UNIVERSITY OF CALIFORNIA,
IRVINE

Context-Based Smoothing for Personalized Prediction Models

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Moshe Lichman

Dissertation Committee:
Professor Padhraic Smyth, Chair
Professor Sharad Mehrotra
Professor Sameer Singh

2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to start by thanking my advisor, Dr. Padhraic Smyth. His guidance, support and mentorship have been invaluable throughout the entire duration of my PhD. Time and time again it felt like our research interest are aligned, which increased the level of excitement in everything that I did and motivated me to come up with new research ideas. I am grateful for having you as my advisor and hope that you will continue doing what you are doing with others as well.

I am grateful to my candidacy and final defense committee members Dr. Sharad Mehrotra, Dr. Sameer Singh, Dr. Alex Ihler, Dr. Rina Dechter, and Dr. Michael Cooper. Thank you to my collaborators, Dimitrios Kotzias, Dr. John Hipp and Christopher Jay Bates for all your hard work and many useful discussions. To the past and current members of the UCI DataLab research group, I have learned a lot from each of you. You have challenged me to be a better researcher and I thank you for that. A special thanks go to Dr. Jimmy Foulds, who has been both a great friend and a mentor throughout my first few years at UCI. Too Nicholas Navaroli, for extremely productive talks and for making long nights at the lab seem like a fun thing. To Dimitrios Kotzias, with which I collaborated in my work and for putting up with my intense personality during hard times — you have been and still are a great friend. And to Eric Nalisnick and Dr. Kevin Bache with whom I have had many conversations regarding both research ideas and personal matters.

I would like to thank my family who supported me from afar with love and motivation. My parents, Dalia and Moti Lichman, who taught me the value of pursuing knowledge from a young age and always pushed me achieve more. And my two sisters, Maya and Inbar, who helped shape me into who I am today and motivated me to enjoy life, even while working hard.

To my beloved partner, Kristin Roher, I want to thank for all the love and support you have given me. The last year has been particularly hard and you have been there for me through good times and through more than a few bad ones. I am forever in your debt and I cannot imagine going through this without you. Thank you.

I owe so much to my amazing co-workers at tracx. I know for a fact that I would not be where I am today without everything you have taught me. You have given me tools, both professionally and outside of work, that one could only dream of getting, and I will use them throughout the rest of my career. So Noam, Niv, Ofer, Shachar, Yaniv, Levona and the rest of the current and past tracx employees — thank you.

Last but not least, during my many years at UCI and the United States I had the fortune of meeting amazing people that referring to them as anything short of family will be inappropriate. Kevin Bache and Kyle Benson, and our incredible psy trance community, Sarah and Yossi Arditi, Jason and Kristin Giannantonio, Denny and Oren Ashkenazi, Levi, Yaniv, Kristi, Rani, Steve and so many others, every moment I spent with you (on or outside the desert dance floor) meant more than I can explain in word, thank you!

This dissertation was supported by a gift from the University Affairs Committee at Xerox Corporation, by the National Science Foundation under award IIS-1320527, by the office of Navel Research/Multidisciplinary University Research Initiative under grant number N00014-08-1-1015 and by Google Faculty Award.

CURRICULUM VITAE

Moshe Lichman

EDUCATION

Doctor of Philosophy in Computer Science **2017**
University of California, Irvine *Irvine, California*

Bachelor of Science in Computational Sciences **2011**
Tel-Aviv University *Tel-Aviv, Israel*

RESEARCH EXPERIENCE

Graduate Research Assistant **2012–2017**
University of California, Irvine *Irvine, California*

REFEREED JOURNAL PUBLICATIONS

John R. Hipp, Christopher Bates, Padhraic Smyth and Moshe Lichman. **Using Social Media to Measure Temporal Ambient Population: Does it Help to Explain Local Crime Rates?**. *Justice Quarterly — In Review.*

REFEREED CONFERENCE PUBLICATIONS

Moshe Lichman and Padhraic Smyth. **Modeling Human Location Data with Mixtures of Kernel Densities**. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014.*

Moshe Lichman, Dimitrios Kotzias and Padhraic Smyth. **Personalized Location Models with Adaptive Mixtures**. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2016.*

Moshe Lichman and Padhraic Smyth. **Prediction of Sparse User-Item Consumption Rates with Zero-Inflated Poisson Regression**. *Proceedings of the 27th International Conference on World Wide Web, 2018 — In Review.*

TEACHING EXPERIENCE

Teaching Assistant - Machine Learning and Data-Mining **2017**
University of California, Irvine *Irvine, California*

Teaching Assistant - Project in Artificial Intelligence **2017**
University of California, Irvine *Irvine, California*

Teaching Assistant - Machine Learning and Data-Mining **2014**
University of California, Irvine *Irvine, California*

PROFESSIONAL EXPERIENCE

Software Engineer / Research Intern **2016**
Google *Irvine, California*

Software Engineer / Research Intern **2015**
Google *Irvine, California*

Research Intern **2014**
Xerox Research Center *Webster, New York*

Research Intern **2013**
Xerox Research Center *Webster, New York*

Data Scientist **2010–2012**
tracx *Tel-Aviv, Israel*

Software Engineer **2009–2010**
Intel *Petah-Tikva, Israel*

Malware Specialist **2008–2009**
RSA Security *Herzliya, Israel*

ABSTRACT OF THE DISSERTATION

Context-Based Smoothing for Personalized Prediction Models

By

Moshe Lichman

Doctor of Philosophy in Computer Science

University of California, Irvine, 2017

Professor Padhraic Smyth, Chair

Software applications that digitally collect and store individual activity data are increasingly prevalent in our daily lives. With the widespread availability of such data, there is a growing demand for predictive models that can provide a personalized experience, i.e., adapt the application to an individual's behavioral patterns. In recent years, such models focused on problems in which the predictive task is to recommend new items that the user has not yet interacted with. Yet, there is also a large number of applications in which users interact with both old (i.e., that a user has interacted with in the past) and new items, such as location tracking services, online music streaming platforms, and e-commerce websites. In this dissertation, we investigate models that can predict an individual's interactions with both previously interacted ("old") and novel ("new") items. We develop novel probabilistic models that combine memory-based components with components that can use contextual information to make predictions that generalize to new items. We show that this approach not only models detailed aspect of both repetitive and novelty-seeking behavior but also infers the balance between the two from data.

For the dissertation's first contribution, we focus on location data in which the observed interactions are in the form of spatial footprints collected by digital services such as location based social media platforms. We predict an individual's future locations using a kernel den-

sity mixture model (mixture-KDE) for continuous events. Using multiple evaluation metrics based on location prediction and anomaly detection, we compare our model to existing methods and baselines based on latent-space representations and find that our mixture-KDE is superior.

Next, we develop a model for discrete location event data. We investigate the use of geo-parcel annotations to discretized continuous event data and show that it creates better personalized data when compared to existing disjoint grid-cell based approaches. We develop a framework that predicts an individual's future locations using an adaptive mixture of past observations and contextual information. We compare our approach to a variety of existing latent-space-based approaches and show that our model outperforms them on multiple evaluation metrics.

In the last chapter, we investigate a broader range of data sets byond location data that describe consumption of both new and old items. We describe a data-driven zero-inflated Poisson model that predicts the rate at which a user will consume a given item. We demonstrate the advantages of our model with a series of quantitative and qualitative experiments. For the former, we compare our model to a variety of state-of-the-art approaches and baselines and again show that our model provides more accurate predictions across multiple data sets from different domains.

Chapter 1

Introduction

Individual-level activity data is increasingly prevalent in the digital world. With the accelerating adoption of technology, an ever-increasing portion of our lives is being digitally collected and stored. This collection provides rich individual-level observations of time-stamped event data of interactions between individuals and a variety of virtual and real items. Examples of such interactions include purchasing books online, online streaming of movies and music, social interactions and even spatial footprints available through check-ins on location based social media platforms.

With the constantly expanding volume and variety of this type of data, there is a rapidly growing demand for applications that provide personalized experiences that can adapt to context. Perhaps the most common example of such individual-level experience is the use of recommender systems for which an individual's past activity within the application influences the recommendation of future items (e.g., movies, shows, music, books, etc.). Another recent example is the case of virtual personal assistants such as the Amazon Alexa, Google Now and Microsoft Cortana. By recording an individual's activity over time, the software can provide more personalized assistance such as providing relevant news, scheduling ap-

pointments, reporting relevant traffic and weather information at the right time, and even automatically adding items to a shopping list based on past purchases.

While the methods being used to produce the individual-level personalization in each of these application can differ widely, they are all based on predictive models of individual person-item-level interactions. In broad terms, this type of modeling provides prediction of a set of items an individual will interact with in the future based on past observations. For example, the recommendation we received from online movie streaming services is based on our past ratings of movies we watched. As a result, the ability to develop accurate individual-level predictive models is becoming increasingly important.

Building accurate individual-level predictive models in a typical digital environment is a challenging task for two primary reasons. First, limited budget (e.g., monetary or time), exposure, or simply preference, limits the number of items with which individuals can interact. As a result, the number of unique items each individual interacts with is only a small fraction of the total set of available items. Second, the behavioral patterns that guide the individual's decision on which items to interact with can be characterized as a mixture of repeated behavior (i.e., interactions with items the individual interacted with in the past) and novelty-seeking behavior that leads to interactions with new "unseen" items. Therefore, a predictive individual-level interaction model must generalize about an individual's behavior and preferences beyond an individual's past observations. This challenge is particularly hard in the context of applications in which the predictive task is also required to predict future interactions with a set of "old" and "new" items. It is then not only required for the model to generalize beyond past observations, but also to infer the balance of an individual's distribution of interest between items which were represented in the data and new items which are only available at prediction time.

1.1 Generalization Using Latent-Space Models

A common approach to addressing the generalization problem is to use individual-level predictive models based on latent-space representations. These models provide a principled way of generalizing for both continuous and discrete event data.

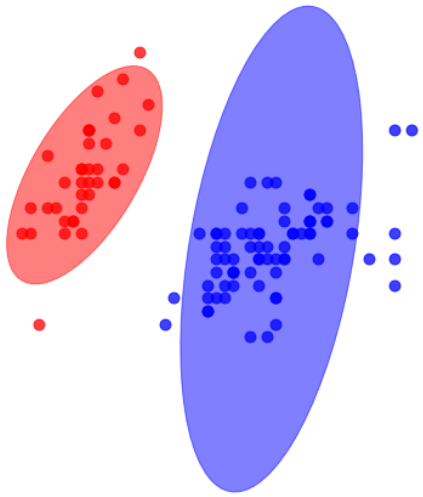


Figure 1.1: Illustration of a 2-dimensional Gaussian Mixture Model with two Gaussian components fitted to synthetic data set. Best viewed in color.

For continuous event data, latent-space models represent any point in the domain in a lower d-dimensional space. Each individual is then represented as a basis over said space. Figure 1.1 shows a simple example of a 2-dimensional continuous space Gaussian Mixture Model (GMM) with 2 components ($d = 2$). Each point in the space is represented by a set fixed d number of mixture components where each component is the Gaussian density function. An individual's preference is represented as a categorical distribution over the d mixture components (for details we refer the reader to [55] chapter 11.2.1). Gaussian Mixture Models are typically used for modeling location-based event data in which each event is an observation of an individual's location in the form of GPS coordinates: $(longitude, latitude)$ [13, 26, 36].

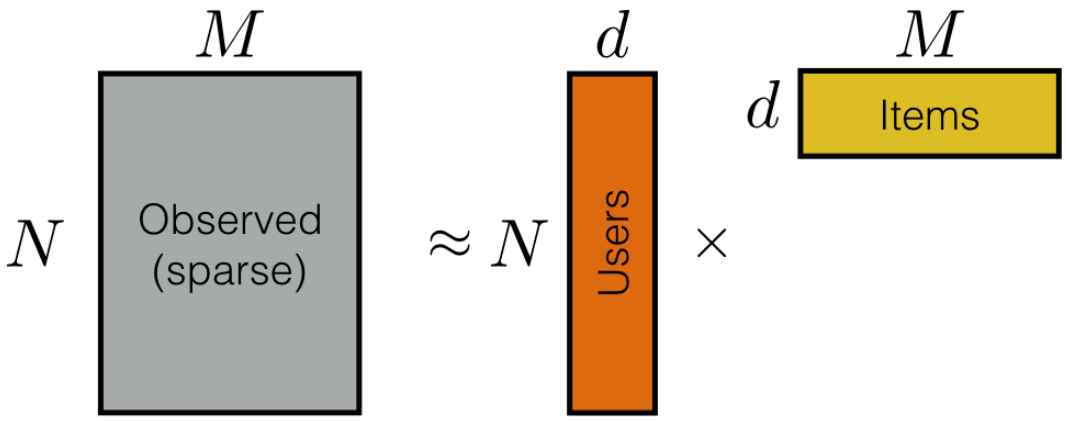


Figure 1.2: Illustration of the Matrix Factorization algorithm. The sparse observations matrix (on the left) is approximated as a product of two dense matrices representing the individual's "category" preferences and the item association of each "category".

Discrete latent space models are perhaps more commonly used for modeling individual-level event data. In those models, the set of possible items can be thought of as a vocabulary with M unique options such as M movies, M restaurants, etc. The event data for N individuals and M items can be represented as an $N \times M$ matrix in which the i, j element corresponds to an interaction (or a number of interactions) between user i and item j. Matrix-factorization approaches, illustrated in Figure 1.2, are widely applied in this context. These latent-space models learn a d-dimensional representation for both items and individual with $d << (M, N)$. The jth element in the d-dimensional item and individual vector represents, roughly, the level of item's association with the jth "category" and the relative individual's preference for the jth "category" respectively (out of d "categories"). Some of the many modeling techniques that are based on matrix-factorization include: PCA, LSI [45], Topic Modeling [4], and Poisson Matrix-Factorization [27].

