Applications of Machine Learning to Location Data

Berk Kapicioglu

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
By the Department of
Computer Science

Advisor: Robert E. Schapire

June 2013

Abstract

Positioning devices are generating location data at an unprecedented pace. Coupled with the right software, these data may enable a virtually unlimited number of valuable services. However, to build such software, there is a need for sophisticated algorithms that can extract the relevant information from location data. In this thesis, we use machine learning to develop such algorithms for three fundamental location-based problems.

First, we introduce a new graphical model for tracking radio-tagged animals and learning their movement patterns. The model provides a principled way to combine radio telemetry data with an arbitrary set of spatial features. We apply our model to real datasets and show that it outperforms the most popular radio telemetry software package used in ecology, produces accurate location estimates, and yields an interpretable model of animal movement.

Second, we develop a novel collaborative ranking framework called Collaborative Local Ranking (CLR), which is designed to solve a ranking problem that occurs frequently in the real-world but has not received enough attention in the scientific community. In this setting, users provide their affinity for items via local preferences among a subset of items instead of global preferences across all items. We justify CLR with a bound on its generalization error and derive an alternating minimization algorithm with runtime guarantees. We apply CLR to a venue recommendation task and demonstrate it outperforms state-of-the-art collaborative ranking methods on real datasets.

Third, we design two Bayesian probabilistic graphical models that predict users' future geographic coordinates based on sparse observations of their past geographic coordinates. Our models intelligently share information across users to infer their locations at any future weekhour, determine the number of significant places and the spatial characteristics of these places, and compute the conditional distributions that describe how users spend their time at these places. We apply our models to real location datasets and demonstrate that, despite the sparsity, they provide accurate representations of users' places and outperform existing methods in estimating users' future locations.

Acknowledgments

During my first week at graduate school, the director of graduate studies told the incoming students that our most valuable asset will be each other. This has certainly been true in my case, and I would like to thank my fellow students Melissa Carroll, Jonathan Chang, Christopher DeCoro, Sean Gerrish, Matt Hoffman, Sina Jafarpour, Indraneel Mukherjee, Gungor Polatkan, Umar Syed, and Chong Wang for enriching my graduate school years and being there when I needed them. Most of my cohort has already graduated, but I am happy that our friendship continues to blossom as we drift further away from graduate school. Among my peers, I would especially like to thank Zafer Barutcuoglu, who has bore the brunt of the burden in molding a newly minted college graduate, with little understanding of how the world works, into someone slightly more mature.

Tamara Broderick, Roland Kays, and Martin Wikelski were my co-authors on the project that ultimately became the first chapter of this thesis. If it wasn't for their serendipitious offer to build models of animal movement, this thesis would not have been about location data.

Chapters 2 and 3 were developed during my collaboration with Sense Networks. I am grateful to Tony Jebara, Christine Lemke, Markus Loecher, and Anand Venkataraman for hiring me despite the Great Recession. Research becomes much easier when you are surrounded by people who share your enthusiasm and make you feel at home, and my colleagues Gregory Davis, Kevin Hannan, David Rosenstrauch, Parth Savani, Ralf Schreijer, Heather Sears, Blake Shaw, Nate Tavel, Jason Nobuyuki Uechi, and Yi-Lun Wu all contributed to creating a highly motivational and collegial environment, without which, this thesis would not have been finished. Among my colleagues at Sense, I am particularly indebted to David Rosenberg, whose perfectionism and passion for science is unmatched in the industry. All of the ideas in this thesis have improved due to David's criticism and constant feedback, and I expect him to be a lifelong role model and collaborator. Lastly, I thank David Petersen for continuing to invest in me and support me, especially during times when I felt overburdened with the responsibilities of graduate school.

I would like to thank David Blei, Jennifer Rexford, and Andrea LaPaugh for being members of my thesis committee and providing me with valuable feedback.

Finally, I would like to thank my advisor, Robert Schapire, for providing me with world-class scholarly advice. Throughout my time in graduate school, Rob has parsed my half-baked ideas, barely carrying any meaning, and returned them back to me with depth and clarity. This thesis is a culmination of my relationship with my advisor, and I also believe and hope that it is also the beginning of future collaborations to come.

Research described in Chapter 1 was supported by the National Science Foundation under Grant 0325463, Smithsonian Tropical Research Institute, Peninsula Foundation, and the National Geographic Society. Researches described in Chapters 2 and 3 were conducted while I was working as a scientist at Sense Networks.

To my brother, Mert,
and my parents, Sait and Zerrin,

Contents

List of Figures

Introduction

During the last few years, there has been a surge in the amount of location data that has been generated by positioning devices, which are devices that determine their users' location. There are three reasons for this surge. First, recent advances in hardware have allowed positioning devices to be smaller and use less power [57], making it easier to embed them into mobile phones. Second, the US government's "Enhanced 911" requirement has mandated US wireless carriers to provide accurate location estimates for emergency calls, accelerating the adoption of positioning technologies by mobile phone manufacturers [22]. Third, widespread adoption of positioning hardware has catalyzed the development of location-aware software, which are software that collect location data to perform their tasks. The culmination of these factors generated an unprecedented amount of location data in a few short years.

The number of valuable services that can be provided with such large amounts of location data is virtually unlimited. For example, with the right software, location data can prevent Alzheimer patients from getting lost, automatically deliver traffic alerts or targeted advertisements based on users' future location, or predict the venues that users will visit and make any necessary reservations. However, collecting large amounts of location data is not sufficient to deliver such services; instead, we need algorithms that can convert raw location data into a more refined form these services require. Depending on the problem, this might mean predicting a user's future location based on her past locations, or it might mean extracting a user's preferences over venues she did not yet visit based on venues she did visit. These algorithms are not trivial to design, since a user's location data is generated by complex processes, and these processes are not necessarily amenable to simple rules.

In this thesis, we use machine learning to design and analyze novel algorithms that leverage location data. As a field that is at the intersection of computer science, statistics, and applied mathematics, machine learning is the study of algorithms that improve their performance based on experience. In the general machine learning setting, one provides the algorithm with a training dataset, the algorithm processes it to extract a relevant hypothesis, and this hypothesis is set aside to be applied to future data. This workflow is preferable to the alternative where researchers extract the hypotheses themselves, because designing such hypotheses manually requires domain-specific knowledge and the hypotheses extracted might not easily transfer to new datasets. In contrast, machine learning methods can more easily adapt to new datasets as long as the assumptions made during the derivations of the algorithms are still valid.

When designing new algorithms, machine learning researchers focus on optimizing their *efficiency* and *generalization ability*. Algorithms are efficient when they are guaranteed to compute a desired solution in a reasonable amount of time, which is an important measure when the size of the training data is large. Algorithms generalize well when the hypotheses they extract from their training data also extend to data that will be processed in the future, such as generalizing from a user's past venue visits to infer her preferences over venues she has not visited yet. In this thesis, we apply these theoretical concepts to each problem we study, in addition to evaluating our algorithms on synthetic and real location datasets.

In Chapter 1, we assist biologists on a fundamental problem in ecology by developing algorithms that model how animals move through their environment. For the past few years, biologists have been studying the plant and animal life that thrives on the Barro Colorado Island, which is a tropical island in the middle of the Panama canal. Some of these animals have been tagged with radio transmitters that allow researchers to track their location, but due to the interference caused by the rainforest canopy, the resulting signal is extremely noisy. Biologists have also logged the species and the location of the trees on the island, providing researchers with valuable features that describe the surrounding geography. In this project, we combine radio telemetry data and geographical data in a principled way using a state space model (SSM) to provide accurate estimates of the animals' locations. Compared to previous work, our SSM comprises a richer movement model that in-

corporates spatial features, generalizing Gaussian random walk models that are associated with Kalman filters, and it yields an interpretable model that enunciates the relationship between the animal and its environment. In order to handle the challenges of incorporating a very large and feature-rich representation of movement, we derive an efficient stochastic gradient descent algorithm for learning the parameters of the model. We demonstrate the algorithm's effectiveness by comparing it to the expectation-maximization (EM) algorithm, both via asymptotic analysis and synthetic experiments. Finally, we apply our model to real datasets and demonstrate that it outperforms the most popular radio telemetry software package used in ecology.

In Chapter 2, we turn our attention to collaborative filtering, and develop a new framework for ranking items. In typical collaborative filtering, a learning algorithm is provided with users' ratings over items they have rated, and in return, the algorithm is expected to predict users' ratings over items they have not yet rated. In contrast to supervised learning algorithms, collaborative filtering algorithms do not require explicit features to be prepared for the users or the items [27], making it easy to apply them to new domains. Furthermore, collaborative filtering algorithms achieve state-of-the-art performance on recommendation tasks, and consequently, are the method of choice for companies such as Amazon [33] and Netflix [5]. Despite their success, existing collaborative filtering methods are not designed for a common type of recommendation problem, one where users only provide local preferences instead of global preferences. This type of feedback exists in many real-world applications, two of which are movie recommendation and venue recommendation. For example, when users choose to watch a movie after reading the reviews of a set of candidate movies, they are implicitly preferring the chosen movie over the reviewed movies, as opposed to preferring the chosen movie over *all* other movies. Similarly, when users choose to visit a venue after exploring nearby venues through a mobile application, they are providing local preferences among nearby venues instead of providing preferences across *all* venues.

We call this problem *collaborative local ranking* and provide a formal characterization of it. We prove a Rademacher-based bound on the generalization error using a hypothesis class of low-rank matrices. We then transform the empirical loss into a tractable form, derive a simple alternating minimization for it, and prove that

each minimization step converges to the correct solution in time *independent* of the size of the training data. Finally, we demonstrate the performance of our algorithm on a venue recommendation task and show that it outperforms state-of-the-art collaborative ranking methods.

In Chapter 3, we develop probabilistic graphical models for location prediction, where the goal is to predict users' future geographic coordinates based on their past geographic coordinates. Location prediction is a problem [8, 19] that underlies virtually all location-based tasks, and in fact, De Domenico et al. [12] recently won Nokia's Mobile Data Challenge by devising a location prediction algorithm that predicts where the user is going to be during the next 24 hours. Despite being a problem of extreme interest, past work on location prediction has focused exclusively on dense datasets, where location data has been collected with high frequency. This is a valid assumption for certain datasets, such as those collected by cellular carriers; however, most location-aware applications can collect location data only when the user is actively using the application, either due to privacy restrictions or to save the device's battery. For example, a reminder application might only record the user's location when running in the foreground, and even though it collects very few location data points from the user during this period, it would nevertheless be expected to remind the user whenever she arrives at the target destination. Similarly, an application which delivers customized news based on the user's location might not record any data when running in the background, but it would nevertheless be expected to deliver news alerts continuously based on the user's estimated location. In fact, as users expect applications to be more intelligent and as location-aware applications collect location data more judiciously, there will be a growing need for location prediction algorithms that can leverage sparse datasets.

In order to estimate users' locations from sparse data, we design two novel Bayesian probabilistic graphical models. Our models intelligently share information across users to infer their locations at any future weekhour even when the user has not been previously observed on that day or at that hour. In addition to estimating users' locations, our models infer the number of significant places for each user, infer the spatial characteristics of these places, and compute the conditional distributions that describe how users spend their time during each weekhour. We derive a fast collapsed Gibbs sampler for estimating model parameters and demon-

strate that our models not only infer significant places accurately, but also achieve state-of-the-art predictive performance in estimating users' future locations.

The research described in Chapter 1 was published as Kapicioglu et al. [26], and it is joint work with Robert Schapire, Martin Wikelski, and Tamara Broderick. The researches described in Chapters 2 and 3 are currently under submission, and they are joint work with David Rosenberg, Robert Schapire, and Tony Jebara.

Chapter 1

Learning Animal Movement Models

1.1 Background

Animals move through their environments in complex ways. Understanding the processes that govern animal movement is a fundamental problem in ecology and has important ramifications in areas such as home-range and territorial dynamics, habitat use and conservation, biological invasions and biological control [24]. Ecologists rely heavily on collecting and analyzing animal movement data to deepen their understanding of these processes.

In recent years, various technological advances, such as radio telemetry systems and the Global Positioning System (GPS), have created new avenues for collecting data from animals [55]. In radio telemetry, animals are tagged with a tiny radio transmitter. At regular time intervals, fixed-location towers in the environment record the signal from the transmitter and use this information to infer the direction of the animal with respect to the tower. In contrast to GPS, radio telemetry systems can use smaller transmitters, can collect data more frequently, are simpler to implement, and can be used under rainforest canopies. An implementation of such a system in Barro Colorado Island, called the Automated Radio Telemetry System Initiative (ARTS), provides researchers access to hundreds of thousands of directional data measurements collected from a variety of animals [10]. However, even though this method has led to a proliferation of directional data measurements,

it has been difficult to harness the full potential of these datasets because: 1) directional measurements obtained through radio telemetry are notoriously noisy; 2) telemetry databases may contain very large amounts of data; and 3) directional measurements are not in a form that is easily interpretable. What is needed are computational tools that would efficiently and accurately convert these large and noisy datasets into a form that would enhance ecological research.

Previously, various sequential probabilistic graphical models have been proposed to solve this problem. Anderson-Sprecher and Lodelter [1, 2] used an iterated extended Kalman filter-smoother to estimate animal locations. Jonsen et al. [25, 24] represented animal movements as correlated random walks and used Bayesian techniques to infer posterior model parameters and animal locations. Ovaskainen [40] used a diffusion approach to model movement in heterogeneous landscapes. Morales et al. [37] fitted multiple random walks to animal movement paths and modeled switching probabilities between them as a function of landscape variates. Patterson et al. [41] and Schick et al. [46] further review past probabilistic graphical models and their applications to ecology and animal movement.

In this work, we propose a new state space model (SSM) approach. In contrast to previous work, our SSM includes a richer and more general animal movement model. It can utilize arbitrary geographical features, such as the population densities of various tree species or the location of water sources, to represent geographically-dependent animal movement models that cannot be represented by previous approaches at this generality. While our animal movement model also generalizes Gaussian random walks, which are models associated with Kalman filters, combining and incorporating spatial features, especially geographical ones, yields an interpretable model that enunciates the relationship between the animal and its environment. Furthermore, a richer model leads to increased accuracy in estimating animal locations from telemetry data.

In order to handle the challenges of incorporating a very large and feature-rich latent state space, we present an efficient stochastic gradient algorithm for learning the parameters of the model. We demonstrate the algorithm's effectiveness in training our model by comparing it to the expectation-maximization (EM) algorithm both via asymptotic analysis and synthetic experiments. We note that, in contrast to sequential Monte Carlo methods such as particle filters, our inference

algorithms allow us to train parameters and infer past estimates based on *all* observations.

Finally, we apply our model to real datasets and demonstrate that it outperforms the most popular radio telemetry software package used in ecology. In conclusion, we show that our methods aid us in producing interpretable results and accurate animal location estimates from large and noisy telemetry datasets.

This chapter is organized as follows. In Section 1.2, we specify the SSM and formalize its parameter and state estimation problems. In Section 1.3, we detail the EM, stochastic gradient descent, and Viterbi algorithms for solving the estimation problems and analyze their time-complexity. We report the results of our experiments on synthetic and real datasets in Section 1.4 and conclude in Section 1.5.

The research described in this chapter was published as [26], and it is joint work with Robert Schapire, Martin Wikelski, and Tamara Broderick.

1.2 A State Space Model

Our telemetry datasets comprise a long sequence of directional measurements that are observed by a small number of fixed-location towers at regular time intervals. Note that all directional measurements in a given telemetry dataset are received from the same animal. We model the data as if it were generated by an SSM. In this section, we describe the SSM by detailing its latent state space as well as its start, transition and observation models. Then, we formalize the parameter and state estimation problems, and argue how solving the estimation problems yields an interpretable model of animal movement and accurate animal location estimates.

Intuitively, the latent state space is the space of all possible animal locations. In order to use the SSM machinery, we discretize a continuous latent state space of animal locations into a finite and discrete space of coordinates $Q \subset \mathbb{R}^2$. In practice, Q is constructed by partitioning a larger state space into finitely many equally sized grid cells and assigning each grid cell midpoint as a coordinate in Q.

The start model generates the first state of the latent animal trajectory. For simplicity's sake, we use the uniform distribution as the start model. Let the latent random state at time step $t \in \{1, \ldots, T\}$ be denoted by $x_t \in Q$. Then, the start model

is

$$p(x_1) = \frac{1}{|Q|}. \tag{1.1}$$

The transition model, a Gibbs distribution, generates the rest of the latent animal location data. Gibbs distributions, also known as the conditional exponential model, have emerged as popular models in machine learning due to their practical success and their theoretical elegance. These distributions are defined using features, which in our case are functions that encode information about the spatial properties of the environment. For example, a feature may encode the distance between two coordinates in the latent state space, or it may encode the minimum distance to a certain tree species. In a slightly different but related setting, it has been shown that the maximum likelihood estimate of a Gibbs distribution is equivalent to the maximum entropy distribution with various constraints imposed by the features [6]. In our case, we adopt the Gibbs distribution as the transition model because it provides a means to incorporate spatial features without making any independence assumptions about them, remains resilient to the extension of feature space by irrelevant features, and generalizes discrete versions of simpler random walk models which have previously been used to model animal movement. More formally, let $f_k : Q \times Q \to \mathbb{R}$ denote the kth feature and $\lambda_k \in \mathbb{R}$ denote the corresponding weight parameter, where $k \in \{1, \ldots, K\}$. We use $\boldsymbol{\lambda}$ as a vector representation of feature weights. Then the transition model is

$$p(x_{t+1}|x_t; \boldsymbol{\lambda}) = \frac{\exp\left(\sum_{k=1}^{K} \lambda_k f_k(x_t, x_{t+1})\right)}{\sum_{x \in Q} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(x_t, x)\right)}. \tag{1.2}$$

The observation model generates the directional measurements observed by the towers. We model the behavior of towers as a von Mises distribution, a circular analogue of the normal distribution [35]. A von Mises distribution is parameterized by μ and κ, roughly the analogs of the mean and the variance of a normal distribution, respectively. Intuitively, a radial bearing is sampled from a von Mises distribution, and it is added as noise to the true bearing that points in the direction of the animal. More formally, fix a radial bearing system shared by all towers (i.e., bearing

$\frac{\pi}{2}$ points north and bearing 0 points west). Let $y_{t,n} \in [-\pi, \pi)$ denote the random variable, a radial bearing, observed by tower $n \in \{1, \ldots, N\}$ at time step t. This is a noisy observation that is supposed to point in the direction of the animal. Let $z_n \in \mathbb{R}^2$ denote the coordinate of tower n and define $h(x, z_n) \in [-\pi, \pi)$ to be the true radial bearing of the vector that points from tower n towards location x. Let $\mu_n \in \mathbb{R}$ and $\kappa_n \geq 0$ denote the parameters of the von Mises distribution for tower n. Let I_0 denote the modified Bessel function of the first kind with order 0, which simply acts as the normalization constant for the von Mises distribution. Finally, let $\boldsymbol{\mu}$, $\boldsymbol{\kappa}$, and $\mathbf{y}_t$ be the vector representations of the corresponding parameters and random variables. Then, the observation model is

$$\begin{aligned} p\left(\boldsymbol{y}_t | x_t; \boldsymbol{\mu}, \boldsymbol{\kappa}\right) &= \prod_{n=1}^{N} p\left(y_{t,n} | x_t; \mu_n, \kappa_n\right) \\ &= \prod_{n=1}^{N} \frac{\exp\left(\kappa \cos\left(y_{t,n} - h(x_t, z_n) - \mu_n\right)\right)}{2\pi I_0(\kappa)}. \end{aligned} \tag{1.3}$$

The factorization occurs because of the conditional independence assumptions that hold between the observations.

The start, transition, and observation models can be used to compute all the marginal and conditional probability distributions of the SSM. Let $\mathbf{y}$ denote the vector of all observations and let $\boldsymbol{\theta} = \left(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\kappa}\right)$ denote the vector of model parameters. As usual, the joint probability distribution of the SSM is

$$\begin{aligned} p(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) &= p\left(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\kappa}\right) \\ &= p(x_1) \prod_{t=1}^{T-1} p(x_{t+1} | x_t; \boldsymbol{\lambda}) \prod_{t=1}^{T} p\left(\boldsymbol{y}_t | x_t; \boldsymbol{\mu}, \boldsymbol{\kappa}\right). \end{aligned} \tag{1.4}$$

Parameter estimation is the problem of estimating the model parameters (i.e. $\boldsymbol{\theta}$). We choose maximum likelihood estimation (MLE) as the method of fitting model parameters to data. We define the problem of parameter estimation as

$$\hat{\boldsymbol{\theta}} = \left(\hat{\boldsymbol{\lambda}}, \hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\kappa}}\right) = \underset{\boldsymbol{\lambda} \in \mathbb{R}^K, \boldsymbol{\mu} \in \mathbb{R}^N, \boldsymbol{\kappa} \geq \mathbf{0}}{\operatorname{argmax}} \log p\left(\mathbf{y}; \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\kappa}\right). \tag{1.5}$$

State estimation, on the other hand, is the problem of estimating values of unobserved random variables. We choose to seek the hidden state sequence that max-

imizes the probability conditioned on the observations and the MLE parameter estimates. More formally, we define the problem of state estimation as

$$\hat{\mathbf{x}} = \underset{\mathbf{x} \in Q^T}{\operatorname{argmax}}\, p\left(\mathbf{x}|\mathbf{y};\hat{\boldsymbol{\theta}}\right). \tag{1.6}$$

Our intention is to use the parameter estimates of the Gibbs distribution as an interpretable model of animal movement, and the state estimates of the latent random variables as animal location estimates. We note that an analysis of Gibbs weights might not conclusively explain the processes governing animal movement, but we hope that it will be a first step in further exploration of these processes.

1.3 Algorithms

In this section, we detail the algorithms for solving the estimation problems and analyze their time-complexity. Later in the chapter, we will see that for the type of datasets we have, where both the cardinality of the latent state space and the number of time steps is large, but the cardinality of the feature space is small, the stochastic gradient algorithm is asymptotically superior to EM. The analysis in this section will allow us to compare EM and the stochastic gradient algorithms.

1.3.1 Expectation Maximization

Algorithm 1.1 EM($\boldsymbol{\theta}^0$, maxTime)

1: $i \leftarrow 0$.
2: **for all** until elapsed time exceeds maxTime **do**
3: $\quad \forall t \in \{1,\ldots,T\}$, compute $\log p\left(x_t|\mathbf{y};\boldsymbol{\theta}^i\right)$.
4: $\quad \forall t \in \{1,\ldots,T-1\}$, compute $\log p\left(x_t, x_{t+1}|\mathbf{y};\boldsymbol{\theta}^i\right)$.
5: $\quad \forall n \in \{1,\ldots,N\}$, compute $\left(\mu_n^{i+1}, \kappa_n^{i+1}\right) \leftarrow \operatorname{argmax}_{\mu_n,\kappa_n} \sum_{t=1}^{T} \mathbb{E}_{x_t|\mathbf{y};\boldsymbol{\theta}^i}\left[\log p\left(y_{t,n}|x_t;\mu_n,\kappa_n\right)\right]$.
6: $\quad \boldsymbol{\lambda}^{i+1} \leftarrow \operatorname{argmax}_{\boldsymbol{\lambda}} \sum_{t=1}^{T-1} \mathbb{E}_{x_t|\mathbf{y};\boldsymbol{\theta}^i}\left[\log p\left(x_{t+1}|x_t;\boldsymbol{\lambda}\right)\right]$.
7: $\quad i \leftarrow i+1$.
8: **return** $\boldsymbol{\theta}^i$.

