# State-Sharing Sparse Hidden Markov Models for Personalized Sequences

Hongzhi Shi[1], Chao Zhang[2], Quanming Yao[3*], Yong Li[1], Funing Sun[4], Depeng Jin[1]
[1] Beijing National Research Center for Information Science and Technology (BNRist), Department of Electronic Engineering, Tsinghua University, Beijing, China.
[2]College of Computing, Georgia Institute of Technology, Atlanta, Georgia, USA
[3] 4Paradigm Inc., Beijing, China; [4] Tencent Inc., Beijing, China
liyong07@tsinghua.edu.cn

## ABSTRACT

Hidden Markov Model (HMM) is a powerful tool that has been widely adopted in sequence modeling tasks, such as mobility analysis, healthcare informatics, and online recommendation. However, using HMM for modeling personalized sequences remains a challenging problem: training a unified HMM with all the sequences often fails to uncover interesting personalized patterns; yet training one HMM for each individual inevitably suffers from data scarcity. We address this challenge by proposing a state-sharing sparse hidden Markov model (S3HMM) that can uncover personalized sequential patterns without suffering from data scarcity. This is achieved by two design principles: (1) all the HMMs in the ensemble share the same set of latent states; and (2) each HMM has its own transition matrix to model the personalized transitions. The result optimization problem for S3HMM becomes nontrivial, because of its two-layer hidden state design and the non-convexity in parameter estimation. We design a new Expectation-Maximization algorithm based, which treats the difference of convex programming as a sub-solver to optimize the non-convex function in the M-step with convergence guarantee. Our experimental results show that, S3HMM can successfully uncover personalized sequential patterns in various applications and outperforms baselines significantly in downstream prediction tasks.

## CCS CONCEPTS

• **Computing methodologies** → **Latent variable models**.

## KEYWORDS

Sequence modeling, hidden Markov model, parameter sharing, sparse estimation.

**ACM Reference Format:**
Hongzhi Shi, Chao Zhang, Quanming Yao, Yong Li, Funing Sun, Depeng Jin. 2019.State-Sharing Sparse Hidden Markov Models for Personalized Sequences. In *The 25th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD'19), August 4–8, 2019, Anchorage, AK, USA.* ACM, NY, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330828

* Corresponding author.


*KDD '19, August 4–8, 2019, Anchorage, AK, USA*

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330828

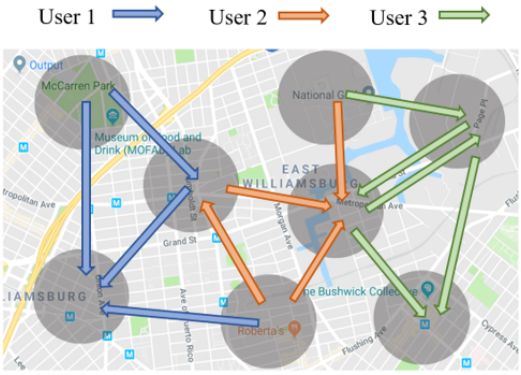



**Figure 1: An example of user mobility modeling with S3HMM. All users share the same set of latent states but have personalized transitions.**

## 1 INTRODUCTION

Personalized sequences are ubiquitous in our daily life, and modeling such sequences is of great importance to various applications. For example, in ride-sharing and transportation systems, modeling users' trajectory sequences can help understand their mobility and improve the effectiveness of the system [16, 18, 25]; in online recommender systems, uncovering the sequential regularities behind people's behaviors is critical to sequential recommendation in the online setting[4]; for targeted advertising, it is important to analyze users' historical purchasing behaviors for more effective personalized advertising strategies [8]. In all these scenarios, an important characteristic of the sequential modeling task is that the sequential patterns are highly *personalized*. Different users can have totally different transition regularities; there is thus a natural need of uncovering *personalized* sequential regularities for task support and decision making.

Hidden Markov Model (HMM) has long been considered as one of the most important models for sequence modeling tasks [9, 11, 15, 19, 25]. Compared with other sequential models such as conditional random fields and recurrent neural networks, HMMs can not only be learned in a totally unsupervised way, but also provides nice interpretability by virtue of its hidden states and

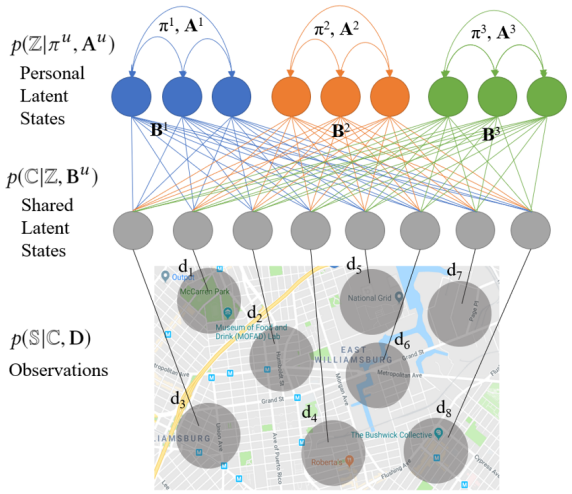



**Figure 2: The two-layer and sparse hidden state design of S3HMM. The shared latent states (middle part) have distributions over the observations and are shared by all the users. Each user has a few personal latent states (top part), which distribute over the shared latent states with the sparsity constraint.**

transitional matrix. Nevertheless, using HMMs for modeling personalized sequences oftentimes faces a key challenge caused by the contradiction between data scarcity and data inconsistency. On one hand, if we train one HMM for each user for personalization, the data are often too scarce to train a reliable model. On the other hand, if we train one HMM for all users, the unified model often fails to reveal interesting personalized patterns because of the conflicting sequential behaviors of different users. Such a dilemma caused by data scarcity and inconsistency remains a critical challenge that has not been solved.

In order to address the above challenge, we propose an extension of HMM. Our model, named state-sharing sparse hidden Markov model (S3HMM), can uncover personalized sequential patterns without suffering from data scarcity. At the high level, S3HMM learns an ensemble of HMMs from the personalized sequences. The design of S3HMM is based on the following principles: (1) all HMMs in the ensemble share the same static set of latent states to avoid suffering from data scarcity; and (2) each HMM maintains its own transition matrix—which is sparse and small—to model personalized transitions. We use Figure 1 to illustrate S3HMM. It shows a mobility modeling task with multiple personalized trajectories. When applying our S3HMM model to such personalized trajectories, the model maintains a personalized transition matrix to reflect the personalized transitions by respecting data inconsistency, but enforces all the users to share the same pool of latent hotspots to learn the hidden states reliably under data scarcity.

We highlight two key features of the model design of S3HMM: **two-layer state-sharing** and **sparsity**. (1) *Two-layer state-sharing.* Straightforwardly, maintaining personalized transition matrix requires $M^2$ parameters for each user, where $M$ is the number of the shared latent states. Such a model design clearly incurs too many parameters and makes the learning procedure intractable because of data scarcity. To tackle this challenge, we introduce a two-layer hidden state design in S3HMM. As shown in Figure 2, given the actual observations (bottom part), the layer of *shared* hidden states (middle part) emit probabilities over the observations and are shared by all the users; and the personalized layer (top part) contains a small number of personal hidden states for each user. Each user transits among a few personal states, which leads to much smaller transition matrices and largely reduces the number of parameters. (2) *Sparsity.* Furthermore, when modeling the distributions of personal states over shared states, we introduce entropy-regularized distributions. The entropy regularization enforces each top-layer state to attend to a few shared states, which further induces sparsity and make the personal states more interpretable.

Optimizing S3HMM becomes nontrivial because of the two-layer hidden state design and the inclusion of the entropy regularization term. As the classic Baum Welch algorithm [1] can no longer be applied here, we propose a new Expectation-Maximization (EM) procedure for the optimization problem. The key challenge in the EM procedure is the sub-problem in the M-step does not have a closed-form solution and becomes non-convex. We thus design a Difference of Convex Programming (DCP) [17, 24], which is guaranteed to converge. Theoretically, we prove the convergence of our optimization algorithm and analyze its computational complexity.

We summarize our contributions as below:

- We propose a novel state-sharing sparse hidden Markov model for modeling personalized sequences. S3HMM not only overcomes data scarcity but also captures the diversity of personalized sequential patterns. Meanwhile, it is highly interpretable by virtue of the sparsity constraint.
- To optimize S3HMM, we design a new EM algorithm which includes DCP as a sub-solver for the M-step. We prove the convergence of our optimization algorithm and analyze its time and space complexity.
- We conduct extensive experiments on the datasets from different domains to demonstrate the generality, effectiveness and efficiency of our model. The experiment results show that our model can successfully reveal personalized sequential patterns, and meanwhile outperforms state-of-the-art models by large margins in downstream prediction tasks.

## 2 PROBLEM FORMULATION

Given a user behavior sequence $\mathbb{S} = r_1 r_2 \cdots r_N$, the $n$-th record of user behavior is defined as a tuple $r_n = <u_n, t_n, e_n>$, where $u_n$ is the user id, $t_n$ is the time stamp and $e_n$ is the user behaviour. In the record, $e_n$ can be either continuous or discrete observations. For the former, $e_n$ is represented as a continuous feature vector, e.g., a two-dimensional vector denoting the user's location in the mobility modeling scenario. For the latter, $e_n$ is a discrete one-hot feature vector, e.g., a one-hot feature vector representing the artist identifier in a music recommender service.