These methods have been shown to be highly accurate in predictive tasks for which the goal is to infer the novelty-seeking behavior of individuals (i.e., interacting with new items).

There are certainly a large number of applications for which such an emphasis is required (such as picking a movie to watch, a book to purchase and so on), making latent-space approaches highly appealing. However, there are also a variety of personalization applications in which the individual's behavior is a mixture of repeated and novelty-seeking behaviors. Two examples in which users exhibit a mixture of repeat and novelty-seeking behavior are: 1) online music streaming applications in which users listen to the same artists habitually and 2) micro-blogging websites such as Twitter and reddit where individuals will engage in the same discussion repeatedly. Surprisingly, there has been little work to date on evaluating latent-space methods in this context.

To that end, in this dissertation we focus on individual-level event data that contains interactions between both "new" and "old" items and develop new approaches for probabilistic predictive modeling. In broad terms, our models are based on a combination of a memory-based component and a generalization component that is estimated from contextual information. In addition to capturing the detailed aspects of both repetitive and novelty-seeking behavior, the use of the two components allows the model to infer the balance between the individual's tendency for novelty-seeking and repetitive behavior. We develop different predictive models based on this approach and apply them to a variety of applications spanning multiple data domains. In terms of predictive power, we find that our models outperform all existing methods and simple baselines, particularly when a significant portion of the interactions in the data can be classified as repetitive behavior.

In the remainder of this chapter, we summarize the individual-level user-item interaction data sets used throughout the dissertation. The chapter concludes with an outline of the remaining dissertation chapters, summarizing the contributions of each chapter.

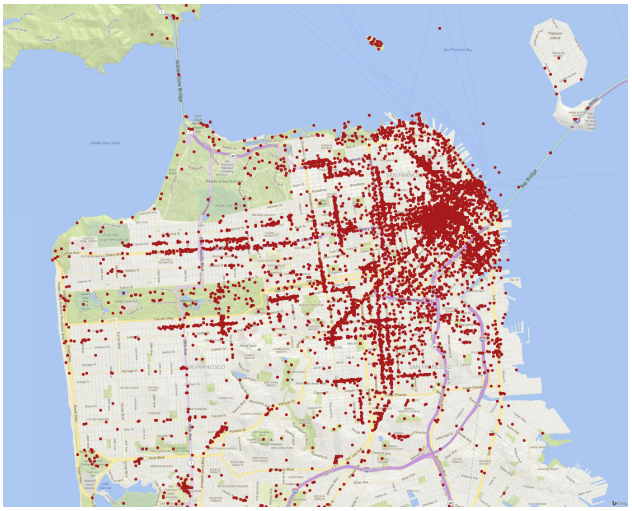


Figure 1.3: Gowalla check-in data over 20 month period in San Francisco, CA.

1.2 Individual-Item Interaction Data

The models and algorithms presented throughout this dissertation are evaluated using multiple individual-level event data sets. Each event in the data consists of tuple (i, j, t) and indicates an interaction between individual i with item j at time t. In chapters two and three, our focus is on location data sets in which an event refers to individual i observed at location j at time t. Figure 1.3 shows an example of location data taken from the widely used Gowalla data set [13] (which will be described later in the dissertation) in the area of San Francisco, California.

In chapter four, we investigate more general non-location data sets. We explore different data domains such as online music streaming, restaurants reviews and online social interactions on micro-blogging websites. We refer to this type of interactions as consumptions and model the number of times (or rate at) which an individual i consumes item j at time t where time t refers to a discrete time window (e.g., day, week, month etc.). Figure 1.4 illustrates this type of data, where each matrix indicates the number of times individual i (row) consumed item j (column) at a certain time t.

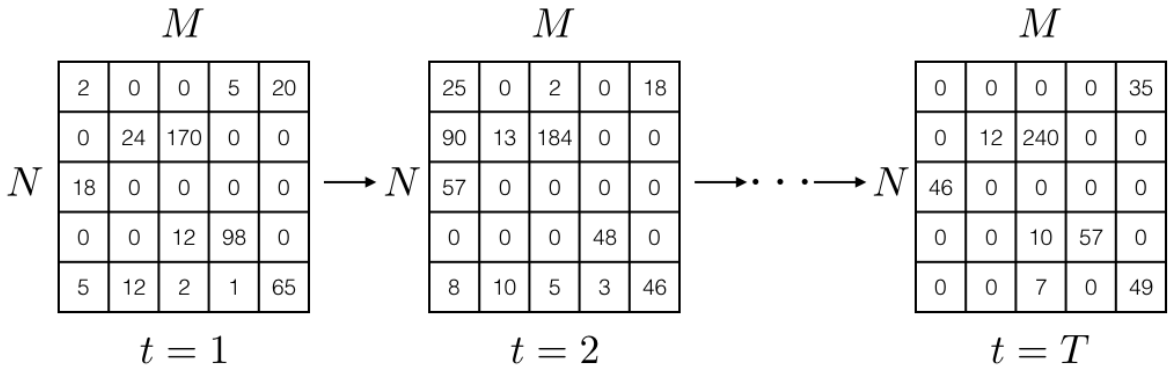


Figure 1.4: Illustration of consumption data over time. Each i, j element in a matrix represents the number of times user i consumed item j at a certain discrete time t.

1.3 Dissertation Outline and Contributions

The structure of the dissertation, including novel contributions, is as follows:

Chapter 2 addresses the problem of modeling continuous-space human location data. The focus of this chapter is to develop predictive models that can infer individuals' future locations. We investigate an adaptive bandwidth kernel density estimation approach and present a mixture of kernel densities model that balances repetitive and novelty-seeking behavior patterns (i.e., individuals both returning to known locations and visiting new locations) by combining individual-level models with contextual information.

Contributions of Chapter 2 include:

- A thorough investigation of the use of kernel density estimation technique for predicting individuals' future locations. This includes investigating multiple approaches for the model parameterization (i.e., different approaches for choosing bandwidths).
- A mixture of kernel densities approach for modeling individual-level human location data that smoothes over multiple pieces of spatial-level information.

- An extensive evaluation of our approach compared to existing methods for continuous-space modeling of human location data.

- An anomaly detection evaluation where we apply our model to the task of simulated fraud detection by detecting inconsistent spatial behavior. Results indicate that the mixture of kernel densities approach produces more accurate results than all compared models.

In **Chapter 3**, we address the problem of modeling human location data in discrete space. In the context of location data, locations are typically categorized into a "vocabulary" of L known point of interest (i.e., restaurants, bars, popular locations). Discrete spatial data (this chapter) and continuous spatial data (previous chapter) are complementary: discrete data can strengthen our inference by combining all events associated with a particular location, while continuous events can provide more precise spatial information. In this chapter, we investigate the use of an adaptive-mixture model that learns a categorical probability distribution over the set of L locations from an individual's historical data. Our model uses contextual information such as spatial proximity, social ties, and overall popularity to generalize to new places and, similarly to Chapter 2, infer the balance between repetitive and novelty-seeking behavior directly from the data.

Contributions of Chapter 3 include:

- A method for discretizing continuous-space location data based on geographic-parcels. We show that this approach provides a more detailed personalization of discrete data than the existing disjoint-cell-based approaches (e.g., grid boxed or cell tower areas).

- The introduction of an adaptive-mixture model framework that learns each individual's balance of explore-exploit behavior as a mixture of individual level models and smoothing components based on contextual information like spatial and social simi-

larities. In addition, this approach automatically infers the optimal level of smoothing for maximal predictive accuracy.

- A comparison between various state-of-the-art discrete-space models using multiple evaluation metrics for predicting future individual-location data in both batch-evaluation and online-evaluation experiments.

In **Chapter 4** we broaden the range of data sets under investigation beyond location data to include online music streaming, restaurants reviews and micro-blogging data sets. In particular we focus on a consumption behavior that is a combination over time of new and "known" items per user. We propose a data-driven zero-inflated Poisson regression approach for modeling sparse consumption patterns consisting of combinations of both new and known items. While zero-inflated Poisson regression models are well-known (e.g., in statistics) this is the first application (that we are aware of) of these types of models to very high-dimension user-item consumption data.

Contributions of Chapter 4 include:

- A thorough investigation of user-item digital consumption rate data sets from three different domains where the individual's behavior consists of a combination of consumption of new and "known" items (repeated consumption).

- A data-driven zero-inflated Poisson regression model for predicting individual-level user-item consumption rates. Our proposed model uses individual-level data and contextual information to model both repetitive and novel behavior and infer the balance from the data.

- Extensive empirical evaluation using three large-scale data sets from different domains, demonstrating systematic improvement in prediction accuracy in both batch and online experiments.

- Qualitative evaluation demonstrating the advantages of our proposed model in the context of predicting sparse user-item consumption rates.

Chapter 2

Modeling Human Location Data with Mixtures of Kernel Densities

Location-based data is increasingly prevalent with the rapid increase and adoption of mobile devices. In this chapter we address the problem of learning spatial density models, focusing specifically on individual-level data. Modeling and predicting a spatial distribution for an individual is a challenging problem given both (a) the typical sparsity of data at the individual level and (b) the heterogeneity of spatial mobility patterns across individuals. We investigate the application of kernel density estimation (KDE) to this problem using a mixture model approach that can interpolate between an individual's data and broader patterns in the population as a whole. The mixture-KDE approach is evaluated on two large geolocation/check-in data sets, from Twitter and Gowalla, with comparisons to non-KDE baselines, using both log-likelihood and detection of simulated identity theft as evaluation metrics. Our experimental results indicate that the mixture-KDE method provides a useful and accurate methodology for capturing and predicting individual-level spatial patterns in the presence of noisy and sparse data.

2.1 Introduction

Human location data is increasingly available in the modern mobile world, often in the form of geolocation tags attached to human behavioral data such as phone calls, text messages, social media activities, and more. With the widespread availability of this data there is increasing interest across a variety of fields of study in creating accurate models to characterize the spatial distributions of populations and individuals. For example, Sadilek et al. [58] analyze the spread of infectious diseases through geolocation data from Twitter, opening up potential new approaches for real-time computational epidemiology. Culotta [16] studies a large variety of health related statistics in different counties in the U.S using Twitter. Cranshaw et al. [14] use Foursquare check-in data to identify local spatial clusters within urban areas, with potential applications in urban planning to economic development and resource allocation. From a commercial perspective, location-based services and personalization are becoming increasingly important to individual mobile device users, with an increasing number of applications that are location-aware, including maps, localized search results, recommender systems, and advertising [66].

In this chapter we focus on the problem of developing accurate individual-level models of spatial location based on geolocated event data. The term "event" here can be interpreted in a broad context—examples include communication events such as phone calls or text messages, check-in events, social media actions, and so on. The goal is to be able to accurately characterize and predict the spatial pattern of an individual's events. The problem is challenging for two main reasons. Firstly, there is often relatively little data for many of the individuals making it difficult to build accurate models at the individual level. Secondly, there is often considerable variety and heterogeneity in the spatial pattern of events of individual users, rendering techniques such as clustering less than ideal for individual-level modeling.

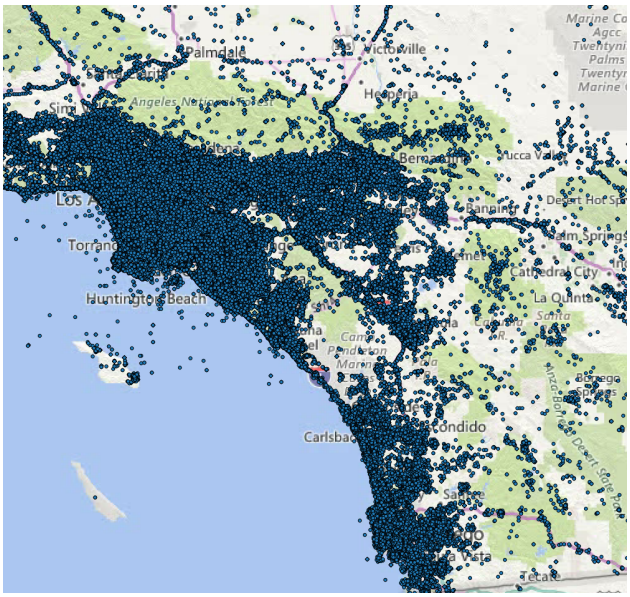


Figure 2.1: Geolocated Twitter tweets over the period of July-August 2013 from all individuals in southern California.

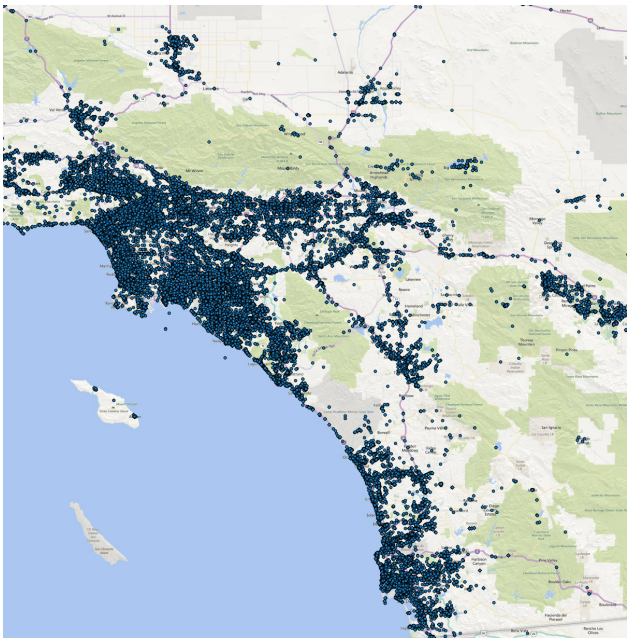


Figure 2.2: Gowalla check-ins over the period of January 2010 to July 2010 from all individuals in southern California.