In Algorithm 1.1, we present the adaptation of EM to our setting. Here, maxTime denotes the maximum allowed running time and $\boldsymbol{\theta}^0 = \left(\boldsymbol{\lambda}^0, \boldsymbol{\mu}^0, \boldsymbol{\kappa}^0\right)$ denotes

the initial parameter settings. Lines 3 and 4 describe what is traditionally called the E-step and involve computations of various conditional log-probabilities. These log-probabilities can be computed using algorithms such as the forward-backward algorithm. Similarly, Lines 5 and 6 describe what is traditionally called the M-step and involve maximization of expected complete log-likelihoods. The maximization problem in Line 5, after substitution of Equation (1.3) and some trigonometric manipulations, can be rewritten as

$$\begin{aligned}\underset{\mu_n,\kappa_n}{\operatorname{argmax}} \sum_{t=1}^{T} \kappa_n \cos \mu_n \mathbb{E}_{x_t|\boldsymbol{y};\boldsymbol{\theta}^i}\left[\cos\left(y_{t,n} - h(x_t, z_n)\right)\right] \\ + \sum_{t=1}^{T} \kappa_n \sin \mu_n \mathbb{E}_{x_t|\boldsymbol{y};\boldsymbol{\theta}^i}\left[\sin\left(y_{t,n} - h(x_t, z_n)\right)\right] - T \log 2\pi I_0(\kappa_n).\end{aligned}$$

In this form, it is equivalent to the problem of finding MLE estimates of von Mises parameters from i.i.d samples, and it can be solved in closed form using techniques from directional statistics [35]. The maximization problem in Line 6 is convex and unconstrained, and it can be solved using numerical optimization algorithms such as BFGS [34, 38].

EM is an iterative algorithm and it is hard to predict beforehand how many iterations are necessary until convergence. Here, we analyze the time-complexity of each iteration for a *direct* implementation of Algorithm 1.1. When it is clear from context, we write Q for $|Q|$. We also remind the reader that Q denotes the latent state space, K denotes the number of features, and T denotes the length of the animal trajectory. Almost all EM computations rely on the transition matrix, which can be computed in $O\left(Q^2 K\right)$. The E-step, Lines 3 and 4, consists of computation of conditional log probabilities, which takes $O\left(Q^2 T\right)$. The M-step, Lines 5 and 6, consists of both iterative and non-iterative optimization algorithms for estimating Gibbs and von Mises parameters, respectively. Optimization of von Mises parameters can be computed in $O(QT)$, whereas optimization of Gibbs parameters is itself conducted iteratively using BFGS. However, in practice, the number of BFGS iterations is small. For each BFGS iteration, it takes $O\left(Q^2 T\right)$ to evaluate the objective function and $O\left(Q^2 K\right)$ to evaluate its gradient. Finally, we note that we suppressed the dependence on N, the number of towers, since it is small. Overall, each EM iteration takes $O\left(Q^2 (K+T)\right)$.

For solving the problem of state estimation, Equation (1.6), the Viterbi algorithm may be used. Viterbi is not iterative and it only needs to be executed once. Once the transition matrix is computed in $O\left(Q^2K\right)$, the most likely path is computed in $O\left(Q^2T\right)$, yielding, like EM, a total running time of $O\left(Q^2(K+T)\right)$.

An important factor in determining both computational efficiency and statistical accuracy is the resolution of the grid used to construct Q, the latent state space. As the diagonal distance between corners of each grid cell in Q gets shorter, the latent state space representation becomes more accurate[1] linearly, whereas the cardinality of the latent state space increases quadratically. Thus, an increase in statistical accuracy at a linear rate is offset by an increase in computational complexity at a *quartic* rate, since EM and Viterbi themselves have time-complexity quadratic in the cardinality of the latent state space. This trade-off leads to a rapid increase in the running time of these algorithms in exchange for small gains in statistical accuracy.

1.3.2 Stochastic Gradient

In this subsection, we propose a stochastic gradient alternative to EM, which under certain conditions, is asymptotically superior to EM. The algorithm is an optimization method that can be used when one has access to a noisy approximation of the gradient of the objective function [49]. Younes [56] used it to estimate parameters of partially observed Markov Random Fields. Delyon et al. [13] proved convergence results for a family of stochastic approximations to EM, including the stochastic gradient algorithm of the form depicted in Algorithm 1.2. In the rest of the subsection, we present our implementation of the stochastic gradient algorithm, analyze its time-complexity and compare it to that of EM, and detail how our implementation differs from other implementations.

The algorithm is presented formally as Algorithm 1.2. Here, $\boldsymbol{\theta}^0$ denotes the initial parameter settings, maxTime denotes the maximum allowed running time, and numBurn denotes the number of Markov Chain Monte Carlo (MCMC) iterations executed before a sample obtained through Gibbs sampling is accepted. The

[1] One measure of accuracy is the diagonal distance between the corners of a grid cell. In the worst-case scenario, a larger grid cell would yield a higher absolute error between the location estimate and the true location even when the true location is estimated by the closest cell midpoint.

Algorithm 1.2 SG($\boldsymbol{\theta}^0$, numBurn, maxTime)

1: $i \leftarrow 0$.
2: **for all** until elapsed time exceeds maxTime **do**
3: Sample $\mathbf{x}^i \sim p\left(.|\mathbf{y};\boldsymbol{\theta}^i\right)$ using randomized Gibbs sampling with a burn-in period of numBurn.
4: $\boldsymbol{\theta}^{i+1} \leftarrow \boldsymbol{\theta}^i + \gamma_i \nabla L_{\mathbf{x}^i}\left(\boldsymbol{\theta}^i\right)$, where γ_i is chosen to satisfy Wolfe conditions with respect to $L_{\mathbf{x}^i}\left(\boldsymbol{\theta}^i\right)$.
5: $i \leftarrow i + 1$.
6: **return** $\boldsymbol{\theta}^i$.

log-likelihood of the complete data, denoted L, is defined as

$$L_{\mathbf{x}}(\boldsymbol{\theta}) = L_{\mathbf{x}}\left(\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\kappa}\right) = \log p\left(\mathbf{x}, \mathbf{y}; \boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\kappa}\right). \tag{1.7}$$

In Line 3, latent random variables are sampled from the conditional distribution using Gibbs sampling, and in line 4, model parameters are updated in the direction of the gradient. We note that the algorithm is a stochastic gradient algorithm because the gradient of the log-likelihood of the complete data, Equation (1.7), is a noisy approximation of the gradient of the objective function, Equation (1.5).

The main difference between our implementation of the stochastic gradient algorithm and previous implementations is the choice of the learning rate γ. Common implementations of the stochastic gradient algorithm use a decreasing learning rate sequence that guarantees the convergence of the algorithm to a stationary point [13]. However, in practice, the choice of the constant factor associated with such sequences significantly influences the speed of convergence and it is a difficult task to identify the optimal factor. We tune the learning rate automatically by choosing one that satisfies the Wolfe conditions. These are line search conditions that guarantee the convergence of gradient updates to an optimum in *deterministic* convex optimization settings [38], but in our case, we use them in a *stochastic* optimization setting. In practice, parameters that satisfy these conditions can be found by providing the objective function and its gradient to line search methods. Even though we lack a proof of convergence, we have observed informally that our criterion for selecting the learning rates ensures better performance than manually choosing learning rates that satisfy the customary assumptions. Furthermore, our approach provides an automated way to set the learning rate based on the dataset.

We conclude the subsection with an analysis of the time complexity of our implementation of Algorithm 1.2. Line 3 takes $O(Q(B+VK))$, where B is the burn-in period and V is the number of unique states visited during sampling. In order to achieve this complexity, we compute the transition probabilities only for the sampled states, and use memoization to store and recall them. Line 4 is conducted iteratively, due to the line search that involves finding the right learning rate, but number of iterations is very small in practice. Computing the objective function takes $O(QVK+T)$, and computing the gradient takes $O(K(QV+T))$.

In conclusion, the choice between EM and the stochastic gradient algorithm depends on the relative size of the latent state space, the feature space, and the number of time steps; however, in our problem, where both the cardinality of the latent state space and the number of time steps is large, but the cardinality of the feature space is small, the stochastic gradient algorithm is asymptotically superior.

An asymptotic comparison of each iteration does not suffice to determine which algorithm to use; thus in the next subsection, we perform empirical tests that compare these algorithms with respect to a variety of performance measures. Also, we note that *both* algorithms may be optimized such that only state transitions to a small neighborhood are taken into account in the computations. This would reduce time-complexity from $O\left(Q^2\right)$ to $O(QR)$, where R is the cardinality of the largest neighborhood. Our implementation of all discussed algorithms take advantage of this optimization.

1.4 Experiments

In this section, we demonstrate our model on both synthetic and real datasets. First, we generate synthetic datasets to evaluate our model in a setting where true feature weights are known. We also use the synthetic datasets to compare the performance of the EM and the stochastic gradient algorithms. Then we use the real datasets to compare our model with LOAS, the most ubiquitous radio telemetry software package used in ecology.

1.4.1 Synthetic Datasets

In order to simulate a real-world application as closely as possible, we created a virtual island that has approximately the same dimensions and the same tower locations as the Barro Colorado Island. We partitioned the virtual island into 3654 grid cells (latent states), where each grid cell is 100×100 meters square. Then we created 10 different animal movement models (transition models). Each animal movement model consisted of 5 features whose function values were generated uniformly at random from $[0,1)$, and each feature's weight was generated uniformly at random from $[-10,10]$. For each animal movement model, we generated an animal path of length 1000. In order to be as realistic as possible, we constrained the animal to move at most 500 meters at each time step and normalized the probabilities of the animal movement model accordingly. We also used the same constraint during the parameter estimation. As for the von Mises parameters of the towers, we set each μ to 0 and κ to 15, which approximates a normal distribution with a standard deviation of 15 degrees. For each of the 10 animal paths, we generated the corresponding noisy bearings, and together with the corresponding features (but without the feature weights), provided them as input to the parameter estimation algorithms.

In the first batch of experiments, we compared the performance of the stochastic gradient algorithm and the EM algorithm. For both algorithms, we set their initial Gibbs weights to 0, which corresponds to a uniform transition model, and their initial κ parameters to 50. For the stochastic gradient algorithm, we used a burn-in period of $100,000$. We executed the algorithms on each dataset for 10 hours. To evaluate their performance, we measured the arithmetic mean error of the location estimates, the Euclidean distance between the learned weights and the true weights, and the log-likelihood of the observed bearings. We used the Euclidean distance between the weights as a measure of the interpretability of the model, where the closer weights were considered more interpretable.

As suggested by the asymptotic analysis, the stochastic gradient algorithm executed many more iterations than EM, and outperformed EM with respect to both the arithmetic mean error of the location estimates and the Euclidean distance between the learned weights and the true weights. The stochastic gradient algorithm also attained a higher log-likelihood than EM. We display the results, averaged

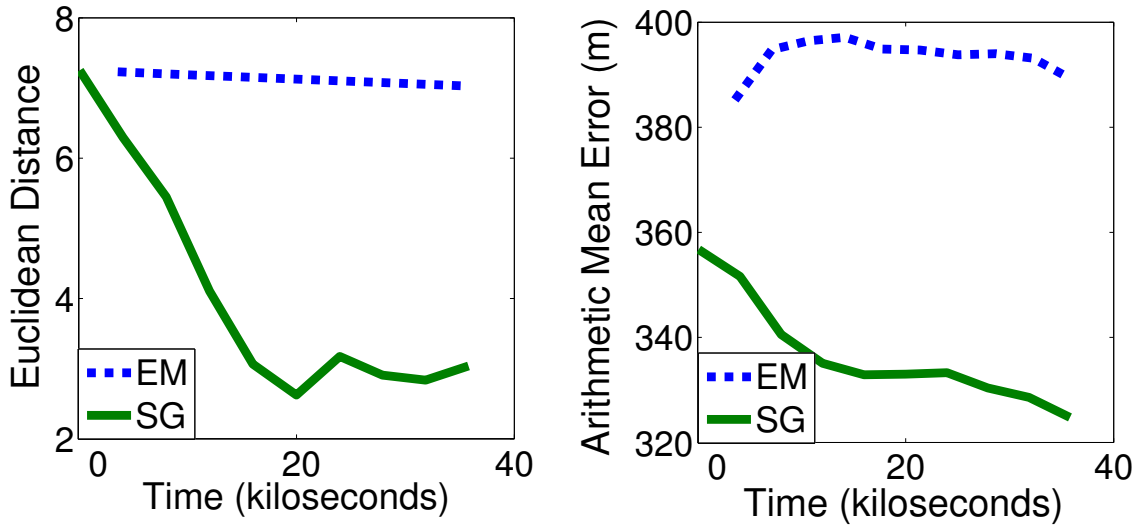

Figure 1.1: A comparison of the performance of EM and the stochastic gradient algorithms. During 10 hours, EM had an average of 10 iterations, whereas stochastic gradient had an average of 500 iterations. The left plot reports the Euclidean distance between the learned weights and the true weights, and the right plot reports the average mean error of the location estimates. Stochastic gradient outperforms EM in both cases.

over the 10 datasets, in Figure 1.1.

In the second batch of experiments, we compared the performance of our animal movement model with a discrete version of the animal movement model used by Anderson-Sprecher [1]. By doing so, we both wanted to demonstrate how our model can generalize previous animal movement models and we wanted to observe whether having a richer model leads to an improvement in the accuracy of location estimates. The animal movement model used by Anderson-Sprecher is an isotropic bivariate Gaussian random walk which was trained using an extended Kalman filter-smoother. His model is defined as

$$p\left(x_{t+1}|x_t;\sigma^2\right) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{\|x_{t+1}-x_t\|^2}{2\sigma^2}\right). \tag{1.8}$$

We can represent the same model approximately as a Gibbs distribution using the single distance-based feature $f_{dist} = -\frac{\|x_{t+1}-x_t\|^2}{2}$. Similar to the first batch of experiments, we executed two copies of the stochastic gradient algorithm on the same datasets; one copy used the features that generated the datasets and the other copy used the single feature that represents the bivariate Gaussian random walk.

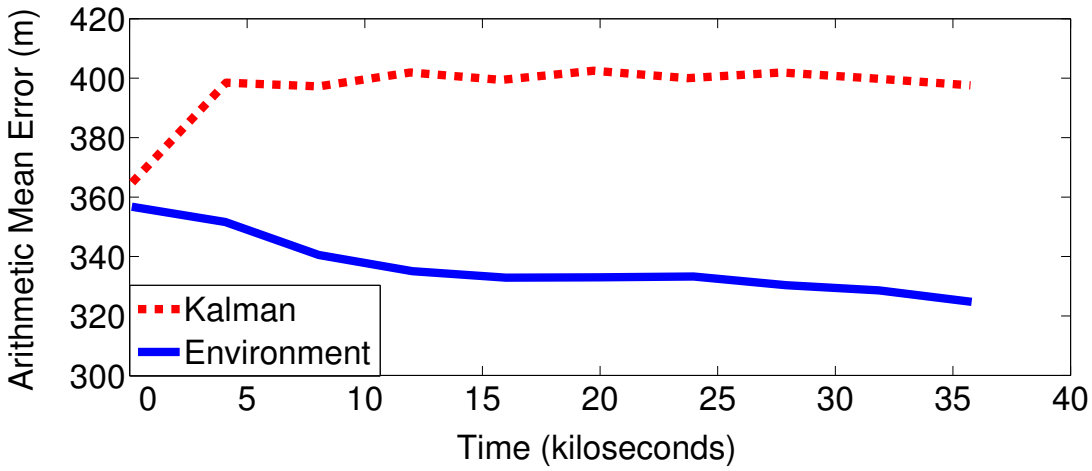

Figure 1.2: Predictive performance of using a richer feature-based (Environment) model versus using a simpler random walk (Kalman) model. The results imply that, if the animal indeed moves around based on environmental features, a model that incorporates such features does yield better location estimates than a simpler random walk model.

The richer model outperformed the Gaussian random walk model with respect to the accuracy of the location estimates. We display the results of these experiments, averaged over 10 datasets, in Figure 1.2.

1.4.2 Real Datasets

We applied our model to the radio telemetry data collected from two sloths, named Chispa and Wendi, that live on the Barro Colorado Island. These sloths were one of the few animals on the island whose true locations were labeled every few days by human researchers via GPS devices. Thus, we were able to use the radio telemetry data to train our algorithms and the GPS data to test them.

The radio telemetry data was collected every 4 minutes for 10 days, yielding approximately 3600 time points. Both datasets had considerable noise in the bearing measurements. In order to apply our model, we discretized the island into 1200 grid cells, each of size 50×50 meters square. As features, we used both the distance-based feature that encodes a bivariate random walk, Equation (1.8), and tree-based features that encode the change in the population density of various tree species across grid cells. In particular, the model included 18 features: 1 distance-based feature and 17 tree-based features. Each of the 17 tree-based features corresponded

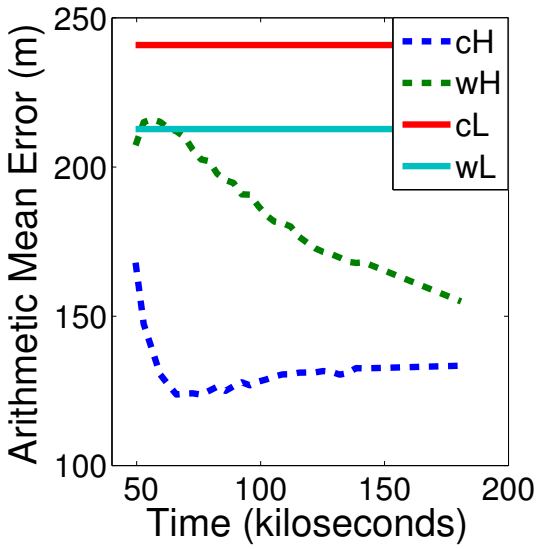

Figure 1.3: Arithmetic mean errors of SSM and LOAS on the two animal datasets. cH and wH represent the performance of the SSM; cL and wL represent the performance of LOAS. The initial letters "c" and "w" refer to the Chispa and Wendi datasets, respectively.

to a different tree species and they were normalized to have values in $[-1,1]$. We initialized the Gibbs weights to 0, the μ parameters to 0, and the κ parameters to 10. We executed 10 different instances of the stochastic gradient algorithm, each using a different random seed, and averaged the results. As a baseline comparison, we used LOAS, which is the most popular radio telemetry software package used in ecology [30].

In Figure 1.3, we display the distance error obtained by SSM and LOAS on the two datasets. For both datasets, SSM outperformed LOAS. In Figure 1.4, we show how the feature weights evolved while training on the Wendi dataset. Our model successfully learned a diminishing movement variance for the sloth, represented in the figure as the growing weight associated with the distance-based feature. It also identified a small portion of the tree species that the sloth seems to have a preference for. In Figure 1.5, we display the true locations, SSM estimates, and the LOAS estimates obtained on the Wendi and Chispa datasets. SSM estimates are clustered much more closely to the true locations than the LOAS estimates.

We also generalized our movement model features to be temporally-dependent. In particular, we partitioned each day into different time zones, and for each such zone, we allowed the animals to move according to a different movement model.

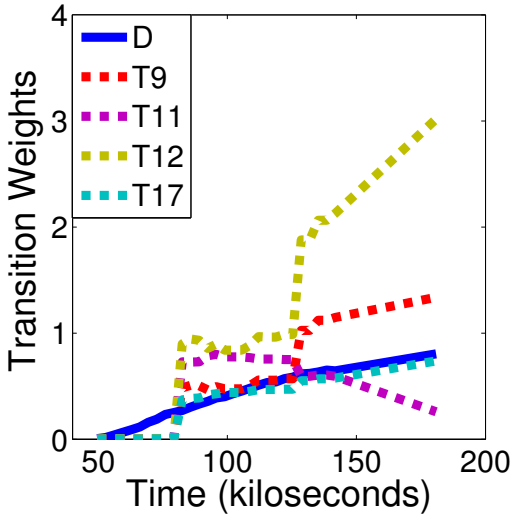

Figure 1.4: The evolution of the transition weights for the Wendi dataset. We only plotted the weights that exceeded the 0.5 threshold. "D" and "T" denote the distance-based and tree-based features, respectively.

The changes in the daily activity of the sloths have already been studied by field biologists; so for our experiments, we set the time zones, "day" and "night", based on biologists' feedback.

In our last experiment, we were interested in evaluating whether feature weights converge to meaningful values, whether they are interpretable, and how different types of features interact with each other. For this purpose, we created an *artificial* tree type, where for each of the very few locations GPS data was available for Wendi, we placed a virtual tree. We defined this tree-feature formally as $f_{tree} = -\frac{\|tree\_dist(x_{t+1}) - tree\_dist(x_t)\|^2}{2}$, where $tree\_dist(x)$ is the Euclidean distance between x and the closest tree of that type. By defining the tree-based feature this way, we set it to the "same" unit as the distance-based feature associated with Equation (1.8), allowing the feature weights to be numerically comparable. After normalizing all feature values to be within $[-1, 1]$, the weights converged to 0.704 for distance-based feature at night and 0.316 during day; -1.279 for tree-based feature at night and -1.312 during day. As expected, interpreting distance-based weights indicate that Wendi moves more during day than night and that the positive magnitude of these weights indicate that the animal is more likely to stay in place in succeeding time steps. Also, tree-based weights correctly indicate that

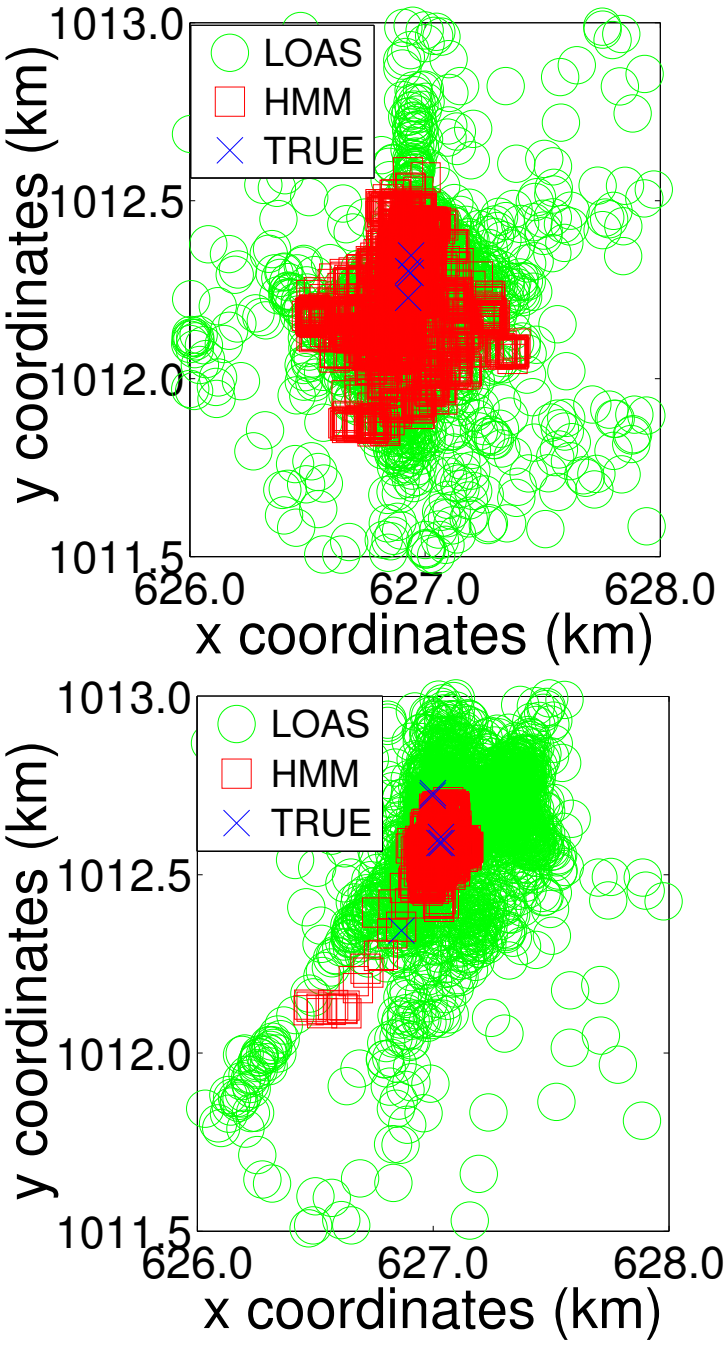

Figure 1.5: Plots of the true locations, SSM estimates, and LOAS estimates on the two animal datasets. SSM estimates are based on the last stochastic gradient iteration. Top plot displays the results for the Wendi dataset and the bottom plot displays the results for the Chispa dataset.

