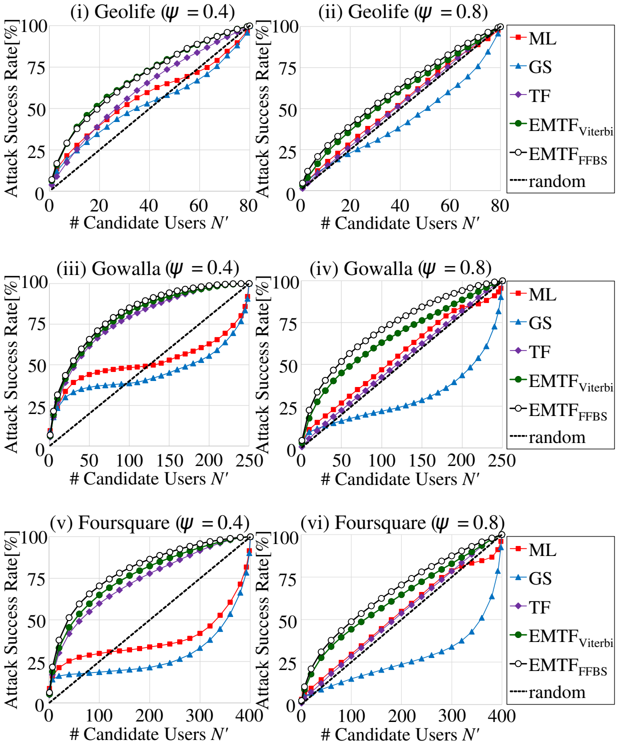


Fig. 10. Relationship between the number of candidate users N' and the attack success rate in the de-anonymization attack (dashed line: random guess).

We secondly evaluated the relationship between the number of candidate users N' and the attack success rate in the de-anonymization attack. Fig. 10 shows the results. We also show in Table 1 (i) the attack success rate in the case when $N' = 10$. It can be seen that **ML** and **GS** provide poor performance. In particular, they perform worse than a random guess when N' is large. This is because many testing traces had transition patterns not in the training traces. Since the personalized transition matrices $\{P_n | n \in [N]\}$ had many *overfitted elements* (as described in Section 1.1), the posterior probability $\Pr(H_n|\mathbf{o})$ in (1) corresponding to the correct answer became very small for these traces. In other words, **ML** and **GS** suffered from the sparse data problem. It can also be seen that **GS** performs worse than **ML**. This is because **GS** did not accurately estimate missing locations, as described in Section 1.1. **TF** outperforms a random guess when the missing probability is $\psi = 0.4$. However, it provides almost the same performance as a random guess when $\psi = 0.8$, since all elements in P_n are *unobserved* in many cases, as shown in Fig. 9.

On the other hand, it can be seen that **EMTF**$_{\mathbf{Viterbi}}$ and **EMTF**$_{\mathbf{FFBS}}$ outperform the existing learning methods in all of the three datasets, and

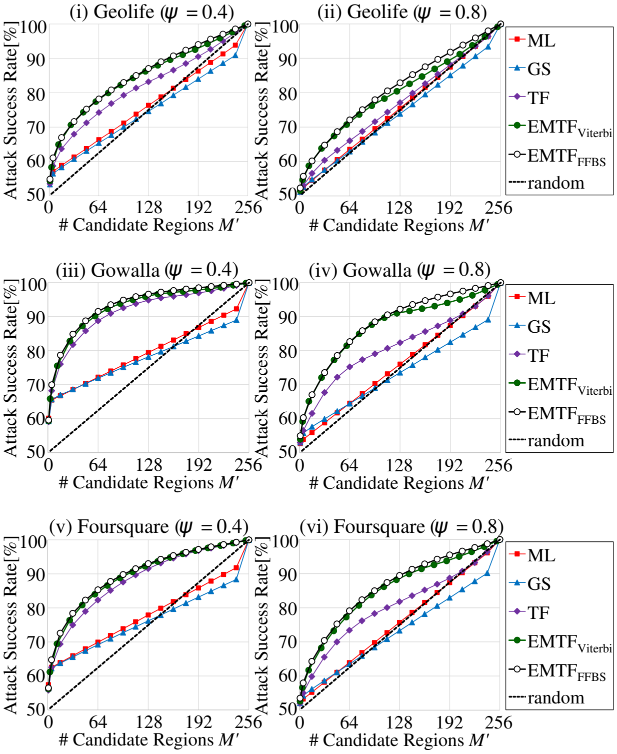


Fig. 11. Relationship between the number of candidate regions M' and the attack success rate in the localization attack (**LGLH**$(0,\ 0.5)$).