Now given a set of user behavior sequences, the personalized sequence modeling problem aims to understanding the underlying regularities behind each user's behaviors: (1) what are the latent states underlying user's sequential behaviors? (2) how does the user sequentially transit between those latent states?

# 3 MODEL DESCRIPTION

## 3.1 Hidden Markov Model

We begin with introducing the classical hidden Markov model (HMM)[14] to help better understand our proposed model. An HMM assumes all the observations in a sequence are governed by a number of latent states, and the transitions among those latent states follow the Markovian assumption, i.e., the visiting probability of next state only depends on the current state. Formally, letting $M$ be the number of latent states, then there are three sets of parameters in the classical HMM:

- A $M$-dimensional vector $\pi \in \mathcal{R}^M$, where $\pi_m = p(z = m)$ defines the initial probability of visiting the $m$-th latent state.
- A $M \times M$ matrix $\mathbf{A} = \{a_{ij}\} \in \mathcal{R}^{M \times M}$, which defines the transition probabilities between $M$ hidden states following the Markovian assumption. The probability of moving from the $i$-th state to the $j$-th state is given by the entry $a_{ij}$, namely $a_{ij} = p(z_n = j | z_{n-1} = i)$.
- A set of parameters $\mathbf{D} = \{d_m\}$, $m = 1, 2, \cdots M$ defining the distributions of latent states over observations, each parameter subset $d_m$ defines a distribution (e.g., Gaussian or multinomial) of the $m$-th latent state over the observations.

## 3.2 Design Philosophy of S3HMM

As aforementioned, using the classical HMM for modeling personalized sequences often faces the dilemma caused by data scarcity and data inconsistency. In our S3HMM model, we address this dilemma by requiring different users to share the same pool of latent states but maintain their personalized transitions among these states. Such a design philosophy aligns well with most practical sequence modeling scenarios: different users have different initial distributions over the latent states ($\pi$) and transition matrices ($\mathbf{A}$) because of their personalized preferences; but their behavior observations are in the same space and often governed by a fixed set of latent states in that space (e.g, a set of hotspots in a city, or a set of music genres).

A straightforward idea to realize the above philosophy is that, we can train individual $\pi^u$ and $\mathbf{A}^u$ for each user and a shared parameter set $\mathbf{D}$ for all the users. However, the number $M$ of latent states can be potentially large (hundreds or thousands), such a straightforward design inevitably leads to a $O(M^2)$ transition matrix for each user. To deal with this challenge, our key observation is that each user usually correlates with a small number instead of all the latent states. This observation motivates a two-layer and sparse hidden state design of S3HMM, as shown in Figure 2. First, in addition to $M$ shared latent states (the middle layer) that governs concrete observations, we include a layer of abstract latent states that are personalized. Namely, each user maintains a small number $K$ of personalized latent states (the top layer) that correlates the user with a few shared latent states. Since $K$ is significantly smaller than $M$ ($K \ll M$), we reduce the number of personalized parameters from $O(M^2)$ to to $O(KM)$. Second, we add a sparsity-induced regularization on the distribution of personalized latent states over the shared latent states. Specifically, we impose an entropy-based sparsity constraint to force each personalized state only emits to a few shared hidden states. Such a sparsity regularization not only facilitates learning the personalized parameters under data scarcity, but also makes the result transition patterns more interpretable.

## 3.3 The S3HMM Model

We are now ready to mathematically formulate our S3HMM model. For all users, we define a set of $M$ shared latent states, which govern the concrete observations. A set of distribution parameters $\mathbf{D} = \{d_m\}$ define the emission from the $M$ latent states to concrete observations. Namely, each parameter subset $d_m$ defines a parameterized distribution of the $m$-th latent states over the observations. Here, we consider two types of emission distributions: (1) Gaussian distribution: $d_m = \{\mu, \Sigma\}$, where $\mu \in \mathcal{R}^2$ and $\Sigma \in \mathcal{R}^{2\times 2}$. For example, we can use Gaussian to model shared hotspots for locations sequences; (2) multinomial distribution: $d_m = \{\theta_h\} \in \mathcal{R}^H$, where $H$ is the number of items. For example, we can use multinomial distributions to model music genres for music listening sequences. The underlying distributions are formed and shared by all users, such as a shopping mall or a residential area for the first case and the types of music for the second case.

We proceed to describe personalized latent states. As shown in Figure 2, for each observed sequence $\mathbb{S} = e_1 e_2 \cdots e_N$ (the bottom layer), we define two levels of latent states: (1) $\mathbb{C} = c_1 c_2 \cdots c_N$ denote the latent states shared by all the users, which governs distributions over the observations; and (2) $\mathbb{Z} = z_1 z_2 \cdots z_N$ denote the personal latent states, which governs a few shared latent states that correlate with the user. As such, for each user $u$, we have three sets of personalized parameters $\mathbf{\Phi^u} = \{\pi^u, \mathbf{A}^u, \mathbf{B}^u\}$:

- A K-dimensional vector $\pi^u$, where $\pi^u_i = p(z_1 = i)$, which defines the initial distribution over personal hidden states;
- A matrix $\mathbf{A}^u = \{a^u_{ij}\} \in \mathcal{R}^{K\times K}$, which defines the transition probabilities among $K$ personal hidden states with $a^u_{ij} = p(z_n = j | z_{n-1} = i)$;
- A matrix $\mathbf{B}^u = \{b^u_{im}\} \in \mathcal{R}^{K\times M}$, which defines the emission probabilities from $i$-th personal hidden state to the $m$-th shared hidden state with $b^u_{im} = p(c_n = m | z_n = i)$.

Taking mobility modeling to exemplify the above model design, the parameters $\pi^{\mathbf{u}}$ and $\mathbf{A}^u$ describe how user $u$ distributes and transits among her abstract personal states (e.g., dining, working, shopping); while $\mathbf{B}^u$ describes how her personal states distribute over the shared hotspots in the physical world (e.g., shopping malls, restaurants, work places). Note that, the number of personal states is usually small, but the number of shared hotspots can be large.

When modeling the distribution of the personal states $\mathbb{Z}$ over the shared latent states $\mathbb{C}$, we impose the sparsity constraint. This is based on the assumption that a user's abstract state usually maps to a small number of shared states. We use entropy-induced sparsity constraints on $\{b^u_{im}\}$ for $m = 1, \cdots, M$, which can measure the diversification of the distribution and encourage the personal states to focus on a few shared states. Formally, we add entropy terms for $\mathbf{B}^u$ and define the overall objective function as follows:

$$\max_{\mathbf{\Phi}, \mathbf{D}} L(\mathbf{\Phi}, \mathbf{D}) = \overbrace{\sum_u \sum_{\mathbb{S} \in \mathbf{J}^u} \log(p(\mathbb{S} | \mathbf{\Phi^u}, \mathbf{D}))}^{\text{log likelihood}} + \overbrace{\sum_u \lambda g(\mathbf{B}^u)}^{\text{sparsity constraint}}, \quad (1)$$

where $\mathbb{S}$ is a sequence of observations $e$, $\mathbf{J}^u$ is the set of user $u$'s behavior sequences and $\lambda \geq 0$ is the coefficient of the sparsity constraint, and

$$g(\mathbf{B}^u) = \sum_{i,m} b^u_{im} \log\left(b^u_{im}\right), \tag{2}$$

$$p(\mathbb{S}|\mathbf{\Phi^u}, \mathbf{D}) = \sum_{\mathbb{Z},\mathbb{C}} p(\mathbb{S}|\mathbb{C}, \mathbf{D}) \cdot p(\mathbb{C}|\mathbb{Z}, \mathbf{B}^u) \cdot p(\mathbb{Z}|\pi^u, \mathbf{A}^u), \tag{3}$$

where $\sum_{\mathbb{Z},\mathbb{C}}$ means $\sum_{z_1=1}^{K}\sum_{z_2=1}^{K}\cdots\sum_{z_N=1}^{K}\sum_{c_1=1}^{M}\sum_{c_2=1}^{M}\cdots\sum_{c_N=1}^{M}$. To summarize, our model contains the user-specific parameters $\mathbf{\Phi^u} = \{\pi^u, \mathbf{A}^u, \mathbf{B}^u\}$ and a set of shared underlying distributions' parameters $\mathbf{D} = \{d_m\}$, of which the optimization process is described in the Section 4.

## 4 OPTIMIZATION

The Baum Welch algorithm [1] is commonly used for learning the parameters of a standard HMM. However, this algorithm is inapplicable to S3HMM, because of the two-layer hidden layer design and the sparsity regularization on $\mathbf{B}^u$ in Equation (1). In this section, we develop an Expectation-Maximization procedure for learning the parameters of S3HMM.

### 4.1 The Expectation-Maximization Framework

In our developed EM optimization procedure, we first use latent variables $\mathbb{Z}$ and $\mathbb{C}$ to find a lower bound of the objective function based on Jensen's inequality [12], and then alternatively optimize the lower bound to update the model parameters and find a new lower bound until convergence.