Wendi has a strong preference for certain neighborhoods on the map (i.e. the true locations), and that the strength of these preferences are indifferent to the time zone.

1.5 Discussion

In this chapter, we presented an SSM approach to locate radio-tagged animals and learn their movement patterns. We presented a model that incorporates both geographical and non-geographical spatial features to improve animal location estimates and provide researchers an interpretable model that enunciates the relationship between the animal and its environment. We showed that the model generalizes discrete versions of random walk models and demonstrated empirically that a richer model improves animal location estimates. We also provided a fast parameter estimation algorithm, the stochastic gradient algorithm, and demonstrated its effectiveness against EM both asymptotically and empirically. Finally, we applied our model to real datasets. Our model outperformed LOAS, the most popular radio telemetry software package in ecology, with respect to the accuracy of the location estimates. Our model also hypothesized that the sloth has a relatively strong preference for certain types of trees. We are currently working on applying our model to other telemetry datasets collected at the Barro Colorado Island.

Chapter 2

Collaborative Ranking of Local Preferences

2.1 Background

Since the early days of the Netflix Prize competition, matrix factorization (MF) [29] has become a popular method for modeling users' preferences over a set of items. MF achieves state-of-the-art performance on very large-scale datasets, such as the Netflix dataset [4], which comprises more than one hundred million ratings. MF also does not require user and item features [28], making it particularly useful for practitioners, as they can easily apply it to new domains without designing domain-specific features.

Most MF methods optimize root-mean-square error (RMSE) [51][44][45], largely because the winner of the Netflix Prize was determined by it. However, for many applications of collaborative filtering, RMSE is inappropriate because actual performance is based on predicting a rank rather than a rating. For example, when choosing which sci-fi movie to watch, a user might wonder how Star Trek ranks against Serenity, and predicting 3.8 for Star Trek and 4.5 for Serenity is relevant only in so far as helping the user infer a relative ranking.

As a remedy, researchers have devised various algorithms for collaborative ranking. Weimer et al. [52] proposed CofiRank, which is an MF method that optimizes ranking measures, such as normalized discounted cumulative gain (NDCG). Rendle et al. [42] analyzed the collaborative ranking problem from a Bayesian perspective

and provided a meta-optimization criterion called Bayesian Personalized Ranking (BPR), which when coupled with MF, reduces into a differentiable version of CofiRank's objective. Balakrishnan and Chopra [3] used a two-stage model to rank; the first stage learns the latent factors via Probabilistic Matrix Factorization (PMF) [44], and the second stage processes these factors as features using supervised regression and ranking algorithms.

Existing collaborative ranking algorithms, including CofiRank and BPR, require users' preferences to be totally ordered, but there are many tasks where this is too restrictive. For example, for a movie recommendation task, when a user browses a small list of sci-fi titles and eventually chooses one, she prefers the chosen movie over the remaining sci-fi movies in that list, but she does not necessarily prefer the chosen movie over *all* other movies. Similarly, for a venue recommendation task, when a user considers nearby restaurants and settles on one, she demonstrates a local preference among a small set of nearby restaurants, rather than a global preference between the chosen restaurant and *all* other restaurants.

We are interested in an algorithm that not only learns from local preferences encountered during training, but also is evaluated according to the local preferences it predicts during deployment. For example, for a venue recommendation task, the user would query the recommendation system when she is at a particular neighborhood, and the system would return a local, not global, ranking over nearby venues. Thus, a successful recommendation system does not just use local preferences it observes during training to piece together a global ranking; what the user ultimately cares about are local rankings themselves.

To tackle these issues, we introduce a matrix factorization framework called Collaborative Local Ranking (CLR). In Section 2.2, we formally set up the problem, and in Section 2.3, we prove a bound on its generalization error using low-rank matrices as hypotheses. In Section 2.4, we describe the CLR objective and explore its relationship with other learning methods. In Section 2.5, we derive our algorithm for training CLR, and analyze its correctness and running time. We present empirical results on a venue recommendation dataset in Section 2.6 and conclude in Section 2.7.

The research described in this chapter is currently under submission and it is joint work with David Rosenberg, Robert Schapire, and Tony Jebara.

2.2 Problem Formulation

In this section, we formally set up the problem by describing our assumptions about the data, the hypothesis space, and the loss function. We assume that data is generated from a sample space $\mathscr{X}$, and each data point consists of a user, a candidate set of items, a local time, and a single item that the user prefers over the remaining candidates (i.e. the label). More formally, let $\mathscr{U} = \{1, \ldots, m\}$ be the set of users, let $\mathscr{V} = \{1, \ldots, n\}$ be the set of items, and let $\mathscr{T} = \{1, \ldots, T\}$ indicate the local time. Then, the sample space is defined as

$$\mathscr{X} = \{(u, C, i, t) \mid u \in \mathscr{U}, C \subseteq \mathscr{V}, i \in C, t \in \mathscr{T}\}. \tag{2.1}$$

Given a training dataset, the learning algorithm's goal is to choose a hypothesis $g : \mathbb{Z}^3 \to \mathbb{R}$. Once the training is complete and the hypothesis is deployed, the recommendation system may be queried with a user u, candidate set C, and local time. In return, the hypothesis assigns a scalar value to each item and induces a ranking over the candidate set. For example, in case of venue recommendation, the user sends the system her geographical coordinates, the system identifies the nearby venues, forms the candidate set, applies the hypothesis, and ranks the candidate venues.

The performance of the recommendation system depends on how high it ranks the venue that the user will ultimately choose. More formally, let $P[\cdot]$ denote probability, let C^i be the set C excluding element i, and let $c \overset{U}{\sim} C$ mean that c is sampled uniformly from C. Then, the *local ranking loss* associated with hypothesis g is

$$L_g(u, C, i, t) = \underset{c \overset{U}{\sim} C^i}{P} [g(u, i, t) - g(u, c, t) \leq 0]. \tag{2.2}$$

Intuitively, the hypothesis assigns a scalar value to each item in the candidate set, and if the highest scalar value is assigned to the correct item i, there is no loss. Otherwise, the loss is proportional to the number of items in the candidate set that are ranked higher than the correct item.

To simplify the discussion, for most of the paper, we will assume that the sample space, hypotheses, and loss functions all exclude references to local time. We will revisit this issue in Section 2.4.

2.3 A Bound on the Generalization Error

Our ultimate goal is to devise algorithms that minimize the generalization error of collaborative ranking. In this section, we focus specifically on deriving a bound on the generalization error, which in turn will influence our algorithmic design.

We assume that the hypothesis class is based on the set of low-rank matrices. Given a low-rank matrix M, let $g_M \in \mathscr{F}$ be the associated hypothesis, where $g_M(u,i) = M_{u,i}$. Throughout the paper, we abuse notation and use g_M and M interchangeably. We assume that data is generated with respect to $\mathscr{D}$, which is an unknown probability distribution over the sample space $\mathscr{X}$, and we let $\mathbb{E}$ denote expectation. Then, the generalization error of hypothesis M is $\mathop{\mathbb{E}}_{(u,C,i)\sim\mathscr{D}} L_M(u,C,i)$, which is the quantity we bound below.

We will derive the generalization bound in two steps. In the first step, we will bound the empirical Rademacher complexity of our loss class, defined below, with respect to samples that contain exactly 2 candidates, and in the second step, we will prove the generalization bound with a reduction to the previous step.

Lemma 2.1. *Let m be the number of users and let n be the number of items. Define $\mathscr{L}_r = \{L_M \mid M \in \mathbb{R}^{m\times n}$ has rank at most $r\}$ as the class of loss functions associated with low-rank matrices. Assume that $S_2 \subseteq \mathscr{X}$ is a set of d samples, where each sample contains exactly 2 candidate items; i.e. if $(u,C,i) \in S_2$, then $|C| = 2$. Let $R_{S_2}(\mathscr{L}_r)$ denote the Rademacher complexity of $\mathscr{L}_r$ with respect to S_2. Then,*

$$R_{S_2}(\mathscr{L}_r) \leq \sqrt{\frac{2r(m+n)\ln\left(\frac{16emn^2}{r(m+n)}\right)}{d}}.$$

Proof. Because each sample in S_2 contains exactly 2 candidates, any hypothesis $L_M \in \mathscr{L}_r$ applied to a sample in S_2 outputs either 0 or 1. Thus, the set of dichotomies that are realized by $\mathscr{L}_r$ on S_2, called $\Pi_{\mathscr{L}_r}(S_2)$, is well-defined. Using Equation (6) from Boucheron et al. [7], we know that $R_{S_2}(\mathscr{L}_r) \leq \sqrt{\frac{2\ln|\Pi_{\mathscr{L}_r}(S_2)|}{d}}$. Let $\mathscr{X}_2 \subseteq \mathscr{X}$ be the set of all samples that contain exactly 2 candidates, $\left|\Pi_{\mathscr{L}_r}(S_2)\right| \leq \left|\Pi_{\mathscr{L}_r}(\mathscr{X}_2)\right|$, so it suffices to bound $\left|\Pi_{\mathscr{L}_r}(\mathscr{X}_2)\right|$.

We bound $\left|\Pi_{\mathscr{L}_r}(\mathscr{X}_2)\right|$ by counting the sign configurations of polynomials using proof techniques that are influenced by Srebro et al. [50]. Let $(u,\{i,j\},i) \in \mathscr{X}_2$ be

a sample and let M be a hypothesis matrix. Because M has rank at most r, it can be written as $M = UV^T$, where $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$. Let $[\![\cdot]\!]$ denote an indicator function that is 1 if and only if its argument is true. Then, the loss on the sample can also be rewritten as $L_M(u,\{i,j\},i) = [\![M_{u,i} - M_{u,j} \le 0]\!] = [\![\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,j} \le 0]\!] = [\![\sum_{a=1}^{r} U_{u,a}\left(V_{i,a} - V_{j,a}\right) \le 0]\!]$. Since cardinality of $\mathscr{X}_2$ is at most $2m\binom{n}{2} \le mn^2$, putting it all together, it follows that $\left|\Pi_{\mathscr{L}_r}(\mathscr{X}_2)\right|$ is bounded by the number of sign configurations of mn^2 polynomials, each of degree at most 2, over $r(m+n)$ variables. Applying Corollary 3 from Srebro et al. [50], we obtain $\left|\Pi_{\mathscr{L}_r}(\mathscr{X}_2)\right| \le \left(\frac{16emn^2}{r(m+n)}\right)^{r(m+n)}$. Taking logarithms and making basic substitutions yield the desired result. □

We proceed to proving the more general result via a reduction to Lemma 2.1.

Theorem 2.2. *Let m be the number of users and let n be the number of items. Assume that S consists of d independently and identically distributed samples chosen from $\mathscr{X}$ with respect to a probability distribution $\mathscr{D}$. Let L_M be the loss function associated with a matrix M, as defined in Equation 2.2. Then, with probability at least $1-\delta$, for any matrix $M \in \mathbb{R}^{m\times n}$ with rank at most r,*

$$\mathop{\mathbb{E}}_{(u,C,i)\sim\mathscr{D}} L_M(u,C,i) \le \mathop{\mathbb{E}}_{(u,C,i)\overset{U}{\sim}S} L_M(u,C,i) + 2\sqrt{\frac{2r(m+n)\ln\left(\frac{16emn}{r}\right)}{d}} + \sqrt{\frac{2\ln\left(\frac{2}{\delta}\right)}{d}}. \quad (2.3)$$

Proof. We will manipulate the definition of Rademacher complexity [7] in order to

use the bound given in Lemma 2.1:

$$
\begin{aligned}
R_S(\mathscr{L}_r) &\doteq \mathop{\mathbb{E}}_{\sigma}\left[\sup_{L_M\in\mathscr{L}_r}\left(\frac{1}{d}\sum_{a=1}^{d}\sigma_a L_M(u_a,C_a,i_a)\right)\right] && \text{(definition of } R_S(\mathscr{L}_r)\text{)}\\
&= \mathop{\mathbb{E}}_{\sigma}\left[\sup_{L_M\in\mathscr{L}_r}\left(\frac{1}{d}\sum_{a=1}^{d}\sigma_a \mathop{\mathbb{E}}_{j_a\overset{U}{\sim}(C_a\setminus\{i_a\})} L_M(u_a,\{i_a,j_a\},i_a)\right)\right] && \text{(definition of } L_M\text{)}\\
&= \mathop{\mathbb{E}}_{\sigma}\left[\sup_{L_M\in\mathscr{L}_r}\left(\mathop{\mathbb{E}}_{j_1,\dots,j_d}\frac{1}{d}\sum_{a=1}^{d}\sigma_a L_M(u_a,\{i_a,j_a\},i_a)\right)\right] && \text{(linearity of expectation)}\\
&\le \mathop{\mathbb{E}}_{\sigma}\left[\mathop{\mathbb{E}}_{j_1,\dots,j_d}\left(\sup_{L_M\in\mathscr{L}_r}\frac{1}{d}\sum_{a=1}^{d}\sigma_a L_M(u_a,\{i_a,j_a\},i_a)\right)\right] && (\sup(\mathbb{E})\le\mathbb{E}(\sup))\\
&= \mathop{\mathbb{E}}_{j_1,\dots,j_d}\left[\mathop{\mathbb{E}}_{\sigma}\left(\sup_{L_M\in\mathscr{L}_r}\frac{1}{d}\sum_{a=1}^{d}\sigma_a L_M(u_a,\{i_a,j_a\},i_a)\right)\right] && \text{(reversing expectation order)}\\
&= \mathop{\mathbb{E}}_{j_1,\dots,j_d}\left[R_{S_2}(\mathscr{L}_r)\right] && \left(\text{definition of } R_{S_2}(\mathscr{L}_r)\right)\\
&\le \sqrt{\frac{2r(m+n)\ln\left(\frac{16emn^2}{r(m+n)}\right)}{d}} && \text{(by Lemma 2.1)}
\end{aligned}
$$

Plugging the bound to Theorem 3.2 in Boucheron et al. [7] proves the theorem. □

Srebro et al. [50] used covering numbers to prove a generalization error bound for collaborative binary classification, which is matrix factorization framework where the matrix entries are binary and the loss function is 0-1. Even though their learning setting is different from ours, we can still compare our bound in Theorem 2.2 with their bound, and it can be seen that they match up to logarithmic factors. Thus, collaborative local ranking maintains the same generalization error as collaborative binary classification in exchange for a small increase in sample size.

We can improve the tightness of our generalization error bound by restricting the sample space $\mathscr{X}$. In our proof of Lemma 2.1, we loosely bounded the cardinality of the sample space with $|\mathscr{X}_2| \le mn^2$, but many collaborative ranking tasks have additional structure which lead to better bounds. For example, in case of venue recommendation, $\mathscr{X}_2$ is restricted because two venues are only assigned to the same candidate set if they are near each other. If we assume that every venue has at most b nearby venues, then $|\mathscr{X}_2| \le mnb$, and the n in the logarithm of Equation 2.3 gets replaced by a small constant. Thus, by making additional assumptions about

the structure of the sample space, the generalization error bound in Theorem 2.2 becomes asymptotically equivalent to the bound provided in [50], even though the latter bound was derived for the simpler collaborative classification setting.

2.4 Collaborative Local Ranking

In this section, we describe CLR by formulating its objective function, which we justify using the generalization error bound derived above. We also make connections between CLR and other learning methods, such as maximum-margin matrix factorization (MMMF) [43] and ranking support vector machine (ranking SVM) [23].

Given the generalization error bound in Theorem 2.2, a reasonable objective is to minimize the empirical local ranking loss, which is the first term on the right hand side of Equation 2.3. However, this function is discontinuous and difficult to minimize with respect to M, so we propose minimizing a more tractable upper bound instead. Let $h(x) = \max(0, 1-x)$ be the hinge function, let M be the hypothesis matrix with rank at most r, and let $M = UV^T$, where $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$. Then, we can bound the empirical local ranking loss as

$$\begin{aligned}
\mathop{\mathbb{E}}_{(u,C,i) \overset{U}{\sim} S} L_M(u,C,i) &= \frac{1}{|S|} \sum_{(u,C,i) \in S} L_M(u,C,i) \\
&= \frac{1}{|S|} \sum_{(u,C,i) \in S} \mathop{P}_{c \overset{U}{\sim} C^i} \left[M_{u,i} - M_{u,c} \leq 0 \right] \\
&= \frac{1}{|S|} \sum_{(u,C,i) \in S} \mathop{\mathbb{E}}_{c \overset{U}{\sim} C^i} [\![M_{u,i} - M_{u,c} \leq 0]\!] \\
&= \frac{1}{|S|} \sum_{(u,C,i) \in S} \frac{1}{|C^i|} \sum_{c \in C^i} [\![\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c} \leq 0]\!] \\
&\leq \frac{1}{|S|} \sum_{(u,C,i) \in S} \frac{1}{|C^i|} \sum_{c \in C^i} h\left(\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c} \right). \qquad (2.4)
\end{aligned}$$

The resulting upper bound is not necessarily convex with respect to U and V jointly, but it is convex with respect to U if V is fixed, and vice versa.

We use trace norm regularization, in addition to the low-rank constraint, to further restrict the hypothesis class. Let $\|\cdot\|_T$ denote the trace norm and let $\|\cdot\|_F$ denote

the Frobenius norm, then we use the equality $\|M\|_T = \min_{U,V,M=UV^T} \frac{1}{2}\left(\|U\|_F^2 + \|V\|_F^2\right)$ given in Lemma 6 of Mazumder et al. [36] to state the objective in its final form.

Definition 2.3 (CLR Objective)**.** Let S be the training data. Let $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{n\times r}$ be the factor matrices, and let $\lambda > 0$ be the regularization parameter. The CLR objective is

$$f^{\mathrm{CLR}}(S;U,V) = \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) + \frac{1}{|S|}\sum_{(u,C,i)\in S}\frac{1}{|C^i|}\sum_{c\in C^i} h\left(\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c}\right).$$

If $\lambda = 0$, then the CLR objective is equivalent to minimizing the bound in Equation 2.4. If $\lambda \neq 0$, then it can be shown that the equivalence still holds, but under the constraint that matrices have bounded trace norm in addition to having bounded rank. Even though minimizing Equation 2.4 is more directly justified by our generalization bound, we use the CLR objective instead, which is more general, to allow additional regularization. Note that the resulting hypothesis still satisfies the assumptions of Theorem 2.2. Rank-truncated trace norms have been used as regularizers in other collaborative learning settings, such as by Foygel et al. [18] and by Rennie and Srebro [43], and have been demonstrated to work well.

We note that the CLR and the ranking SVM [23] objectives are closely related. If V is fixed and we only need to minimize U, then each row of V acts as a feature vector for the corresponding item, each row of U acts as a separate linear predictor, and the CLR objective decomposes into solving simultaneous ranking SVM problems. In particular, let $S_u = \{(a,C,i) \in S \mid a = u\}$ be the examples that correspond to user u, let U_u denote row u of U, and let f^{rSVM} denote the objective function of ranking SVM, then

$$\begin{aligned} f^{\mathrm{CLR}}(S;U,V) &= \frac{\lambda}{2}\|U\|_F^2 + \frac{1}{|S|}\sum_{(u,C,i)\in S}\frac{1}{|C^i|}\sum_{c\in C^i} h\left(\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c}\right) \\ &= \sum_{u=1}^{m}\left[\frac{\lambda}{2}\|U_u\|_F^2 + \frac{1}{|S|}\sum_{(u,C,i)\in S_u}\frac{1}{|C^i|}\sum_{c\in C^i} h\left(\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c}\right)\right] \\ &= \sum_{u=1}^{m} f^{\mathrm{rSVM}}(S_u;U_u,V). \end{aligned}$$

This correspondence is not surprising, since in the ranking SVM setting, items

have features, the training data consists of binary orderings between items, and the ranking SVM objective itself is a convex upper bound on the number of misordered pairs. Additionally, just like optimizing the CLR objective decomposes into solving simultaneous ranking SVM objectives, optimizing the MMMF [43] objective also decomposes into solving simultaneous binary SVM objectives, and this observation influences how we design our algorithms in the next section.

Lastly, we extend the CLR objective to incorporate features. In the context of venue recommendation, such features might be obtained from an external database and indicate the venue type or the query time stamp. We assume an extended sample space, and given a sample $(u, C, i, t) \in \mathscr{X}$, we let t denote the query time stamp.

Definition 2.4 (CLR.F Objective)**.** Let S be the training data. Given a sample $(u, C, i, t) \in S$, assume that $F_{u,i,t} \in \mathbb{R}^q$ denotes the corresponding feature vector. Let $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$ be the factor matrices, let $w \in \mathbb{R}^q$ be the feature coefficients, and let $\lambda, \gamma > 0$ be the regularization parameters. The CLR.F objective is

$$f^{\text{CLR.F}}(S; U, V, w) = \frac{\lambda}{2}\left(\|U\|_F^2 + \|V\|_F^2\right) + \frac{\gamma}{2}\|w\|^2 + \frac{1}{|S|}\sum_{(u,C,i,t)\in S}\frac{1}{\left|C^i\right|}\sum_{c \in C^i} h\left(\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c} + w^T\left(F_{u,i,t} - F_{u,c,t}\right)\right).$$

2.5 Algorithms

2.5.1 Derivation

In this subsection, we derive and describe our algorithm for optimizing the CLR objective. As we noted before, the CLR objective is not necessarily jointly convex in U and V, but it is convex in U when V is fixed and vice versa. We minimize the CLR objective using alternating minimization, where we sequentially alternate between solving the convex subproblems, and we solve each such subproblem using projected stochastic subgradient descent. The pseudocode is depicted in Algorithm 2.1.

Now, we derive the projected stochastic subgradient descent algorithm for minimizing V while keeping U fixed. At each iteration, the algorithm approximates

Algorithm 2.1 Alternating minimization for optimizing the CLR objective.