The primary contribution of this chapter is a systematic approach for individual-level modeling of geolocation data based on kernel density estimates. Kernel density approaches have been relatively unexplored to date in the context of spatial modeling of geolocation data. While in principle they provide a flexible and general framework for spatial density estimation, direct application at the level of individual event data will tend to lead to significant over-fitting (because of the sparsity of data at the individual level). We propose a hierarchical extension of the traditional kernel approach that can avoid the over-fitting problem by systematically smoothing an individual's data towards the population data. We demonstrate how adaptive bandwidth techniques provide better quality density estimates compared to fixed bandwidths. Using two large geolocation data sets from Twitter and Gowalla (see Figures 2.1 and 2.2) we show that the proposed kernel method is significantly more accurate than baselines such as Gaussian mixtures for such data, in terms of the quality of predictions on out-of-sample data.

This chapter is organized as follows. Section 2.2 provides an overview of existing approaches for modeling human location data and elaborates on the challenges of developing spatial models in practice. In section 2.3 we review kernel density estimation and discuss the use of adaptive bandwidth methods and in section 2.4 we describe our proposed mixture-KDE approach for modeling and predicting individuals' locations. In section 2.5 we present empirical experiments using two different geospatial/check-in data sets and using both test log-likelihood and accuracy in detection of simulated identity theft. Section 2.6 discusses scalability and online algorithms for the proposed approach and we conclude with a brief discussion in Section 2.7.

2.2 Background and Related Work

2.2.1 Notation and Problem Definition

In this chapter we consider data available in the form of individual-level geotagged events, $\mathbf{E} = \{E_1, \ldots, E_N\}$ where $E_i = \{e_i^1, \ldots, e_i^{n_i}\}$ and e_i^j is the jth event for the ith individual, $1 \leq i \leq N$. Each event e_i^j consists of a tuple $< i, x, y, t >$, where x and y are longitude and latitude respectively and t is a time-stamp, e.g., geotagged tweets based on GPS location.

One approach in analyzing such data is to focus on the problem of sequentially predicting a user's behavior in terms of their short-term trajectory, e.g., predicting where a user i's next event e_i^{j+1} is likely to occur in terms of location $< x, y >$ given their event history $\{e_i^1, \ldots, e_i^j\}$, where events may be minutes or hours apart (e.g., Song et al. [65] and Scellato et al. [59]). In this chapter we focus on a different problem, that of modeling a user's spatial patterns over a longer time-period in a more aggregate sense. Specifically, we focus on learning probability density models of the form $f_i(x, y)$ that represent the spatial density of user i's events. Given an event has occurred for individual i, the probability that it lies in any area A is $\int\int f_i(x, y)dxdy$ where the integral is over the region defined by A (see also [26]). In this context we focus in this chapter on modeling $f_i(x, y)$ rather than $f_i(x, y, t)$, and only use the time dimension t to order the data, e.g. for online training and prediction. In principle it should be possible to extend the 2-dimensional spatial modeling methods proposed in this chapter to include the temporal dimension, allowing for inclusion of circadian and calendar-dependent aspects of an individual's behavior.

2.2.2 Modeling of Discretized Locations

A widely-used approach in location modeling is to restrict attention to a finite set of known fixed locations, effectively discretizing space and turning the problem into a multivariate data analysis problem where each location represents a single dimension[1]. One can use such representations to generate a sparse matrix consisting of individuals as rows and locations as columns, where each element i, j contains the count of the number of events for individual i that occurred at location j. The locations (columns) can be defined in different ways. For example, one can define the columns by identifying a set of specific locations such as shops, restaurants, and so forth (e.g., see [9, 12, 14]). An alternative approach is to discretize the spatial domain into disjoint cells (e.g., via clustering), and then associate a discrete set of venues with each cell (as in Cranshaw et al. [15] who used Foursquare check-in venues) or to aggregate the counts of geolocated events within each cell (as in Lee et al. [46] and Frias-Martinez et al. [24]). The advantage of these discretized representations is that they allow the application of broad set of multivariate data analysis tools, such as clustering techniques or matrix factorization methods. However, they do not explicitly encode spatial semantics and, as such, do not provide the ability to make predictions in continuous space, which is a primary aim in our work in this chapter.

2.2.3 Continuous Spatial Models

In the context of continuous models, a number of authors have explored such models for individual location data in prior work. For example, Gonzalez et al. [26] and Brockmann et al. [7] explored general distributional patterns of human mobility from location data. Eagle and Pentland [20], Li et al. [49], and Cho et al. [13] demonstrated how different aspects of individuals' daily routines can be effectively extracted from traces of location data.

[1] In fact we pursue this discretization approach later in Chapter 3 in this thesis.

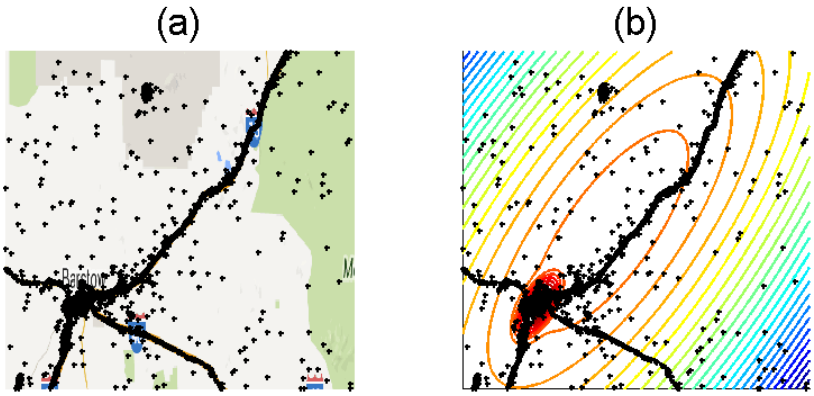


Figure 2.3: On the left (a): Geotagged events in the area between Los Angeles and Las Vegas near the city of Barstow, CA. (b): The contour lines of a Gaussian mixture model with 2 components. Figure best viewed in color.

A simple approach to modeling an individual's spatial density $f_i(x, y)$ is to use a single Gaussian density function, i.e.,

$$\begin{aligned} f_G(x, y|\underline{\mu}_i, \Sigma_i) &= f_G(e_i|\underline{\mu}_i, \Sigma_i) \\ &= \frac{1}{(2\pi)\,|\Sigma_i|^{\frac{1}{2}}} e^{-\frac{1}{2}(e_i-\underline{\mu}_i)'\Sigma_i^{-1}(e_i-\underline{\mu}_i)} \end{aligned} \tag{2.1}$$

where $e_i = (x, y)$ is a 2-dimensional longitude-latitude pair, $\underline{\mu}_i$ is a 2-dimensional mean vector, and Σ_i is a 2×2 covariance matrix. The unimodal and elliptical density contours of a single Gaussian are too simple to accurately represent human location data in practice. With this in mind, a finite mixture of C Gaussian densities can provide additional flexibility, defined as

$$f_{MG}(e_i|\theta_i) = \sum_{c=1}^{C} \pi_{ic} f_G(e|\underline{\mu}_{ic}, \Sigma_{ic}) \tag{2.2}$$

with parameters θ_i consisting of the C mixing weights $\pi_1, \ldots, \pi_C$, $\sum_c \pi_c = 1$, and means $\underline{\mu}_{ic}$ and covariance matrices Σ_{ic}, $1 \leq c \leq C$. For example, as described in Cho et al. [13], a two-component ($C = 2$) spatial model may be a useful model for capturing the bimodal variation due to "home" and "work" components in an individual's spatial data.

While the mixture model can provide additional modeling power beyond that of a single Gaussian, it has a number of practical limitations. Firstly, the number of components C required for an accurate density model may vary considerably across different individuals, and automatically and reliably determining the number of components is a non-trivial problem. Secondly, the number of data points per individual is usually skewed towards small counts. For example, in our Twitter data set, whiche will be described in a later section, 60% of the individuals have associated with them 5 or fewer events over a 2 month period (July and August 2013). This makes it challenging, if not impossible, to fit mixture models, even if the number of components C for each individual is known and fixed. A third limitation of the Gaussian mixture approach is a more pragmatic one. Human mobility is constrained by our environment resulting in sharp transitions in spatial densities, due both to natural topography (mountains, oceans) and man-made artifacts (roads, city centers, etc.). Figure 2.3(a) shows Twitter data for a region near Barstow, California. The spatial density of the data shows significant local variation, including regions of high density for the town of Barstow (bottom left), for the military base (top center), and along the various major roads, with very low density in the surrounding desert. Figure 2.3(b) shows a fitted mixture density model with two components: it is unable to capture many of the high density patterns and "wastes" considerable probability mass over sparsely populated desert regions.

2.3 Kernel Density Estimation for Location Data

To address these limitations, we investigate the use of kernel density estimation (KDE) methods as outlined in detail in the next section. There has been limited prior work investigating the application of KDE methods in the context of human location data. Zhang and Chow [70] illustrated the advantages of KDE techniques (over Gaussian mixture models) for data from location-based social networks, but used 1-dimensional kernel densities

on distances rather than 2-dimensional spatial models, and Hasan et al. [30] illustrated the use of 2-dimensional spatial KDE models for exploratory analysis of check-in data. KDE methods have also been used in application areas such as epidemiology [3], ecology [22], and marketing [19], for modeling spatial densities of *populations* of individual entities, but not for modeling spatial densities of individuals themselves.

2.3.1 Kernel Density Estimation

Kernel density estimation is a non-parametric method for estimating a density function from a random sample of data [61]. Let $E = \{e^1, \ldots, e^n\}$ be a set of historical events where $e^j =< x, y >$ is a two-dimensional location, $1 \leq j \leq n$, and where we have suppressed any dependence on individual i for the moment and dropped dependence on time t. We will refer to E as the training data set. A simple approach for estimating a bivariate density function from such data is to use a single fixed bandwidth h for both spatial dimensions and a Gaussian kernel function $K(\cdot)$. This results in a bivariate KDE of the following form:

$$f_{KD}(e|E,h) = \frac{1}{n}\sum_{j=1}^{n} K_h(e, e^j) \tag{2.3}$$

$$K_h(e, e^j) = \frac{1}{2\pi h}\exp\left(-\frac{1}{2}(e - e^j)^t \Sigma_h^{-1}(e - e^j)\right) \tag{2.4}$$

$$\Sigma_h = \begin{pmatrix} h & 0 \\ 0 & h \end{pmatrix}$$

where e is the location for which we wish to compute the density and $h > 0$ is a fixed scalar bandwidth parameter for all events in E. It is well known that the resulting density estimate

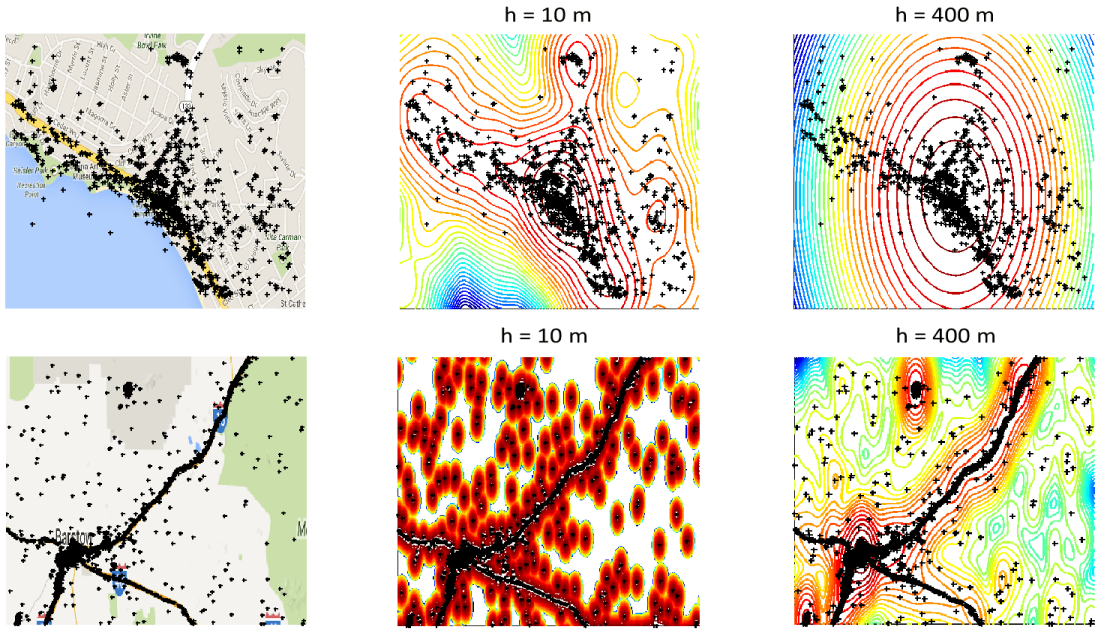


Figure 2.4: Left plots: Events in the city of Laguna Beach in southern California (top row) and in an area between Los Angeles and Las Vegas (bottom row). Points on the map represent observed events. Middle plots: Contour of the log-probability of a KDE model with fixed $h = 10$ meters. Right Plots: Contour of the log-probability of a KDE model with fixed $h = 400$ meters. Best viewed in color.

f_{KD} can be highly sensitive to the value of the bandwidth h, producing densities that are sharply peaked around the training data points e^j when h is too small, and producing an overly smooth estimate that may omit important structure in the data (such as multiple modes) when h is too large [61].