The following lemma first establishes a lower bound for the original objective function $L(\mathbf{\Phi}, \mathbf{D})$. Due to the space limit, we leave all proofs for Lemmas and Propositions in Appendix A.

Lemma 4.1. *Let* $\bar{p}(\mathbb{Z},\mathbb{C}) = p\left(\mathbb{Z},\mathbb{C}|\mathbb{S},\mathbf{\Phi}^u_{old},\mathbf{D}_{old}\right)$*, then, for* (1) *it is given by:* $L(\mathbf{\Phi},\mathbf{D}) \geq L'_1(\mathbf{\Phi},\mathbf{D}) - L'_2$ *where*

$$L'_1(\mathbf{\Phi},\mathbf{D}) = \sum_u \left( \sum_{\mathbb{S}\in\mathbf{J}^u} \sum_{\mathbb{Z},\mathbb{C}} \bar{p}(\mathbb{Z},\mathbb{C}) \log p(\mathbb{S},\mathbb{Z},\mathbb{C}|\mathbf{\Phi^u},\mathbf{D}) + \lambda g(\mathbf{B}^u) \right),$$

$$L'_2 = \sum_u \sum_{\mathbb{S}\in\mathbf{J}^u} \sum_{\mathbb{Z},\mathbb{C}} \bar{p}(\mathbb{Z},\mathbb{C}) \log \bar{p}(\mathbb{Z},\mathbb{C}),$$

*where* $L'_2$ *is a constant, which is independent w.r.t.* $\mathbf{\Phi}$ *and* $\mathbf{D}$.

Based on the above lower bound for the objective function, Algorithm 1 sketches the EM algorithm for learning the parameters of S3HMM. We alternate between the two steps: (1) E-step (step 3): calculating the posterior probability $\bar{p}(\mathbb{Z},\mathbb{C})$ for all sequences of all users; (2) M-step (step 4): updating the parameters $\mathbf{\Phi}$ by maximizing the lower bound. We will shortly detail how we update the parameters in the M-step (Algorithm 2) in Section 4.2.

In the E-step, we define three auxiliary variables, i.e., $\xi_n(i,j) = p(z_{n+1} = j, z_n = i|\mathbb{S},\mathbf{\Phi},\mathbf{D})$ where $n = 1, 2..., N-1$, $\gamma_n(i) = p(z_n = i|\mathbb{S},\mathbf{\Phi},\mathbf{D})$ and $\rho_n(i,m) = p(z_n = i, c_n = m|\mathbb{S},\mathbf{\Phi},\mathbf{D})$ where $n = 1, 2..., N$. Then, $\bar{p}(\mathbb{Z},\mathbb{C})$ can be estimated using above three auxiliary variables. In the M-step, using $\xi_n(i,j)$, $\gamma_n(i)$ and $\rho_n(i,m)$ to replace

**Algorithm 1 Expectation Maximization for learning S3HMM.**

**Require:** $J^u$ for all $u$: the set of trajectories of all users $u$;
1: initialize $\{\pi^u\}$, $\{\mathbf{A}^u\}$, $\{\mathbf{B}^u\}$ and $\mathbf{D}$, set threshold $\Delta L > 0$;
2: **while** true **do**
3: *E-step*: update $\bar{p}(\mathbb{Z},\mathbb{C})$;
4: *M-step*: update $\{\pi^u\}$, $\{\mathbf{A}^u\}$ and $\mathbf{D}$ based on $L'_1$ separately, $\{\mathbf{B}^u\}$ based on $L'_1$ using Algorithm 2;
5: **if** $L(\mathbf{\Phi},\mathbf{D}) - L(\mathbf{\Phi_{old}},\mathbf{D_{old}}) < \Delta L$ **then**
6: break;
7: **end if**
8: **end while**
9: **return** model parameters $\{\pi^u\}$, $\{\mathbf{A}^u\}$, $\{\mathbf{B}^u\}$ and $\mathbf{D}$.

$\bar{p}(\mathbb{Z},\mathbb{C})$, $L'_1(\mathbf{\Phi},\mathbf{D})$ becomes:

$$L'_1(\mathbf{\Phi},\mathbf{D}) = \sum_{u,i} \sum_{\mathbb{S}\in\mathbf{J}^u} \gamma_1(i) \log \pi^u_i \tag{4}$$

$$+ \sum_{u,i,j} \sum_{\mathbb{S}\in\mathbf{J}^u} \sum_{n=1}^{N-1} \xi_n(i,j) \log a^u_{ij} \tag{5}$$

$$+ \sum_{u,i,m} \left( \sum_{\mathbb{S}\in\mathbf{J}^u} \sum_{n=1}^{N} \rho_n(i,m) \log b^u_{im} + \lambda b^u_{im} \log\left(b^u_{im}\right) \right) \tag{6}$$

$$+ \sum_{u,i,m} \sum_{\mathbb{S}\in\mathbf{J}^u} \sum_{n=1}^{N} \rho_n(i,m) \log p(e_n|d_m). \tag{7}$$

Then, we maximize $L'_1(\mathbf{\Phi},\mathbf{D}|\mathbb{Z},\mathbb{C})$ by finding the optimal $\mathbf{\Phi}$ and $\mathbf{D}$. We split $L'_1(\mathbf{\Phi},\mathbf{D}|\mathbb{Z},\mathbb{C})$ into four terms in (4)-(7) which are the functions of $\{\pi^u\}$, $\{\mathbf{A}^u\}$, $\{\mathbf{B}^u\}$ and $\mathbf{D}$, respectively. Since $L'_1$ w.r.t. $\{\pi^u\}$, $\{\mathbf{A}^u\}$, $\mathbf{D}$ are concave without any other additional terms, we can obtain estimation equations which are similar to traditional Baum Welch algorithm [1]. We find that only the estimation of $\{\mathbf{B}^u\}$ is influenced by the sparsity term. Furthermore, in Section 4.2, we will show the optimization problem w.r.t. $\{\mathbf{B}^u\}$ is not always concave, which is significantly different from the traditional EM algorithm, requiring non-convex optimization techniques in M-step. Overall, we report the process of the our designed algorithm in Algorithm 1, of which the details are described in Appendix B.

### 4.2 DCP for Optimizing w.r.t. $\{\mathbf{B}^u\}$

For simplicity, let $\bar{\rho}_m = \sum_{\mathbb{S}\in\mathbf{J}^u}\sum_{n=1}^{N}\rho_n(i,m)$, $b_m = b^u_{im}$ and $\mathbf{b} = \{b_m\}$. Following the tradition of optimization, the problem of finding $\mathbf{b}$ for each $i$ and $u$, i.e., (6), can be transformed into such a minimization problem from a maximization problem:

$$\min_{\mathbf{b}} \check{f}(\mathbf{b}) + \hat{f}(\mathbf{b}), \text{ s.t. } \sum_m b_m = 1, \tag{8}$$

where

$$\check{f}(\mathbf{b}) = -\sum_m \bar{\rho}_m \log b_m, \quad \hat{f}(\mathbf{b}) = -\lambda \sum_m b_m \log b_m,$$

$\bar{\rho}_m$ has been estimated in E-step, and $\lambda > 0$ is predefined as the sparsity coefficient.

Proposition 4.2. *There exists* $\lambda > 0$ *such that the* (8) *is a non-convex optimization problem.*

To optimize such a non-convex function of $\mathbf{b}$ with a convergence guarantee, we use difference of convex programming (DCP) [17, 24], which is motivated by Lemma 4.3.

Lemma 4.3. *Equation* (8) *can be decomposed into the addition of a convex term* $\check{f}$ *and a concave term* $\hat{f}$.

DCP is a general and powerful framework for solving non-convex problems, which has been successfully applied in many applications, e.g., learning sparse vectors [21] and low-rank matrices [22]. According to DCP, we need to minimize a sequence of convex upper bound $f^{(t+1)}(b)$ by linearizing the concave term locally, i.e.,

$$f^{(t+1)}(\mathbf{b}) = \check{f}(\mathbf{b}) + (\mathbf{b} - \mathbf{b}^{(t)})^\top \nabla \hat{f}(\mathbf{b}^{(t)}), \text{ s.t. } \sum_m b_m = 1. \quad (9)$$

How to solve (9) efficiently is the key for achieving fast speed of DCP. To meet this goal, we transform (9) into a one-dimensional problem in Proposition 4.4 below.

Proposition 4.4. *There exists an* $\eta$ *such that:*

$$\sum_m -\bar{\rho}_m / \lambda(\log b_m^{(t)} + 1 - \eta) = 1, \quad (10)$$

*and the optimal solution for* (9) *can be obtained from* $\eta$ *by* $b_m^* = -\bar{\rho}_m / \lambda(\log b_m^{(t)} + 1 - \eta)$ *for* $m = 1, \cdots, M$.

Equation (10) is a simple one-dimensional programming problem, which can be efficiently solved, e.g., using bisection method. The whole DCP for solving (8) is summarized in Algorithm 2.