Require: Training data $S \subseteq \mathcal{X}$, regularization parameter $\lambda > 0$, rank constraint r, number of iterations T.

1: $U_1 \leftarrow$ Sample matrix uniformly at random from $\left[-\frac{1}{\sqrt{\lambda m r}}, \frac{1}{\sqrt{\lambda m r}}\right]^{m \times r}$.
2: $V_1 \leftarrow$ Sample matrix uniformly at random from $\left[-\frac{1}{\sqrt{\lambda n r}}, \frac{1}{\sqrt{\lambda n r}}\right]^{n \times r}$.
3: **for all** t from 1 to $T-1$ **do**
4: $\quad U_{t+1} \leftarrow \arg\min_{U} f^{\mathrm{CLR}}(S; U, V_t)$
5: $\quad V_{t+1} \leftarrow \arg\min_{V} f^{\mathrm{CLR}}(S; U_{t+1}, V)$
6: **return** U_T, V_T.

the objective function based on an example selected at random, updates the weight vector using the approximate subgradient, and projects the weights onto a bounded ball. Let $(u, C, i) \in S$ be an example, then the corresponding approximate objective function is

$$f^{\mathrm{CLR}}((u, C, i); U, V) = \frac{\lambda}{2}\|V\|_F^2 + \frac{1}{\left|C^i\right|} \sum_{c \in C^i} h\left(\left(UV^T\right)_{u,i} - \left(UV^T\right)_{u,c}\right).$$

We introduce various matrix notation to help us define the approximate subgradients. Given a matrix M, let $M_{k,\cdot}$ denote row k of M. Define the matrix $\hat{M}^{p,q,z}$, for $p \neq q$, as

$$\hat{M}^{p,q,z}_{s,\cdot} = \begin{cases} M_{z,\cdot} & \text{for } s = p, \\ -M_{z,\cdot} & \text{for } s = q, \\ 0 & \text{otherwise,} \end{cases} \tag{2.5}$$

and define the matrix $\check{M}^{p,q,z}_{s,\cdot}$ as

$$\check{M}^{p,q,z}_{s,\cdot} = \begin{cases} M_{p,\cdot} - M_{q,\cdot} & \text{for } s = z, \\ 0 & \text{otherwise.} \end{cases} \tag{2.6}$$

Then, the subgradient of the approximate objective function with respect to V is

$$\nabla_V f^{\mathrm{CLR}}((u,C,i);U,V) \;=\; \lambda V - \frac{1}{|C^i|}\sum_{c\in C^i} \mathbb{I}\Big(UV^T\Big)_{u,i} - \Big(UV^T\Big)_{u,c} \;<\; 1\mathbb{I}\hat{U}^{i,c,u}. \quad (2.7)$$

Setting $\eta_t = \frac{1}{\lambda t}$ as the learning rate at iteration t, the approximate subgradient update becomes $V_{t+1} = V_t - \eta_t \nabla_V f^{\mathrm{CLR}}((u,C,i);U,V)$. After the update, the weights are projected onto a ball with radius $\frac{1}{\sqrt{\lambda}}$. The pseudocode for optimizing both convex subproblems is depicted in Algorithms 2.2 and 2.3. We prove the correctness of the algorithms and bound their running time in the next subsection.

Algorithm 2.2 Projected stochastic subgradient descent for optimizing U.

Require: Factors $V \in \mathbb{R}^{n\times r}$, training data S, regularization parameter λ, rank constraint r, number of iterations T.

$U_1 \leftarrow 0^{m\times r}$
for all t from 1 to $T-1$ **do**
  Choose $(u,C,i) \in S$ uniformly at random.
  $\eta_t \leftarrow \frac{1}{\lambda t}$
  $C^+ \leftarrow \Big\{c \in C^i \mid \big(U_t V^T\big)_{u,i} - \big(U_t V^T\big)_{u,c} < 1\Big\}$
  $U_{t+1} \leftarrow \big(1-\eta_t\lambda\big)U_t + \frac{\eta_t}{|C^i|}\sum_{c\in C^+} \check{V}^{i,c,u}$
  $U_{t+1} \leftarrow \min\Big\{1, \frac{1}{\sqrt{\lambda}\|U_{t+1}\|_F}\Big\} U_{t+1}$
return U_T.

Algorithm 2.3 Projected stochastic subgradient descent for optimizing V.

Require: Factors $U \in \mathbb{R}^{m\times r}$, training data S, regularization parameter λ, rank constraint r, number of iterations T.

$V_1 \leftarrow 0^{n\times r}$
for all t from 1 to $T-1$ **do**
  Choose $(u,C,i) \in S$ uniformly at random.
  $\eta_t \leftarrow \frac{1}{\lambda t}$
  $C^+ \leftarrow \Big\{c \in C^i \mid \big(UV_t^T\big)_{u,i} - \big(UV_t^T\big)_{u,c} < 1\Big\}$
  $V_{t+1} \leftarrow \big(1-\eta_t\lambda\big)V_t + \frac{\eta_t}{|C^i|}\sum_{c\in C^+} \hat{U}^{i,c,u}$
  $V_{t+1} \leftarrow \min\Big\{1, \frac{1}{\sqrt{\lambda}\|V_{t+1}\|_F}\Big\} V_{t+1}$
return V_T.

We conclude the subsection by commenting on the implementation details. A

naive implementation of Algorithm 2.2 would execute each iteration in time $\Omega(bmnr)$, where b denotes the size of the largest candidate set. One can reduce the running time of each iteration considerably by normalizing the matrix efficiently. We represent U as a triplet (W, a, v), where a is a scalar, $U = aW$, and $\|U\|_F = v$. It can be verified that, using the new representation, a single iteration in both Algorithms 2.2 and 2.3 can be executed in time $O(br)$.

2.5.2 Analysis

In this subsection, we analyze the running time and correctness of our algorithms. In particular, we prove that projected stochastic subgradient descent converges to the correct solution in time *independent* of the size of the training data. We do not bound the total number of alternating minimization steps a priori; nevertheless, we demonstrate that our methods are especially suitable for large datasets since they solve each individual minimization problem efficiently.

The convex subproblems we analyze have the general form

$$\min_{X \in D} f(X; \ell) = \min_{X \in D} \frac{\lambda}{2} \|X\|_F^2 + \frac{1}{|S|} \sum_{(u,C,i) \in S} \ell(X; (u, C, i)). \tag{2.8}$$

One can obtain the individual subproblems by specifying the domain D and the loss function ℓ. For example, in case of Algorithm 2.2, the corresponding minimization problem is specified by

$$\min_{X \in \mathbb{R}^{m \times r}} f(X; \ell_V) \text{ where } \ell_V(X; (u, C, i)) = \frac{1}{|C^i|} \sum_{c \in C^i} h\left(\left(XV^T\right)_{u,i} - \left(XV^T\right)_{u,c}\right), \tag{2.9}$$

and in case of Algorithm 2.3, it is specified by

$$\min_{X \in \mathbb{R}^{n \times r}} f(X; \ell_U) \text{ where } \ell_U(X; (u, C, i)) = \frac{1}{|C^i|} \sum_{c \in C^i} h\left(\left(UX^T\right)_{u,i} - \left(UX^T\right)_{u,c}\right). \tag{2.10}$$

Let $U^\star = \arg\min_U f(U; \ell_V)$ and $V^\star = \arg\min_V f(V; \ell_U)$ denote the solution matrices of Equations 2.9 and 2.10, respectively. Also, given a general convex loss ℓ and domain D, let $\bar{X} \in D$ be an ϵ-accurate solution for the corresponding minimization problem if $f\left(\bar{X}; \ell\right) \leq \min_{X \in D} f(X; \ell) + \epsilon$.

In the remainder of this subsection, we show that Algorithms 2.2 and 2.3 are adaptations of the Pegasos [47] algorithm to the CLR setting. Then, we prove certain properties that are prerequisites for obtaining Pegasos's performance guarantees. In particular, we show that the approximate subgradients computed by Algorithms 2.2 and 2.3 are bounded and the loss functions associated with Equations 2.9 and 2.10 are convex. In the end, we plug these properties into a theorem proved by Shalev-Shwartz et al. to show that our algorithms reach an ϵ-accurate solution with respect to their corresponding minimization problems in $\tilde{O}\left(\frac{1}{\lambda^2\epsilon}\right)$ iterations.

Lemma 2.5. $\left\|U^\star\right\| \leq \frac{1}{\sqrt{\lambda}}$ *and* $\left\|V^\star\right\| \leq \frac{1}{\sqrt{\lambda}}$.

Proof. One can obtain the bounds on the norms of the optimal solutions by examining the dual form of the optimization problems and applying the strong duality theorem. Equations 2.9 and 2.10 can both be represented as

$$\min_{v \in D} \frac{1}{2}\|v\|^2 + \sum_{k=1}^{K} e_k h(f_k(v)), \tag{2.11}$$

where $e_k = \frac{1}{\lambda|S||C_k|}$ is a constant, h is the hinge function, D is a Euclidean space, and f_k is a linear function. We rewrite Equation 2.11 as a constrained optimization problem

$$\begin{aligned} \min_{v \in D, \xi \in \mathbb{R}^K} \quad & \frac{1}{2}\|v\|^2 + \sum_{k=1}^{K} e_k \xi_k & & \\ \text{subject to} \quad & \xi_k \geq 1 - f_k(v), & & k = 1, \ldots K, \\ & \xi_k \geq 0, & & k = 1, \ldots K. \end{aligned} \tag{2.12}$$

The Lagrangian of this problem is

$$\begin{aligned} L(v, \xi, \alpha, \beta) &= \frac{1}{2}\|v\|^2 + \sum_{k=1}^{K} e_k \xi_k + \sum_{k=1}^{K} \alpha_k (1 - f_k(v) - \xi_k) - \sum_{k=1}^{K} \beta_k \xi_k \\ &= \frac{1}{2}\|v\|^2 + \sum_{k=1}^{K} \xi_k (e_k - \alpha_k - \beta_k) + \sum_{k=1}^{K} \alpha_k (1 - f_k(v)), \end{aligned}$$

and its dual function is

$$g(\alpha, \beta) = \inf_{v, \xi} L(v, \xi, \alpha, \beta).$$

Since $L\left(v,\xi,\alpha,\beta\right)$ is convex and differentiable with respect to v and ξ, the necessary and sufficient conditions for minimizing v and ξ are

$$\begin{aligned} \nabla_v L = 0 \qquad & \Leftrightarrow \qquad v = \sum_{k=1}^{K} \alpha_k \nabla_v f_k(v), \\ \nabla_\xi L = 0 \qquad & \Leftrightarrow \qquad e = \alpha + \beta. \end{aligned} \tag{2.13}$$

We plug these conditions back into the dual function and obtain

$$\begin{aligned} g\left(\alpha,\beta\right) &= \inf_{v,\xi} L\left(v,\xi,\alpha,\beta\right) \\ &= \frac{1}{2}\left\| \sum_{k=1}^{K} \alpha_k \nabla_v f_k(v) \right\|^2 + \sum_{k=1}^{K} \alpha_k \left(1 - f_k\left(\sum_{k=1}^{K} \alpha_k \nabla_v f_k(v)\right)\right) \\ &= \frac{1}{2}\left\| \sum_{k=1}^{K} \alpha_k \nabla_v f_k(v) \right\|^2 + \sum_{k=1}^{K} \alpha_k - \sum_{k=1}^{K} \alpha_k f_k\left(\sum_{k=1}^{K} \alpha_k \nabla_v f_k(v)\right). \end{aligned} \tag{2.14}$$

Since f_k is a linear function, we let $f_k(v) = \vec{k} \cdot v$, where $\vec{k}$ is a constant vector, and $\nabla_v f_k(v) = \vec{k}$. Then,

$$\begin{aligned} \left\| \sum_{k=1}^{K} \alpha_k \nabla_v f_k(v) \right\|^2 &= \left\| \sum_{k=1}^{K} \alpha_k \vec{k} \right\|^2 \\ &= \left(\sum_{k=1}^{K} \alpha_k \vec{k}\right) \cdot \left(\sum_{k=1}^{K} \alpha_k \vec{k}\right) \\ &= \sum_{k=1}^{K} \alpha_k \vec{k} \cdot \left(\sum_{k=1}^{K} \alpha_k \vec{k}\right) \\ &= \sum_{k=1}^{K} \alpha_k f_k \left(\sum_{k=1}^{K} \alpha_k \vec{k}\right) \\ &= \sum_{k=1}^{K} \alpha_k f_k \left(\sum_{k=1}^{K} \alpha_k \nabla_v f_k(v)\right). \end{aligned} \tag{2.15}$$

Simplifying Equation 2.14 using Equation 2.15 yields

$$\begin{aligned} g(\alpha,\beta) &= -\frac{1}{2}\left\|\sum_{k=1}^{K}\alpha_k\nabla_v f_k(v)\right\|^2 + \sum_{k=1}^{K}\alpha_k. \\ &= -\frac{1}{2}\left\|\sum_{k=1}^{K}\alpha_k\vec{k}\right\|^2 + \sum_{k=1}^{K}\alpha_k \end{aligned} \tag{2.16}$$

Finally, we combine Equations 2.13 and 2.16, and obtain the dual form of Equation 2.12,

$$\begin{aligned} &\max_{\alpha} && -\frac{1}{2}\left\|\sum_{k=1}^{K}\alpha_k\vec{k}\right\|^2 + \sum_{k=1}^{K}\alpha_k && \\ &\text{subject to} && 0 \le \alpha_k \le e_k, && k = 1,\dots K. \end{aligned} \tag{2.17}$$

The primal problem is convex, its constraints are linear, and the domain of its objective is open; thus, Slater's condition holds and strong duality is obtained. Furthermore, the primal problem has differentiable objective and constraint functions, which implies that $(v^\star,\xi^\star)$ is primal optimal and $(\alpha^\star,\beta^\star)$ is dual optimal if and only if these points satisfy the Karush-Kuhn-Tucker (KKT) conditions. It follows that

$$v^\star = \sum_{k=1}^{K}\alpha_k^\star\vec{k}. \tag{2.18}$$

Note that we defined $e_k = \frac{1}{\lambda|S||C_k|}$, where $\sum_{k=1}^{K} e_k = \frac{1}{\lambda}$, and the constraints of the dual problem imply $0 \le \alpha_k \le e_k$; thus, $\sum_{k=1}^{K}\alpha_k^\star \le \frac{1}{\lambda}$. Because of strong duality, there is no duality gap, and the primal and dual objectives are equal at the optimum,

$$\begin{aligned} \frac{1}{2}\left\|v^\star\right\|^2 + \sum_{k=1}^{K} e_k\xi_k^\star &= -\frac{1}{2}\left\|\sum_{k=1}^{K}\alpha_k^\star\vec{k}\right\|^2 + \sum_{k=1}^{K}\alpha_k^\star \\ &= -\frac{1}{2}\left\|v^\star\right\|^2 + \sum_{k=1}^{K}\alpha_k^\star && \text{(by Equation 2.18)} \\ &\le -\frac{1}{2}\left\|v^\star\right\|^2 + \frac{1}{\lambda} \\ \Rightarrow \left\|v^\star\right\|^2 &\le \frac{1}{\lambda} \end{aligned}$$

This proves the lemma. □

Given the bounds in Lemma 2.5, it can be verified that Algorithms 2.2 and 2.3 are adaptations of the Pegasos [47] algorithm for optimizing Equations 2.9 and 2.10, respectively. It still remains to show that Pegasos's performance guarantees hold in our case.

Lemma 2.6. *In Algorithms 2.2 and 2.3, the approximate subgradients have norm at most* $\sqrt{\lambda} + 2\sqrt{\frac{1}{\lambda}}$.

Proof. The approximate subgradient for Algorithm 2.3 is depicted in Equation 2.7. Due to the projection step, $\|V\|_F \leq \frac{1}{\sqrt{\lambda}}$, and it follows that $\|\lambda V\|_F \leq \sqrt{\lambda}$. The term $\hat{U}^{i,c,u}$ is constructed using Equation 2.5, and it can be verified that $\left\|\hat{U}^{i,c,u}\right\|_F \leq \sqrt{2}\|U\|_F \leq \sqrt{\frac{2}{\lambda}}$. Using triangle inequality, one can bound Equation 2.7 with $\sqrt{\lambda} + \sqrt{\frac{2}{\lambda}}$. A similar argument can be made for the approximate subgradient of Algorithm 2.2, yielding the slightly higher upper bound given in the lemma statement. □

We combine the lemmas to obtain the correctness and running time guarantees for our algorithms.

Lemma 2.7. *Let* $\lambda \leq \frac{1}{4}$*, let* T *be the total number of iterations of Algorithm 2.2, and let* U_t *denote the parameter computed by the algorithm at iteration* t*. Let* $\bar{U} = \frac{1}{T}\sum_{t=1}^{T} U_t$ *denote the average of the parameters produced by the algorithm. Then, with probability at least* $1-\delta$*,*

$$f\left(\bar{U}; \ell_V\right) \leq f\left(U^\star; \ell_V\right) + \frac{21\left(\sqrt{\lambda} + 2\sqrt{\frac{1}{\lambda}}\right)^2 \ln\left(\frac{T}{\delta}\right)}{\lambda T}.$$

The analogous result holds for Algorithm 2.3 as well.

Proof. First, for each loss function ℓ_V and ℓ_U, variables are linearly combined, composed with the convex hinge function, and then averaged. All these operations preserve convexity, hence both loss functions are convex. Second, we have argued above that Algorithms 2.2 and 2.3 are adaptations of the Pegasos [47] algorithm for optimizing Equations 2.9 and 2.10, respectively. Third, in Lemma 2.6, we proved a bound on the approximate subgradients of both algorithms. Plugging these three

results into Corollary 2 in Shwartz et al. [47] yields the statement of the theorem. □

The theorem below gives a bound in terms of individual parameters rather than average parameters.

Theorem 2.8. *Assume that the conditions and the bound in Lemma 2.7 hold. Let t be an iteration index selected uniformly at random from $\{1,\ldots,T\}$. Then, with probability at least $\frac{1}{2}$,*

$$f(U_t;\ell_V) \leq f\left(U^\star;\ell_V\right) + \frac{42\left(\sqrt{\lambda}+2\sqrt{\frac{1}{\lambda}}\right)^2 \ln\left(\frac{T}{\delta}\right)}{\lambda T}.$$

The analogous result holds for Algorithm 2.3 as well.

Proof. The result follows directly from combining Lemma 2.7 with Lemma 3 in Shwartz et al. [47]. □

Thus, with high probability, our algorithms reach an ϵ-accurate solution in $\tilde{O}\left(\frac{1}{\lambda^2\epsilon}\right)$ iterations. Since we argued in subsection 2.5.1 that the running time of each stochastic update is $O(br)$, it follows that a complete run of projected stochastic subgradient descent takes $\tilde{O}\left(\frac{br}{\lambda^2\epsilon}\right)$ time, and the running time is *independent* of the size of the training data.

2.6 Experiments

In this section, we provide an empirical analysis of the CLR framework by applying it to a venue recommendation task. We assess CLR's generalization and running time performance by comparing it against CofiRank, an algorithm which is representative of the state-of-the-art in collaborative ranking.

We created our dataset by collecting publicly available "check-ins" via the Twitter and Foursquare APIs. A check-in is a virtual announcement where a user shares her whereabouts with other people on her social network. A check-in contains a user ID, a venue ID (e.g. "Shake Shack"), and a local time; optionally, it may also contain a text (e.g. "Trying the shackburger!!"), photos, and other metadata. We collected public check-ins that occurred in New York City during the 9 month

period ranging from January 2011 to September 2011. We filtered our dataset to only include users who have checked in at least 5 times and venues which have been checked into at least 5 times, yielding a total of 13750 users, 11700 venues, and 269597 check-ins. We note that all the check-ins we collected were shared with the entire Internet, so our dataset does not contain any private information.

At a high level, our goal is to build a venue recommendation application for mobile platforms, where the user specifies a geographical region she is interested in exploring, and the application returns a personalized ranking over the venues in that region. If such an application already existed, it would have been an ideal fit for the CLR setting for the following reasons: 1) the regions explored by the users are generated by an unknown probability distribution; 2) when checking into a venue, the user implicitly prefers it over the remaining venues in the explored region; 3) the application's goal is to provide a relevant ranking over the venues by minimizing the local ranking loss. Even though the Foursquare dataset we collected is not guaranteed to be generated under similar assumptions, we use it as a proxy regardless, and assume that when a user checks into a venue, she is implicitly preferring that venue over the remaining venues in a specified radius. We note that similar assumptions about implicit preferences have been made by Joachims [23] in the supervised learning setting, where the author introduced ranking SVMs to rerank search engine results using clickthrough logs.

We partition the check-ins chronologically, and place the earliest 60% of check-ins to train, the subsequent 20% to validation, and final 20% to test sets. Each check-in corresponds to a tuple (u,i), where u is the user and i is the checked-in venue. In order to apply CLR, we augment each check-in with a candidate set C, which consists of venues within a specified distance to i, and form the CLR sample (u,C,i). We train a variety of CLR and CofiRank models on a combined train and validation set, determine the best performing model parameters using the validation set, and report the performance of the chosen parameters on the test set. The model parameters include the rank $r \in \{2,5,10,20\}$, CofiRank regularization parameter $\lambda \in \{10,100,1000,10000\}$, and CLR regularization parameter $\lambda \in \{0.01,0.001,0.0001,0.00001,0.000001\}$. For testing, we form queries with a user u and a candidate set C, hide the label i, and measure the algorithms' performances accordingly. We vary the radius that determines the size of the candidate

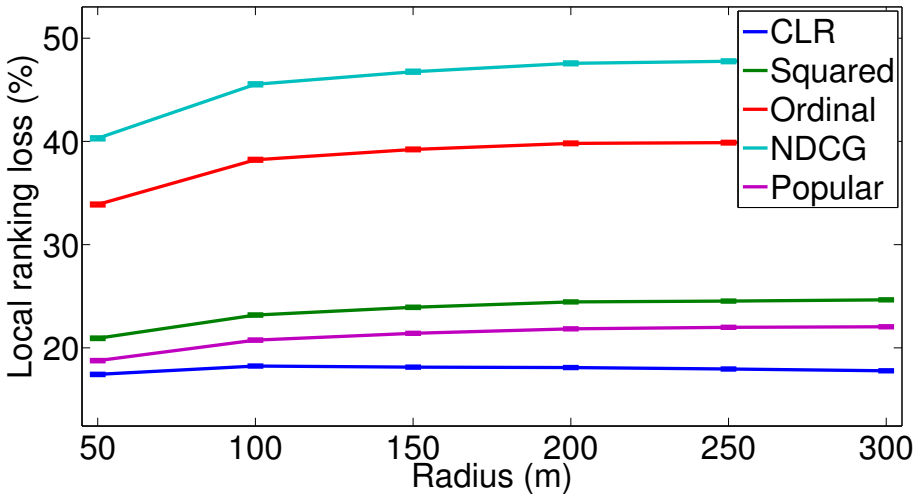

Figure 2.1: The local ranking loss of various algorithms on the held-out test set. CofiRank performs best when coupled with squared loss, "Popular" performs better than CofiRank, and CLR outperforms them all. The error bars correspond to standard error and are barely noticeable.

set C from 50m to 300m, in 50m increments.