There are a number of techniques that can be used to evaluate the quality of a particular value for the bandwidth h. One straightforward data-driven option is to measure the log-probability (or log-likelihood) of a set of test data points not used in constructing the density estimate, i.e.,

$$L(h) = \frac{1}{n_t} \sum_{r=1}^{n_t} \log f_{KD}(e^r | E, h) \tag{2.5}$$

where the n_t events e^r are data points not included in the training data E (e.g., a validation

set). Larger values of $L(h)$ are preferred since it means that higher probability is being assigned to new unseen data. Hence, a simple approach to bandwidth selection (at least for a single bandwidth parameter h) is to perform a grid-search on h using a validation set. We will use the above out-of-sample log-probability score function $L(h)$ later in the chapter for both bandwidth selection and for comparing different types of density models.

One could also use various "plug-in" estimates for h, such as that of Silverman [61, pages 86-88]. These estimates are optimal (e.g. in a mean integrated squared error sense) if the true underlying density being estimated is Gaussian, and can work well in practice for other related smooth non-Gaussian densities. However, for spatial location data we found that "plug-in" estimates for h were much too large and significantly oversmoothed the KDE results in a very poor fit due to the highly multimodal nature of location data (supporting results are provided later in this section).

At this point in the discussion it is worth noting that, in addition to its advantages in terms of flexibility, kernel density estimation has some well known drawbacks that have tended to limit its use in practice in the past, particularly in machine learning and data mining. The first drawback is that it is particularly susceptible to the curse of dimensionality, essentially requiring an exponential number of data points as a function of data dimensionality d. This is not an issue for location-data modeling since we are in the low-dimensional regime of $d = 2$. A second drawback of kernel densities (as with related "neighbor-based" methods) is the need to store all of the training data in memory at prediction time. This was arguably a relevant point 10 years or more ago when memory was relatively expensive, but in current times it is relatively inexpensive (both computationally and financially) to keep millions (or even hundreds of millions) of points accessible in main memory at prediction time. We will return to this point in more detail later in the chapter—here it is sufficient to note that kernel density estimation is practical for 2-dimensional problems with millions of data points.

2.3.2 The Adaptive Bandwidth Method

A limitation of the approach described above is that the smoothing is homogeneous, i.e., the amount of smoothing is constant through the 2-dimensional region since the bandwidth h is fixed for all events. This does not reflect the realities of human-location data where dense urban areas will tend to have high event density and sparsely-populated rural areas will have low event density. This limitation is clearly visible in Figure 2.4. The two plots in the center use the same small fixed bandwidth of $h = 10$ meters, which works well for the relatively dense area of Laguna Beach (upper plot), but works poorly (overfits) for the rural area near Barstow, CA in the lower plot. If the bandwidth is increased to $h = 400$ meters, as in the two plots to the right, we find that this produces a more acceptable result in the lower plot (the rural area) but is vastly oversmoothing in the upper plot.

One approach to address this issue is to use an adaptive kernel bandwidth, several methods of which have been proposed in the literature, that relaxes the assumption of a constant fixed bandwidth parameter. Breiman et al. [6] suggested adapting the kernel bandwidth h^j to each data point e^j. Using this idea, we let h^j be the Euclidean distance to the kth nearest neighbor to e^j in the training data. Hence we can define an adaptive bandwidth kernel density estimate as:

$$f_{KD}(e|E) = \frac{1}{n}\sum_{j=1}^{n} K_{h^j}(e, e^j) \tag{2.6}$$

where K_{h^j} is defined as in Equation 2.4 replacing h with h^j.

Table 2.1 shows the results from a series of tests on a validation data set, comparing the fixed and adaptive bandwidth approaches using different values for (a) the fixed bandwidth h, (b) the "plug in" estimate, and (c) the number of neighbors k (for the adaptive method). We trained the models using 100,000 randomly selected events from our Twitter data set

	Bandwidth	$\log P$
Fixed	$h = 10^{-2}$	-0.592
	$h = 10^{-3}$	-0.157
	$h = 10^{-4}$	0.139
	$h = 10^{-5}$	-0.326
	$h =$ "plug in"	-8.273
Adaptive	$k = 2$	0.046
	$k = 5$	**1.275**
	$k = 10$	1.196
	$k = 20$	0.354

Table 2.1: Average log-probability scores on held-out events, comparing the fixed, "plug in" estimate, and the adaptive approaches for kernel density estimation for Twitter geolocation data. Higher scores are better. Best performing method indicated in bold font.

(described in more detail later in the chapter) and then computed the log-probability score (Equation 2.5) using a set of $n_t = 100,000$ randomly selected held-out events. From the results, we can see that the adaptive bandwidth models dominate the performance of the fixed bandwidth methods, including the "plug-in" estimate for the bandwidth value.

2.4 Modeling an Individual's Location Data

So far, our predictive model $f_{KD}(e|E, h)$, does not depend on the identity of an individual i. However, our primary goal in this work is to be able to build accurate predictive spatial density models at the individual level.

2.4.1 Mixtures of Kernel Density Models

To address this task, we could apply the adaptive kernel density methods described above at the level of an individual (rather than for aggregations of events across a population of individuals), computing $f_{KD}(e|E_i)$ in Equation 2.6 where we now condition on just the

individual's event data E_i rather than the events for the population E.

A significant challenge with building individual-level models in this manner is the "cold-start" problem, given that we typically have very little data for many of the individuals for whom we wish to make predictions. To address this data sparsity problem we propose a multi-scale kernel model where we use a mixture of (a) an individual's potentially noisy kernel density estimate with (b) more robust coarse-scale models[2]. More specifically we define a mixture-KDE for individual i as

$$P_{MKD}(e|E) = \sum_{c=1}^{C} \pi_c f_{KD}(e|E^c) \tag{2.7}$$

where $\pi_1, \ldots, \pi_C$ are non-negative mixing weights with $\sum_c \pi_c = 1$, and $f_{KD}(e|E^c)$ is the cth component of the mixture. Here component c is a kernel density estimate computed as a function only of a subset of points (or events) E^c. The component density estimates, f_{KD}, can be any density model, including fixed or adaptive bandwidth KDE. We use adaptive bandwidth KDE, with $k = 5$ (following Table 1), for all of the components in the mixture-KDE used in this chapter.

As a specific example consider a model for individual i where $C = 2$, with the first component being the individual-level kernel density with $E^1 = E_i$, and the second component being a population-level kernel density estimate with $E^2 = E$. This mixture will have the effect of smoothing the individual's density towards the population density, with more or less smoothing depending on the relative size of the $\boldsymbol{\pi}$ weights. Note that this mixture is significantly different in nature to the use of a Gaussian mixture for an individual's data (e.g., as in [13]). In that approach, each component typically represents a different spatial location around areas which an individual's activity is centered (such as "home" and "work"), whereas in the mixture-KDE each mixture component is responsible for a broader

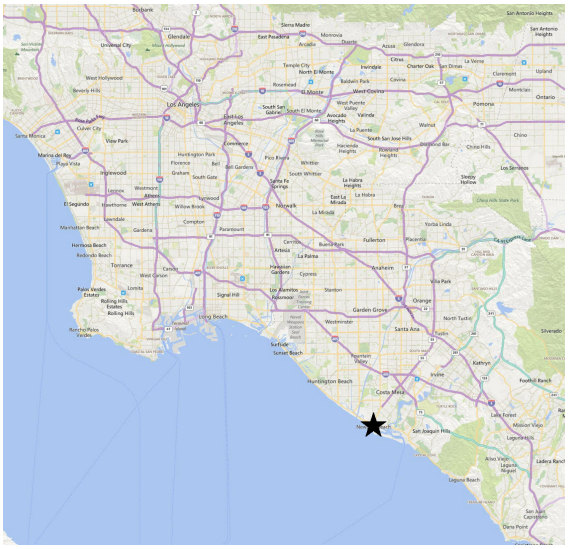
[2]A potential alternative option would be a Bayesian hierarchical model—however Bayesian modeling with kernel densities is not straightforward given that kernels do not have parameters that can be "shrunk" as in the usual Bayesian approach.

spatial scale of activity.

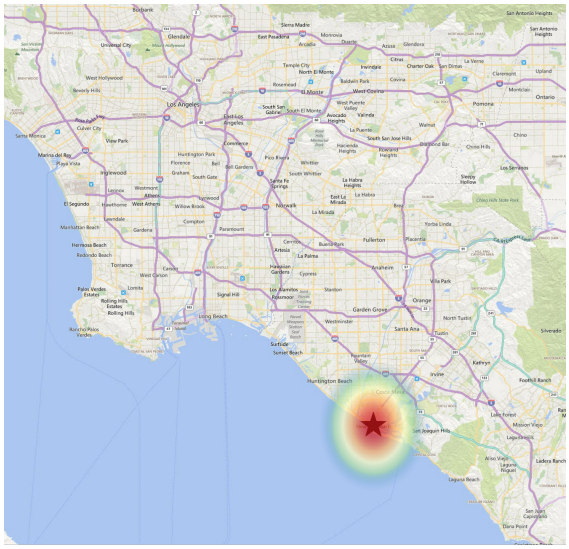
For C components, where $C > 2$, we can have the first component be an individual level density, and the Cth component be the population density, where the intermediate components $c = 2, \ldots, C-1$ can represent different spatial scales (such as neighborhoods, cities, states, and even countries). Given that the data sets we are using in this chapter are from the southern California region, we chose to use a 3-level model ($C = 3$), with the first and last components being the individual and population level models respectively, and the $c = 2$ model representing (approximately) the "home city" for an individual. To find the spatial area that represents the individual "home city" we divided the area of southern California into 81 regions corresponding to a 9×9 array of equal-sized grid boxes. Each individual i was associated with the region that contains the majority of their individual events—thus, the $c = 2$ component in the model represents the scale of a local region or city. A similar approach was used in prior work for finding the "home" location of an individual [13].

This process of selecting additional spatial components, that are "between" the individual and the full population, is somewhat arbitrary—we chose a single intermediate component in the work below, but other choices could be explored for other applications.

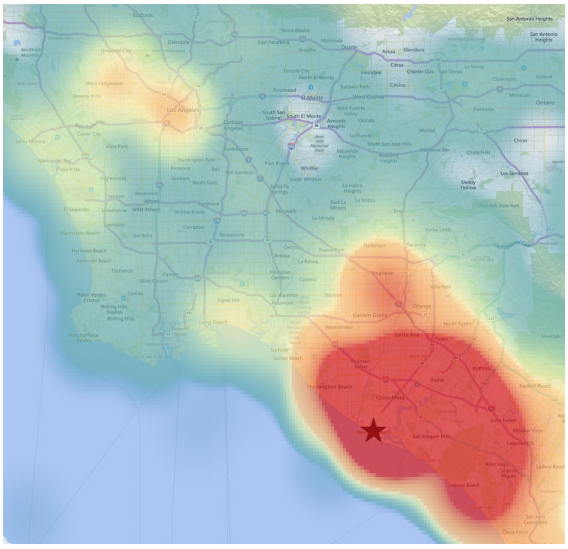
To illustrate the differences between the kernel densities of the spatial scales we plot in Figure 2.5 the heat maps representation of each density for a randomly selected user from the Twitter data set. The $c = 2$ KDE, roughly corresponding to the "home city" of Newport Beach, California for this individual, is shown in subplot (c). First, the individual-level KDE focuses all the density around the few events that were observed. Next, the spatial-scale KDE that represents the "home city" is computed based on the events from individuals with the same "home city." As a result, the estimated density is smoothed toward more areas around Newport Beach, California. Finally, the third spatial-scale component, corresponding to the entire population, smooth the density toward popular areas in southern California, particularly around Los Angeles, California, and shown in subplot (d).



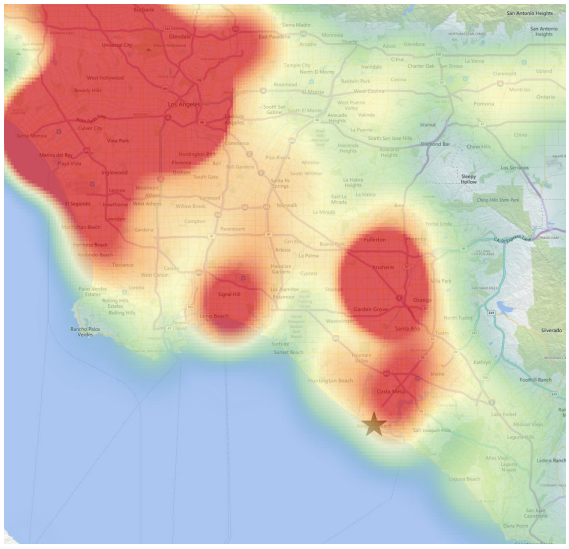
(a) Individual Data



(b) Individual KDE



(c) "Home City" KDE



(d) Population KDE

Figure 2.5: Top left: Location observations for a randomly selected individual from the Twitter data set. All observations are located in Newport Beach, California. Top right: Heat map representation of the learned individual-level kernel density estimation. Bottom left: Heat map representations of the kernel density estimations for the $c = 2$ KDE, roughly corresponding to the "home city" of Newport Beach, California. Bottom left: Heat map representation of the population KDE ($c = 3$) learned using observations from the entire population in southern California. Best viewed in color.

From a generative perspective, one interpretation of Equation 2.7 above for the mixture-KDE is as follows. Assuming that the first component is based only on individual i's data, individual i has a probability π_1 of generating events in the future in a manner similar to

his/her past behavior, and a probability π_c ($c = 2, \ldots, C$) of generating events in accordance with the larger "subpopulations" defined by events E^c. In this chapter, in the absence of additional metadata about the individuals, we defined the larger subpopulations solely on spatial characteristics (component 2 being roughly a city, and component 3 being the whole southern California region). However, one could also define the larger subpopulations based on metadata (if it were available), such as demographics, social ties, and so forth.