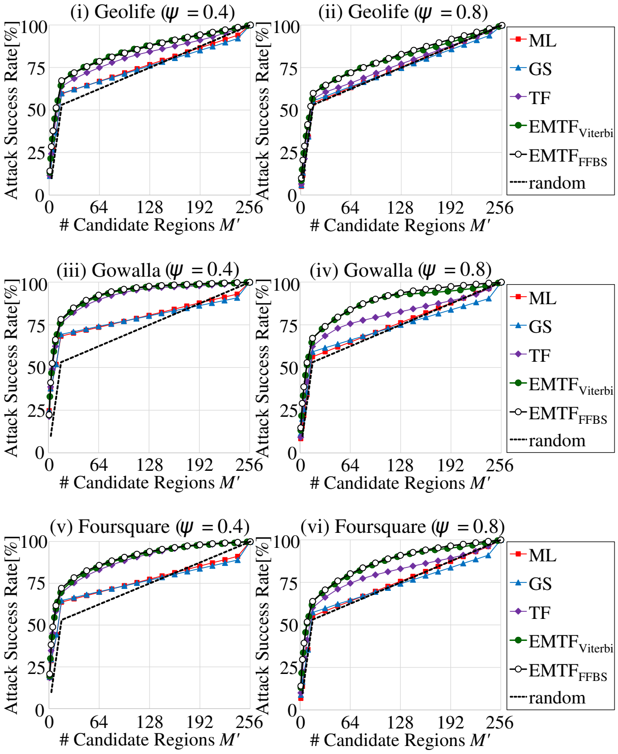


Fig. 12. Relationship between the number of candidate regions M' and the attack success rate in the localization attack (**LGLH**$(2,\ 0.5)$).

significantly outperform a random guess even when $\psi = 0.8$, which shows the effectiveness of our learning methods. It can also be seen that $\mathbf{EMTF_{FFBS}}$ outperforms $\mathbf{EMTF_{Viterbi}}$ in some cases (e.g. Fig. 10 (iv), (v), and (vi)), since $\mathbf{EMTF_{FFBS}}$ approximates a distribution of missing locations $Q(\mathbf{z})$ more accurately, as described in Section 4.3. For example, when $\psi = 0.8$ and $N' = 10$, the attack success rate of $\mathbf{EMTF_{FFBS}}$ is 22.6% in the Gowalla dataset (see Table 1(i)), which is 5.7 times as much as that of a random guess ($10/250 = 4\%$).

We thirdly evaluated the relationship between the number of candidate regions M' and the attack success rate in the localization attack. Fig. 11 and 12 show the results in the case when the testing traces are obfuscated using **LGLH**$(0,\ 0.5)$ and **LGLH**$(2,\ 0.5)$, respectively. Note that the dashed line represents the performance of a random guess that chooses M' regions from a merged region (or all regions when the region is deleted). For example, when $M' = 10$, the attack success rate is 52.0% ($= 0.5 + 0.5 \times 10/256$) and 33.2% ($= 0.5 \times 10/16 + 0.5 \times 10/256$) in **LGLH**$(0,\ 0.5)$ and **LGLH**$(2,\ 0.5)$, respectively[8]. We also show in Table 1 (ii) and (iii) the attack success rate in the case when $M' = 10$.

It can be seen that **ML** and **GS** provide poor performance, while $\mathbf{EMTF_{Viterbi}}$ and $\mathbf{EMTF_{FFBS}}$ outperform the existing learning methods in all of the three datasets, and significantly outperform a random guess even when $\psi = 0.8$, as in the case of the de-anonymization attack. What is different from the case of the de-anonymization attack is that $\mathbf{EMTF_{Viterbi}}$ provides almost the same performance as $\mathbf{EMTF_{FFBS}}$ in all cases. This indicates that the Viterbi algorithm provides a good enough approximation in the case of the localization attack. It can also be seen that **TF** outperforms a random guess even when $\psi = 0.8$ (unlike the case of the de-anonymization attack). This is because **TF** trained feature vectors $\mathbf{u}_i^{(c)}$ and $\mathbf{v}_j^{(c)}$ in (4), which are common to all of the target users, from all of

[8] We included unobfuscated regions when computing the attack success rate in **LGLH**$(0,\ 0.5)$, because we would like to evaluate the attack success rate using the same testing data as in the case of **LGLH**$(2,\ 0.5)$ (i.e. $90N$ regions).