Figure 2.1 displays the local ranking loss of various algorithms on the held-out test set. We compare CLR against both CofiRank and a baseline algorithm called "Popular", which simply outputs the most frequently checked in venue from the candidate set. We execute CofiRank with three different loss functions: ordinal, NDCG, and squared [53]. CofiRank performs best when coupled with the squared loss, where it effectively mimics MMMF [43]; however, the "Popular" baseline outperforms CofiRank. Our algorithm, CLR, outperforms all methods and achieves the lowest local ranking loss. We note that, unlike competing methods, CLR does not suffer an additional loss when the radius and size of its candidate set increases.

Figure 2.2 displays the recall@k performance on the held-out test set, for $k = \{1, 5, 10\}$. The order of algorithms with respect to recall performance is exactly the same as their order with respect to local ranking loss. For recall@1, the "Popular" baseline outperforms all versions of CofiRank, but as the radius of candidate sets increase beyond 150m, the performance of CofiRank-squared matches that of "Popular". In contrast, CLR outperforms them all. For recall@5 and recall@10, performance of CLR and "Popular" coincide up to 100m radius, but CLR outperforms competing methods for higher radii.

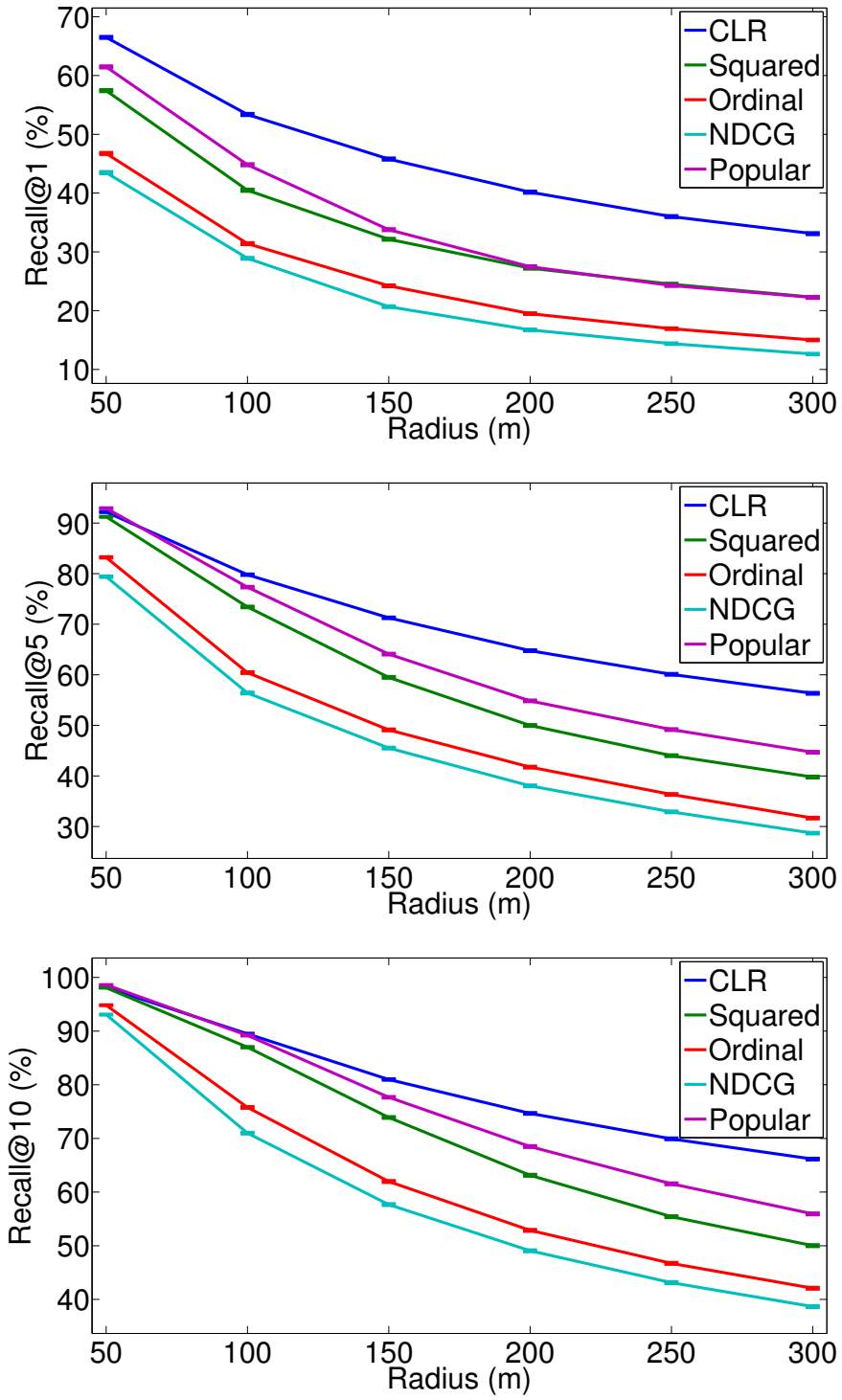

Figure 2.2: Recall@k of various algorithms on the held-out test set, for $k = \{1, 5, 10\}$. CLR outperforms all competing methods. The error bars correspond to standard error and are barely noticeable.

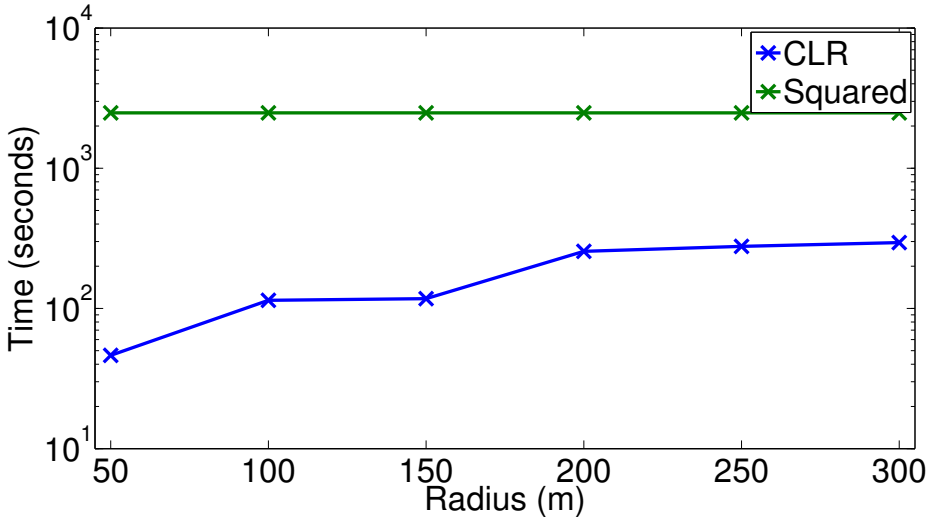

Figure 2.3: Training time of CLR and CofiRank in log-scale. We report the results for CofiRank-squared, which is the best performing CofiRank variant. CLR trains orders of magnitude faster than CofiRank, even when we increase the size of the radius and candidate sets.

We also empirically analyze the running time of CofiRank and CLR. For both algorithms, for each radius, we choose the parameter settings which perform the best on the validation set, and record their respective training time. We plot the results using log-scale in Figure 2.3. Even though the running time of CLR has a linear dependence on the size of the candidate sets; nevertheless, CLR trains orders of magnitude faster than CofiRank, as suggested by our theoretical analysis in Subsection 2.5.2.

2.7 Discussion

In this chapter, we formulated a new collaborative ranking framework, called Collaborative Local Ranking, which allows us to formulate a wide variety of real-world ranking tasks in a collaborative setting. We justified CLR with a bound on its generalization error. We also derived a simple alternating minimization algorithm and showed that each minimization step can be efficiently computed in time independent of the size of the training data. We applied CLR to a venue recommendation task and demonstrated that it outperforms state-of-the-art collaborative ranking methods, such as CofiRank, both in terms of generalization performance and run-

ning time.

Chapter 3

Collaborative Place Models

3.1 Background

During the last three years, positioning devices that measure and record our locations have become ubiquitous. The most common positioning device, the smartphone, is projected to be used by a billion people in the near future [11]. This surge in positioning devices has increased the availability of location data, and provided scientists with new research opportunities, such as building location-based recommendation systems [21, 58, 59], analyzing human mobility patterns [8, 19, 48], and modeling the spread of diseases [16].

For many location-based tasks, a fundamental problem is predicting users' future locations. For example, a navigation application that has access to traffic conditions can warn the user about when to depart, without requiring any input from the user, as long as the application can accurately predict the arrival time and location of user's next destination. Similarly, a restaurant application can provide a list of recommended venues, even reserve them while space is still available, by modeling the probable locations the user might visit in the evening. Predicting future locations based on past ones is a difficult problem, and to solve it, one needs efficient algorithms that identify the structures and routines hidden in users' past locations.

Complicating the problem further, in many real-world applications, location datasets are sparse. Due to privacy considerations [54] and high energy consumption of positioning hardware [39], most mobile phone users only allow applications

to log their location when the application is active, but not when it is running in the background. Consequently, compared to previous datasets used in location prediction research, many real-world datasets have much less information available for each user, as depicted in Figure 3.1. Even when users share very little with their applications, they nevertheless expect these applications to infer as much as possible; thus, there is a need for location prediction algorithms that are designed with sparsity in mind.

There have been various studies on location prediction. Gao et al. designed a Markov model that takes temporal context into account in order to predict user's location in the immediate future. Cho et al. [9] proposed a two-state mixture of Gaussians that leverages the social relationships between users, but they limited their model to only represent "home" and "work". De Domenico et al. presented a location prediction algorithm that won the Nokia Mobile Data Challenge (MDC), where the dataset consisted of tens of thousands of GPS observations, collected every few minutes and over the span of a year. Their algorithm exploited the high density of the location dataset, as well as the social relationships between users, by comparing all historical trajectories that spanned a day and making a forecast based on the similarities between such trajectories.

In addition to location prediction, there has also been studies on algorithms that detect significant places and routines in location data. Eagle and Pentland [14] applied eigendecomposition to the Reality Mining dataset, where all locations were already labeled as "home" or "work, and extracted users' daily routines. Farrahi and Gatica-Perez [17] used the same dataset, but extracted the routines using Latent Dirichlet Allocation (LDA) instead of eigendecomposition. Liao et al. [31, 32] proposed a hierarchical conditional random field to identify activities and significant places from the users' GPS traces, and since their algorithm was supervised, it required locations to be manually labeled for training.

In this chapter, we propose two unsupervised Bayesian probabilistic models for location prediction that are designed specifically for sparse location datasets. Both models assume that each user is characterized by a number of place clusters, whose spatial extents, such as their means and covariances, are determined probabilistically from the data. Similar to a mixed-membership model [15], we allow the place clusters to be global to each user, but restrict user's preferences over the clusters

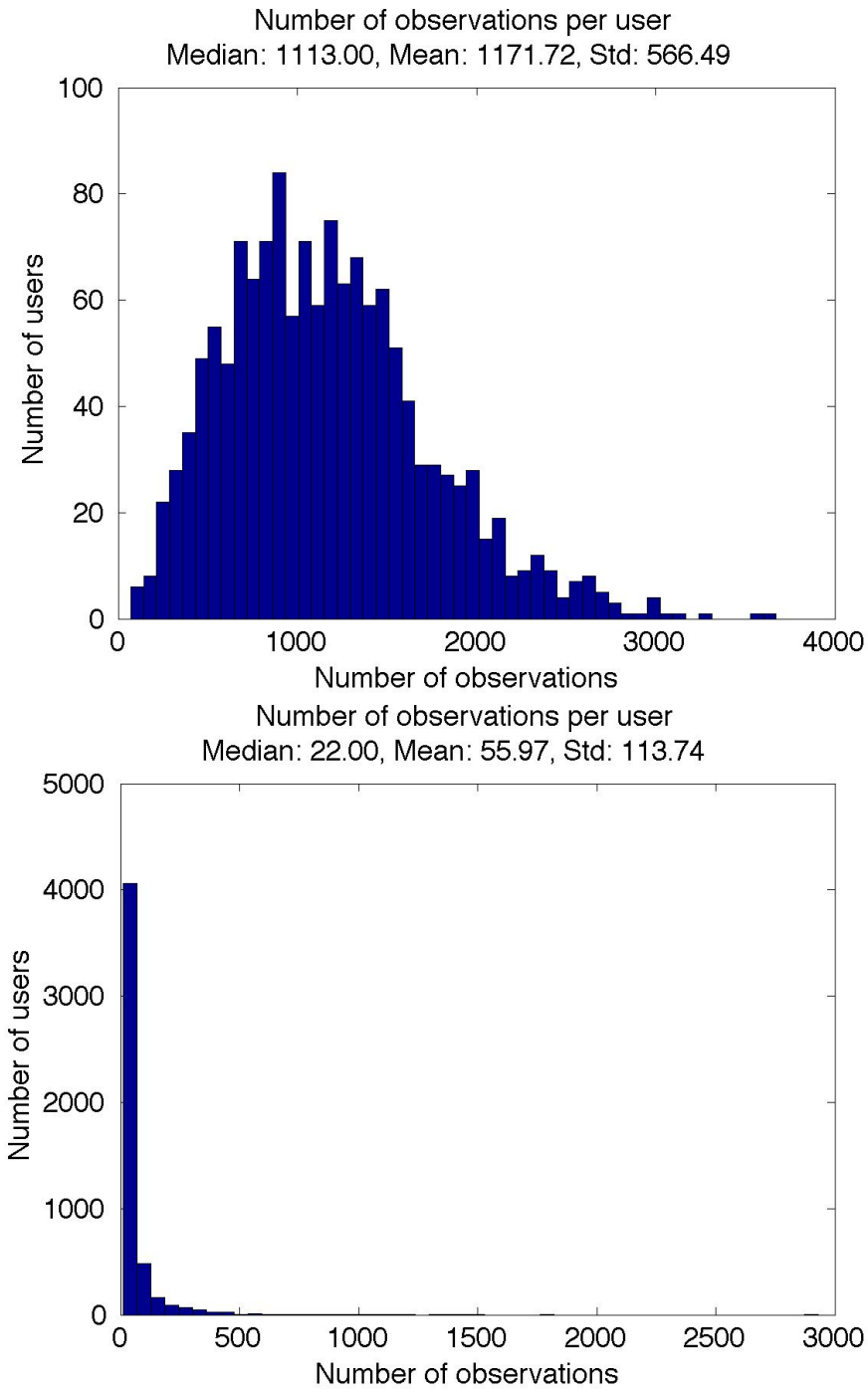

Figure 3.1: Histograms of number of observations per user for a dense and a sparse dataset. The top plot is based on a random sample of 1395 mobile carrier users and is representative of dense datasets used in previous work on location prediction. The bottom plot is based on a random sample of 5007 smartphone application users and is the sparse dataset used in this chapter. On average, the dense dataset has 21 times more observations per user than the sparse dataset.

to be local to each weekhour. For example, our models may learn that a user partitions her time between four places, two of which are in West Village and midtown Manhattan, determine the spatial extent of these places, infer that on thursday 7pm the user has a high likelihood of visiting West Village and a low likelihood of visiting midtown Manhattan, and achieve these inferences in an unsupervised manner from the observed geographic coordinates alone.

A source of difficulty is that, due to sparsity, each user has many weekhours for which no location has been observed before. Since our goal is to construct a week-long profile of users' whereabouts, our models need to infer the place distributions for the missing weekhours, which they accomplish by sharing information across users. It is the details of how users share this information that sets the two models apart.

Our first model, called Constrained Place Model (CPM), assumes that for each weekhour all users have the same place distribution, even though the spatial extents of these places remain unique to each user. For example, users Rob and Tony may both have a work cluster, with different means and covariances, and even though their probability of being at work varies for each weekhour, Tony's probability remains equal to Rob's probability. By assuming that the place distributions are the same for all users, CPM is able to estimate place distributions for *all weekhours*, even when data is sparse.

Since constraining all users to have the same place distribution may be too restrictive, we propose a second model called Smooth Place Model (SPM). This model assumes that, for each user, each weekhour's place distribution is a convex combination of user-specific factor distributions, and that the coefficients of these convex combinations are shared across all users. As a result, SPM allows users to have different place distributions for each weekhour, different number of places for each user, as well as different spatial extents for each place. Despite its flexibility, SPM accurately constructs place distributions for hours and days where the user has not been observed before, in addition to providing state-of-the-art performance in sparse location prediction.

The chapter proceeds as follows. In Section 3.2, we provide a formal description of our models. In Section 3.3, we derive the inference algorithms, which include collapsed Gibbs sampling for posterior inference. In Section 3.4, we provide de-

tails of our experimental setup and apply our algorithms to a sparse smartphone application dataset and a dense mobile carrier dataset. We conclude in Section 3.5.

The research described in this chapter is currently under submission and it is joint work with David Rosenberg, Robert Schapire, and Tony Jebara.

3.2 Collaborative Place Models

In this section, we provide a formal description of our models, CPM and SPM, and describe their generative process. Both models comprise a spatial component, which represents the inferred place clusters, and a temporal component, which represents the inferred place distributions for each weekhour. Both CPM and SPM have the same spatial components, but they have different temporal components. The models are depicted in Figure 3.2 using the graphical model representation.

3.2.1 Constrained Place Model (CPM)

Here, we detail the temporal and spatial components of CPM, and discuss the intuition behind some of the technical decisions we have made about the model. The temporal component of CPM constrains all users to have the same place distribution for a given weekhour. Let w represent an hour in a week, ranging from 1 to 168, and let k represent a place index, ranging from 1 to K. In case of CPM, all users have exactly K places, but we will relax this requirement for SPM. Let $\boldsymbol{\theta}_w \in \mathbb{R}^K$ denote a Dirichlet random variable that represents, intuitively, users' global place preferences at weekhour w. These preferences are used to generate $z_{u,w,n}$, the latent place assignment for user u and observation n at weekhour w. The place assignment is in turn used to sample the observed coordinates $\boldsymbol{\ell}_{u,w,n}$ from the corresponding place cluster.

The spatial component of CPM models the K place clusters, whose means and covariances are unique to each user. Intuitively, we expect each place cluster to correspond to locations such as "home", "work", and "gym". Given a user u and a place index k, each place cluster is characterized by a bivariate normal distribution, with mean $\boldsymbol{\phi}_u^k$ and covariance Σ_u^k. The observed coordinates $\boldsymbol{\ell}$ are considered to be noisy observations sampled from these place clusters.

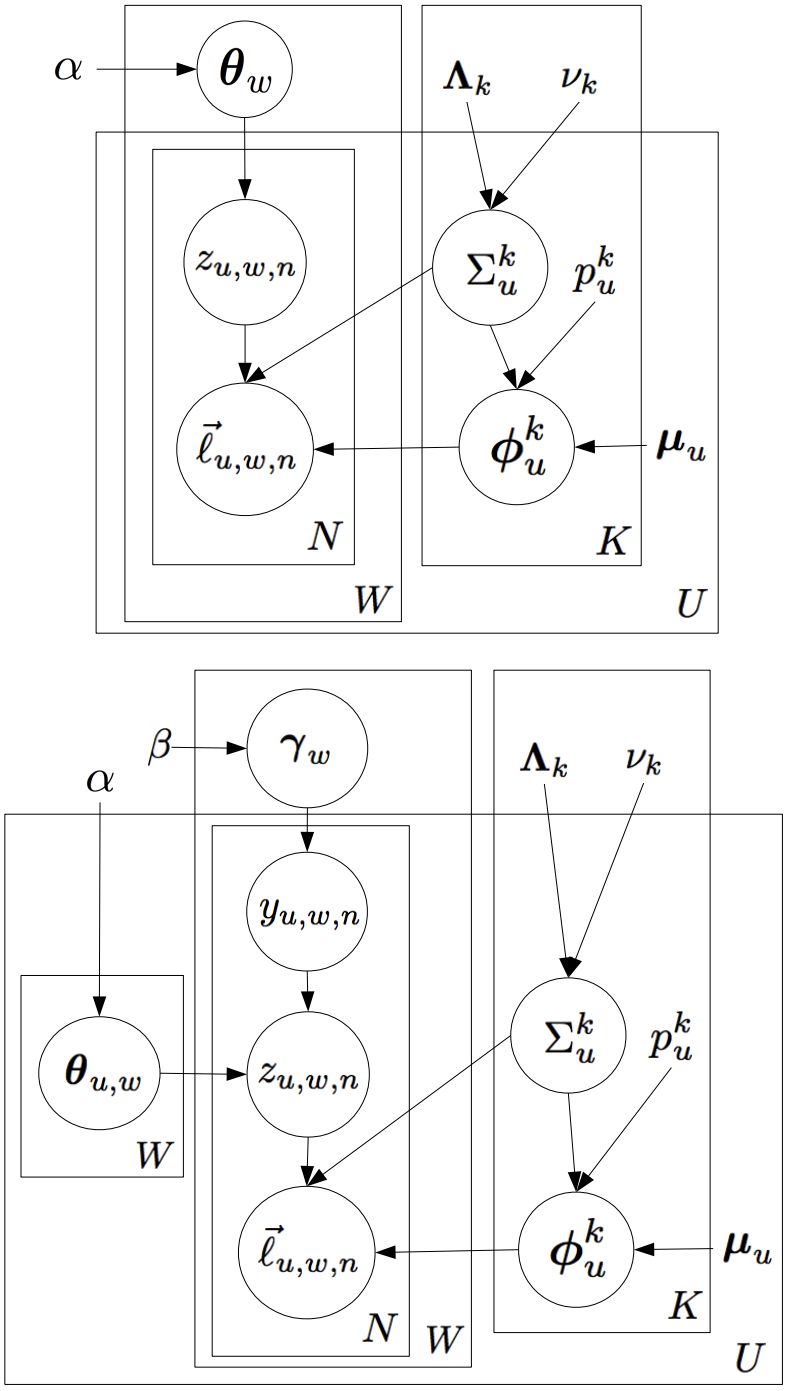

Figure 3.2: Graphical model representations of CPM (top) and SPM (bottom). CPM assumes that all users share the same place distribution for each weekhour. SPM assumes that all users share the same correlation between the place distributions.

Our models use conjugate priors because of the computational advantages they provide in Bayesian inference. Since $\boldsymbol{\theta}_w$ is the parameter of a categorical distribution, we set its prior to a symmetric Dirichlet with concentration parameter $\alpha > 0$ and let $\text{Dirichlet}_K(\alpha)$ denote the K-dimensional distribution. In practice, we set α to 1 and make all parameters in Dirichlet's support equally probable, a reasonable assumption given the lack of prior information.

We set the normal distribution's prior to the normal-inverse-Wishart (NIW) distribution. Among its parameters, $\Lambda \in \mathbb{R}^{2\times 2}$ is a positive definite scale matrix, ν>1 indicates the degrees of freedom, and together they define the distribution over the covariance matrix. The prior mean of the mean, $\boldsymbol{\mu}_u$, is customized for each user u and is computed as the mean of user's historical locations. The p parameter is a function of the user and place, and it ensures that the covariance of the mean matches the empirical covariance of user's past location data.

In location datasets, a user sometimes logs locations that are one-offs and are not representative of the user's regular location profile. To ensure that such outliers do not affect how the model infers user's regular place clusters, we designate place K as a special outlier place and set its covariance's prior mean, as determined by Λ_K, to be very large. As for the covariance of regular clusters, we set the remaining scale matrices, Λ_{-K}, such that each coordinate of the covariance's prior mean has a standard deviation of 250 meters. This is a reasonable size for a place cluster, as it is large enough to encapsulate both the potential inaccuracies of location hardware and the inherent noise in users' locations, but small enough to ensure that each place cluster corresponds to a single intuitive place, such as "home" or "work".