**Algorithm 2** DCP for solving (8).

1: initialize $\mathbf{b}^{(1)}$;
2: **for** $t = 1, \cdots, T$ **do**
3: transform (9) into (10) with current $\mathbf{b}^{(t)}$;
4: obtain $\mathbf{b}^{(t+1)}$ by using bisection method solving (10);
5: **end for**
6: **return** $\mathbf{b}^{(T)}$;

## 5 THEORETICAL ANALYSIS

**Complexity Analysis.** We now analyze the time and space complexities of our proposed model. There are several parameters that determine the computational complexity of our model: the number of top-layer hidden states $K$ for each user, the number of middle-layer hidden states $M$ for all users, and the sum of all sequences lengths $L$. The time complexity for one iteration in Algorithm 1 is then $O\left((K^2 + KM)L\right)$, which is approximately $O(KML)$ due to $K \ll M$. For the space complexity, all the variables involved in Algorithm 1 have to be stored. Among them, $\xi_n(i, j)$ and $\rho_n(i, m)$ are the dominating variables that determine the space complexity. The space complexity is then given by $O\left((K^2 + KM)L\right)$, which is approximately $O(KML)$. Table 1 summarizes the space and time complexity of our model and two main existing models.

**Convergence Analysis.** There are two significant differences between our optimization algorithm and the traditional EM algorithm for HMM training. First, our objective function (1) is not just a log likelihood but with an additional entropy term. To tackle this issue, we utilize the Jensen's inequality [12] to derive the lower bound.

**Table 1: The time and space complexity of our model and two previous models. For *Gmove* [25], $G$ is the number of user groups, and $I$ is the number of total iterations for the HMM training in the inner loop.**

| Method | Time / iteration | Space |
|---|---|---|
| ***HMM*** [6] | $O(M^2L)$ | $O(M^2L)$ |
| ***Gmove*** [25] | $O(GM^2LI)$ | $O(GM^2LI)$ |
| ***S3HMM*** | $O(KML)$ | $O(KML)$ |

Second, as optimizing the lower bound w.r.t. $\{\mathbf{B}^u\}$ is a non-convex in the M-step, we use DCP in the inner loop and use $\{\mathbf{B}^u_{old}\}$ as initial points to ensure the lower bound is non-decreasing. We prove that such treatments guarantee the convergence of our optimization procedure.

Theorem 5.1. *The objective value* $L(\mathbf{\Phi}, \mathbf{D})$ *produced by Algorithm 1 is non-decreasing, and converges to a limit point.*

Proof. First, we prove that the overall objective function is non-decreasing during the iterations. We use $h(\pi)$, $h(\mathbf{A})$, $h(\mathbf{B})$, $h(\mathbf{D})$ to denote the four terms (4)-(7), respectively. Because $h(\pi)$, $h(\mathbf{A})$ and $h(\mathbf{D})$ are convex, we can easily get the global maximum and we have $h(\pi) \geq h(\pi_{old})$, $h(\mathbf{A}) \geq h(\mathbf{A}_{old})$ and $h(\mathbf{D}) \geq h(\mathbf{D}_{old})$. However, $h(\mathbf{B})$ is non-convex, to avoid that converge to another local maximum, we use $\mathbf{B}_{old}$ as the initial point to iterate by DCP. Since DCP has convergence guarantee[17, 24], we have $h(\mathbf{B}) \geq h(\mathbf{B}_{old})$. Based on Lemma 4.1 and above analysis, we get:

$$\begin{aligned} L(\mathbf{\Phi}, \mathbf{D}) \geq & h(\pi) + h(\mathbf{A}) + h(\mathbf{B}) + h(\mathbf{D}) + L_2' \\ \geq & h(\pi_{old}) + h(\mathbf{A}_{old}) + h(\mathbf{B}_{old}) + h(\mathbf{D}_{old}) + L_2' \\ = & L(\mathbf{\Phi}_{old}, \mathbf{D}_{old}). \end{aligned}$$

Thus, the sequence of $L(\mathbf{\Phi}, \mathbf{D})$ is non-decreasing.

Then, to prove that $L(\mathbf{\Phi}, \mathbf{D})$ is limited, we only need to prove that all (2) and (3) have upper bounds. Since (2) is the opposite of entropy, (2) is no more than 0. The specific form of (3) is given by,

$$p(\mathbb{S}|\mathbf{\Phi}^u, \mathbf{D}) = \sum_{\mathbb{Z}, \mathbb{C}} \pi^u_{z_1} \cdot \prod_{n=1}^{N-1} a^u_{z_n, z_{n+1}} \cdot \prod_{n=1}^{N} b^u_{z_n, c_n} \cdot \prod_{n=1}^{N} p(e_n | d_{c_n})$$

Since all four types of terms, $\pi^u_{z_1}$, $a^u_{z_n, z_{n+1}}$, $b^u_{z_n, c_n}$ and $p(e_n|d_{c_n})$, are limited, after a limited number of addition and multiplication operations on these terms, the result $p(\mathbb{S}|\mathbf{\Phi}^u, \mathbf{D})$ is still limited. Thus, the overall objective function $L(\mathbf{\Phi}, \mathbf{D})$ is also limited. □

## 6 EVALUATION

### 6.1 Experiments Setup

*6.1.1 Datasets.* We evaluate our model with three datasets from different domains. The details of the three datasets are described as follows:

***Twitter***: This is a public dataset [25] collected on *Twitter* from Aug. 1st to Nov. 30th, 2014 in Los Angeles, which consists million-scale geo-tagged tweets. We extract the effective transitions such that the time gap is less than 6 hours. After preprocessing, there are approximately 8 thousand users and 30 thousand effective transitions. We use this dataset for personalized mobility modeling and location prediction.

***Wechat***: This is a dense trajectory dataset [7] collected from about 5 thousand active Wechat users from Sept. 17th to Oct. 31st, 2016. Being collected on Wechat's map platform, each trajectory is sampled at a five-minute rate and reflects the user's real-life mobility in the physical world. After preprocessing, we obtain 0.16 million clean transitions. We again use this dense trajectory dataset for personalized mobility modeling and location prediction.

***Lastfm*** : This is a public dataset collected from Last.FM [2] from Jan. 1st to Apr. 1st, 2008, which consists 1 thousand users' music listening records. Different from the previous datasets, the observations in this dataset is discrete one-hot feature vectors. After preprocessing, there are approximately 30 thousand effective transitions. We use this dataset for modeling personalized music listening behaviors. For quantitative evaluation, we use the prediction of next artist the user will listen to as the evaluation task.

*6.1.2 Compared Methods.* We compare S3HMM with the following baselines:

- ***HMM*** [6] is the classical HMM that trains one unified HMM for all the sequences. The result HMM is used for sequential prediction for all the users.
- ***Gmove*** [25] is a group-level hidden Markov model. It jointly clusters users into groups and learns an HMM for each user group. At prediction time, it softly infers the group membership of the target users and aggregates the predictions from the user's belonging groups.
- ***Pipeline*** [11] is a simple baseline for learning personalized sequential models. It first clusters all the observations to form high-level states. Then for each user, it selects $K$ states that correlate with the user's observations most and infers a transition matrix for the selected $K$ states. In our implementation, we use k-means to cluster the observations (either user locations or one-hot feature vectors).

Note that our baselines are the classic HMM and its variants. While there are other models (e.g., recurrent neural networks [7, 20]) designed specifically for location prediction and sequential recommendation on our datasets, we do not compare with them, because we do not claim S3HMM will achieve state-of-the-art performance on these specific tasks. Rather, we consider S3HMM as a general-to-use HMM variant that can better uncover personalized sequential patterns and improve downstream prediction tasks. In addition to the above baselines, we also study several variants of our S3HMM model to verify its components:

- ***S3HMM***-$\infty$ is variant of S3HMM that has hard assignment of personal states over shared states. When inferring the distributions personal states over shared states, it chooses the shared state that has the largest probability and sets its probability entry to 1. It is equivalent to our model when $\lambda \to +\infty$.
- ***S3HMM***-0: It is also a special case of our proposed model when $\lambda = 0$. In other words, this model imposes no sparsity constraints.
- ***S3HMM***-$\lambda$: It is our proposed S3HMM model in which $\lambda$ is the parameter controlling the strength of the sparsity constraint.

There are three important parameters in S3HMM: (1) the number of shared latent states $M$, (2) the number of personal latent states $K$, and (3) the strength of the sparsity constraint $\lambda$. By default, we set $K = 5$ for all dataset, $M = 500$ for *Wechat* and *Lastfm* and $M = 1000$ for *Twitter*, and $\lambda = 50$. We study the effects of these parameters later in Section 6.6. The parameters of baseline methods are tuned to achieve the best performance using a validation dataset.

## 6.2 Qualitative Studies

In this subsection, we investigate the learned S3HMM model qualitatively to verify its effectiveness in learning personalized sequential models. Below, we first show how S3HMM learns state-sharing and sparse personalized models, then perform a set of case studies to examine these personalized models. Due to the space limit, we only show the results on the *Wechat* dataset, but similar phenomena are also observed on *Twitter* and Music.