Another way to interpret the mixture-KDE model is that it provides a relatively simple weighting mechanism to allow us to upweight data points from individual i in the kernel density estimate and downweight points that don't belong to the individual. Given that kernel densities are defined as weighted sums over all points, the mixture-KDE can be thought of as a form of kernel density estimate where the points are given an additional level of weighting depending on what component they belong to. As a sidenote, in the results presented here we allow data points to belong to multiple sets E^c, e.g., a data point for individual i can be included in the kernel density estimate for all C components. The other option would be to only allow points to belong to a single component. Either option is fine in theory: from the weighted kernel perspective it only changes (in effect) the manner in which we are assigning weights to each point in the overall weighted sum.

2.4.2 Training the Model

Given a set of components $f_{KD}(e|E^c), c = 1 \ldots C$, the next step is to compute the mixing weights $\pi_1, \ldots, \pi_C$ in Equation 2.7. To do so, we randomly sample a validation set, disjoint from the training set, and use it to learn the weights as follows. For each event in the validation set, its density value under each component c is computed (using the training set for the KDE computation). This results in a fixed set of component density values on the validation data points and we can optimize over the convex set of mixing weights π_c to find the highest-

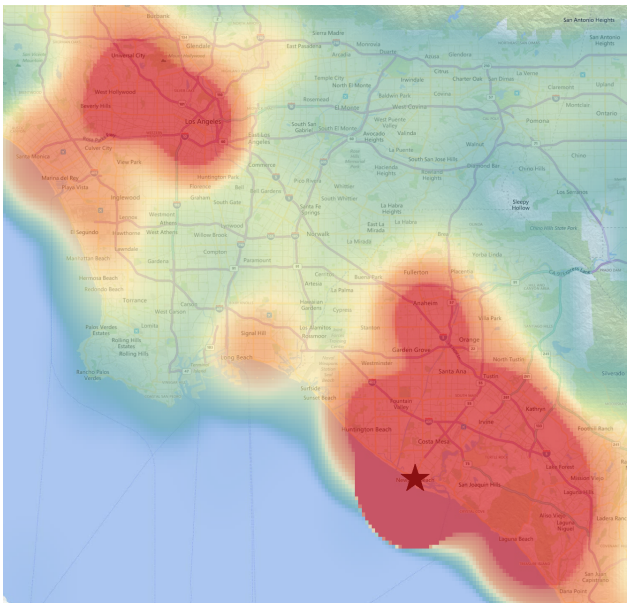


Figure 2.6: Heat map representation of the mixture of kernel densities for selected user in the Twitter data set from southern California. Best viewed in color.

likelihood combination. We used the Expectation-Maximization (EM) algorithm since it is easy to implement and converges quickly (one could use other optimization techniques such as gradient descent)—this is in essence the same as using EM for learning finite mixtures (such as Gaussian mixtures) but where the parameters of the component densities are known and fixed and one is just learning the mixture weights (see also Smyth and Wolpert [64]).

In Figure 2.6 we plot the resulting mixture of kernel densities learned for the randomly selected individual that was used in Figure 2.5. From the figure we can see that the resulting density is a mixture of all three spatial scale components, with high density around the individual's observed events, the areas nearby, and the most popular areas in southern California (i.e., Downtown Los Angeles and Disneyworld, California).

2.5 Experiments and Results

2.5.1 Data Sets

For our experiments, we use two geolocation/check-in data sets: **Twitter** and **Gowalla**. Twitter is a popular micro-blogging service that allows the posting of short texts (up to 140 characters) and pictures. Using the Twitter Streaming API[3] we collected over 4 million public tweets from 230,450 unique individuals in the southern California area over the period of July-August 2013. In the experiments in this chapter we use data only from weekdays, i.e., Monday through Friday. To remove repeated and bursty events we replaced tweets occurring with the same hour and within 50 meters of each other with a single effective tweet. Figure 2.1 shows the spatial distribution of our data set. The Gowalla data is the same data used by Cho et al. [13], containing $145,558$ events from $7,653$ unique individuals on weekdays between January and October, 2010, in the southern California area as shown in Figure 2.2. Training, validation, and test sets were extracted from both data sets for our experiments (details provided later).

2.5.2 Evaluation using Log-Likelihood

The training set for the Twitter data consists of all events recorded for the month of July, 2013. The test set for Twitter is all of the events in August 2013 for a randomly selected set of 2000 individuals, selected from the individuals that have at least 2 events in July. To create the Gowalla training set we used the data from the months of January to June, 2010. The test set for Gowalla is all of the events from the months of July to October, 2010 for a randomly selected set of 1000 individuals, selected from the individuals that have at least 2 events in the months of January to June, 2010. A validation set was generated for

[3] `https://developer.twitter.com/en/docs/tweets/filter-realtime/overview.html`

each of the Twitter and Gowalla data sets by setting aside approximately 37,000 and 25,000 randomly selected events from each training data set.

We built models on the data from our training data set and computed the test set log-probability score of the events in the test set, under each model. We report results both in terms of the mean and median log-likelihood per event, denoted as e-log P, and the mean and median log-likelihood per individual, denoted as i-log P (the latter giving equal weight to individuals, the former to events).

2.5.3 Mixture of Kernel Densities Individual vs. Global Mixing Weights

We first compare two versions of the mixture of kernel densities model: one which uses globally-shared mixing weights $\boldsymbol{\pi}$ and another which uses individually learned mixing weights $\boldsymbol{\pi}_i$. For both models, we learn the individual kernel density components on the training set and optimize the mixing weights on the validation data set.

For the mixture model with globally-shared mixing weights, we learn a single set of mixing weights, $\boldsymbol{\pi}$, which is shared across all users. In the M-step of the learning algorithm, we use a flat Dirichlet prior on the weights with a strength parameter that is optimized on the validation data set. For the individually-learned mixing weights model, we learn a unique set of mixing-weights for each individual in each M-step. We use an empirical Bayes prior proportional to the average mixing weights across all users with a strength parameter that is optimized using the validation data set. The use of such prior allows users with small amount of historical data to effectively "borrow" information from users with more historical information.

Table 2.2 shows that in terms of mean log-likelihood, the mixture of kernel densities model

	i-log P		e-log P	
Data set	*indiv*-$\boldsymbol{\pi}_i$	*global*-$\boldsymbol{\pi}$	*indiv*-$\boldsymbol{\pi}_i$	*global*-$\boldsymbol{\pi}$
Twitter	4.065	**4.279**	**5.352**	5.293
Gowalla	5.847	**6.599**	6.323	**6.568**
Avg.	4.956	**5.439**	5.838	**5.931**

Table 2.2: Mean i-log P and e-log P on the test data for Mixture-KDE with individually and globally learned mixing weights across the different data sets and on average. Higher results are better. Best performing methods indicated in bold font.

with globally-learned mixture weights outperforms the model with individually learned mixing weights on average by a small margin. The results for median log-likelihood were consistent with the mean log-likelihood results and were omitted from the table for clarity.

Further analysis of the results showed that users for whom the global mixing weights approach performed better had relatively small validation data sets at optimization time. As a result, the quality of their individually-learned mixing weights was low which leads them to underperform the globally-learned mixing weights model. In real life applications however, when the amount of data available for each user is significantly larger, it is possible that the individually-learned mixing-weights mixture of kernel densities model or a hybrid approach will outperform the globally-learned mixing-weights mixture approach.

In addition, the model complexity (i.e., number of learned parameters) of the global mixing-weights mixture model is much lower than that of the individual-learned mixing-weights version as the number of mixing weights grows linearly with the number of users. For these reasons we focus on global mixing-weights model throughout the remainder of this chapter.

2.5.4 Comparing Mixture-KDE and Existing Methods

We also compare our mixture of kernel densities model to existing individual-level continuous space models. By an "individual-level model" we mean a model that is fit to each individual i using only their data E_i, and then used for predicting future events for that individual i.

Gaussian: A Gaussian density model. We used maximum likelihood estimates for the mean and maximum a posteriori (MAP) estimates for the covariance matrices (e.g., see [55], chapter 4.6.2):

$$\mu^i = \hat{\mu}^i_{MLE}, \qquad \Sigma^i = \lambda\Sigma^i_0 + (1-\lambda)\hat{\Sigma}^i_{MLE}, \qquad \lambda = \frac{n_0}{n_0 + n_i}$$

where the prior parameters were set to $n_0 = 3$ and $\sigma = 5$ kilometers (for the diagonal on the prior covariance) via a grid search over the log-likelihood on a validation set across all individuals.

Gaussian Mixture Model (GMM): Two different Gaussian mixture models with $C = 2$ and $C = 4$ components. The models were fit using the EM algorithm. Again, we used maximum likelihood to estimate the μ_c and MAP estimates for the Σ_c. Parameters for the priors were also determined on a validation set.

Fixed KDE: A fixed bandwidth kernel density estimate using Equation 2.3. The value of the fixed bandwidth, h, was determined via grid search using the validation set. For both metrics i-log P and e-log P the best value was approximately corresponding to $h = 5.3$ kilometers, for all users.

Adaptive KDE: An adaptive kernel density estimate using Equation 2.6. The nearest-neighbor value of k was determined via grid search using the validation set. For both metrics i-log P and e-log P the best value was $k = 5$, for all users.

In addition, for our log-likelihood experiments we evaluated a single "global" population model (Population KDE) using an adaptive kernel density estimate ($k = 5$) based on **all** data points in the training set (i.e., not an individual-level model).

For our mixture-KDE model we used 3 components corresponding to the individual i, the "home city" and the full population, respectively. Each component is an adaptive bandwidth

	i-log P		e-log P	
Model	Mean	Median	Mean	Median
Gaussian	-0.586	0.151	-0.242	1.357
GMM (C = 2)	-0.469	0.221	0.001	1.676
GMM (C = 4)	-0.474	0.279	-0.015	1.712
Fixed KDE	-1.025	-0.714	-0.869	-0.688
Adaptive KDE	-7.154	0.446	$(*)$	-4.908
Population KDE	2.014	0.563	0.784	0.237
Mixture-KDE	**4.279**	**4.312**	**5.293**	**6.302**

Table 2.3: Average log-probabilities on the test data for individuals and events from the Twitter data set. Higher scores are better. Best performing methods indicated in bold font.

	i-log P		e-log P	
Model	Mean	Median	Mean	Median
Gaussian	-1.513	0.008	-1.133	1.027
GMM (C = 2)	-1.532	0.308	-0.956	1.412
GMM (C = 4)	-1.522	-0.479	-0.958	1.471
Fixed KDE	-1.136	-0.749	-1.092	-0.742
Adaptive KDE	$(*)$	-0.247	-3.288	1.355
Population KDE	3.388	1.237	4.021	0.923
Mixture-KDE	**6.599**	**6.296**	**6.568**	**2.619**

Table 2.4: Average log-probabilities on the test data for individuals and events from the Gowalla data set. Higher scores are better. Best performing methods indicated in bold font.

KDE with $k = 5$ neighbors. Using EM to determine the mixture weights resulted in the following values for the π's: 0.85 for π_1 (the individual level), 0.12 for π_2 (the region level), and 0.03 for π_3 (the population level) for the Twitter data set, and $\pi_1 = 0.50$, $\pi_2 = 0.32$ and $\pi_3 = 0.18$ for the Gowalla data set.

The mean and median scores, for both individual and event-level scores, are shown in Tables 2.3 and 2.4 for the Twitter and Gowalla data sets respectively. A $(*)$ indicates that the test log-likelihood was not computed due to numerical underflow. The mixture-KDE model clearly outperforms all other methods, on both data sets, for all metrics, assigning significantly higher log-probability to the test events than any of the other modeling approaches. Figures 2.7 and 2.8 show the comparison between the mixture-KDE approach and the GMM (on the left) and fixed KDE (on the right) when looking at each individual separately, again

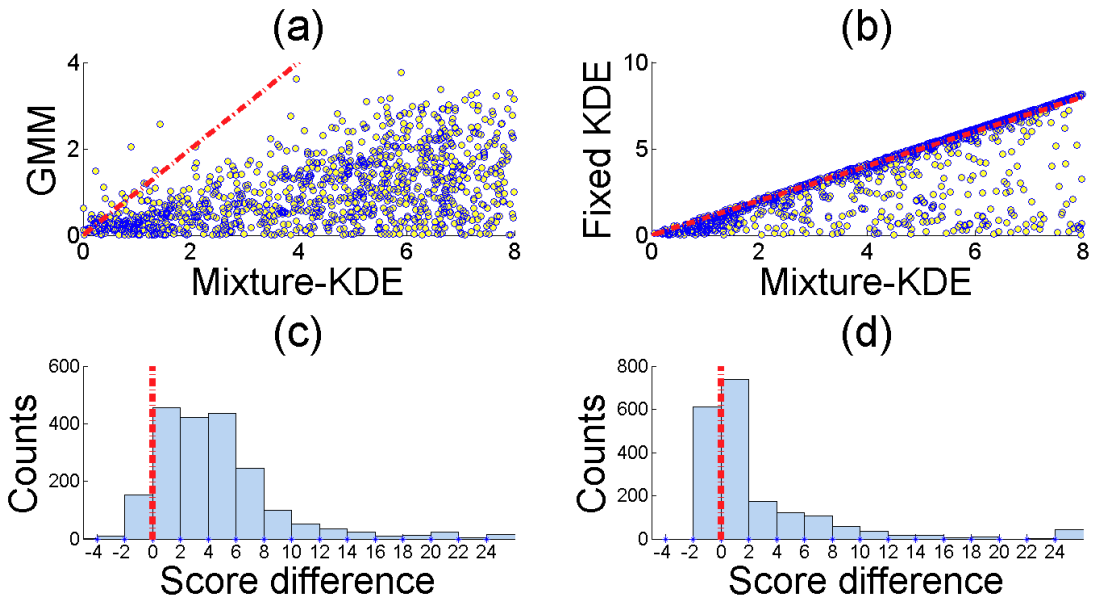


Figure 2.7: Upper plots (a) and (b): scatter plots for a sample of test set log-probability scores for Twitter individuals with (a) individual Gaussian mixtures ($C = 4$) versus the mixture-KDE, and (b) individual fixed-KDE versus the mixture-KDE. Lower plots (c) and (d): histograms of the score differences per event for the mixture-KDE minus the score of the corresponding model on the y-axis in the upper plot.

clearly showing the improvement of the mixture-KDE approach over the other methods.