Let $\mathcal{N}$ denote the normal distribution and IW denote the inverse-Wishart distribution. The generative process of CPM is described in more formal terms below.

1. For each weekhour w, draw a distribution over places $\boldsymbol{\theta}_w \sim \text{Dirichlet}_K(\alpha)$.
2. For each user u and each place k,
   (a) Draw a place covariance $\Sigma_u^k \sim IW(\Lambda_k, v)$.
   (b) Draw a place mean $\boldsymbol{\phi}_u^k \sim \mathcal{N}\left(\boldsymbol{\mu}_u, \frac{\Sigma_u^k}{p_k^u}\right)$.
3. For each user u, weekhour w, and observation index n,

(a) Draw a place assignment $z_{u,w,n} \sim \text{Categorical}(\boldsymbol{\theta}_w)$.

(b) Draw a location $\boldsymbol{\ell}_{u,w,n} \sim \mathcal{N}\left(\boldsymbol{\phi}_u^{z_{u,w,n}}, \Sigma_u^{z_{u,w,n}}\right)$.

3.2.2 Smooth Place Model (SPM)

In this subsection, we turn our attention to SPM, and specify its spatial and temporal components. SPM's spatial component is essentially identical to that of CPM, except SPM does not restrict all users to have the same number of places, whereas CPM does. In contrast, SPM's temporal component is quite different from that of CPM, and that is what sets the two models apart.

SPM's temporal component assumes that, for each user u and weekhour w, the corresponding place distribution is a convex combination of F factor distributions. The factor distributions are unique to each user, but the coefficients of the convex combination are shared across all users. This way, SPM is able to infer each user's place distribution for weekhours where the user has not been observed before, as well as predict each user's future geographic coordinates accurately despite sparsity. Furthermore, by not restricting each user to have the same number of places or each weekhour to have the same distribution over places, SPM allows a more realistic and flexible representation of users' location profiles.

For each user u and factor index f, let $\boldsymbol{\theta}_u^f \in \mathbb{R}^{K_u}$ be a Dirichlet random variable that represents the corresponding factor distribution, where K_u denotes the number of places. Each factor distribution is essentially a place distribution and indicates user's preferences over places. $\boldsymbol{\gamma}_w \in \mathbb{R}^F$ denotes a Dirichlet random variable that represents the global coefficients, which when combined with the factor distributions, result in user's place distributions for weekhour w. $\boldsymbol{\gamma}_w$ generates $y_{u,w,n}$, the factor assignment for user u and observation n at weekhour w, and the factor assignment generates the place index $z_{u,w,n}$ from the factor distribution $\boldsymbol{\theta}_u^{y_{u,w,n}}$. The place index is in turn used to sample the observed coordinates $\boldsymbol{\ell}_{u,w,n}$ from the corresponding place cluster.

We abuse notation and let $\text{Dirichlet}_K(\cdot)$ denote a symmetric Dirichlet if its parameter is a scalar and a general Dirichlet if its parameter is a vector. The generative process of SPM is described in more formal terms below.

1. For each weekhour w, draw a distribution over factors $\boldsymbol{\gamma}_w \sim \text{Dirichlet}_F\left(\boldsymbol{\beta}_w\right)$.

2. For each user u and factor f, draw a factor distribution $\boldsymbol{\theta}_u^f \sim \text{Dirichlet}_K(\alpha)$.

3. For each user u and place k,

   (a) Draw a place covariance $\Sigma_u^k \sim IW(\Lambda_k, v)$.

   (b) Draw a place mean $\boldsymbol{\phi}_u^k \sim \mathcal{N}\left(\boldsymbol{\mu}_u, \frac{\Sigma_u^k}{p_k^u}\right)$.

4. For each user u, weekhour w, and observation index n,

   (a) Draw a factor assignment $y_{u,w,n} \sim \text{Categorical}(\boldsymbol{\gamma}_w)$.

   (b) Draw a place assignment $z_{u,w,n} \sim \text{Categorical}(\boldsymbol{\theta}_u^{y_{u,w,n}})$.

   (c) Draw a location $\boldsymbol{\ell}_{u,w,n} \sim \mathcal{N}\left(\boldsymbol{\phi}_u^{z_{u,w,n}}, \Sigma_u^{z_{u,w,n}}\right)$.

3.3 Posterior inference and parameter estimation

In this section, we derive algorithms that compute the posterior expectations of the hidden variables conditioned on the observed geographic coordinates. In particular, we use a strategy popularized by Griffiths and Steyvers [20], derive a collapsed Gibbs sampler to sample from the posterior distribution of the categorical random variables, and use these samples to approximate the posterior expectations of the non-categorical random variables. In subsequent sections, we will show how these posteriors are used to compute users' significant places, to analyze the way users partition their time across these places, and to predict users' future locations.

Since the derivations for CPM and SPM are fairly similar, we avoid duplication and only present the derivation for SPM. In Lemmas 3.1 and 3.2, we derive the collapsed Gibbs sampler for variables $\boldsymbol{z}$ and $\boldsymbol{y}$, respectively. Then, in Lemmas 3.3, 3.4, and 3.5, we use the samples obtained via the sampler to approximate the posterior expectations of the variables $\boldsymbol{\gamma}$, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, and Σ.

Given a vector $\boldsymbol{x}$ and an index k, let $\boldsymbol{x}_{-k}$ indicate all the entries of the vector excluding the one at index k. For Lemmas 3.1 and 3.2, assume $i = (u, w, n)$ denotes the index of the variable that will be sampled.

Lemma 3.1. *The unnormalized probability of z_i conditioned on the observed location data and remaining categorical variables is*

$$p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}, \boldsymbol{\ell}\right) \propto t_{\tilde{v}_k^u - 1}\left(\boldsymbol{\ell}_i \mid \tilde{\boldsymbol{\mu}}_k^u, \frac{\tilde{\Lambda}_u^k\left(\tilde{p}_k^u + 1\right)}{\tilde{p}_k^u\left(\tilde{v}_k^u - 1\right)}\right)\left(\alpha + \tilde{m}_{u,\cdot}^{k,f}\right).$$

The parameters $\tilde{v}_k^u$, $\tilde{\boldsymbol{\mu}}_k^u$, $\tilde{\Lambda}_u^k$, and $\tilde{p}_k^u$ are defined in the proof. t denotes the bivariate t-distribution and $\tilde{m}_{u,\cdot}^{k,f}$ denotes counts, both of which are defined in the appendix.

Proof. We decompose the probability into two components using Bayes' theorem:

$$\begin{aligned}
p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}, \boldsymbol{\ell}\right) &= p\left(\boldsymbol{\ell}_i \mid z_i = k, y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}, \boldsymbol{\ell}_{-i}\right) \\
&\quad \times \frac{p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}, \boldsymbol{\ell}_{-i}\right)}{p\left(\boldsymbol{\ell}_i \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}, \boldsymbol{\ell}_{-i}\right)} \\
&= p\left(\boldsymbol{\ell}_i \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}\right) \\
&\quad \times \frac{p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right)}{p\left(\boldsymbol{\ell}_i \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}, \boldsymbol{\ell}_{-i}\right)} \\
&\propto p\left(\boldsymbol{\ell}_i \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}\right) \qquad (3.1) \\
&\quad \times p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right). \qquad (3.2)
\end{aligned}$$

In the first part of the derivation, we operate on (3.1). We augment it with $\boldsymbol{\phi}$ and Σ:

$$\begin{aligned}
p\left(\boldsymbol{\ell}_i \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}\right) &= \int\int p\left(\boldsymbol{\ell}_i \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}, \boldsymbol{\phi}_u^k, \Sigma_u^k\right) \\
&\quad \times p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}\right) \mathrm{d}\boldsymbol{\phi}_u^k \, \mathrm{d}\Sigma_u^k \\
&= \int\int p\left(\boldsymbol{\ell}_i \mid z_i = k, \boldsymbol{\phi}_u^k, \Sigma_u^k\right) \\
&\quad \times p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}\right) \mathrm{d}\boldsymbol{\phi}_u^k \, \mathrm{d}\Sigma_u^k \\
&= \int\int \mathcal{N}\left(\boldsymbol{\ell}_i \mid \boldsymbol{\phi}_u^k, \Sigma_u^k\right) \qquad (3.3) \\
&\quad \times p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}\right) \mathrm{d}\boldsymbol{\phi}_u^k \, \mathrm{d}\Sigma_u^k. \qquad (3.4)
\end{aligned}$$

We convert (3.4) into a more tractable form:

$$\begin{aligned}
p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}\right) &= p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{\tilde{M}_{u,\cdot}^{k,\cdot}}\right) \\
&= p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{\tilde{M}_{u,\cdot}^{k,\cdot}}\right) \\
&= p\left(\boldsymbol{\ell}_{\tilde{M}_{u,\cdot}^{k,\cdot}} \mid \boldsymbol{\phi}_u^k, \Sigma_u^k, \boldsymbol{z}_{-i}\right) \frac{p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid \boldsymbol{z}_{-i}\right)}{p\left(\boldsymbol{\ell}_{\tilde{M}_{u,\cdot}^{k,\cdot}} \mid \boldsymbol{z}_{-i}\right)} \\
&\propto p\left(\boldsymbol{\ell}_{\tilde{M}_{u,\cdot}^{k,\cdot}} \mid \boldsymbol{\phi}_u^k, \Sigma_u^k, \boldsymbol{z}_{-i}\right) p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k\right) \\
&= \left(\prod_{j \in \tilde{M}_{u,\cdot}^{k,\cdot}} p\left(\boldsymbol{\ell}_j \mid \boldsymbol{\phi}_u^k, \Sigma_u^k, \boldsymbol{z}_{-i}\right)\right) p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k\right) \\
&= \left(\prod_{j \in \tilde{M}_{u,\cdot}^{k,\cdot}} p\left(\boldsymbol{\ell}_j \mid \boldsymbol{\phi}_u^k, \Sigma_u^k, z_j\right)\right) p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k\right) \\
&= \left(\prod_{j \in \tilde{M}_{u,\cdot}^{k,\cdot}} \mathcal{N}\left(\boldsymbol{\ell}_j \mid \boldsymbol{\phi}_u^k, \Sigma_u^k\right)\right) \mathcal{N}\left(\boldsymbol{\phi}_u^k \mid \boldsymbol{\mu}_u, \frac{\Sigma_u^k}{p_k^u}\right) \\
&\quad \times IW\left(\Sigma_u^k \mid \Lambda_k, v\right) \\
\Longrightarrow p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}\right) &\propto \left(\prod_{j \in \tilde{M}_{u,\cdot}^{k,\cdot}} \mathcal{N}\left(\boldsymbol{\ell}_j \mid \boldsymbol{\phi}_u^k, \Sigma_u^k\right)\right) \mathcal{N}\left(\boldsymbol{\phi}_u^k \mid \boldsymbol{\mu}_u, \frac{\Sigma_u^k}{p_k^u}\right) \\
&\quad \times IW\left(\Sigma_u^k \mid \Lambda_k, v\right).
\end{aligned}$$

Since the normal-inverse-Wishart distribution is the conjugate prior of the multivariate normal distribution, the posterior is also a normal-inverse-Wishart distribution,

$$p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}\right) = \mathcal{N}\left(\boldsymbol{\phi}_u^k \mid \tilde{\boldsymbol{\mu}}_k^u, \frac{\Sigma_u^k}{\tilde{p}_k^u}\right) IW\left(\Sigma_u^k \mid \tilde{\Lambda}_u^k, \tilde{v}_k^u\right), \tag{3.5}$$

whose parameters are defined as

$$
\begin{aligned}
\tilde{p}_k^u &= p_k^u + \tilde{m}_{u,\cdot}^{k,\cdot}, \\
\tilde{v}_k^u &= \nu + \tilde{m}_{u,\cdot}^{k,\cdot}, \\
\tilde{\boldsymbol{\ell}}_k^u &= \frac{1}{\tilde{m}_{u,\cdot}^{k,\cdot}} \sum_{j \in \tilde{M}_{u,\cdot}^{k,\cdot}} \boldsymbol{\ell}_j, \\
\tilde{\boldsymbol{\mu}}_k^u &= \frac{p_k^u \boldsymbol{\mu}_u + \tilde{m}_{u,\cdot}^{k,\cdot} \tilde{\boldsymbol{\ell}}_k^u}{\tilde{p}_k^u}, \\
\tilde{S}_u^k &= \sum_{j \in \tilde{M}_{u,\cdot}^{k,\cdot}} \left(\boldsymbol{\ell}_j - \tilde{\boldsymbol{\ell}}_k^u\right)\left(\boldsymbol{\ell}_j - \tilde{\boldsymbol{\ell}}_k^u\right)^T, \\
\tilde{\Lambda}_u^k &= \Lambda_k + \tilde{S}_u^k + \frac{p_k^u \tilde{m}_{u,\cdot}^{k,\cdot}}{p_k^u + \tilde{m}_{u,\cdot}^{k,\cdot}} \left(\tilde{\boldsymbol{\ell}}_k^u - \boldsymbol{\mu}_u\right)\left(\tilde{\boldsymbol{\ell}}_k^u - \boldsymbol{\mu}_u\right)^T.
\end{aligned}
$$

We rewrite (3.1) by combining (3.3), (3.4), and (3.5) to obtain

$$
\begin{aligned}
p(\boldsymbol{\ell}_i \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}) &= \int\int \mathcal{N}\left(\boldsymbol{\ell}_i \mid \boldsymbol{\phi}_u^k, \Sigma_u^k\right) \\
&\quad \times p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}\right) \mathrm{d}\boldsymbol{\phi}_u^k \, \mathrm{d}\Sigma_u^k \\
&= \int\int \mathcal{N}\left(\boldsymbol{\ell}_i \mid \boldsymbol{\phi}_u^k, \Sigma_u^k\right) \mathcal{N}\left(\boldsymbol{\phi}_u^k \mid \tilde{\boldsymbol{\mu}}_k^u, \frac{\Sigma_u^k}{\tilde{p}_k^u}\right) \\
&\quad \times IW\left(\Sigma_u^k \mid \tilde{\Lambda}_u^k, \tilde{v}_k^u\right) \mathrm{d}\boldsymbol{\phi}_u^k \, \mathrm{d}\Sigma_u^k \\
&= t_{\tilde{v}_k^u - 1}\left(\boldsymbol{\ell}_i \mid \tilde{\boldsymbol{\mu}}_k^u, \frac{\tilde{\Lambda}_u^k\left(\tilde{p}_k^u + 1\right)}{\tilde{p}_k^u\left(\tilde{v}_k^u - 1\right)}\right), \qquad (3.6)
\end{aligned}
$$

where t is the 2-dimensional t-distribution.

Now, we move onto the second part of the derivation. We operate on (3.2) and augment it with $\boldsymbol{\theta}$:

$$
\begin{aligned}
p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right) &= \int p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}, \boldsymbol{\theta}_u^f\right) p\left(\boldsymbol{\theta}_u^f \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right) \mathrm{d}\boldsymbol{\theta}_u^f \\
&= \int p\left(z_i = k \mid y_i = f, \boldsymbol{\theta}_u^f\right) \qquad (3.7) \\
&\quad \times p\left(\boldsymbol{\theta}_u^f \mid y_i = f, \boldsymbol{y}_{-i}, \boldsymbol{z}_{-i}\right) \mathrm{d}\boldsymbol{\theta}_u^f. \qquad (3.8)
\end{aligned}
$$

We convert (3.8) into a more tractable form,

$$\begin{aligned}
p\left(\boldsymbol{\theta}_u^f \mid y_i = f, \boldsymbol{y}_{-i}, \boldsymbol{z}_{-i}\right) &= p\left(\boldsymbol{\theta}_u^f \mid y_i = f, \boldsymbol{y}_{-i}, \boldsymbol{z}_{\tilde{M}_{u,\cdot}^{\cdot,f}}\right) \\
&= \frac{p\left(\boldsymbol{z}_{\tilde{M}_{u,\cdot}^{\cdot,f}} \mid y_i = f, \boldsymbol{y}_{-i}, \boldsymbol{\theta}_u^f\right) p\left(y_i = f, \boldsymbol{y}_{-i}, \boldsymbol{\theta}_u^f\right)}{p\left(y_i = f, \boldsymbol{y}_{-i}, \boldsymbol{z}_{\tilde{M}_{u,\cdot}^{\cdot,f}}\right)} \\
&\propto \prod_{j \in \tilde{M}_{u,\cdot}^{\cdot,f}} p\left(z_j \mid y_i = f, \boldsymbol{y}_{-i}, \boldsymbol{\theta}_u^f\right) p\left(\boldsymbol{\theta}_u^f\right) \\
&= \prod_{j \in \tilde{M}_{u,\cdot}^{\cdot,f}} p\left(z_j \mid y_j, \boldsymbol{\theta}_u^f\right) p\left(\boldsymbol{\theta}_u^f\right) \\
&= \prod_{j \in \tilde{M}_{u,\cdot}^{\cdot,f}} \text{Categorical}\left(z_j \mid \boldsymbol{\theta}_u^f\right) \text{Dirichlet}_K\left(\boldsymbol{\theta}_u^f \mid \alpha\right) \\
\implies p\left(\boldsymbol{\theta}_u^f \mid y_i = f, \boldsymbol{y}_{-i}, \boldsymbol{z}_{-i}\right) &= \text{Dirichlet}_K\left(\boldsymbol{\theta}_u^f \mid \alpha + \tilde{m}_{u,\cdot}^{1,f}, \ldots, \alpha + \tilde{m}_{u,\cdot}^{K,f}\right), \qquad (3.9)
\end{aligned}$$

where the last step follows because Dirichlet distribution is the conjugate prior of the categorical distribution.

We rewrite (3.2) by combining (3.7), (3.8), and (3.9):

$$\begin{aligned}
p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right) &= \int p\left(z_i = k \mid y_i = f, \boldsymbol{\theta}_u^f\right) p\left(\boldsymbol{\theta}_u^f \mid y_i = f, \boldsymbol{y}_{-i}, \boldsymbol{z}_{-i}\right) \mathrm{d}\boldsymbol{\theta}_u^f \\
&= \int \theta_{u,k}^f \text{Dirichlet}_K\left(\boldsymbol{\theta}_u^f \mid \alpha + \tilde{m}_{u,\cdot}^{1,f}, \ldots, \alpha + \tilde{m}_{u,\cdot}^{K,f}\right) \mathrm{d}\boldsymbol{\theta}_u^f \\
&= \frac{\alpha + \tilde{m}_{u,\cdot}^{k,f}}{K\alpha + \tilde{m}_{u,\cdot}^{\cdot,f}}. \qquad (3.10)
\end{aligned}$$

Finally, we combine (3.1), (3.2), (3.6), and (3.10) to obtain the unnormalized probability distribution:

$$\begin{aligned}
p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}, \boldsymbol{\ell}\right) &\propto p\left(\boldsymbol{\ell}_i \mid z_i = k, \boldsymbol{z}_{-i}, \boldsymbol{\ell}_{-i}\right) \\
&\quad \times p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right) \\
&= t_{\tilde{v}_k^u - 1}\left(\boldsymbol{\ell}_i \mid \tilde{\boldsymbol{\mu}}_k^u, \frac{\tilde{\Lambda}_u^k\left(\tilde{p}_k^u + 1\right)}{\tilde{p}_k^u\left(\tilde{v}_k^u - 1\right)}\right) \frac{\alpha + \tilde{m}_{u,\cdot}^{k,f}}{K\alpha + \tilde{m}_{u,\cdot}^{\cdot,f}}.
\end{aligned}$$

□

Lemma 3.2. *The unnormalized probability of y_i conditioned on the observed location data and remaining categorical variables is*

$$p\left(y_i = f \mid z_i = k, \boldsymbol{y}_{-i}, \boldsymbol{z}_{-i}, \boldsymbol{\ell}\right) \propto \frac{\alpha + \tilde{m}_{u,\cdot}^{k,f}}{K\alpha + \tilde{m}_{u,\cdot}^{\cdot,f}} \left(\beta_{w,f} + \tilde{m}_{\cdot,w}^{\cdot,f}\right),$$

where counts $\tilde{m}_{u,\cdot}^{k,f}$, $\tilde{m}_{u,\cdot}^{\cdot,f}$, and $\tilde{m}_{\cdot,w}^{\cdot,f}$ are defined in the appendix.

Proof. We decompose the probability into two components using Bayes' theorem:

$$\begin{aligned}
p\left(y_i = f \mid z_i = k, \boldsymbol{y}_{-i}, \boldsymbol{z}_{-i}, \boldsymbol{\ell}\right) &= p\left(y_i = f \mid z_i = k, \boldsymbol{y}_{-i}, \boldsymbol{z}_{-i}\right) \\
&= p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right) \frac{p\left(y_i = f \mid \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right)}{p\left(z_i = k \mid \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right)} \\
&\propto p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right) && (3.11) \\
&\quad \times p\left(y_i = f \mid \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right). && (3.12)
\end{aligned}$$

Since (3.11) is equal to (3.2), we rewrite it using (3.10)

$$p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right) = \frac{\alpha + \tilde{m}_{u,\cdot}^{k,f}}{K\alpha + \tilde{m}_{u,\cdot}^{\cdot,f}}. \tag{3.13}$$

We operate on (3.12) and augment it with $\boldsymbol{\gamma}$:

$$\begin{aligned}
p\left(y_i = f \mid \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right) &= p\left(y_i = f \mid \boldsymbol{y}_{-i}\right) \\
&= \int p\left(y_i = f \mid \boldsymbol{y}_{-i}, \boldsymbol{\gamma}_w\right) p\left(\boldsymbol{\gamma}_w \mid \boldsymbol{y}_{-i}\right) \mathrm{d}\boldsymbol{\gamma}_w \\
&= \int p\left(y_i = f \mid \boldsymbol{\gamma}_w\right) p\left(\boldsymbol{\gamma}_w \mid \boldsymbol{y}_{-i}\right) \mathrm{d}\boldsymbol{\gamma}_w \\
&= \int \gamma_{w,f} && (3.14) \\
&\quad \times p\left(\boldsymbol{\gamma}_w \mid \boldsymbol{y}_{-i}\right) \mathrm{d}\boldsymbol{\gamma}_w. && (3.15)
\end{aligned}$$

We convert (3.15) into a more tractable form,

$$\begin{aligned}
p\left(\boldsymbol{\gamma}_w \mid \boldsymbol{y}_{-i}\right) &= p\left(\boldsymbol{\gamma}_w \mid \boldsymbol{y}_{\tilde{M}_{\cdot,w}^{\cdot,\cdot}}\right) \\
&= p\left(\boldsymbol{y}_{\tilde{M}_{\cdot,w}^{\cdot,\cdot}} \mid \boldsymbol{\gamma}_w\right) \frac{p\left(\boldsymbol{\gamma}_w\right)}{p\left(\boldsymbol{y}_{\tilde{M}_{\cdot,w}^{\cdot,\cdot}}\right)} \\
&\propto p\left(\boldsymbol{y}_{\tilde{M}_{\cdot,w}^{\cdot,\cdot}} \mid \boldsymbol{\gamma}_w\right) p\left(\boldsymbol{\gamma}_w\right) \\
&= \prod_{j \in \tilde{M}_{\cdot,w}^{\cdot,\cdot}} p\left(y_j \mid \boldsymbol{\gamma}_w\right) p\left(\boldsymbol{\gamma}_w\right) \\
&= \prod_{j \in \tilde{M}_{\cdot,w}^{\cdot,\cdot}} \text{Categorical}\left(y_j \mid \boldsymbol{\gamma}_w\right) \text{Dirichlet}_F\left(\boldsymbol{\gamma}_w \mid \boldsymbol{\beta}_w\right) \\
\implies p\left(\boldsymbol{\gamma}_w \mid \boldsymbol{y}_{-i}\right) &= \text{Dirichlet}_F\left(\boldsymbol{\gamma}_w \mid \beta_{w,1} + \tilde{m}_{\cdot,w}^{\cdot,1}, \ldots, \beta_{w,F} + \tilde{m}_{\cdot,w}^{\cdot,F}\right), \qquad (3.16)
\end{aligned}$$

where the last step follows because Dirichlet distribution is the conjugate prior of the categorical distribution.