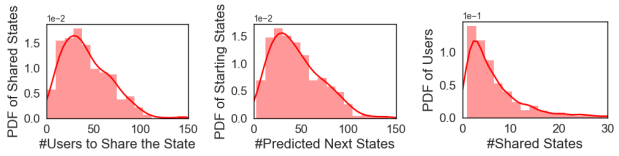
**State-sharing and sparse properties of S3HMM.** We train S3HMM on *Wechat* and analyze the learned model to verify our state-sharing and sparse model design. Figure 3 shows three sets of qualitative analysis of our model.

We first analyze whether the underlying latent states are indeed shared by different users. For this purpose, we count the number of the users that distribute on each underlying state with probability more than $1/M$. Figure 3(a) shows the histogram of the underlying states w.r.t. the number of users sharing a common state. As shown, we observe that more than 99% underlying states are shared by at least 10 users, and most underlying states are shared by $10 - 70$ users. This verifies S3HMM can effectively uncover the shared latent states. It is because of such parameter sharing that allows S3HMM to jointly learn with all the user sequences to address data scarcity.

We then examine whether different users have diverse transition patterns. Starting from the same underlying state, we compute the next state that users are most likely to visit based on their transition matrices. Figure 3(b) plots the histogram of the number of the states that users may visit. As shown, with the same starting state, the users can have totally different transition patterns: there are often $10 - 60$ possibilities that different users may visit. This phenomenon shows the necessity of learning personalized transitions and the effectiveness of S3HMM in doing this.

Finally, we examine whether S3HMM produces sparse models for different users. This is done by counting how many underlying shared states each user distributes on. Figure 3(c) shows the results. We can see the number of shared states that each user focuses on is mostly no more than 10, which verifies effectiveness of the sparsity constraint of S3HMM.

**Case Study.** We proceed to study the personalized sequential models learned for two different users in the *Wechat* dataset. Figure 4 plots their shared latent states and transition patterns (i.e., the transition matrix $\mathbf{A}$). Note that, each axis in the heatmap should be the index of the top-layer hidden state, because we add sparsity constraint, each top-layer state mainly focuses on one middle-layer hidden state. For the ease of visualization, we index the axis in the heatmap with middle-layer hidden states, thus each entry in the heatmap has direct physical mapping to the hotspots in the left figure.




(a) #Users. (b) #Predicted Next States. (c) #Shared States.

**Figure 3: Analysis of the learned S3HMM model learned from the *Wechat* dataset: (a) most underlying states are shared by at least ten users; (b) different users have different transition patterns and may visit a diverse set of next states from the same starting state; (c) each user focuses on only a small number of latent states.**

One can see from Figure 4 that S3HMM successfully captures different transition patterns for different users and the transitions makes sense intuitively. For user A, there are five highly correlated states: a residential area (Huilongguan), a university (Peking University), a tech park (Zhongguancun), a bus station (Sihui) and a remote village. She is likely to move from her university to the tech part, and from the tech park to her residential area. This is possibly because she is a university student interning at the tech park. User B's transition patterns are simpler: she is routinely commuting between her home and the working place. Finally, we see that these two users share common underlying states (e.g., the tech park) but have totally different transition patterns, which verifies the effectiveness of S3HMM. In contrast, if we train one unified HMM for the users, such data conflicts will make the result model hard to interpret and less effective for predictive tasks.

## 6.3 Performance Comparison

In this subsection, we evaluate the performance of S3HMM for sequential prediction tasks. Given a user sequence $\mathbb{S} = e_1 e_2 \cdots e_N$, the task aims at predicting $e_N$ based on its preceding sequence $e_1 e_2 \cdots e_{N-1}$. To evaluate the performance of different sequential models, we check whether the ground truth $e_N$ appears in the top-$k$ rank list generated by a sequential model. We compute the accuracy as $k$ varies. On the *Wechat* and *Twitter* datasets, we predict the next location a user is likely to visit following the setup in [25]; on the *Lastfm* dataset, we predict the artist the user is likely to listen to given her preceding music listening behaviors. For all the sequences, we use first 70% for training, the subsequent 20% for validation, and the rest for testing.

*6.3.1 Performance Comparison.* Figure 5 shows the sequential prediction performance of different models on the three datasets. As shown, S3HMM consistently outperforms all the baselines on the datasets: (1) On the *Twitter* dataset, S3HMM outperforms the strongest baseline *Gmove* by large margins, improving the absolute accuracy by up to 20%; (2) On the *Wechat* dataset, *Pipeline* is the best-performing baseline, yet S3HMM improves its absolute accuracy by 19.5%, 30.4%, 34.6%, 36.3%, 36.6% as $k$ varies; (3) Similar significant improvements can be observed on the *Lastfm* dataset as well, where S3HMM outperforms the baselines by more than 20% in absolute accuracy.

Comparing the performance of different variants of S3HMM (*S3HMM-λ*, *S3HMM-0*, and *S3HMM–∞*), we can see *S3HMM-λ* is consistently the best. This empirically verifies that imposing the sparsity constraint can improve the performance of the model, probably because it serves as a regularization term that better learn the model under data scarcity. In contrast, *S3HMM–∞* is too rigid and suffers from accuracy loss, while *S3HMM-0* does not enjoy the benefits of sparsity regularization. A side benefit of the sparsity constraint is that it makes the result model easily interpretable, as shown in Section 6.2.

*6.3.2 Performance on Different Users Groups.* In this set of experiments, we study the performance of our model on different user groups with the *Wechat* dataset. We partition all the users into different groups based on three criteria and study how the performance of S3HMM varies: (1) the number of transitions for training; (2) the total number of visited places; (3) the entropy of user trajectory[5]. Investigating the performance on different user groups can help understand the robustness of the model. Figure 6 shows the performance of our model and the classic *HMM*. Across all the different user groups, our method stably outperforms *HMM* with more than 10% improvements in absolute accuracy.

## 6.4 Convergence

In this set of experiments, we verify that our method converges empirically. Figure 7 shows the convergence plots on the three datasets. We can observe that on all the three datasets, the objective function $L(\mathbf{\Phi}, \mathbf{D})$ monotonously increases and gradually converges after a small number of iterations.

## 6.5 Running time comparison

Table 2 reports the training time of different models on the three datasets. As expected, the classical HMM is the fastest in terms of its training time, but comes with the cost of poor predictive accuracy. The running time of *S3HMM-∞*, *S3HMM-0* and *S3HMM-λ* are similar, and are all much faster than the *Gmove* model.

**Table 2: Time costs of different models.**

| Methods | *HMM* | *Gmove* | *Pipeline* | *S3HMM-∞* | *S3HMM-0* | *S3HMM-λ* |
|---|---|---|---|---|---|---|
| ***Twitter*** | 90s | 3758s | 578s | 2615s | 2383s | 2398s |
| ***Wechat*** | 293s | 17757s | 2122s | 16253s | 9424s | 9450s |
| ***Lastfm*** | 474s | 39039s | 2366s | 36489s | 34290s | 34258s |

## 6.6 Effects of parameters

In this subsection, we study the effects of different parameters on the performance of S3HMM. There are three key parameters: (1) the number of shared hidden states $M$, (2) the number of personal states $K$, and (3) the strength of sparsity $\lambda$. For each parameter, we study the performance of our model when that parameter varies, while the other two parameters are fixed at their default values. Due to the space limit, we only report the results about parameters on the *Wechat* dataset.

**Effect of $M$.** Figure 8(a) shows the effect of $M$ on the performance of S3HMM. We observe that when $M$ is small, the performance of S3HMM improves significantly with $M$ and then gradually becomes stable. This is expected, as more shared states provide finer granularity in characterizing the observation space, but such an effect