2.5.5 Evaluation using Simulated Identity Theft

We now compare the different models by using a simulated real-world application based on identity theft. Over 8 million people are victims of identity theft every year in the United States alone, with an annual cost exceeding 4 billion dollars and over 32 million hours spent by individuals to resolve the problem [21]. To simulate identify theft we replaced the geolocation events for an individual over a specific time-window with the geolocation events of a *different individual* and then tested the ability of our models to detect the resulting anomalous spatial patterns of events.

We used the Twitter data set for our experiment since the Gowalla data set did not show any significant differences for this problem between the different models (single Gaussian,

mixtures of Gaussians, various forms of KDEs). We believe this may be due to the fact that our data set for Gowalla has fewer individuals than Twitter, and that these individuals use many of the same checkins, limiting the effectiveness of Gowalla data for detecting "identity switching".

Focusing on the Twitter data set, we defined the training set to be all events in the month of July, 2013. The test set consists of two types of event streams in August, 2013: events for normal "control" individuals and events with simulated identity theft. The control individuals are a randomly selected set of 950 individuals that have at least 2 events in July and at least 10 events in August. The individuals with simulated identity theft correspond to a set of 50 randomly selected individuals with at least 2 events in July. For each of these 50, we then replaced their real test data events in August, with the set of events from a different randomly selected individual, among individuals who have at least 10 events in August. In this manner, our test data has 50 event sets where the spatial distribution for each sets will in general look different (in a realistic manner, as if a different individual were using the Twitter account), compared to the event sets for the "true" individual in the training data from July.

To evaluate the different models we computed a surprise index S_i for each individual i, defined as the negative log-probability of individual i's events in the test data set relative to a model constructed on historical data:

$$S_i = -\frac{1}{n_i} \sum_{r=1}^{n_i} \log \hat{f}_i(e_i^r) \tag{2.8}$$

where e_i^r is the rth event in the test data set for individual i, and $\hat{f}_i$ is the density estimate for individual i constructed using the training data. The larger the surprise score S_i, the more anomalous the events e_i^r are relative to the model $\hat{f}_i$. In these experiments we used all of the models that we used in the log-likelihood experiments as described earlier in the chapter,

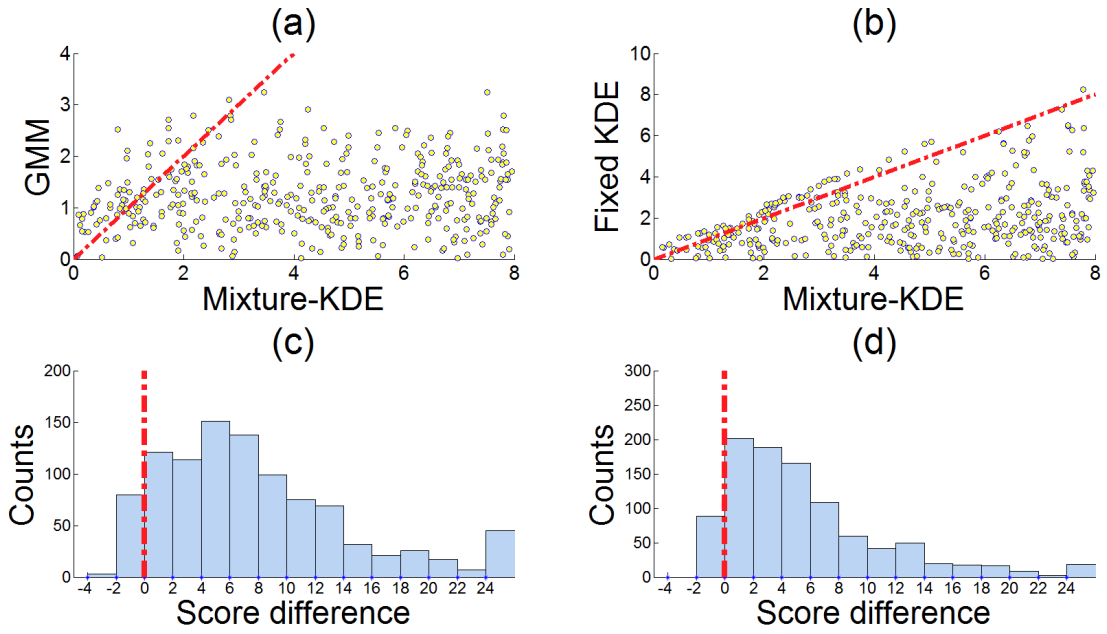


Figure 2.8: Upper plots (a) and (b): scatter plots for a sample of test set log-probability scores for Gowalla individuals with (a) individual Gaussian mixtures ($C = 4$) versus the mixture-KDE, and (b) individual fixed-KDE versus the mixture-KDE. Lower plots (c) and (d): histograms of the score differences per event for the mixture-KDE minus the score of the corresponding model on the y-axis in the upper plot.

except for the population model which is unable to generate rankings at the individual level.

We used a grid search on the validation set (defined in a similar manner to the training-test setup above) to determine various parameters of the models, where the parameter values were selected to optimize precision on the identity theft task. In general, optimizing for precision results in different parameter values for the various models compared to optimizing for likelihood. The priors of the Gaussian mixture model resulted in $n_0 = 0.1$ and $\sigma = 5$ kilometers. The optimal parameters for the individual KDE were estimated to be a fixed bandwidth of $h = 3$ kilometers and $k = 5$ for the adaptive method. The optimal mixture weights for the mixture-KDE model were $\pi_1 = 0.90$ (the individual level), $\pi_2 = 0.08$ (the region level), and $\pi_3 = 0.02$ for the population model.

For each model, we then ranked all of the individuals in the test data by their surprise index S_i and computed precision relative to the known ground truth in terms of which

	$n_{te} = 1$	$n_{te} = 5$	$n_{te} = 10$
Gaussian	**0.612**	0.470	0.325
GMM (C=2)	0.320	0.240	0.160
GMM (C=4)	0.240	0.170	0.130
Fixed KDE	0.500	0.500	0.460
Adaptive KDE	0.432	0.309	0.274
Mixture-KDE	0.531	**0.747**	**0.816**

Table 2.5: Average precision (over 50 runs) for the top 20 ranked individuals in the test data, as a function of the number of observed test events n_{te} per individual. Higher scores are better. Best performing methods indicated in bold font.

	$n_{te} = 1$	$n_{te} = 5$	$n_{te} = 10$
Gaussian	0.379	0.341	0.240
GMM (C=2)	0.260	0.220	0.150
GMM (C=4)	0.240	0.190	0.150
Fixed KDE	0.330	0.370	0.320
Adaptive KDE	0.296	0.188	0.142
Mixture-KDE	**0.459**	**0.589**	**0.644**

Table 2.6: Average precision (over 50 runs) for the top 20 ranked individuals in the test data, as a function of the number of observed test events per individual, for "cold-start" individuals (as defined in the text). Higher scores are better. Best performing methods indicated in bold font.

individuals correspond to simulated identity theft and which to the controls. Table 2.5 shows the precision at 20, the fraction of simulated identity theft cases correctly detected in the top 20 ranked individuals. These precision numbers are the result of averaging over 50 different randomly generated test sets, using the methodology described earlier. The rows correspond to different models and the columns correspond to 3 different scenarios: computing the surprise-index per individual based on their first n_{te} events (in time) for each individual in the test set, with $n_{te} = 1, n_{te} = 5, n_{te} = 10$. Table 2.6 shows the same information for a "cold-start" scenario, where now test sets are generated for simulated identity theft and normal individuals (using the same general procedure as before) who are constrained to have between 2 and 5 events in their training data (compared to any number greater than or equal to 2 for the general case).

The results in the two tables show that the mixture-KDE model dominates all of the other

methods, with a Wilcoxon signed rank p-value of $p < 0.02$, except for the Gaussian model in the non-cold-case situation for $n_{te} = 1$. This may be due to the fact that for a simulated identity theft case, a sampled new event has a high probability of coming from a popular area.

The mixture-KDE model improves as it sees more events in the test set (as n_{te} increases from left to right in the table). However, the other methods all decrease in precision as n_{te} increases. On closer inspection we found that this was being caused by their sensitivity to false alarms, i.e., with more data points per individual there is a higher chance that a control individual (a false alarm) will have an event in the test data that is not close spatially to the individual's events in the training data, resulting in a high-surprise score and a high rank for that individual. The mixture-KDE is more robust to this type of variation, consistent with results earlier in the chapter in terms of log-likelihood.

2.6 Scalability and Online Computation

2.6.1 Scalability

Our experience suggests that kernel density models are quite scalable for two-dimensional data, and can likely be scaled to millions of individuals and hundreds of millions of events relatively easily. To compute the density of a new point e, given a training data set E, we need to compute the contribution of each training point to e's density, as shown in Equation 2.3. In general, storing the N training data points requires $\mathcal{O}(N)$ space and computing the density for a new event will result in time complexity $\mathcal{O}(dN)$ where d is the dimension of the data (here $d = 2$). In our implementation of the KDE models for the results in this chapter we used k-d trees, as described in [28], to further speedup our KDE implementation. This effectively computes only contributions from nearby points to e, based on a k-d tree

	i-log P		e-log P	
Model	Mean	Median	Mean	Median
Mixture-KDE	4.279	4.321	5.293	6.302
Online	**4.892**	**4.913**	**5.788**	**6.531**

	i-log P		e-log P	
Model	Mean	Median	Mean	Median
Mixture-KDE	6.599	6.296	6.568	2.619
Online	**6.987**	**6.742**	**7.432**	**3.022**

Table 2.7: Predictive log-probability scores, averaged over individuals and events for a) Twitter (top table) and b) Gowalla (bottom table). "Online" is the online version of the Mixture-KDE model as described in the text. Higher scores are better. Best performing methods indicated in bold font.

partition of the 2-dimensional space, resulting in a significant reduction in computation time. We implemented our algorithm for kernel density estimation in Java[4] and ran the code on an 8-core 2.4GHz Intel Xeon CPU with 8 hyper threads and 48 GB of RAM memory. Using one million events as training data points, the average time for computing the density of a new event is 8 milliseconds, making the model tractable for large data sets. Additional speed-ups could be achieved for example by distributed computation since the density contribution from different subsets of points are independent from one another. Hence, one can "split" the training data set into disjoint groups and aggregate the density contributions in parallel.

2.6.2 Online Prediction

The results presented up to this point in the chapter have used a batch approach to training and testing. A useful feature of the kernel density approach, including the mixture-KDE, is that it is quite easy to implement an *online* version that can sequentially be updated as new events arrive in streaming fashion. When a new event e arrives we simply add it to the training set. For the adaptive bandwidth approach, every time the training set changes, we need to find the k neighbors of the new point, as well as potentially needing to find new

[4]The code is available for download at `http://www.datalab.uci.edu/resources/`

neighbors for all N existing points. In practice, however, only a very small fraction of existing points will need to have their neighbors updated. Other parameters of the mixture-KDE model, such as mixture weights for the components, are likely to change relatively slowly over time, and can be periodically updated on validation subsets.

Table 2.7 shows predictive log-probability scores for the same training and test data used earlier for likelihood experiments, but now, each sequential test event is included in the training data in an online fashion before computing the log-probability of the next event. The online model shows a significant systematic improvement in predictive scores compared to the batch model, suggesting that online adaptation is beneficial with human location data. This certainly makes intuitive sense, as we expect individual behavior to be non-stationary and changing over time. In a practical application of an online model one would likely incorporate some downweighting (e.g., via exponential weighting) or windowing of events that are further back in time, allowing the model to adapt to changes in individual behavior.

2.7 Conclusions

The primary contributions of this chapter include the following:

- We investigated the use of kernel density estimation technique for modeling individual-level human location data.
- We explored and evaluated different approaches for selecting the model parameters (i.e., kernel bandwidth).
- We developed a mixture of kernel densities model that provides a principled way to learn and model the balance between exploitation and novelty seeking in the context of continuous space.

- We applied this framework to the problem of predicting individuals' future locations using two large scale data sets from Twitter and Gowalla, showing that our approach outperforms existing baselines.

- We evaluated our model using the task of fraud detection, based on spatial anomaly detection, showing again that our model outperforms existing approaches.

Chapter 3

Personalized Location Models with Adaptive Mixtures

In the previous chapter we presented a mixture approach for modeling repetitive and novelty-seeking behavioral patterns as observed in *continuous-space* location data. In this chapter, we extend this work and develop a general adaptive mixture framework, using techniques from the previous chapter, for individual-level *discrete-space* location data. The model we develop in this chapter adpatively combines individual-level data with different smoothing components derived from contextual information such as spatial distance and popularity. In a series of experiments with Twitter geolocation data and Gowalla check-in data we demonstrate that our proposed approach can be significantly more accurate than more traditional smoothing and latent-space models based on matrix factorization techniques. The improvement in performance over the existing approaches is pronounced and may be explained by the tendency of dimension reduction methods to over-smooth and not retain enough detail at the individual level.