We rewrite (3.12) by combining (3.14), (3.15), and (3.16):

$$\begin{aligned}
p\left(y_i = f \mid \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right) &= \int \gamma_{w,f}\, p\left(\boldsymbol{\gamma}_w \mid \boldsymbol{y}_{-i}\right) \mathrm{d}\boldsymbol{\gamma}_w \\
&= \int \gamma_{w,f} \text{Dirichlet}_F\left(\boldsymbol{\gamma}_w \mid \beta_{w,1} + \tilde{m}_{\cdot,w}^{\cdot,1}, \ldots, \beta_{w,F} + \tilde{m}_{\cdot,w}^{\cdot,F}\right) \mathrm{d}\boldsymbol{\gamma}_w \\
&= \frac{\beta_{w,f} + \tilde{m}_{\cdot,w}^{\cdot,f}}{\sum_f \left(\beta_{w,f} + \tilde{m}_{\cdot,w}^{\cdot,f}\right)}. \qquad (3.17)
\end{aligned}$$

Finally, we combine (3.11), (3.12), (3.13), and (3.17) to obtain the unnormalized probability distribution:

$$\begin{aligned}
p\left(y_i = f \mid z_i = k, \boldsymbol{y}_{-i}, \boldsymbol{z}_{-i}, \boldsymbol{\ell}\right) &\propto p\left(z_i = k \mid y_i = f, \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right) \\
&\quad \times p\left(y_i = f \mid \boldsymbol{z}_{-i}, \boldsymbol{y}_{-i}\right) \\
&= \frac{\alpha + \tilde{m}_{u,\cdot}^{k,f}}{K\alpha + \tilde{m}_{u,\cdot}^{\cdot,f}} \frac{\beta_{w,f} + \tilde{m}_{\cdot,w}^{\cdot,f}}{\sum_f \left(\beta_{w,f} + \tilde{m}_{\cdot,w}^{\cdot,f}\right)}.
\end{aligned}$$

□

Lemmas 3.1 and 3.2 described a collapsed Gibbs sampler for sampling the pos-

teriors of the categorical random variables. Below, Lemmas 3.3, 3.4, and 3.5 show how these samples, denoted as $\hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{z}}$, can be used to approximate the posterior expectations of $\boldsymbol{\gamma}$, $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, and Σ.

Lemma 3.3. *The expectation of $\boldsymbol{\gamma}$ given the observed geographical coordinates and the posterior samples is*

$$\hat{\gamma}_{w,f} = \mathbb{E}\left[\gamma_{w,f} \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right] = \frac{\beta_{w,f} + m_{\cdot,w}^{\cdot,f}}{\sum_{f}\left(\beta_{w,f} + m_{\cdot,w}^{\cdot,f}\right)},$$

where count $m_{\cdot,w}^{\cdot,f}$ is defined in the appendix.

Proof.

$$\begin{aligned}
p\left(\boldsymbol{\gamma}_w \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right) &= p\left(\boldsymbol{\gamma}_w \mid \hat{\boldsymbol{y}}_{M_{\cdot,w}^{\cdot,\cdot}}\right) \\
&= \frac{p\left(\hat{\boldsymbol{y}}_{M_{\cdot,w}^{\cdot,\cdot}} \mid \boldsymbol{\gamma}_w\right) p\left(\boldsymbol{\gamma}_w\right)}{p\left(\hat{\boldsymbol{y}}_{M_{\cdot,w}^{\cdot,\cdot}}\right)} \\
&= \frac{p\left(\boldsymbol{\gamma}_w\right) \prod_{j \in M_{\cdot,w}^{\cdot,\cdot}} p\left(\hat{y}_j \mid \boldsymbol{\gamma}_w\right)}{p\left(\hat{\boldsymbol{y}}_{M_{\cdot,w}^{\cdot,\cdot}}\right)} \\
&\propto \text{Dirichlet}_F\left(\boldsymbol{\gamma}_w \mid \boldsymbol{\beta}_w\right) \prod_{j \in M_{\cdot,w}^{\cdot,\cdot}} \text{Categorical}\left(\hat{y}_j \mid \boldsymbol{\gamma}_w\right) \\
&= \text{Dirichlet}_F\left(\boldsymbol{\gamma}_w \mid \beta_{w,1} + m_{\cdot,w}^{\cdot,1}, \ldots, \beta_{w,F} + m_{\cdot,w}^{\cdot,F}\right) \\
\implies \hat{\gamma}_{w,f} = \mathbb{E}\left[\gamma_{w,f} \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right] &= \frac{\beta_{w,f} + m_{\cdot,w}^{\cdot,f}}{\sum_{f}\left(\beta_{w,f} + m_{\cdot,w}^{\cdot,f}\right)}.
\end{aligned}$$

□

Lemma 3.4. *The expectation of $\boldsymbol{\theta}$ given the observed geographical coordinates and the posterior samples is*

$$\hat{\theta}_{u,k}^{f} = \mathbb{E}\left[\theta_{u,k}^{f} \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right] = \frac{\alpha + m_{u,\cdot}^{k,f}}{K\alpha + m_{u,\cdot}^{\cdot,f}},$$

where counts $m_{u,\cdot}^{k,f}$ and $m_{u,\cdot}^{\cdot,f}$ are defined in the appendix.

Proof.

$$
\begin{aligned}
p\left(\boldsymbol{\theta}_u^f \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right) &= p\left(\boldsymbol{\theta}_u^f \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}\right) \\
&= p\left(\boldsymbol{\theta}_u^f \mid \hat{\boldsymbol{z}}_{M_{u,\cdot}^{\cdot,f}}, \hat{\boldsymbol{y}}\right) \\
&= \frac{p\left(\hat{\boldsymbol{z}}_{M_{u,\cdot}^{\cdot,f}} \mid \boldsymbol{\theta}_u^f, \hat{\boldsymbol{y}}\right) p\left(\boldsymbol{\theta}_u^f \mid \hat{\boldsymbol{y}}\right)}{p\left(\hat{\boldsymbol{z}}_{M_{u,\cdot}^{\cdot,f}} \mid \hat{\boldsymbol{y}}\right)} \\
&= \frac{p\left(\boldsymbol{\theta}_u^f\right) \prod\limits_{j \in M_{u,\cdot}^{\cdot,f}} p\left(\hat{z}_j \mid \boldsymbol{\theta}_u^f, \hat{\boldsymbol{y}}\right)}{p\left(\hat{\boldsymbol{z}}_{M_{u,\cdot}^{\cdot,f}} \mid \hat{\boldsymbol{y}}\right)} \\
&\propto \text{Dirichlet}_K\left(\boldsymbol{\theta}_u^f \mid \alpha\right) \prod_{j \in M_{u,\cdot}^{\cdot,f}} \text{Categorical}\left(\hat{z}_j \mid \boldsymbol{\theta}_u^f\right) \\
&= \text{Dirichlet}_K\left(\boldsymbol{\theta}_u^f \mid \alpha + m_{u,\cdot}^{1,f}, \ldots, \alpha + m_{u,\cdot}^{K,f}\right) \\
\implies \hat{\theta}_{u,k}^f = \mathbb{E}\left[\theta_{u,k}^f \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right] &= \frac{\alpha + m_{u,\cdot}^{k,f}}{K\alpha + m_{u,\cdot}^{\cdot,f}}.
\end{aligned}
$$

□

Lemma 3.5. *The expectations of $\boldsymbol{\phi}$ and Σ given the observed geographical coordinates and the posterior samples is*

$$\hat{\boldsymbol{\phi}}_u^k = \mathbb{E}\left[\boldsymbol{\phi}_u^k \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right] = \hat{\boldsymbol{\mu}}_k^u$$

and

$$\hat{\Sigma}_u^k = \mathbb{E}\left[\Sigma_u^k \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right] = \frac{\hat{\Lambda}_u^k}{\hat{v}_k^u - 3}.$$

Parameters $\hat{\boldsymbol{\mu}}_k^u$, $\hat{\Lambda}_u^k$, and $\hat{v}_k^u$ are defined in the proof of Lemma 3.1.

Proof.

$$
\begin{aligned}
p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right) &= p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right) \\
&= p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid \hat{\boldsymbol{z}}, \boldsymbol{\ell}_{M_{u,\cdot}^{k,\cdot}}\right) \\
&= \frac{p\left(\boldsymbol{\ell}_{M_{u,\cdot}^{k,\cdot}} \mid \boldsymbol{\phi}_u^k, \Sigma_u^k, \hat{\boldsymbol{z}}\right) p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid \hat{\boldsymbol{z}}\right)}{p\left(\boldsymbol{\ell}_{M_{u,\cdot}^{k,\cdot}} \mid \hat{\boldsymbol{z}}\right)} \\
&= \frac{\prod_{j \in M_{u,\cdot}^{k,\cdot}} p\left(\boldsymbol{\ell}_j \mid \boldsymbol{\phi}_u^k, \Sigma_u^k, \hat{\boldsymbol{z}}\right) p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k\right)}{p\left(\boldsymbol{\ell}_{M_{u,\cdot}^{k,\cdot}} \mid \hat{\boldsymbol{z}}\right)} \\
&= \frac{\mathcal{N}\left(\boldsymbol{\phi}_u^k \mid \boldsymbol{\mu}_u, \frac{\Sigma_u^k}{p_k^u}\right) IW\left(\Sigma_u^k \mid \Lambda_k, \upsilon\right) \prod_{j \in M_{u,\cdot}^{k,\cdot}} \mathcal{N}\left(\boldsymbol{\ell}_j \mid \boldsymbol{\phi}_u^k, \Sigma_u^k\right)}{p\left(\boldsymbol{\ell}_{M_{u,\cdot}^{k,\cdot}} \mid \hat{\boldsymbol{z}}\right)} \\
&\propto \mathcal{N}\left(\boldsymbol{\phi}_u^k \mid \boldsymbol{\mu}_u, \frac{\Sigma_u^k}{p_k^u}\right) IW\left(\Sigma_u^k \mid \Lambda_k, \upsilon\right) \prod_{j \in M_{u,\cdot}^{k,\cdot}} \mathcal{N}\left(\boldsymbol{\ell}_j \mid \boldsymbol{\phi}_u^k, \Sigma_u^k\right) \\
\implies p\left(\boldsymbol{\phi}_u^k, \Sigma_u^k \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right) &= \mathcal{N}\left(\boldsymbol{\phi}_u^k \mid \hat{\boldsymbol{\mu}}_k^u, \frac{\Sigma_u^k}{\hat{p}_k^u}\right) IW\left(\Sigma_u^k \mid \hat{\Lambda}_u^k, \hat{\upsilon}_k^u\right) \\
\implies p\left(\boldsymbol{\phi}_u^k \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right) &= t_{\hat{\upsilon}_k^u - 1}\left(\boldsymbol{\phi}_u^k \mid \hat{\boldsymbol{\mu}}_k^u, \frac{\hat{\Lambda}_u^k}{\hat{p}_k^u\left(\hat{\upsilon}_k^u - 1\right)}\right) \\
\implies \hat{\boldsymbol{\phi}}_u^k = \mathbb{E}\left[\boldsymbol{\phi}_u^k \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right] &= \hat{\boldsymbol{\mu}}_k^u \\
\implies p\left(\Sigma_u^k \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right) &= IW\left(\Sigma_u^k \mid \hat{\Lambda}_u^k, \hat{\upsilon}_k^u\right) \\
\implies \hat{\Sigma}_u^k = \mathbb{E}\left[\Sigma_u^k \mid \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}, \boldsymbol{\ell}\right] &= \frac{\hat{\Lambda}_u^k}{\hat{\upsilon}_k^u - 3}.
\end{aligned}
$$

□

3.4 Experiments

We demonstrate the empirical capabilities of our models by applying them to two datasets, a dense cellular carrier dataset and a sparse mobile application dataset, whose density profiles are depicted in Figure 3.1. We assess our models quantita-

tively by measuring their location prediction performance and held-out likelihood. We also assess our models qualitatively by inspecting their inferred places clusters and the temporal distributions they associate with these clusters.

The sparse mobile application dataset consists of 5007 users and the dense cellular carrier dataset consists of 1000 users. For both dataset, each data point comprises a user ID, local time, and geographic coordinates (i.e. latitude and longitude). During preprocessing, we check if a user has logged multiple observations during the same hour, and if so, we replace these observations with their geometric median, computed using Weiszfeld's algorithm. We then sort the data chronologically, and for each user, we partition the earlier 90% as train data and the subsequent 10% as test data.

When we applied CPM to our datasets, we observed that constraining all users to have the same place distribution yields worse predictive performance than baseline methods. Thus, we combined CPM and SPM to form a two-stage approach. In the first stage, we train a separate CPM for each user and each $K \in \{1, \ldots, 10\}$, where K denotes the number of places, and for each user, we select the K parameter where the corresponding model has the highest train likelihood. In the second stage, we train a single SPM, and for each user, we use the K parameter that was determined in the previous stage.

We evaluate a method's location prediction performance by measuring the distance between its predicted coordinate and the actual observed coordinate. Let $\hat{\boldsymbol{\gamma}}$, $\hat{\boldsymbol{\theta}}$, $\hat{\boldsymbol{\phi}}$, and $\hat{\Sigma}$ be the posterior expectations computed in Section 3.3. Given a user u and a weekhour w, our method makes a prediction by first computing the place cluster with the highest posterior weight, $\hat{k}_{u,w} = \text{argmax}_k \sum_f \hat{\gamma}_{w,f} \hat{\theta}^f_{u,k}$, and then outputting the posterior mean of that place cluster, $\hat{\boldsymbol{\phi}}_u^{\hat{k}_{u,w}}$. In contrast, the baseline method, which we call the user weekhour mode (UWM), partitions the coordinate space and assigns each observation in the train data to a grid cell. For a given user and weekhour, UWM makes a prediction by outputting the corresponding mode, which is the centroid of the grid cell with the highest number of historical observations. If a user has no previous observation at the query weekhour, UWM outputs the global mode for that user instead. The grid cell edge size is chosen optimally from $\{100, 250, 500\}$ to minimize the median distance error.

In Figure 3.3, we display the error distributions of SPM and UWM on held-out

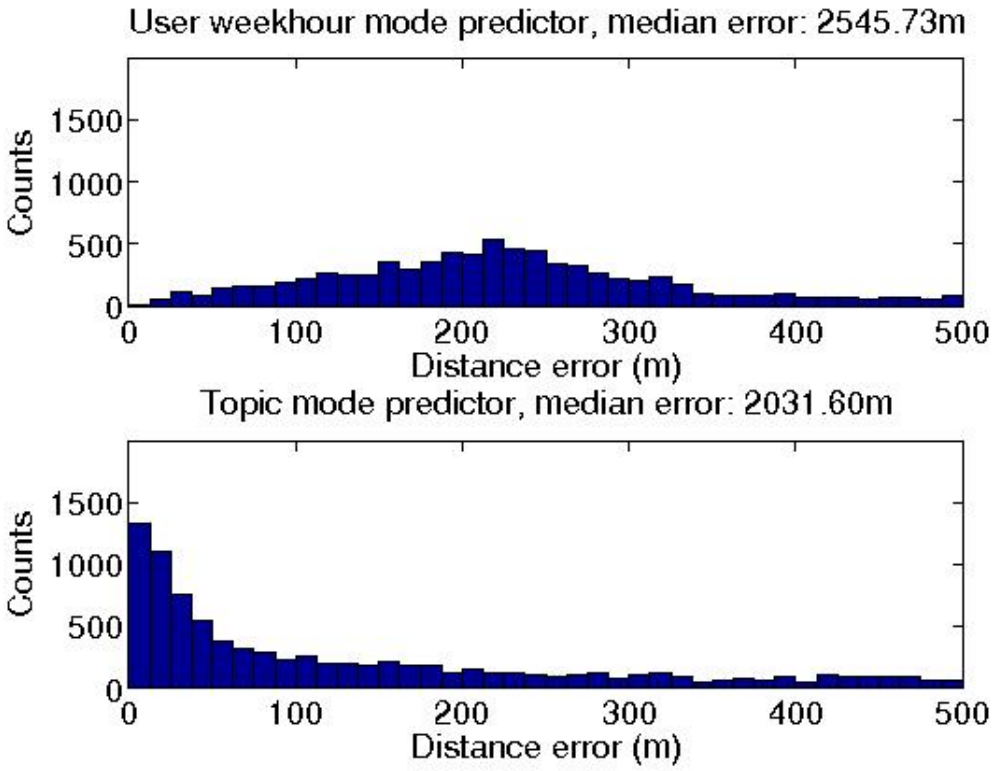

Figure 3.3: Error distributions of UWM (top) and SPM (bottom) on held-out data. UWM yields a median distance error of 2546 meters, whereas SPM yields a median distance error of 2032 meters, a reduction in error of 20%. Furthermore, compared to UWM, SPM predicts many more examples with error less than 100 meters.

data. UWM yields a median distance error of 2546 meters, whereas SPM yields a median distance error of 2032 meters, a reduction in error of 20%. Furthermore, due to discretization, UWM predicts much fewer examples than SPM as having error less than 100 meters. If we increase UWM's grid cell resolution, the discretization effect is diminished, but then UWM's median distance error takes a hit. In contrast to UWM, SPM does not suffer from such trade-offs, and yields both lower median distance error and more predictions with high accuracy.

We evaluate our method qualitatively by analyzing the inferred place clusters and temporal distributions. We focus our analysis to a single user who gave us permission to dissect his location data publicly. CPM obtains its highest train likelihood when it partitions user's location data into 8 places. In Figure 3.4, we depict user's posterior place distributions, computed using $\sum_f \hat{\gamma}_{w,f} \hat{\boldsymbol{\theta}}_u^f$. The top plot corre-

sponds to CPM, and because location data is sparse, CPM learns place distributions that are jagged. In contrast, SPM copes with sparsity by sharing information across users, and consequently, learns place distributions that are smooth. In fact, SPM infers weekhour patterns that are reminiscient of home (Place 0) and work (Place 1), even though the user did not log any location data during most weekhours.

In Figure 3.5, we display the spatial cluster associated with place 0. The blue circles represent user's historical observations and the blue pin represents the cluster center inferred by our model. Since our model is Bayesian and the predictive distribution associated with the cluster is a robust t-distribution, the outlier ping at the figure's far right side is not able to sway the cluster center too much. Furthermore, as depicted in Figure 3.4, the temporal distribution associated with place 0 is similar to a home distribution; it dips during typical work hours and peaks otherwise. We verified that the cluster center, represented as a blue pin, is within 50 meters of the user's apartment.

In Figure 3.6, we display the spatial cluster associated with place 1. The yellow circles represent user's historical observations and the yellow pin represents the cluster center inferred by our model. In Figure 3.4, the temporal distribution associated with place 1 is similar to a work distribution; it peaks during typical work hours and dips otherwise. We verified that the cluster center, represented as a yellow pin, is within 50 meters of the user's work place.

In Figure 3.7, we display the spatial cluster associated with place 3. The orange circles represent user's historical observations and the orange pin represents the cluster center inferred by our model. The posterior covariance associated with place 3 is much larger than the ones associated with previous places; the previous places covered only couple blocks whereas place 3 covers many more. A closer look at Figure 3.4 reveals that probability of place 3 peaks during the weekend. In fact, we verified that the user spends leisure time in this area.

When the goal is location prediction, our method always predicts a single geographic coordinate, the posterior mean of the most probable place cluster, associated with the query user and weekhour. However, our model actually constructs a much richer representation of how users relate to their geography and creates a probability distribution over the entire space of geographic coordinates. For example, for the user above, even though place 0 dominates all other place clusters

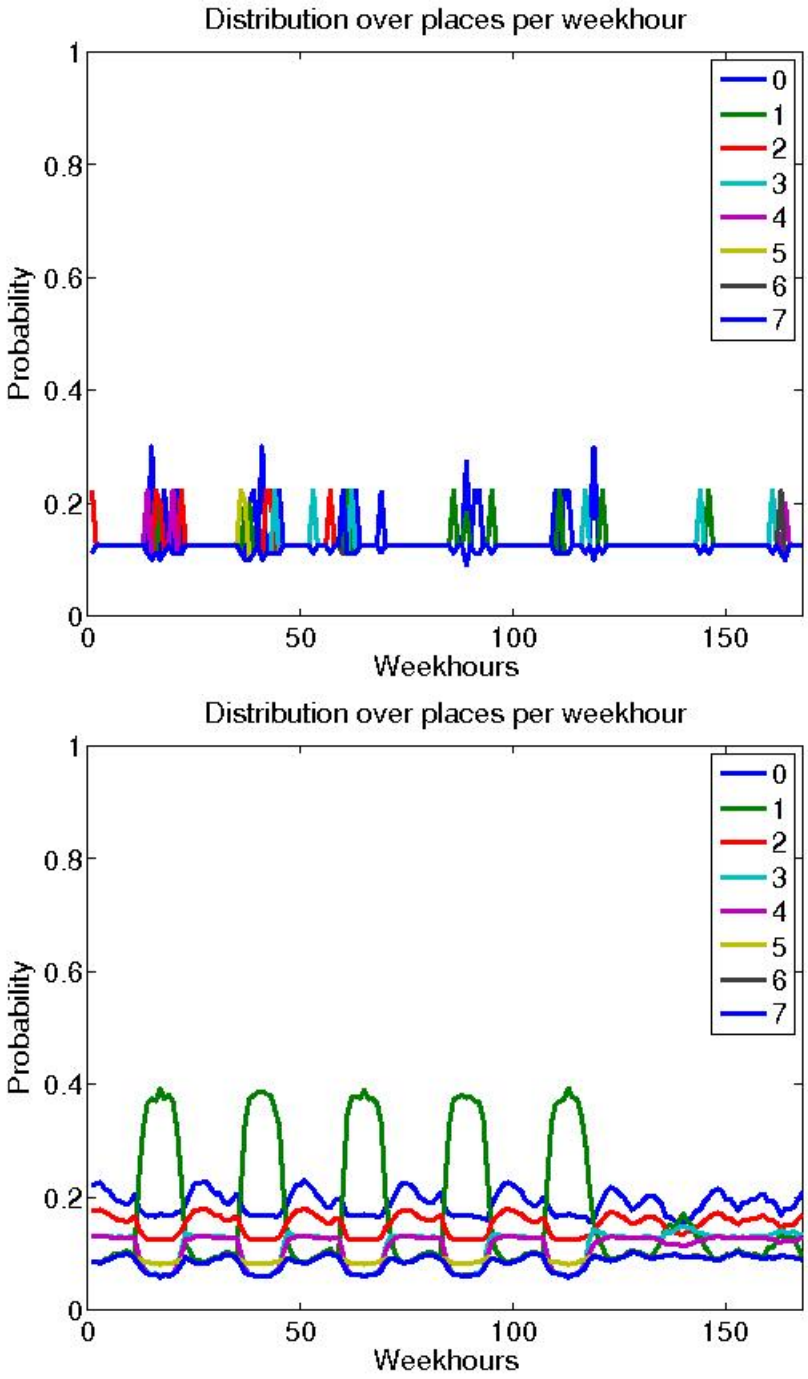

Figure 3.4: Place distributions inferred by CPM (top) and SPM (bottom). Because of sparsity, CPM learns place distributions that are jagged. In contrast, SPM shares information across users and learns place distributions that are smooth. Furthermore, SPM infers weekhour patterns that are reminiscient of home (Place 0) and work (Place 1), even though the user was not observed during most weekhours.