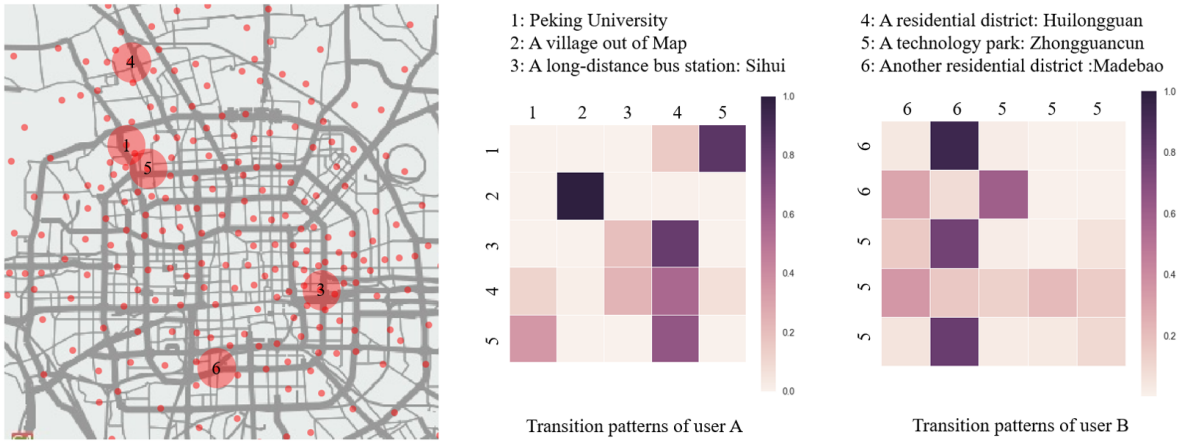
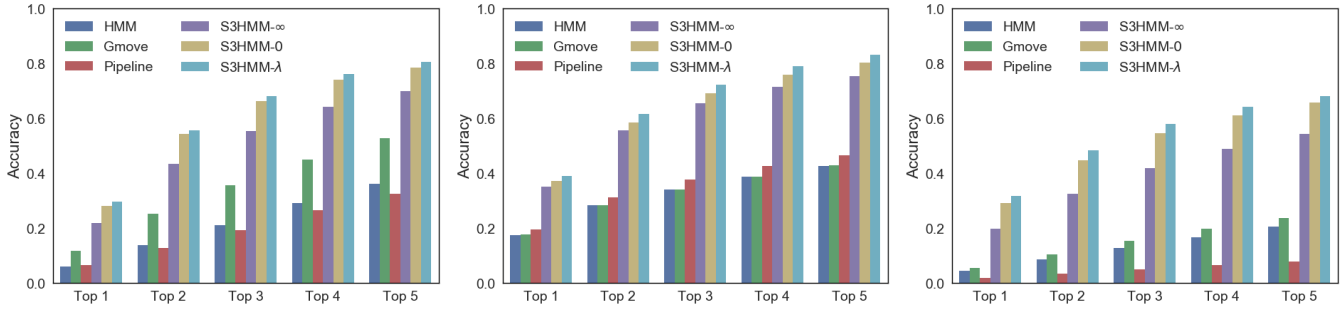
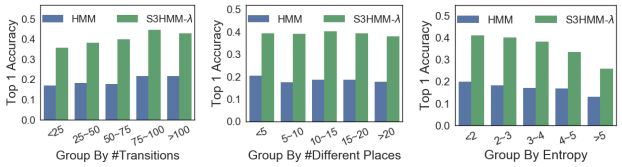



**Figure 4: Case study for two users on the *Wechat* dataset: we show the shared underlying latent states with red circles in the left, and show the transition patterns of the two users in the right heat map (darker color represents higher probability).**



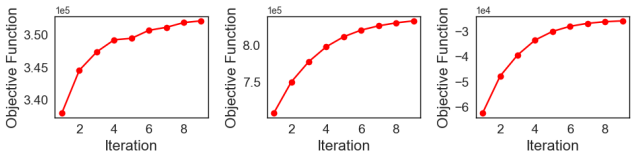

(a) *Twitter.* (b) *Wechat.* (c) *Lastfm.*

**Figure 5: The accuracies of top-$k$ prediction of different sequential models.**



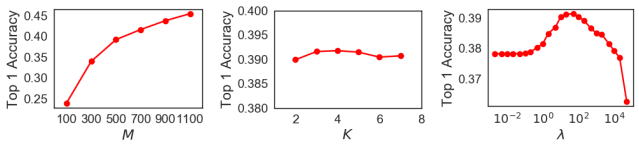

(a) #Transitions. (b) #Different Places. (c) Trajectory Entropy.

**Figure 6: Prediction accuracies of different user groups.**



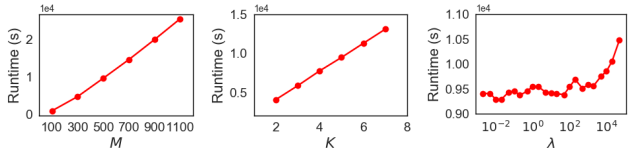

(a) *Twitter.* (b) *Wechat.* (c) *Lastfm.*

**Figure 7: The convergence in terms of $L(\Phi, \mathrm{D})$ for *S3HMM-λ*.**




(a) Accuracy w.r.t. $M$. (b) Accuracy w.r.t. $K$. (c) Accuracy w.r.t. $\lambda$.

**Figure 8: Top-1 prediction accuracy when varying different parameters.**




(a) Runtime w.r.t. $M$. (b) Runtime w.r.t. $K$. (c) Runtime w.r.t. $\lambda$.

**Figure 9: Time cost when varying different parameters.**

gradually saturates when $M$ is large enough. Figure 9(a) shows the running time of S3HMM, from which we can see S3HMM scales linearly with $M$.

**Effect of $K$.** Figure 8(b) shows the effectiveness of S3HMM when $K$ varies. The performance of S3HMM is rather stable when $K$ varies from 2 to 7. This indicates that two personal states is often enough to capture the transition patterns for most Wechat users. Figure 9(b) shows the training time of S3HMM as $K$ increases. Similar to $M$, S3HMM scales linearly with $K$.

**Effect of $\lambda$.** Figure 8(c) shows the effect of $\lambda$ on the performance of S3HMM. When $\lambda < 1$, there is little change in the prediction accuracy as we vary $\lambda$. However, when $\lambda > 1$, the accuracy first increase before it reaches its maximum at the point of $\lambda = 50$, then deteriorates after $\lambda > 50$. From Figure 9(c), we can observe that the running time slightly increases with $\lambda$. This is because a

larger $\lambda$ requires more iterations in the DCP procedure for sparse estimation.

## 7 RELATED WORK

Hidden Markov model (HMM) is a powerful model for modeling sequences because it can both discover the intrinsic underlying distributions and learn the transition patterns among the distributions [9, 11, 15, 19, 25]. However, most existing studies on HMMs for sequence modeling focus on learning one HMM for a long sequence or a collection of sequences, thus not able to learn personalized HMMs. Gmove [25] learns "semi-personalized" HMMs from a collection of user sequences. It jointly performs user clustering and group-level HMM training, thus clustering users with similar transition patterns into the same group and learning an HMM for that group. However, such a group-level HMM training is still too rigid for obtaining user-level personalized models. SSHMM [19] learns an HMM with one-layer hidden states sharing underlying distributions, which partly solves the problem of data scarcity. However, its structure of one-layer hidden state leads to a very large transition matrix for each user and lack of interpretability of personalized states. In contrast, in our S3HMM model, we design a two-layer hidden state model, allowing for fully capturing individual-level patterns without suffering from data scarcity with high efficiency and interpretability.

Sequential pattern mining methods have been studied extensively [3, 13] , aiming to discover frequent sub-sequences from a sequence database. While our model also discovers sequential patterns from input sequences, we adopt a statistical approach rather than rule-based pattern mining. Furthermore, none of these pattern mining methods can uncover personalized sequential patterns.

Recurrent neural networks have been popular in recent studies for various sequential prediction tasks, such as location prediction [7, 10] and basket recommendation [23]. Our work is orthogonal to them, as our model aims at descriptive modeling instead of predictive modeling. Instead of designing an optimized model for a specific prediction task, we aim to develop a general-purpose tool for uncovering personalized sequential patterns. Our developed model is general-to-use for various sequence modeling tasks and is highly interpretable.

## 8 CONCLUSION

We proposed an extension of hidden Markov model for modeling personalized sequences. Our model S3HMM addressed the challenge of learning personalized hidden Markov models under severe data scarcity. This is achieved by assuming a pool of latent states shared by all the users, but meanwhile allow users to maintain a small number of personal latent states. The personal states distribute over the shared states with sparsity constraints, thus allowing for learning reliable personal models with limited data. We have developed a new optimization algorithm for estimating the parameters of the model and proved its convergence theoretically. Through experiments on three real-life datasets, we found S3HMM can successfully uncover personalized and interpretable sequential patterns, and significantly outperforms baseline models for downstream predictive tasks. S3HMM can serve as a general-to-use model for modeling personalized sequential behaviors under data scarcity in various domains. An interesting future work is to extend the idea of S3HMM to deep neural networks, for building personalized predictive models under data scarcity.

## ACKNOWLEDGMENTS

This work was supported in part by the National Nature Science Foundation of China under 61861136003, 61621091 and 61673237, Beijing National Research Center for Information Science and Technology under 20031887521, and research fund of Tsinghua University - Tencent Joint Laboratory for Internet Innovation Technology.