Table 1. Attack success rate [%] in the case when the number of candidates is 10 (**E$_{\text{V}}$**: **EMTF$_{\text{Viterbi}}$**, **E$_{\text{F}}$**: **EMTF$_{\text{FFBS}}$**). The best performance is marked with an asterisk (*).

(i) de-anonymization attack ($N' = 10$)

	ML	GS	TF	E$_{\text{V}}$	E$_{\text{F}}$
Geolife ($\psi = 0.4$)	26.2	23.4	23.6	36.3*	36.1
Geolife ($\psi = 0.8$)	16.3	14.2	14.1	22.1	26.0*
Gowalla ($\psi = 0.4$)	24.0	23.4	27.2	29.3	31.5*
Gowalla ($\psi = 0.8$)	11.0	9.54	5.59	17.8	22.6*
Foursquare ($\psi = 0.4$)	17.7	15.0	20.6	23.0	26.6*
Foursquare ($\psi = 0.8$)	7.18	5.60	4.62	11.3	13.5*

(ii) localization attack (**LGLH**(0, 0.5), $M' = 10$)

	ML	GS	TF	E$_{\text{V}}$	E$_{\text{F}}$
Geolife ($\psi = 0.4$)	57.9	57.4	61.6	63.9	64.5*
Geolife ($\psi = 0.8$)	53.3	53.7	54.8	58.0	58.0*
Gowalla ($\psi = 0.4$)	66.0	66.4	72.6	74.2	74.9*
Gowalla ($\psi = 0.8$)	54.8	56.8	58.9	64.0*	63.8
Foursquare ($\psi = 0.4$)	63.2	63.1	66.3	68.2	69.1*
Foursquare ($\psi = 0.8$)	54.1	55.3	57.4	60.7	61.2*

(iii) localization attack (**LGLH**(2, 0.5), $M' = 10$)

	ML	GS	TF	E$_{\text{V}}$	E$_{\text{F}}$
Geolife ($\psi = 0.4$)	40.1	39.3	47.7	53.2	54.0*
Geolife ($\psi = 0.8$)	34.2	34.4	37.4	43.0	44.4*
Gowalla ($\psi = 0.4$)	51.9	52.0	64.9	67.3	68.7*
Gowalla ($\psi = 0.8$)	37.2	37.9	44.4	54.0	55.5*
Foursquare ($\psi = 0.4$)	44.3	44.3	60.6	62.0	63.9*
Foursquare ($\psi = 0.8$)	35.5	35.4	43.3	53.1	54.0*

Table 2. Running time of the E-step and M-step in **EMTF$_{\text{Viterbi}}$** and **EMTF$_{\text{FFBS}}$** (second).

	EMTF$_{\text{Viterbi}}$		EMTF$_{\text{FFBS}}$	
	E-step	M-step	E-step	M-step
Geolife ($\psi = 0.4$)	0.063	78	0.031	1072
Geolife ($\psi = 0.8$)	0.11	79	0.078	1068
Gowalla ($\psi = 0.4$)	0.22	260	0.093	3761
Gowalla ($\psi = 0.8$)	0.36	263	0.22	3882
Foursquare ($\psi = 0.4$)	0.34	552	0.17	7019
Foursquare ($\psi = 0.8$)	0.59	550	0.34	7539

the training traces. Each transition matrix P_n captured these common features, and the adversary utilized them for specifying a region.

We have so far showed that our learning methods are effective in both the de-anonymization attack and the localization attack. It should be noted, however, that tensor factorization is effective only when the amount of training data is small. It is reported in [25] that **ML** outperformed **TF** when there were 9 training traces (each of which comprises 10 locations) per user. We also confirmed that **GS** outperformed **ML**, **TF**, and our learning methods when there were 9 training traces per user and the missing probability was $\psi = 0.4$[9]. However, since many users do not disclose much location information to the public in their daily lives, nor disclose their locations continuously (but rather sporadically), the amount of training data can be very small and many locations can be missing from the training traces (as described in Section 1). Our learning methods can be a threat even in such a desperate situation for the adversary (i.e. even when there is one training trace and the missing probability is $\psi = 0.8$).