3.1 Introduction

In recent years we have gained the ability to record human behavior in increasingly fine-grained detail. Such data can provide significant insights and predictions related to human behavior at the population level. At the individual level, this type of data holds the promise of *personalization:* delivering information, products, and treatments in a manner that is optimized for each specific individual.

A key challenge in personalization is being able to generalize about an individual beyond their historical data. For location data we can anticipate that individuals are often likely to visit new locations for which we have zero counts in their historical data. The challenge is how to model their propensity to visit new locations (including which new locations) while respecting their tendency to revisit locations from their past.

This problem is particularly challenging in the presence of sparse data. Assume for example that we are representing our data as an $N \times M$ matrix where N is the number of individuals in the population and M is the number of locations. Each entry i, j is the number of times we have seen an event (such as a check-in) from individual i at location j. These data matrices are typically highly sparse with as few as 0.01% (or less) of the entries being non-zero depending on the context. Furthermore, the total number of data points per individual (per row) is often highly skewed, with relatively little data for many individuals and a long tail of a relatively small number of individuals with large amounts of data. As an example, consider the data shown in Figure 3.1 which shows the probability that a randomly selected individual visits L unique locations in both Twitter and Gowalla data sets (both data sets will be discussed later in section 3.3). We see the usual long tail of individuals with large numbers of observations, but for most individuals there are very few observations.

In this chapter we consider the problem of generalizing from sparse location data to learn models that can make accurate future predictions at the individual level. Our goal is to

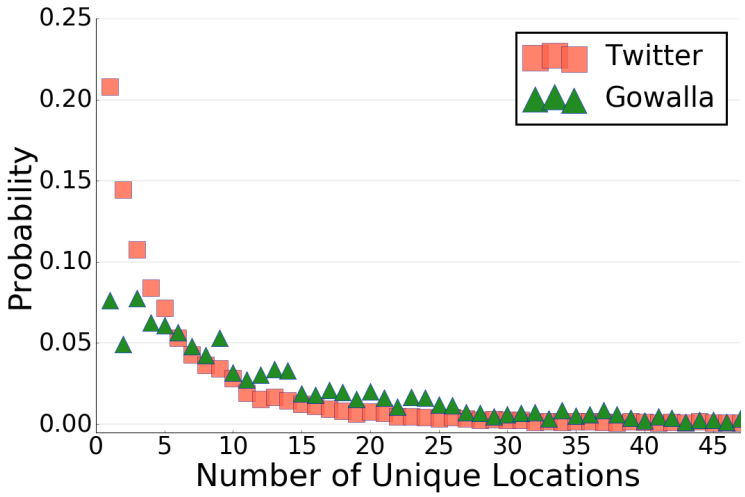


Figure 3.1: The probability that a randomly selected individual visits L unique locations, where L is the x-axis, for both Twitter and the Gowalla data. Values of $L > 50$ where omitted from the plot for clarity. Best viewed in color.

construct for each individual i a model in the form of a multinomial (categorical) distribution $\boldsymbol{\theta}_i$ over M locations, where θ_{ij} is the probability that the location is j given that individual i generated an event, with $\sum_{j=1}^{M} \theta_{ij} = 1$.

A central idea in building personalized models is to "borrow strength" across individuals. This way we can leverage patterns at the population level to build more accurate models at the individual level. A simple example of this is the smoothing of a frequency-based estimate of $\boldsymbol{\theta}_i$ towards a prior distribution as specified by a Dirichlet prior, using a hierarchical or empirical Bayesian framework (see for example [25]).

This type of approach is often very effective. It will typically provide better predictive power than using only the individual's own data. However, it does make a strong implicit assumption of homogeneity, i.e., that our prior expectation is that each individual can be modeled as the deviation from a common global pattern of behavior. Thus, a weakness of these types of smoothing methods is that they do not accommodate enough heterogeneity at the individual level. For location data in particular, data at the individual-level often exhibits high variation, such as locations that are relatively unique to one individual or to a small set of individuals.

There are a variety of other approaches that rely on "borrowing strength" across individuals for addressing the sparse individual data problem. The approach of collaborative filtering for example uses similarity between individuals to share information (e.g., [67]). Techniques such as matrix factorization [14, 12] and latent space models [9, 32] map the original M-dimensional count vector for each individual to a latent and more dense d-dimensional space, where $d << M$. The latent space is represented by d basis functions estimated from the observed count matrix, and individuals are represented as linear combinations of these globally-derived basis functions. One limitation of these approaches is their reliance on co-occurrence data which can limit their effectiveness in the presence of sparse location data [12, 47]—as we will see later in the chapter, they can be improved upon by incorporating individual-specific information.

We address these problems by proposing a flexible framework that can trade-off global and local information to learn an effective predictive location model for each individual. The primary contributions of our work are (1) an adaptive mixture model framework that infers appropriate levels of smoothing, and (2) experimental results in both batch and online modes demonstrating systematic improvements in prediction accuracy (compared to traditional global methods) on a range of large-scale location data sets.

In the remainder of the chapter we discuss related work in Section 3.2 and describe the Twitter and Gowalla data sets that we use in our experiments in Section 3.3. In Section 3.4 we outline our mixture modeling approach and present the learning algorithm in Section 3.5. Section 3.6 describes our experimental setup used to evaluate our proposed model and existing approaches. In Section 3.7 we describe a series of experiments in batch mode and in Section 3.8 we describe a framework and experimental results for online day-by-day learning. Section 3.9 discusses the scalability of the approach and we conclude with a brief summary in Section 3.10.

3.2 Background

As discussed in the introduction, our focus in this chapter is on building models from location data in the form of event tuples $\langle i, x, y, t \rangle$ where i is the individual who generated the event, (x, y) is the location (e.g. longitude and latitude from GPS) and t is a discrete timestamp (e.g., day, week, etc.). From such data we wish to create individualized location models. We will assume that the locations (x, y) can be mapped to a discrete set of M locations. This is a standard approach in location modeling and will be discussed in more detail in Section 3.3. In contrast to Chapter 2, here we model locations discretely. Both continuous and discrete representations have their strengths and weaknesses. Continuous representations allow for very fine-grained resolution - but the location-based discrete locations we use here have the advantage of "borrowing strength" locally (e.g., all tweets within an airport are categorized as such) in a way that a continuous isotropic representation cannot.

There are three broad strands of relevant work on learning individual-level location models that we can distinguish in terms of the time-scale of the predictions being made for an individual's location. The first line of work deals with *instantaneous* location prediction where there is metadata about an event (such as the text of a social media post) and the goal is to predict the specific location of the individual when the event occurred, e.g., [40]. The second line of work focuses on the time-scale of *minutes to hours*, predicting where an individual or a vehicle is likely to be within a relatively short time-frame (e.g., [73, 5]). This type of work often takes advantage of relatively high sampling rates for the location data, such as GPS measurements of a vehicle every few seconds, and typically casts the problem as some form of predictive trajectory modeling.

Our focus is on the third broad class of predictive location modeling, in which the time-scale is that of learning about an individual's typical behavior over days and weeks. Here we are interested in learning predictive distributions that characterize the locations where an

individual is likely to visit in the future, often based on relatively sparse historical data such as occasional geolocated social media posts or check-ins. The data here is often in the form of a matrix of N individuals by M locations where cell i, j contains a count of the number of times individual i visited location j. From this data we wish to infer predictive distributions $\boldsymbol{\theta}_i$ over the M locations for each individual.

As mentioned earlier, matrix factorization and collaborative filtering techniques are widely used in this context. Both however, can oversmooth individual-level details and potentially lead to suboptimal predictive performance (e.g., see [12]).

A common approach for preserving individual-level information is to use geographical preferences related to each individual. As shown in [12, 13] individuals tend to visit locations within a certain proximity and the further a location is from where an individual was observed in the past the lower the probability for the individual to visit those locations in the future. This inherent characteristic of human mobility data can be used to constrain the traditional methods in order to allow for additional personalization. One way of doing so, is to learn a geographical preference for each individual independently, using each individual's historical data and geographical information such as pairwise distances between locations. When combined with the smoothed predictions from collaborative filtering or matrix factorization methods, the geographical component lowers the score of locations that are far away from the individual and increase for those that are close by [12, 70, 71]. However, these methods use the individual information to learn geographical context and apply it to the smoothed check-in data; they do not provide a direct method to control the degree to which the geographical context component affects the predictions.

An alternative and more systematic approach is to use the geographical preferences for each individual during the learning process, e.g., during matrix factorization [53, 50, 47, 54]. These methods project the data into a lower dimensional space while taking into account the individual geographical preference. While this approach improves location prediction

over direct matrix factorization, it does not take advantage of other available sources of information such as using social ties to infer the similarities between individuals. In addition, as with the previously discussed approaches, the individual-level data is only used to learn geographical preferences and is not used directly.

Our contribution in this context is to develop new approaches that are better at retaining the details of each individual's behavior. In particular, we develop a mixture modeling framework that learns a model for each individual based on their own historical data, but also learns to blend this model with information from the individual geographical preferences as well as other individuals in a manner that tries to optimize the predictive ability of the model.

3.3 Geolocation Data Sets

We consider two different data sets in our experiments, from Twitter and Gowalla. The Twitter data set contain tweets from two different regions: Orange County (California), and New York and referred to as `TwOC` and `TwNY` respectively. The tweets were gathered using the Twitter API[1] from May 2015 to February 2016, selecting tweets that have geolocation (GPS) coordinates for each tweet. The Gowalla data set (obtained from [13]) contain check-ins from San Francisco and New York, and denoted as `GoSF` and `GoNY` respectively.

For the experimental results reported later in the chapter we retained individuals with information from at least 5 unique days and locations with 3 events or more. This was done primarily to reduce the computation time in the experiments (by truncating the long tails of sparse individuals and locations) particularly for some of the baseline methods using matrix factorization. For the purpose of this work, we defined the discrete time window t as a single day starting at 4AM and ending at 3:59AM (local time). This step resulted in 4 different

[1] `https://developer.twitter.com/en/docs/tweets/filter-realtime/overview.html`

	Events	Individuals	Locations	Days
TwOC	449306	13559	11347	240
TwNY	690801	30320	11260	240
GoSF	138934	2593	7706	507
GoNY	77761	1831	7692	441

Table 3.1: Number of unique events, individuals, locations and days for the different data sets.

data sets presented in Table 3.1.

3.3.1 Defining a Vocabulary of Locations

In location-based modeling the GPS coordinates of events are typically categorized into a "vocabulary" of M locations in one of two different ways. In the first method locations are mapped to a set of known "high volume" locations such as businesses and public venues (e.g., [13, 72]). A disadvantage of this approach for personalization is that it can ignore a lot of potentially relevant data for individuals such as events associated with homes or work places. An alternative approach is to partition the continuous two-dimensional space into M disjoint cells using a regular grid, boundaries defined by cellular towers or applying partition techniques on the data (e.g., [35, 43]), and to then assign each GPS measurement to the cell it is contained in. This has the advantage of including all of the available data. However a disadvantage is that the resulting cells can be somewhat arbitrary in terms of what space they cover and they do not necessarily correspond to meaningful locations. Another issue (for cells based on regular grids in particular) is that there is a trade-off between the spatial resolution of the grid (how large the cells are) and how much data is available per cell — cells may need to be quite large in order to have enough data points per cell for modeling purposes.

Given the limitations of these discretization methods in this work, we used publicly-available

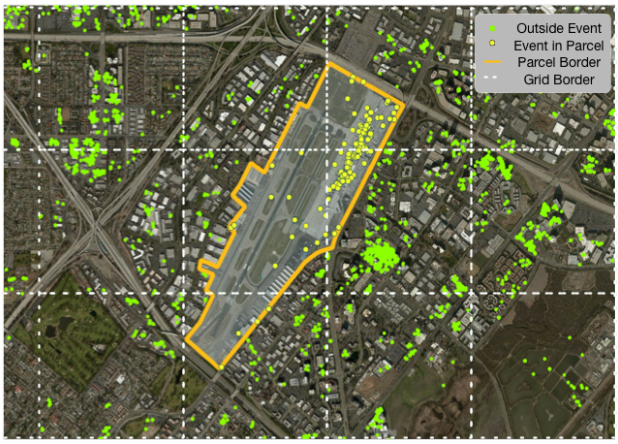


Figure 3.2: Area around the Orange County airport (SNA) in Santa Ana, California showing both the parcel boundary for the airport and grid-based boundaries, as well as Twitter events within the airport parcel. Best viewed in color.

geoparcel information to partition space into cells (for the Twitter data sets—the Gowalla data is already discretized). Geoparcels are a set of disjoint polygons on a map where each polygon represents a specific property or a lot (such as a house, a stadium, a store/shop, an airport etc.). The polygons vary in size and shape depending on the function of the property (e.g., typically smaller polygons for houses, much larger polygons for sport stadiums and theme parks) providing a natural trade-off between spatial granularity and number of data points per cell. In addition the geoparcels have a direct semantic interpretation allowing for ease of interpretation of the model's spatial vocabulary.

In our work we analyze two areas using geoparcels: Orange County, CA and New York, NY. The parcels for Orange County are publicly available from the Southern California Association of Government website[2]. The data for New York was obtained through the OpenStreetMap API[3].