## REFERENCES

[1] Leonard E Baum, Ted Petrie, George Soules, and Norman Weiss. 1970. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The annals of mathematical statistics* (1970).
[2] Òscar Celma Herrada. 2009. Music recommendation and discovery in the long tail. (2009).
[3] Meng Chen, Xiaohui Yu, and Yang Liu. 2015. Mining moving patterns for predicting next location. *Information Systems* (2015).
[4] Zhiyong Cheng, Jialie Shen, Lei Zhu, Mohan S Kankanhalli, and Liqiang Nie. 2017. Exploiting Music Play Sequence for Music Recommendation.. In *IJCAI*.
[5] Justin Cranshaw, Eran Toch, Jason Hong, Aniket Kittur, and Norman Sadeh. 2010. Bridging the gap between physical location and online social networks. In *Ubicomp*.
[6] Budhaditya Deb and Prithwish Basu. 2015. Discovering latent semantic structure in human mobility traces. In *EWSN*.
[7] Jie Feng, Yong Li, Chao Zhang, Funing Sun, Fanchao Meng, Ang Guo, and Depeng Jin. 2018. DeepMove: Predicting Human Mobility with Attentional Recurrent Networks. In *WWW*.
[8] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *KDD*.
[9] Tao Li, Minsoo Choi, Kaiming Fu, and Lei Lin. 2018. Music sequence prediction with mixture hidden markov models. *arXiv preprint arXiv:1809.00842* (2018).
[10] Dongliang Liao, Weiqing Liu, Yuan Zhong, Jing Li, and Guowei Wang. 2018. Predicting Activity and Location with Multi-task Context Aware Recurrent Neural Network. In *IJCAI*.
[11] Wesley Mathew, Ruben Raposo, and Bruno Martins. 2012. Predicting future locations with hidden Markov models. In *Ubicomp*.
[12] Thomas Minka. 1998. Expectation-Maximization as lower bound maximization.
[13] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. 2009. Wherenext: a location predictor on trajectory pattern mining. In *KDD*.
[14] Nasser M Nasrabadi. 2007. Pattern recognition and machine learning. *Journal of electronic imaging* 16, 4 (2007), 049901.
[15] Hongzhi Shi, Hancheng Cao, Xiangxin Zhou, Yong Li, Chao Zhang, Vassilis Kostakos, Funing Sun, and Fanchao Meng. 2019. Semantics-Aware Hidden Markov Model for Human Mobility. In *SDM*.
[16] Hongzhi Shi and Yong Li. 2018. Discovering Periodic Patterns for Large Scale Mobile Traffic Data: Method and Applications. *IEEE Transactions on Mobile Computing* 17, 10 (2018), 2266–2278.
[17] Pham Dinh Tao et al. 2005. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Annals of operations research* (2005).
[18] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to Estimate the Travel Time. In *KDD*.
[19] Tong Xia, Yue Yu, Fengli Xu, Funing Sun, Diansheng Guo, and Yong Li. 2019. Understanding Urban Dynamics by State-sharing Hidden Markov Model. In *WWW*.
[20] Di Yao, Chao Zhang, Jianhui Huang, and Jingping Bi. 2017. SERM: A recurrent model for next location prediction in semantic trajectories. In *CIKM*.
[21] Quanming Yao and James T Kwok. 2017. Efficient learning with a family of nonconvex regularizers by redistributing nonconvexity. *Journal of Machine Learning Research* (2017).
[22] Quanming Yao, James T Kwok, Taifeng Wang, and Tie-Yan Liu. 2018. Large-Scale Low-Rank Matrix Learning with Nonconvex Regularizers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2018).
[23] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *SIGIR*.
[24] Alan L Yuille and Anand Rangarajan. 2003. The concave-convex procedure. *Neural computation* (2003).
[25] Chao Zhang, Keyang Zhang, Quan Yuan, Luming Zhang, Tim Hanratty, and Jiawei Han. 2016. Gmove: Group-level mobility modeling using geo-tagged social media. In *KDD*.

# A PROOF

## A.1 Lemma 4.1

Proof. By utilizing latent variables $\mathbb{Z}$ and $\mathbb{C}$, we find a lower bound of the first term in $L(\mathbf{\Phi}, \mathbf{D})$, i.e., (1) based on Jensen's inequality [12]. For simplicity, let $\bar{p}(\mathbb{Z}, \mathbb{C}) = p\left(\mathbb{Z}, \mathbb{C} | \mathbb{S}, \mathbf{\Phi}^u_{old}, \mathbf{D}_{old}\right)$, then we have:

$$\begin{aligned}
&\sum_u \sum_{\mathbb{S} \in \mathbf{J}^u} \log\left(p(\mathbb{S}|\mathbf{\Phi}^u, \mathbf{D})\right) \\
&= \sum_u \sum_{\mathbb{S} \in \mathbf{J}^u} \log\left(\sum_{\mathbb{Z},\mathbb{C}} p(\mathbb{S}, \mathbb{Z}, \mathbb{C}|\mathbf{\Phi}^u, \mathbf{D})\right) \\
&= \sum_u \sum_{\mathbb{S} \in \mathbf{J}^u} \log\left(\sum_{\mathbb{Z},\mathbb{C}} \bar{p}(\mathbb{Z}, \mathbb{C}) \frac{p(\mathbb{S}, \mathbb{Z}, \mathbb{C}|\mathbf{\Phi}^u, \mathbf{D})}{\bar{p}(\mathbb{Z}, \mathbb{C})}\right) \\
&\geq \sum_u \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{\mathbb{Z},\mathbb{C}} \bar{p}(\mathbb{Z}, \mathbb{C}) \log \frac{p(\mathbb{S}, \mathbb{Z}, \mathbb{C}|\mathbf{\Phi}^u, \mathbf{D})}{\bar{p}(\mathbb{Z}, \mathbb{C})} \\
&= \sum_u \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{\mathbb{Z},\mathbb{C}} \bar{p}(\mathbb{Z}, \mathbb{C}) \log p(\mathbb{S}, \mathbb{Z}, \mathbb{C}|\mathbf{\Phi}^u, \mathbf{D}) \\
&\quad - \sum_u \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{\mathbb{Z},\mathbb{C}} \bar{p}(\mathbb{Z}, \mathbb{C}) \log \bar{p}(\mathbb{Z}, \mathbb{C}).
\end{aligned}$$

Also, $L(\mathbf{\Phi}, \mathbf{D})$ is defined as,

$$L(\mathbf{\Phi}, \mathbf{D}) = \sum_u \sum_{\mathbb{S} \in \mathbf{J}^u} \log(p(\mathbb{S}|\mathbf{\Phi}^{\mathbf{u}}, \mathbf{D})) + \sum_u \lambda g(\mathbf{B}^u).$$

Thus, $L(\mathbf{\Phi}, \mathbf{D})$ can be given by: $L(\mathbf{\Phi}, \mathbf{D}) \geq L'_1(\mathbf{\Phi}, \mathbf{D}) - L'_2$ where

$$\begin{aligned}
L'_1(\mathbf{\Phi}, \mathbf{D}) &= \sum_u \left( \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{\mathbb{Z},\mathbb{C}} \bar{p}(\mathbb{Z}, \mathbb{C}) \log p(\mathbb{S}, \mathbb{Z}, \mathbb{C}|\mathbf{\Phi}^{\mathbf{u}}, \mathbf{D}) + \lambda g(\mathbf{B}^u) \right), \\
L'_2 &= \sum_u \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{\mathbb{Z},\mathbb{C}} \bar{p}(\mathbb{Z}, \mathbb{C}) \log \bar{p}(\mathbb{Z}, \mathbb{C}),
\end{aligned}$$

where $L'_2$ is independent w.r.t. $\mathbf{\Phi}$ and $\mathbf{D}$. □

## A.2 Proposition 4.2

Proof. The second-order derivative of $f(\mathbf{b})$ w.r.t. $\mathbf{b}$ is given by,

$$f''(\mathbf{b}) = \begin{bmatrix} \bar{\rho}_1/b_1^2 - \lambda/b_1 & 0 & \cdots & 0 \\ 0 & \bar{\rho}_2/b_2^2 - \lambda/b_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \bar{\rho}_M/b_M^2 - \lambda/b_M \end{bmatrix}.$$

We can observe that if there exists $m$ such that $\lambda > \bar{\rho}_m/b_m$, we will obtain $\bar{\rho}_m/b_m^2 - \lambda/b_m < 0$, i.e., the matrix is not positive semi-definite. So the problem of minimizing $f''(\mathbf{b})$ can be a non-convex problem. □

## A.3 Lemma 4.3

Proof. We can split the objective function of $\mathbf{b}$ into two terms given by,

$$f(\mathbf{b}) = \check{f}(\mathbf{b}) + \hat{f}(\mathbf{b}),$$

where $\check{f}(\mathbf{b}) = -\sum_m \bar{\rho}_m \log b_m$ and $\hat{f}(\mathbf{b}) = -\lambda \sum_m b_m \log b_m$. The second-order derivative of $\check{f}(\mathbf{b})$ w.r.t. $\mathbf{b}$ is given by,

$$\check{f}''(\mathbf{b}) = \begin{bmatrix} \bar{\rho}_1/b_1^2 & 0 & \cdots & 0 \\ 0 & \bar{\rho}_2/b_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \bar{\rho}_M/b_M^2 \end{bmatrix}.$$

We find that $\check{f}''(\mathbf{b})$ is a diagonal matrix. Since $\bar{\rho}_m > 0$ and $b_m > 0$, we can get $\bar{\rho}_1/b_1^2 > 0$ for $m = 1, 2, \cdots, M$, i.e., all the diagonal entries are larger than 0. So $\check{f}''(\mathbf{b})$ is a positive definite matrix, and $\check{f}(\mathbf{b})$ is convex. Similarly, the second-order derivative of $\hat{f}(\mathbf{b})$ w.r.t. $\mathbf{b}$ is given by,

$$\hat{f}''(\mathbf{b}) = \begin{bmatrix} -\lambda/b_1 & 0 & \cdots & 0 \\ 0 & -\lambda/b_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & -\lambda/b_M \end{bmatrix}.$$

Since the predefined coefficient $\lambda > 0$ and $b_m > 0$ for $m = 1, 2, \cdots, M$, all the diagonal entries are less than 0. Thus, $\hat{f}''(\mathbf{b})$ is negative definite, and $\hat{f}(\mathbf{b})$ is concave. □