We finally measured the time required for running the E-step and M-step in **EMTF$_{\text{Viterbi}}$** and **EMTF$_{\text{FFBS}}$** on an Intel Xeon CPU E5-2620 v3 (2.40 GHz, 6 cores, 12 logical processors) with 32 GB RAM. Table 2 shows the results in the case when there is one training trace per user, and when we set the number of iterations η in ALS to $\eta = 50$ (we confirmed that model parameters converged before 50 iterations in most cases). The running time of the E-step increases roughly in proportion to the missing probability ψ, which is consistent with our time complexity analysis in Sections 4.2 and 4.3. However, the running time of the E-step is negligibly small compared to that of the M-step. Regarding the running time of the M-step, **EMTF$_{\text{FFBS}}$** requires about 10 times as much as **EMTF$_{\text{Viterbi}}$**, since the number of samples S is $S = 10$. In other words, there is a trade-off between the accuracy of approximation and the running time, as described in Section 4.3. Nonetheless, the running time of one EM cycle in **EMTF$_{\text{FFBS}}$** is only about two hours even in the Foursquare dataset, which contains the largest number of target users ($N = 400$) (note that it is computationally infeasible to perform EMTF, as described in Section 4.2). Thus we can say that our learning methods successfully reduced the learning time.

[9] We also found that **EMTF$_{\text{FFBS}}$** provided the worst performance in the de-anonymization attack in this case due to the overfitting problem described in Section 4.3. Specifically, since there were many missing locations, many estimated locations in $\mathbf{z}_s$ produced a rare transition, whose count in $A_o^{\mathbf{z}_s}$ was only one. By deleting these estimated locations from $\mathbf{z}_s$, the performance of **EMTF$_{\text{FFBS}}$** in the de-anonymization attack was significantly improved (it was better than or almost equal to that of **EMTF$_{\text{Viterbi}}$**). However, it was still worse than that of **GS**.

6 Related Work

6.1 Location Privacy

Location privacy has been widely studied in the literature (see [13, 19] for surveys on this field). In the following, we review the previous work closely related to this paper.

One of the most popular approaches to location privacy attacks is based on a Markov chain model. Many studies showed that this model de-anonymizes traces or infers locations with very high accuracy when the amount of training data is very large. For example, Mulder *et al.* [24] de-anonymized traces of 100 users at the success rate of 77% to 88% when they used traces in a period of one month as training data. Gambs *et al.* [11] de-anonymized traces of 59 users in the Geolife dataset [48] at the success rate of about 45% by using a half of the dataset (over two years) as training data. Shokri *et al.* [36] showed that an adversary who knows personalized transition matrices $\{P_n | n \in [N]\}$ can de-anonymize and de-obfuscate traces with higher accuracy than the one who knows a stationary distribution over regions. Song *et al.* [38] compared various location predictors, which infer a future location based on disclosed locations[10], and showed that a low-order Markov predictor outperformed the others when more than 1000 locations were used as training data. In reality, however, many users would not disclose many locations nor two temporally-continuous locations, as described in Section 1. Thus we proposed EMTF and two approximate learning methods, and showed that the proposed methods can solve both the sparse data problem and the missing location problem.

It is known that the adversary can de-anonymize traces with very high accuracy if the traces contain home/work locations or places where the users meet their friends. Golle and Partridge [14] showed that a majority of the U.S. working population can be uniquely identified from home/work location pairs at the granularity of census blocks. Freudiger *et al.* [10] extended this work, and showed that home/work location pairs can be identified from 20 LBS queries, most of which are made from home/workplace, at the success rate of 65% to 75%. Srivatsa and Hicks [39] deanonymized traces by matching a contact graph, which represents a pattern of meeting between users in the anonymized traces, against a social network graph. Ji *et al.* [17] proposed a graph-based de-anonymization attack that does not require a mapping of landmark nodes. However, users who are worried even a little about their privacy may not use LBS from home/workplace, nor with their friends in a social network graph, when they use LBS continuously. We therefore did not use such auxiliary information, and showed that location privacy attacks can be a threat even in this case.