Figure 3.5: User's spatial cluster associated with place 0. The blue circles represent user's historical observations and the blue pin represents the cluster center inferred by our model. Since the predictive distribution associated with the cluster is a robust t-distribution, the outlier ping at the figure's far right side is not able to sway the cluster center too much. Furthermore, we verified that the cluster center, represented by a blue pin, is within 50 meters of the user's apartment.

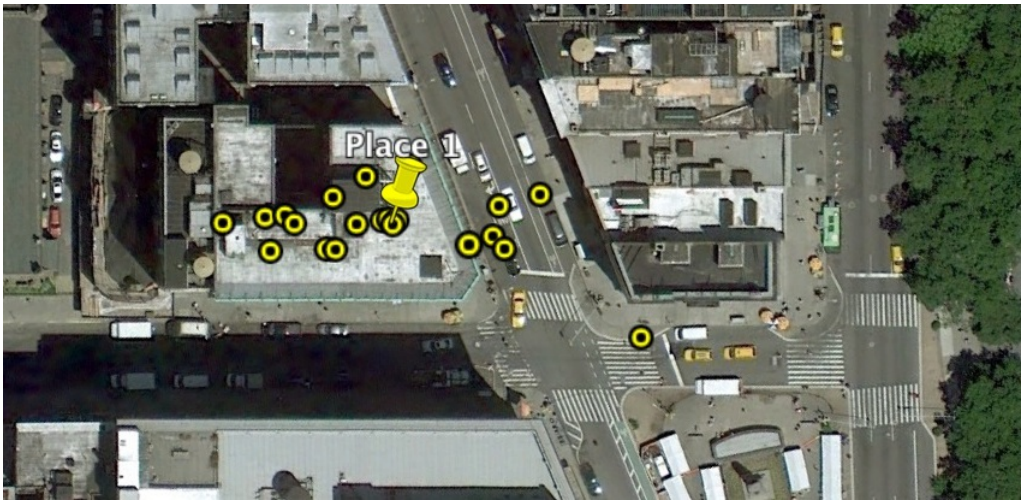

Figure 3.6: User's spatial cluster associated with place 1. The yellow circles represent user's historical observations and the yellow pin represents the cluster center inferred by our model. We verified that the cluster center, represented by a yellow pin, is within 50 meters of the user's work place.

Figure 3.7: User's spatial cluster associated with place 3. The orange circles represent user's historical observations and the orange pin represents the cluster center inferred by our model. The posterior covariance associated with place 3 is much larger than the ones associated with previous places. We verified that the user spends leisure time in this area.

during the weekend, the model nevertheless assigns a significant probability to place 2. Thus, even if we do not use the place clusters other than the top one for location prediction, learning a full probabilistic representation of users' whereabouts is important, as it enables many applications that make decisions based on rich geographic profiles rather than simple point estimates.

3.5 Discussion

In this chapter, we developed two unsupervised Bayesian graphical models, both of which construct a complete profile of users' whereabouts based on sparse geographic coordinates. We demonstrated that our models outperform a simple but strong baseline with respect to predicting users' future geographic coordinates. We also showed that, aside from location prediction, our models identify users' significant places and infer temporal descriptions of how users spend their time in them. We provided a quantitative analysis of our method on a sparse mobile application dataset, as well as qualitatively demonstrating the place clusters and hourly place

distributions inferred by our method on a particular user.

3.6 Appendix

3.6.1 Miscellaneous notation

Throughout the chapter, we use various notation to represent sets of indices and their cardinalities. Vectors $\boldsymbol{y}$ and $\boldsymbol{z}$ denote the factor and place assignments in SPM, respectively. Each vector entry is identified by a tuple index (u,w,n), where $u \in \{1,\ldots,U\}$ is a user, $w \in \{1,\ldots,W\}$ is a weekhour, and $n \in \{1,\ldots,N_{u,w}\}$ is an iteration index.

We refer to subset of indices using

$$M_{u_0,w_0}^{k_0,f_0} = \left\{(\dot{u},\dot{w},\dot{n}) \mid z_{\dot{u},\dot{w},\dot{n}} = k_0, y_{\dot{u},\dot{w},\dot{n}} = f_0, \dot{u} = u_0, \dot{w} = w_0\right\},$$

where u_0 denotes the user, w_0 denotes the weekhour, k_0 denotes the place, and f_0 denotes the factor. If we want the subset of indices to be unrestricted with respect to a category, we use the placeholder "$\cdot$". For example,

$$M_{u_0,w_0}^{\cdot,f_0} = \left\{(\dot{u},\dot{w},\dot{n}) \mid y_{\dot{u},\dot{w},\dot{n}} = f_0, \dot{u} = u_0, \dot{w} = w_0\right\}$$

has no constraints with respect to places.

Given a subset of indices denoted by M, the lowercase $m = |M|$ denotes its cardinality. For example, given a set of indices

$$M_{u_0,\cdot}^{\cdot,f_0} = \left\{(\dot{u},\dot{w},\dot{n}) \mid z_{\dot{u},\dot{w},\dot{n}} = \cdot, y_{\dot{u},\dot{w},\dot{n}} = f_0, \dot{u} = u_0, \dot{w} = w_0\right\},$$

its cardinality is

$$m_{u_0,\cdot}^{\cdot,f_0} = \left|M_{u_0,\cdot}^{\cdot,f_0}\right|.$$

For the collapsed Gibbs sampler, the sets of indices and cardinalities used in the derivations exclude the index that will be sampled. We use "$\sim$" to modify sets or cardinalities for this exclusion. Let (u,w,n) denote the index that will be sampled, then given an index set M, let $\tilde{M} = M - \{(u,w,n)\}$ represent the excluding set and

let $\tilde{m} = \left|\tilde{M}\right|$ represent the corresponding cardinality. For example,

$$\tilde{M}^{\cdot,f_0}_{u_0,\cdot} = M^{\cdot,f_0}_{u_0,\cdot} - \{(u,w,n)\}$$

and

$$\tilde{m}^{\cdot,f_0}_{u_0,\cdot} = \left|\tilde{M}^{\cdot,f_0}_{u_0,\cdot}\right|.$$

In the proof of Lemma 3.1, parameters $\tilde{v}^u_k$, $\tilde{\boldsymbol{\mu}}^u_k$, $\tilde{\Lambda}^k_u$, and $\tilde{p}^u_k$ are defined using cardinalities that exclude the current index (u,w,n). Similarly, in the proof of Lemma 3.5, parameters $\hat{\boldsymbol{\mu}}^u_k$, $\hat{\Lambda}^k_u$, and $\hat{v}^u_k$ are defined like their wiggly versions, but the counts used in their definition do not exclude the current index.

3.6.2 Probability distributions

Let Γ_2 denote a 2-dimensional gamma function, defined as

$$\Gamma_2(a) = \pi^{\frac{1}{2}} \prod_{j=1}^{2} \Gamma\left(a + \frac{1-j}{2}\right).$$

Let $\nu > 1$ and let $\Lambda \in \mathbb{R}^{2\times 2}$ be a positive definite scale matrix. The inverse-Wishart distribution, which is the conjugate prior to the multivariate normal distribution, is defined as

$$IW(\Sigma \mid \Lambda, \nu) = \frac{|\Lambda|^{\frac{\nu}{2}}}{2^{\nu}\Gamma_2\left(\frac{\nu}{2}\right)} |\Sigma|^{\frac{-\nu-3}{2}} \exp\left(-\frac{1}{2}\operatorname{tr}\left(\Lambda\Sigma^{-1}\right)\right).$$

Let $\Sigma \in \mathbb{R}^{2\times 2}$ be a positive definite covariance matrix and let $\boldsymbol{\mu} \in \mathbb{R}^2$ denote a mean vector. The multivariate normal distribution is defined as

$$\mathcal{N}\left(\boldsymbol{\ell} \mid \boldsymbol{\mu}, \Sigma\right) = (2\pi)^{-1} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\left(\boldsymbol{\ell}-\boldsymbol{\mu}\right)^T \Sigma^{-1}\left(\boldsymbol{\ell}-\boldsymbol{\mu}\right)\right).$$

Let $\nu > 1$ and let $\Sigma \in \mathbb{R}^{2\times 2}$, then the 2-dimensional t-distribution is defined as

$$t_v\left(x \mid \boldsymbol{\mu}, \Sigma\right) = \frac{\Gamma\left(\frac{\nu}{2}+1\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{|\Sigma|^{-\frac{1}{2}}}{\nu\pi} \left(1 + \frac{1}{\nu}\left(x-\boldsymbol{\mu}\right)^T \Sigma^{-1}\left(x-\boldsymbol{\mu}\right)\right)^{-\frac{\nu}{2}-1}.$$

Conclusion

In this thesis, we developed rigorous characterizations of three fundamental problems that involve location data, proposed novel machine learning methods for them, and justified these methods theoretically and empirically.

In Chapter 1, we presented a probabilistic graphical model that locates radio-tagged animals and learns their movement patterns. Our method incorporated both geographical and non-geographical spatial features and provided researchers with an interpretable model that enunciates the relationship between the animal and its environment. We showed that our model generalizes random walk and demonstrated empirically that a richer model improves animal location estimates. We also provided a fast parameter estimation algorithm and demonstrated its effectiveness both asymptotically and empirically. We applied our model to real datasets and demonstrated that it outperforms the most popular radio telemetry software package in ecology.

In Chapter 2, we formulated a new collaborative ranking framework, called Collaborative Local Ranking (CLR), which allowed us to formulate a wide variety of real-world ranking tasks that involve local and implicit feedback. We justified CLR with a bound on its generalization error. We also derived a simple alternating minimization algorithm and showed that each minimization step can be efficiently computed in time independent of the size of the training data. We applied CLR to a venue recommendation task and demonstrated that it outperforms state-of-the-art collaborative ranking methods, both in terms of generalization performance and running time.

In Chapter 3, we developed two unsupervised Bayesian graphical models, both of which construct a complete profile of users' whereabouts based on sparse geographic coordinates. We demonstrated that our models outperform a simple but

strong baseline with respect to predicting users' future geographic coordinates. We also showed that, aside from location prediction, our models identify users' significant places and infer temporal descriptions of how users spend their time in them. We provided a quantitative analysis of our method on a sparse mobile application dataset, as well as qualitatively demonstrating the place clusters and hourly place distributions inferred by our method on specific users.

There are a multitude of ways that our research can be improved upon. In Chapter 2, we described CLR as a general collaborative ranking framework, and applying it to domains that do not involve location data would strengthen its case. In Chapter 3, we used a computationally expensive search procedure to determine the optimal number of places for each user, and designing non-parametric versions of our models with stochastic inference could improve both the efficiency and generalization capability of our methods. We leave the resolution of these open problems to future work.

Bibliography

[1] Richard Anderson-Sprecher. Robust estimates of wildlife location using telemetry data. *Biometrics*, 1994.

[2] Richard Anderson-Sprecher and Johannes Ledolter. State-space analysis of wildlife telemetry data. *Journal of the American Statistical Association*, 1991.

[3] Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 143–152, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-0747-5. doi: 10.1145/2124295.2124314. URL http://dx.doi.org/10.1145/2124295.2124314.

[4] Robert M. Bell and Yehuda Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9(2):75–79, December 2007. ISSN 1931-0145. doi: 10.1145/1345448.1345465. URL http://dx.doi.org/10.1145/1345448.1345465.

[5] James Bennett and Stan Lanning. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, August 2007. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.117.8094.

[6] Adam L. Berger, Vincent J. Della Pietra, and Stephen A. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 1996.

[7] Stéphane Boucheron, Olivier Bousquet, and Gabor Lugosi. Theory of classification : A survey of some recent advances.

ESAIM: Probability and Statistics, 9:323–375, 2005. URL http://cat.inist.fr/?aModele=afficheN&cpsidt=17367966.

[8] D. Brockmann, L. Hufnagel, and T. Geisel. The scaling laws of human travel. *Nature*, 439(7075):462–465, January 2006. ISSN 0028-0836. doi: 10.1038/nature04292. URL http://dx.doi.org/10.1038/nature04292.

[9] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. 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, pages 1082–1090, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020579. URL http://dx.doi.org/10.1145/2020408.2020579.

[10] Margaret C. Crofoot, Ian C. Gilby, Martin C. Wikelski, and Roland W. Kays. Interaction location outweighs the competitive advantage of numerical superiority in Cebus capucinus intergroup contests. In *Proceedings of the National Academy of Sciences of the United States of America*, 2008.

[11] William R. Davie. *Telephony*, pages 251–264. Focal Press, 2012. ISBN 9780240824567. doi: 10.1016/b978-0-240-81475-9.50018-9. URL http://dx.doi.org/10.1016/b978-0-240-81475-9.50018-9.

[12] Manlio De Domenico, Antonio Lima, and Mirco Musolesi. Interdependence and predictability of human mobility and social interactions, October 2012. URL http://arxiv.org/abs/1210.2376.

[13] Bernard Delyon, Marc Lavielle, and Eric Moulines. Convergence of a stochastic approximation version of the EM algorithm. *Annals of Statistics*, 1999.

[14] Nathan Eagle and Alex S. Pentland. Eigenbehaviors: identifying structure in routine. *Behavioral Ecology and Sociobiology*, 63(7):1057–1066, May 2009. ISSN 0340-5443. doi: 10.1007/s00265-009-0739-0. URL http://dx.doi.org/10.1007/s00265-009-0739-0.

[15] Elena Erosheva, Stephen Fienberg, and John Lafferty. Mixed-membership models of scientific publications. *Proceedings of the National Academy*

of Sciences of the United States of America, 101(Suppl 1):5220–5227, April 2004. ISSN 1091-6490. doi: 10.1073/pnas.0307760101. URL http://dx.doi.org/10.1073/pnas.0307760101.

[16] Stephen Eubank, Hasan Guclu, V. S. Anil Kumar, Madhav V. Marathe, Aravind Srinivasan, Zoltan Toroczkai, and Nan Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429(6988): 180–184, May 2004. ISSN 0028-0836. doi: 10.1038/nature02541. URL http://dx.doi.org/10.1038/nature02541.

[17] Katayoun Farrahi and Daniel G. Perez. Discovering routines from large-scale human locations using probabilistic topic models. *ACM Trans. Intell. Syst. Technol.*, 2(1), January 2011. ISSN 2157-6904. doi: 10.1145/1889681.1889684. URL http://dx.doi.org/10.1145/1889681.1889684.

[18] Rina Foygel, Ruslan Salakhutdinov, Ohad Shamir, and Nati Srebro. Learning with the weighted trace-norm under arbitrary sampling distributions. In John S. Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, Kilian Q. Weinberger, John S. Taylor, Richard S. Zemel, Peter L. Bartlett, Fernando C. N. Pereira, and Kilian Q. Weinberger, editors, *NIPS*, pages 2133–2141, 2011. URL http://dblp.uni-trier.de/rec/bibtex/conf/nips/FoygelSSS11.

[19] Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008. ISSN 0028-0836. doi: 10.1038/nature06958. URL http://dx.doi.org/10.1038/nature06958.

[20] Thomas L. Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228–5235, April 2004. ISSN 1091-6490. doi: 10.1073/pnas.0307752101. URL http://dx.doi.org/10.1073/pnas.0307752101.

[21] Qiang Hao, Rui Cai, Changhu Wang, Rong Xiao, Jiang M. Yang, Yanwei Pang, and Lei Zhang. Equip tourists with knowledge mined from travelogues. In *Proceedings of the 19th international conference on*

World wide web, WWW '10, pages 401–410, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772732. URL http://dx.doi.org/10.1145/1772690.1772732.

[22] Mike Hazas, James Scott, and John Krumm. Location-Aware computing comes of age. *Computer*, 37(2):95–97, February 2004. ISSN 0018-9162. doi: 10.1109/MC.2004.1266301. URL http://dx.doi.org/10.1109/MC.2004.1266301.

[23] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '02, pages 133–142, New York, NY, USA, 2002. ACM. ISBN 1-58113-567-X. doi: 10.1145/775047.775067. URL http://dx.doi.org/10.1145/775047.775067.

[24] Ian D. Jonsen, Ransom A. Myers, and Joanna Mills Flemming. Meta-analysis of animal movement using state-space models. *Ecology*, 2003.

[25] Ian D. Jonsen, Joanna Mills Flemming, and Ransom A. Myers. Robust state-space modeling of animal movement data. *Ecology*, 2005.

[26] Berk Kapicioglu, Robert E. Schapire, Martin Wikelski, and Tamara Broderick. Combining spatial and telemetric features for learning animal movement models. In *Uncertainty in Artificial Intelligence*, 2010.

[27] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1401944. URL http://dx.doi.org/10.1145/1401890.1401944.

[28] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, chapter 5, pages 145–186. Springer US, Boston, MA, 2011. ISBN 978-0-387-85819-7. doi: 10.1007/978-0-387-85820-3_5. URL http://dx.doi.org/10.1007/978-0-387-85820-3_5.

[29] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.263. URL http://dx.doi.org/10.1109/MC.2009.263.

[30] Russell V. Lenth. On finding the source of a signal. *Technometrics*, 1981.

[31] Lin Liao, Dieter Fox, and Henry Kautz. Location-based activity recognition. In *In Advances in Neural Information Processing Systems (NIPS*, pages 787–794, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.108.7802.

[32] Lin Liao, Dieter Fox, and Henry Kautz. Extracting places and activities from GPS traces using hierarchical conditional random fields. *Int. J. Rob. Res.*, 26 (1):119–134, January 2007. ISSN 0278-3649. doi: 10.1177/0278364907073775. URL http://dx.doi.org/10.1177/0278364907073775.

[33] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, January 2003. ISSN 1089-7801. doi: 10.1109/MIC.2003.1167344. URL http://dx.doi.org/10.1109/MIC.2003.1167344.

[34] Robert Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the Sixth Conference on Natural Language Learning*, 2002.

[35] Kanti V. Mardia and Peter E. Jupp. *Directional Statistics*. Wiley, 1999.

[36] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322, August 2010. URL http://www.jmlr.org/papers/volume11/mazumder10a/mazumder10a.pdf.

[37] Juan Manuel Morales, Daniel T. Haydon, Jacqui Frair, Kent E. Holsinger, and John M. Fryxell. Extracting more out of relocation data: building movement models as mixtures of random walks. *Ecology*, 2004.

[38] Jorge Nocedal and Stephen Wright. *Numerical Optimization*. Springer, 2006.

[39] T. O. Oshin, S. Poslad, and A. Ma. Improving the Energy-Efficiency of GPS based location sensing smartphone applications. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on*, pages 1698–1705. IEEE, June 2012. ISBN 978-1-4673-2172-3. doi: 10.1109/trustcom.2012.184. URL http://dx.doi.org/10.1109/trustcom.2012.184.

[40] Otso Ovaskainen. Habitat-specific movement parameters estimated using mark-recapture data and a diffusion model. *Ecology*, 2004.

[41] Toby A. Patterson, Len Thomas, Chris Wilcox, Otso Ovaskainen, and Jason Matthiopoulos. Statespace models of individual animal movement. *Trends in Ecology and Evolution*, 2008.

[42] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars S. Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press. ISBN 978-0-9749039-5-8. URL http://portal.acm.org/citation.cfm?id=1795167.

[43] Jasson D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pages 713–719, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102441. URL http://dx.doi.org/10.1145/1102351.1102441.

[44] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[45] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the International Conference on Machine Learning*, volume 25, 2008.

[46] Robert S. Schick, Scott R. Loarie, Fernando Colchero, Benjamin D. Best, Andre Boustany, Dalia A. Conde, Patrick N. Halpin, Lucas N. Joppa, Catherine M.

McClellan, and James S. Clarks. Understanding movement data and movement processes: current and emerging directions. *Ecology Letters*, 2008.

[47] Shai S. Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: primal estimated sub-gradient solver for SVM. *Math. Program.*, 127(1):3–30, March 2011. ISSN 0025-5610. doi: 10.1007/s10107-010-0420-4. URL http://dx.doi.org/10.1007/s10107-010-0420-4.

[48] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-László Barabási. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, February 2010. ISSN 1095-9203. doi: 10.1126/science.1177170. URL http://dx.doi.org/10.1126/science.1177170.

[49] James C. Spall. *Introduction to Stochastic Search and Optimization*. Wiley-Interscience, 2003.

[50] Nathan Srebro, Noga Alon, and Tommi Jaakkola. Generalization error bounds for collaborative prediction with Low-Rank matrices. In *NIPS*, 2004. URL http://dblp.uni-trier.de/rec/bibtex/conf/nips/SrebroAJ04.

[51] Nathan Srebro, Jason D. M. Rennie, and Tommi S. Jaakola. Maximum-Margin matrix factorization. In *Advances in Neural Information Processing Systems 17*, volume 17, pages 1329–1336, 2005. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.59.118.

[52] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alex J. Smola. COFI RANK - maximum margin matrix factorization for collaborative ranking. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *NIPS*. MIT Press, 2007.

[53] Markus Weimer, Alexandroos Karatzoglou, and Alex Smola. Improving maximum margin matrix factorization. In *ECML*, 2008. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.140.4647.

[54] Marius Wernke, Pavel Skvortsov, Frank Dürr, and Kurt Rothermel. A classification of location privacy attacks and approaches.

pages 1–13, 2012. doi: 10.1007/s00779-012-0633-z. URL http://dx.doi.org/10.1007/s00779-012-0633-z.

[55] Martin Wikelski, Roland W. Kays, N. Jeremy Kasdin, Kasper Thorup, James A. Smith, and George W. Swenson Jr. Going wild: what a global small-animal tracking system could do for experimental biologists. *Journal of Experimental Biology*, 2007.

[56] Laurent Younes. Parametric inference for imperfectly observed Gibbsian fields. *Probability Theory and Related Fields*, 1989.

[57] Yilin Zhao. Standardization of mobile phone positioning for 3G systems. *Communications Magazine, IEEE*, 40(7):108–116, July 2002. ISSN 0163-6804. doi: 10.1109/MCOM.2002.1018015. URL http://dx.doi.org/10.1109/MCOM.2002.1018015.

[58] Vincent W. Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative location and activity recommendations with GPS history data. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 1029–1038, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: 10.1145/1772690.1772795. URL http://dx.doi.org/10.1145/1772690.1772795.

[59] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei Y. Ma. Mining interesting locations and travel sequences from GPS trajectories. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 791–800, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-487-4. doi: 10.1145/1526709.1526816. URL http://dx.doi.org/10.1145/1526709.1526816.