## A.4 Proposition 4.4

Proof. The convex upper bound to minimize is given by,

$$\begin{aligned}
f^{(t+1)}(\mathbf{b}) &= \check{f}(\mathbf{b}) + (\mathbf{b} - \mathbf{b}^{(t)})^\top \nabla \hat{f}(\mathbf{b}^{(t)}) \\
&= -\sum_m \bar{\rho}_m \log b_m - \lambda \sum_m b_m (\log b_m^{(t)} + 1) \\
\text{s.t.} \ &\sum_m b_m = 1.
\end{aligned}$$

We utilized Lagrange multiplier to transform such a constrained optimization problem into an unconstrained optimization problem given by,

$$\min_{\mathbf{b}, \eta} -\sum_m \bar{\rho}_m \log b_m - \lambda \sum_m b_m (\log b_m^{(t)} + 1) - \eta \left(\sum_m b_m - 1\right), \quad (11)$$

By letting the derivative of the objective in (11) w.r.t. $b_m$ be zero, we can obtain,

$$b_m = -\bar{\rho}_m / \lambda (\log b_m^{(t)} + 1 - \eta), \quad (12)$$

for $m = 1, \cdots, M$. We replace $b_m$ in $\sum_m b_m = 1$ by (12) and obtain

$$\sum_m -\bar{\rho}_m / \lambda (\log b_m^{(t)} + 1 - \eta) = 1.$$

Therefore, we transform the $M$-dimensional problem into a one-dimensional problem. Since $\bar{\rho}_k > 0$ and $\lambda > 0$, when $\eta > \log b_k^{(t)} + 1$ for $k = 1, 2, \cdots, M$, $\sum_m -\bar{\rho}_m / \lambda (\log b_m^{(t)} + 1 - \eta)$ monotonously decreases with $\eta$. Since

$$\sum_m -\bar{\rho}_m / \lambda (\log b_m^{(t)} + 1 - \eta) < \sum_m -\bar{\rho}_m / \lambda (\log \max b_m^{(t)} + 1 - \eta) = 1,$$

we set the higher bound as,

$$\eta < \sum_{m=1}^{M} \bar{\rho}_m / \lambda + \log(\max b_m) + 1.$$

We denote that

$$m_{max} = \arg\max_m b_m,$$

so that

$$\sum_m -\bar{\rho}_m/\lambda(\log b_m^{(t)} + 1 - \eta) > -\bar{\rho}_{m_{max}}/\lambda(\log b_{m_{max}}^{(t)} + 1 - \eta) = 1.$$

We set the lower bound as

$$\eta > \bar{\rho}_{m_{max}}/\lambda + \log(b_{m_{max}}) + 1.$$

Thus, there is a unique solution in this interval. So, we can solve the equation of $\eta$ and then get $\mathbf{b}^{(t+1)}$ by bisection method. □

# B THE DETAILED EM ALGORITHM

We alternatively conduct E-step and M-step until the objective function $L(\mathbf{\Phi}, \mathbf{D})$ converges as shown in algorithm 1. We introduce how to estimate the posterior probabilities in E-step and how to infer parameters in M-step in detail as follows.

## B.1 E-Step

For each sequence $\mathbb{S}$, we first define three auxiliary probabilities to calculate:

$$\xi_n(i,j) = p(z_{n+1} = j, z_n = i|\mathbb{S}, \mathbf{\Phi}^u, \mathbf{D}),$$

where $n = 1, 2, \cdots, N-1$, and

$$\gamma_n(i) = p(z_n = i|\mathbb{S}, \mathbf{\Phi}^u, \mathbf{D}),$$

$$\rho_n(i,m) = p(z_n = i, c_n = m|\mathbb{S}, \mathbf{\Phi}^u, \mathbf{D}),$$

where $n = 1, 2, \cdots, N$. We use a forward-backward procedure to calculate these three probabilities. The forward probability $\alpha_n(i)$ is defined as

$$\alpha_n(i) = p(e_1, e_2, \cdots, e_n, z_n = i|\mathbf{\Phi}^u, \mathbf{D}).$$

The initial values are

$$\alpha_1(i) = \pi_i \sum_{m=1}^{M} b_{im}^u p(e_1|d_m).$$

Then $\alpha_{n+1}(j)$ can be calculated as follows,

$$\alpha_{n+1}(j) = \sum_{i=1}^{K} (\alpha_n(i) a_{ij}^u \sum_{m=1}^{M} b_{jm} p(e_{n+1}|d_m)).$$

The backward probability is defined as

$$\beta_n(i) = p(e_{n+1}, e_{n+2}, \cdots, e_N|z_n = i, \mathbf{\Phi}^u, \mathbf{D}).$$

The initial values are given by $\beta_N(i) = 1$. Then, $\beta_n(i)$ can be calculated by $\beta_{n+1}(j)$

$$\beta_n(i) = \sum_{j=1}^{K} \left( a_{ij}^u \sum_{m=1}^{M} b_{jm}^u p(e_{n+1}|d_m) \right) \beta_{n+1}(j).$$

Based on $\alpha_n(i)$ and $\beta_n(j)$, $\xi_n(i,j)$ can be calculated as

$$\xi_n(i,j) = \frac{\alpha_n(i) a_{ij}^u \left( \sum_{m=1}^{M} b_{jm}^u p(e_{n+1}|d_m) \right) \beta_{n+1}(j)}{\sum_{k=1}^{K} \sum_{l=1}^{K} \alpha_n(k) a_{kl}^u \left( \sum_{m=1}^{M} b_{jm}^u p(e_{n+1}|d_m) \right) \beta_{n+1}(l)}.$$

Finally, $\gamma_n(i)$ and $\rho_n(i,m)$ can be calculated as:

$$\gamma_n(i) = \frac{\alpha_n(i)\beta_n(i)}{\sum_{j=1}^{K} \alpha_n(j)\beta_n(j)},$$

$$\rho_n(i,m) = \frac{\gamma_n(i) b_{im}^u p(e_{n+1}|d_m)}{\sum_{l=1}^{M} b_{il}^u p(e_{n+1}|d_l)}.$$

## B.2 M-Step

Based on $\xi_n(i,j)$, $\gamma_n(i)$ and $\rho_n(i,m)$, the parameters of HMM (except for the parameters of the underlying distributions) can be updated by the following formulas,

$$\pi_i^u = \sum_{\mathbb{S} \in \mathbf{J}^u} \gamma_1(i),$$

$$a_{ij}^u = \frac{\sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{n=1}^{N-1} \xi_n(i,j)}{\sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{n=1}^{N-1} \gamma_n(i)}.$$

If there is no sparsity constraint ($\lambda = 0$), then we can calculate $b_{im}^u$ as,

$$b_{im}^u = \frac{\sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{n=1}^{N} \rho_n(i,m)}{\sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{n=1}^{N} \gamma_n(i)}.$$

If the assignment is hard ($\lambda \to +\infty$), then $b_{im}^u$ can be calculated as below,

$$b_{im}^u = \begin{cases} 1, & \text{if } m = argmax_{m'} \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{n=1}^{N} \rho_n(i,m'), \\ 0, & \text{otherwise.} \end{cases}$$

Otherwise, we need to conduct an non-convex optimization process to estimate $b_{im}^u$ as described in Section 4.2.

The updating formulas of the parameters of the underlying distributions are divided into two cases as follows,

(1) For sequences with underlying Gaussian distributions, $e_n$ is a continuous vector (e.g., for location sequences, $e_n = (l_o, l_a)$, where $l_o$ is the longitude and $l_a$ is the latitude), the $d_m = \{\mu_m, \Sigma_m\}$ can be estimated by:

$$\mu_m = \frac{\sum_{u \in \mathbf{U}} \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{n=1}^{N} \sum_{i=1}^{K} \rho_n(i,m) e_n}{\sum_{m=1}^{M} \sum_{u \in \mathbf{U}} \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{n=1}^{N} \sum_{i=1}^{K} \rho_n(i,m)},$$

$$\Sigma_m = \frac{\sum_{u \in \mathbf{U}} \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{n=1}^{N} \sum_{i=1}^{K} \rho_n(i,m)(e_n - \mu_m)(e_n - \mu_m)^{\mathrm{T}}}{\sum_{m=1}^{M} \sum_{u \in \mathbf{U}} \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{n=1}^{N} \sum_{i=1}^{K} \rho_n(i,m)}.$$

(2) For sequences with underlying multinomial distributions, $e_n$ is a one-hot vector, and $d_m = \{\theta_h\}$ can be estimated by:

$$\theta_m = \frac{\sum_{u \in \mathbf{U}} \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{n=1}^{N} \sum_{i=1}^{K} \rho_n(i,m) e_n}{\sum_{m=1}^{M} \sum_{u \in \mathbf{U}} \sum_{\mathbb{S} \in \mathbf{J}^u} \sum_{n=1}^{N} \sum_{i=1}^{K} \rho_n(i,m)}.$$

Specifically, for music listening sequences, $e_n$ denotes the one-hot vector identifying which artist the user listens to. When the training process terminates, we obtain the parameters of the model.