Another important work is the one presented by de Montjoye *et al.* [23]. They studied how unique mobility traces are, and showed that as few as four locations in a “testing” trace were enough to uniquely characterize 95% of the traces amongst one and a half million people. However, they (implicitly) assume that the adversary has enough “training” traces, and the four locations are included in the training traces. We emphasize again that the amount of training data can be very small and many locations can be missing (i.e. training locations can be sporadically disclosed) in practice. This is the the reason that we proposed EMTF and two approximate learning methods.

Finally, a recent work by Narain *et al.* [29] proposed a side-channel attack that infers actual locations of a target user using only *zero-permission mobile sensors* such as an accelerometer, gyroscope, and magnetometer. They considered a scenario, in which the adversary distributes a mobile app that is seemingly innocuous but collects zero-permission sensor data continuously from the victim. As a method to infer actual locations using the sensor data, they proposed the maximum likelihood route identification algorithm on a graph, and showed that it can identify a route with high accuracy. Our learning methods may be used to further de-anonymize the route (i.e. link the route to the victim’s identity) from a small number of sporadic training locations that are made public (e.g. via LBSN).

6.2 Learning Algorithms (EM Algorithm and Matrix/Tensor Factorization)

There are some studies that combined the EM algorithm with matrix (or tensor) factorization. For example, some studies used the EM algorithm to train model parameters in matrix (or tensor) factorization while estimating missing elements in a matrix (or a third-order tensor) [1, 46]. Another example is the work of Wang *et al.* [42], which expresses a Laplace distribution as a scaled mixture of Gaussians and uses the EM algorithm to perform

[10] Note that a location predictor can be used to infer a sensitive location [22, 44], and such an attack can be regarded as a special case of the localization attack [25].

matrix factorization based on the l_1 loss. However, all of the above studies did not intend to estimate missing values in time-series data such as mobility traces. Anandkumar *et al.* [3] utilized the fact that low-order moments in latent variable models (e.g. HMM) can be written as low-rank tensors. However, this method does not consider a multi-user scenario and estimates *one transition matrix* via tensor factorization. Therefore, this method needs to estimate each personalized transition matrix independently, and suffers from the sparse data problem (in the same way as **ML** and **GS**).

In summary, no studies have incorporated tensor factorization to the EM algorithm to solve the sparse data problem and the missing location problem simultaneously. Thus in this paper, we firstly proposed EMTF, which trains model parameters in tensor factorization while estimating missing locations using the Forward-Backward algorithm. Since the time complexity of EMTF is exponential in the number of missing locations, we proposed two approximate learning methods: $\mathbf{EMTF_{Viterbi}}$ and $\mathbf{EMTF_{FFBS}}$.

7 Conclusion

In this paper, we proposed EMTF and two approximate learning methods to solve both the sparse data problem and the missing location problem. We applied our learning methods to two representative location privacy attacks (i.e. the de-anonymization attack and the localization attack), and evaluated them using three real datasets. The experimental results showed that our learning methods significantly outperform a random guess in all of the three datasets, even when there is only one training trace composed of 10 locations per user, and each location is missing with probability 80% (all elements are *unobserved*, as matrices P_2 and P_3 in Fig. 1, in more than 70% of the cases).

Although we considered only missing locations in training traces, our learning methods can be generalized to a case where training traces are obfuscated (e.g. by adding noise, deleting some locations, merging regions). As future work, we would like to evaluate our learning methods in such a more general scenario. In addition, we would also like to evaluate a state-of-the-art obfuscation method (e.g. perturbation method satisfying geo-indistiguishability [4]) applied to sporadic training locations as a countermeasure against location privacy attacks using our learning methods.

Acknowledgement: The author would like to thank Jacob Schuldt (AIST), Atsunori Kanemura (AIST), and Hideitsu Hino (University of Tsukuba) for technical comments on this paper. The author would also like to thank the reviewers for many valuable comments. This study was supported in part by JSPS KAKENHI 16K16069.

References

[1] Acar E, Dunlavy DM, Kolda TG, Morup M (2011) Scalable tensor factorizations for incomplete data. Chemometrics and Intelligent Laboratory Systems 106(1):41–56

[2] Albright R, Cox J, Duling D, Langville AN, Meyer CD (2014) Algorithms, initializations, and convergence for the nonnegative matrix factorization. SAS Technical Report pp 1–18

[3] Anandkumar A, Ge R, Hsu D, Kakade SM, Telgarsky M (2014) Tensor decompositions for learning latent variable models. Journal of Machine Learning Research 15(1):2773–2832

[4] Andrés ME, Bordenabe NE, Chatzikokolakis K, Palamidessi C (2013) Geo-indistinguishability: Differential privacy for location-based systems. In: Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS'13), pp 901–914

[5] Bishop C (2006) Pattern Recognition and Machine Learning. Springer

[6] Cheng Z, Caverlee J, Lee K, Sui DZ (2011) Exploring millions of footprints in location sharing services. Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM'11) pp 81–88

[7] Cho E, Myers SA, Leskovec J (2011) Friendship and mobility: User movement in location-based social networks. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11), pp 1082–1090

[8] Cichocki A, Zdunek R, Phan AH, Amari S (2009) Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation. Wiley

[9] Eagle N, Pentland A, Lazer D (2009) Inferring friendship network structure by using mobile phone data. Proceedings of the National Academy of Sciences (PNAS) 106(36):15,274–15,278

[10] Freudiger J, Shokri R, Hubaux JP (2011) Evaluating the privacy risk of location-based services. In: Proceedings of the 15th international conference on Financial Cryptography and Data Security (FC'11), pp 31–46

[11] Gambs S, Killijian MO, Núñez del Prado Cortez M (2014) De-anonymization attack on geolocated data. Journal of Computer and System Sciences 80(8):1597–1614

[12] Gedik B, Liu L (2008) Protecting location privacy with personalized k-anonymity: Architecture and algorithms. IEEE Transactions on Mobile Computing 7(1):1–18

[13] Ghinita G (2013) Privacy for Location-based Services. Morgan & Claypool Publishers

[14] Golle P, Partridge K (2009) On the anonymity of home/work location pairs. In: Proceedings of the 7th International Conference on Pervasive Computing (Pervasive'09), pp 390–397

[15] Hoh B, Gruteser M, Xiong H, Alrabady A (2010) Achieving guaranteed anonymity in GPS traces via uncertainty-aware path cloaking. IEEE Transactions on Mobile Computing 9(8):1089–1107

[16] Hull B, Bychkovsky V, Zhang Y, Chen K, Goraczko M (2006) CarTel: A distributed mobile sensor computing system. In: Proceedings of the 4th International Conference on Embedded Networked Sensor Systems (SenSys'06), pp 125–138

[17] Ji S, Li W, Srivatsa M, Beyah R (2014) Structural data de-anonymization: Quantification, practice, and implications. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS'14), pp 1040–1053

[18] Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. IEEE Computer 42(8):30–37

[19] Krumm J (2009) A survey of computational location privacy. Personal and Ubiquitous Computing 13(6):391–399

[20] Kuleshov V, Chaganty AT, Liang P (2015) Tensor factorization via matrix factorization. In: Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS'15), pp 507–516

[21] Matsuo Y, Okazaki N, Izumi K, Nakamura Y, Nishimura T, Hasida K (2007) Inferring long-term user properties based on users' location history. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI'07), pp 2159–2165

[22] Minami K, Borisov N (2010) Protecting location privacy against inference attacks. In: Proceedings of the 9th Annual ACM Workshop on Privacy in the Electronic Society (WPES'10), pp 123–126

[23] de Montjoye YA, Hidalgo AC, Verleysen M, Blondel VD (2013) Unique in the Crowd: The Privacy Bounds of Human Mobility. Scientific Reports, 3(1376):1–5

[24] Mulder YD, Danezis G, Batina L, Preneel B (2008) Identification via location-profiling in GSM networks. In: Proceedings of the 7th ACM Workshop on Privacy in the Electronic Society (WPES'08), pp 23–32

[25] Murakami T, Watanabe H (2016) Localization attacks using matrix and tensor factorization. IEEE Transactions on Information Forensics and Security 11(8):1647–1660

[26] Murakami T, Kanemura A, Hino H (2017) Group sparsity tensor factorization for Re-identification of Open Mobility Traces. IEEE Transactions on Information Forensics and Security 12(3):689–704

[27] Murakami T, Kanemura A, Hino H (2015) Group Sparsity Tensor Factorization for De-anonymization of Mobility Traces. In: Proceedings of the 14th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'15), pp 621–629

[28] Murphy KP (2012) Machine Learning: A Probabilistic Perspective, The MIT Press, chap 17, p 619

[29] Narain S, Vo-Huu TD, Block K, and Noubir G (2016) Inferring User Routes and Locations using Zero-Permission Mobile Sensors. In: Proceedings of the 2016 IEEE Symposium on Security and Privacy (S&P'16), pp 397–413

[30] Please Rob Me - Raising Awareness about Over-sharing (2010) http://pleaserobme.com/

[31] Rabiner LR (1989) A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of IEEE 77(2):257–286

[32] Rendle S, Shmidt-Thieme L (2010) Pairwise interaction tensor factorization for personalized tag recommendation. In: Proceedings of the 3rd ACM International Conference on Web Search and Data Mining (WSDM'10), pp 81–90

[33] Rendle S, Freudenthaler C, Shmidt-Thieme L (2010) Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th International Conference on World Wide Web (WWW'10), pp 811–820

[34] Shekhar S, Evans MR, Gunturi V, Yang K (2012) Spatial big-data challenges intersecting mobility and cloud computing. In: Proceedings of the 11th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'12), pp 1–12

[35] Shokri R, Theodorakopoulos G, Boudec JYL, Hubaux JP (2011) Quantifying location privacy. In: Proceedings of the 2011 IEEE Symposium on Security and Privacy (S&P'11), pp 247–262

[36] Shokri R, Theodorakopoulos G, Danezis G, Hubaux JP, Boudec JYL (2011) Quantifying location privacy: The case of sporadic location exposure. In: Proceedings of the 11th International Conference on Privacy Enhancing Technologies (PETS'11), pp 57–76

[37] Shokri R, Theodorakopoulos G, Troncoso C, Hubaux JP, Boudec JYL (2012) Protecting location privacy: Optimal strategy against localization attacks. Proceedings of the 2012 ACM Conference on Computer and Communications Security (CCS'12) pp 617–627

[38] Song L, Kotz D, Jain R, He X (2006) Evaluating next-cell predictors with extensive wi-fi mobility data. IEEE Transactions on Mobile Computing 5(12):1633–1649

[39] Srivatsa M, Hicks M (2012) Deanonymizing mobility traces: Using social networks as a side-channel. In: Proceedings of the 2012 ACM conference on Computer and communications security (CCS'12), pp 628–637

[40] Vaccari A (2014) Introducing a new optional feature called nearby friends. http://newsroom.fb.com/news/2014/04/introducing-a-new-optional-feature-called-nearby-friends/

[41] Voelcker J (2006) Stalked by satellite: An alarming rise in GPS-enabled harassment. IEEE Spectrum 47(7):15–16

[42] Wang N, Yao T, Wang J, Yeung DY (2012) A probabilistic approach to robust matrix factorization. In: Proceedings of the 12th European Conference on Computer Vision (ECCV'12), vol 7578, pp 126–139

[43] Webroot Inc (2010) Webroot survey finds geolocation apps prevalent amongst mobile device users, but 55% concerned about loss of privacy. http://www.webroot.com/ca/en/company/press-room/releases/social-networks-mobile-security

[44] Xue AY, Zhang R, Zheng Y, Xie X, Huang J, Xu Z (2013) Destination prediction by sub-trajectory synthesis and pri-

vacy protection against such prediction. In: Proceedings of the 2013 IEEE International Conference on Data Engineering (ICDE'13), pp 254–265

[45] Yang D, Zhang D, Zheng VW, Yu Z (2015) Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. IEEE Transactions on Systems, Man, and Cybernetics: Systems 45(1):129–142

[46] Zhang S, Wang W, Ford J, Makedon F (2006) Learning from incomplete ratings usig non-negative matrix factorization. In: Proceedings of the 6th SIAM International Conference on Data Mining (SDM'06), pp 548–552

[47] Zheng Y, Zhang L, Xie X, Ma WY (2009) Mining interesting locations and travel sequences from GPS trajectories. In: Proceedings of the 18th International Conference on World Wide Web (WWW'09), pp 791–800

[48] Zheng Y, Xie X, Ma WY (2010) GeoLife: A collaborative social networking service among user, location and trajectory. IEEE Data Engineering Bulletin 32(2):32–40

A The Gibbs Sampling Method and the ML Estimation Method

In this appendix, we discuss the relationship between the Gibbs sampling method in [35] and the ML estimation method [11, 24, 36, 44]. Shokri *et al.* [35] assumed that all of the prior information available to the adversary are encoded in one of two ways: in the form of training traces or as a matrix of transition counts that are not encoded as traces. Although we consider only the first way of encoding (i.e. we assume that all of the prior information are encoded in the form of training traces), our discussion in this appendix can be extended to the case when the adversary has, in addition to training traces, transition counts that are not encoded as traces.

The learning method in [35] alternates between sampling a personalized transition matrix $P_n^{(l)}$ and sampling missing locations using the Gibbs sampling method until convergence ($l = 1, 2, \cdots$). To sample $P_n^{(l)}$, they assumed that each row of P_n is independent of the others, and assumed the Dirichlet prior distribution for each row of P_n. Let $\mathbf{p_{n,i}} = (p_{n,i,1}, \cdots, p_{n,i,M})$ be the i-th row of P_n. The probability density of the Dirichlet distribution for variables $\mathbf{p_{n,i}} = (p_{n,i,1}, \cdots, p_{n,i,M})$ with parameters $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_M)$ is given by

$$\mathrm{Dir}(\mathbf{p_{n,i}}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{j=1}^{M} \alpha_j)}{\Gamma(\alpha_1) \cdots \Gamma(\alpha_M)} \prod_{j=1}^{M} (p_{n,i,j})^{\alpha_j - 1}, \quad (17)$$

where Γ denotes the gamma function. It should be noted that $\mathrm{Dir}(\mathbf{p_{n,i}}|\boldsymbol{\alpha})$ in (17) is a probability density function for $\mathbf{p_{n,i}} = (p_{n,i,1}, \cdots, p_{n,i,M})$ in the case when the event that user u_n moves from region x_i to x_j is observed $\alpha_j - 1$ times ($1 \leq j \leq M$).

Let $\mathbf{x}_n$ (resp. $\mathbf{z}_n$) be a vector that consists of observed locations (resp. missing locations) in training traces of user u_n. Let further $\{\mathbf{x}_n, \mathbf{z}_n^{(l)}\}$ be "complete" training traces of u_n at the l-th iteration of the Gibbs sampling, and $c_{nij}^{(l)}$ be the number of transitions from region x_i to x_j in the traces $\{\mathbf{x}_n, \mathbf{z}_n^{(l)}\}$. Based on the meaning of $\mathrm{Dir}(\mathbf{p_{n,i}}|\boldsymbol{\alpha})$ explained above, it would be natural to set the parameters $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_M)$ as follows[11]:

$$\alpha_j = c_{nij}^{(l-1)} + 1 \quad (1 \leq j \leq M). \quad (18)$$

Then, the mode of the Dirichlet distribution is written as follows:

$$\frac{\alpha_j - 1}{\sum_{j=1}^{M} \alpha_j - M} = \frac{c_{nij}^{(l-1)}}{\sum_{j=1}^{M} c_{nij}^{(l-1)}}. \quad (19)$$

We can discuss the relationship between the Gibbs sampling method in [35] and the ML estimation method [11, 24, 36, 44] based on (19). Consider the case when none of the locations are missing from training traces, and let c_{nij} be the number of transitions from region x_i to x_j in the training traces of user u_n. Then, it follows from (19) that the mode of the Dirichlet distribution can be written as $c_{nij}/(\sum_{j=1}^{M} c_{nij})$, which is equivalent to the ML estimate. Since the mode is the value at which its probability density function has a maximum value, the Gibbs sampling method in [35] performs almost as well as the ML estimation method [11, 24, 36, 44] in this case.

Thus, even if the adversary "perfectly" estimates missing locations, the Gibbs sampling method performs almost as well as the ML estimation method without missing locations. For example, even if the adversary perfectly estimates missing locations in our experiments in Section 5, the Gibbs sampling method in [35] performs almost as well as the ML estimation with only ten training locations for each user, and suffers from the sparse data problem. In addition, the adversary cannot accurately estimate missing locations in this case, since $P_n^{(l)}$ is not accurate. This is the reason that the Gibbs sampling method performed worse than the ML estimation method in our experiments.

[11] In [35], they also added a *mobility constraint* parameter ϵ_{ij}, which takes a very small positive number if it is possible to move from region x_i to x_j in one time instant (otherwise, $\epsilon_{ij} = 0$). However, we do not add ϵ_{ij} to α_j in this paper for simplicity.