As an example, Figure 3.2 shows the parcel boundary for the Orange County airport, capturing precisely the extent of the airport footprint. Overlaid on this image are boundaries of cells in a regular grid where each cell roughly covers the same area as the airport parcel. We

[2] `https://www.scag.ca.gov/`
[3] `http://wiki.openstreetmap.org/wiki/API`

can see that the grids are less precise, consisting of a mix of events from different locations (e.g., from the airport and from nearby business and residential locations)—and the data from a large location such as the airport is fragmented into multiple different cells in the grid representation.

In our experiments with Twitter data in Section 3.7 we use these parcel-based cells to define our set of M locations for each region.

3.4 Model Description

3.4.1 Adaptive Smoothing with Mixtures

The key ideas behind our modeling approach are to build models for individuals that (a) smooth individual-level information towards population patterns, geographic constraints, and social contexts, and (b) learn to combine these sources in a manner that optimizes predictive performance.

Let $\boldsymbol{\theta}_i = [\theta_{i1}, \ldots, \theta_{iM}], \sum_j \theta_{ij} = 1$, be an unknown probability distribution over M locations for individual i. To estimate this distribution from data we could use the maximum likelihood estimate $\theta_{ij}^{MLE} = \frac{r_{ij}}{r_i}$, where r_{ij} is the historical count for individual i and location j and $r_i = \sum_j r_{ij}$. This estimate is problematic with sparse data — a count $r_{ij} = 0$ should not necessarily imply that the probability of an individual visiting location j in the future is 0.

An alternative approach is the Bayesian framework where the $\boldsymbol{\theta}_i$ parameters are treated as random variables with a prior. In particular, given a Dirichlet prior with parameters $\alpha_1, \ldots, \alpha_M$, the mean posterior estimate (MPE) (and also the predictive probability) for the

probability of category j given individual i can be written as [25]:

$$\hat{\theta}_{ij} = \frac{r_{ij} + \alpha_j}{r_i + \alpha} = \gamma_i \frac{r_{ij}}{r_i} + (1 - \gamma_i)\frac{\alpha_j}{\alpha} \tag{3.1}$$

with $r_i = \sum_{j=1}^{M} r_{ij}$, $\alpha = \sum_{j=1}^{M} \alpha_j$, and where $\gamma_i = \frac{r_i}{r_i + \alpha}$. Thus, the posterior mean can be viewed as a convex combination of individual data and a prior, where the mixing coefficient γ_i serves to smooth the individual estimate towards the prior.

A natural idea in this context is to define individual models of the form $\boldsymbol{\theta}_i = \pi \boldsymbol{\theta}_i^I + (1-\pi)\boldsymbol{\theta}^P$ where $\boldsymbol{\theta}_i^I$ is the maximum likelihood (frequency) estimate given individual i's data , $\boldsymbol{\theta}^P$ is an empirical prior distribution based on pooling the location counts across all individuals, i.e., $\theta_j^P = \frac{\sum_i r_{ij}}{\sum_i r_i}$, and where $\pi \in [0, 1]$ is a mixing coefficient. This general idea has been used extensively in language modeling for example, where adaptive smoothing based on mixtures is commonly used and found to be effective for building predictive models tailored to specific individuals or corpora [1, 23].

While this adaptive mixing provides some power over direct smoothing, as discussed earlier in the chapter it nonetheless tends to pull all estimates in the same direction, i.e., towards the global $\boldsymbol{\theta}^P$ in our example above. What we would ideally like is to smooth towards parts of the data matrix that are more relevant to individual i. If we have some knowledge about the similarities of our rows (individuals) and our columns (locations) then two additional smoothing strategies immediately suggest themselves, namely smoothing using data from individuals that are similar to individual i, and smoothing using data from locations (columns) that are similar to the columns that are present in individual i's data. In particular, in this chapter we use social network friendships to measure similarities between individuals and spatial distance for similarities of locations. The use of side-information for similarities between rows and columns has also been exploited in the past in the context of location data matrix factorization and recommender systems. Here we are using similar ideas but without

doing dimension reduction, and instead model the categorical distributions directly in the original space. As we will see later in the experimental results, this allows us to retain details about each individual that tend to get blurred in dimension-reduction techniques.

3.4.2 Defining the Mixture Components

Given the above we can set up a model (described below) where there are 4 components: one "unsmoothed individual" component, a population component, and two components based on row and column smoothing. We represent each individual as a convex combination (finite mixture) of component distributions as follows:

$$\boldsymbol{\theta}_i = \sum_{k \in \{I,P,L,S\}} \pi_k \, \boldsymbol{\theta}_i^k \tag{3.2}$$

where the π_k are mixture weights with $0 \leq \pi_k \leq 1, \sum_k \pi_k = 1$, and where the components are each a categorical distribution over the M items (locations), with elements θ_{ij}^k for each component defined as follows:

Individual:

$$\theta_{ij}^I = \frac{r_{ij}}{r_i}$$

where r_{ij} is the count for individual i and location j and $r_i = \sum_{j=1}^M r_{ij}$ is the total count for individual i. This is the maximum likelihood (frequency-based) estimate of an individual's distribution.

Population Smoothing:

$$\theta_j^P = \frac{r_j + \alpha'}{r + M\alpha'}$$

where $r_j = \sum_{i=1}^N r_{ij}$ is the count for location j across all individuals, $r = \sum_{j=1}^M r_j$ is the sum of all counts (across all locations and all individuals), and α' is the parameter of a flat

symmetric Dirichlet smoothing prior (to allow for non-zero probability mass on locations that have no counts in the historical data). Note that this is the only component that does not vary by individual i. In the results later in the chapter we used $\alpha' = \frac{10}{M}$. An alternative will be to optimize this value (e.g., via grid search) as part of the model learning procedure.

Location (Column) and Social (Rows) Smoothing:

In addition to the standard individual and population smoothing we use two additional components for individual-based smoothing that allow for generalization toward nearby locations and locations that are popular among individuals that share social ties. We use kernel diffusion weight matrices [41] $K^L(j, j')$ and $K^S(i, i')$ for column and row smoothing respectively, where $K^L(j, j') \geq 0$ is a non-negative similarity kernel defined for all pairs of locations $1 \leq j, j' \leq M$, and $K^S(i, i') \geq 0$ is defined similarly for rows, $1 \leq i, i' \leq N$. This type of diffusion approach is commonly used to propagate information across similar items in a variety of contexts (e.g., see [74, 48]). The particular form of the diffusion weights is not critical—below we use relatively simple measures based on location and social distance.

For the location smoothing we define

$$r^L_{ij} = \frac{1}{Z_j} \sum_{j', j' \neq j} K^L(j, j') r_{ij'}$$

where $Z_j = \sum_{j, j' \neq j} K^L(j, j')$ is a normalization constant and the r^L_{ij} values can be viewed as smoothed pseudocounts for a particular location (column) j, obtained from a sum of weighted counts from similar locations (similar columns), with the weights being a function of location-to-location similarity.

From these pseudocounts we can estimate a predictive distribution for each individual i and location j as

$$\theta^L_{ij} = \frac{r^L_{ij}}{\sum_j r^L_{ij}}.$$

In the results in this chapter, for location (column) smoothing, we use the spatial distance between two locations as an indicator of similarity. The (j, j')th element in $K^L(j, j')$ is proportional to the probability density function of a univariate Gaussian model of the distance (in kilometers) between j and j' (or equal to 0 for $j = j'$) using a fixed bandwidth σ. Later in this chapter, we used $\sigma = 2km$ in our experiments. This was based on initial data analysis in which we observed that two kilometers was approximately the distance individuals traveled to new locations.

For the social smoothing we define

$$r_{ij}^S = \frac{1}{Z_i} \sum_{i', i' \neq j} K^S(i, i') r_{i'j}$$

where $Z_i = \sum_{i, i' \neq i} K^S(i, i')$ is a normalization constant and the r_{ij}^S values can be viewed as smoothed pseudocounts for a particular individual (row) i, obtained from a sum of weighted counts from similar individuals (rows), with the weights being a function of individual-to-individual similarity. From these pseudocounts we can again estimate a predictive distribution for each individual i and location j as

$$\theta_{ij}^S = \frac{r_{ij}^S}{\sum_j r_{ij}^S}.$$

The social (row) smoothing is estimated using the social graph. Each similarity $K^S(i, i')$ is proportional to the cosine similarity between the two binary vectors that represent the individuals' social ties.

Kernel-based approaches for location and social smoothing have been used for location prediction problems in the past (e.g., [70, 71]) where a similarity matrix is created using the data or external information (geographic distances) and then used to smooth the individual's data in collaborative filtering fashion. Our approach can be viewed as similar in the sense that we use location and social similarity kernels, but different in that we learn to combine

these sources of information by learning mixture weights in an adaptive manner.

To ensure that our approach is scalable we do not work with the full dense or $M \times M$ location kernel matrix, but instead use sparse versions of these matrices that in effect set to 0 any small similarity values below a small threshold value. The social kernel matrix is naturally sparse.

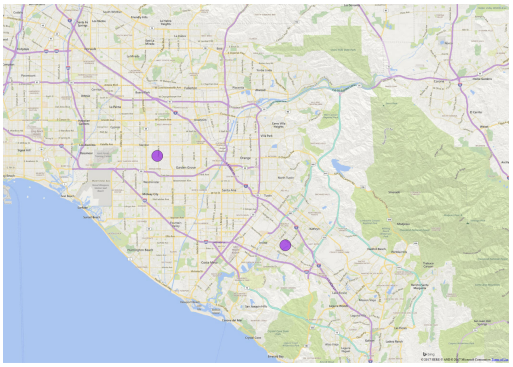
To illustrate the effect of each smoothing component on an individual we plot in Figure 3.3 the top 100 locations with the highest probability under each component for a randomly selected user from the Twitter data set. The plot shows that without smoothing the model only puts probability mass in the "known" locations, which will likely not be effective at prediction time.

3.5 Learning Individual Models

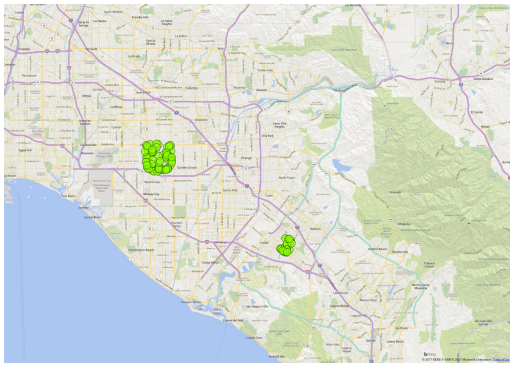
We explore both batch learning and online (day by day) learning of individual-level models in this chapter. We begin below by discussing the batch learning approach and return to online learning later. In our experiments we chronologically partition our data into 3 disjoint subsets:

- Training data D_{tr}, for estimating the components in the mixture model (as described above);
- Validation data D_v, for estimating the mixing weights π (described below);
- Test data D_{te}, which is holdout data used for evaluation of different models (as described later in the experimental results section).

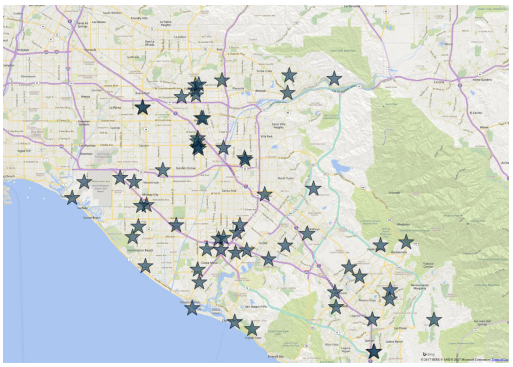
In the first step of learning the training data D_{tr} is used to compute the four component



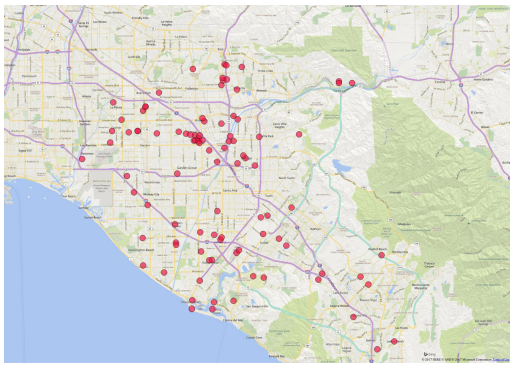
(a) Individual Component



(b) Location Smoothing Component



(c) Social Smoothing Component



(d) Population Smoothing Component

Figure 3.3: Top 100 locations with the highest probability in the individual component and for each smoothing component for a randomly selected user with two observed locations in the twitter data set from Orange County, California.

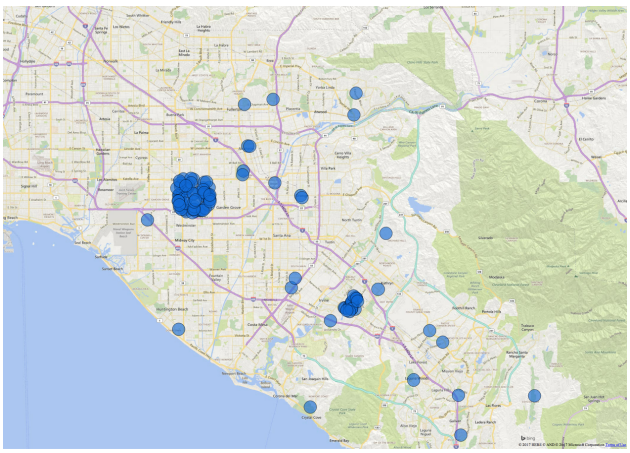


Figure 3.4: Top 100 locations with the highest probability according to the adaptive mixture model for a randomly selected user in the twitter data set from Orange County, California.

distributions $\boldsymbol{\theta}_i^I, \boldsymbol{\theta}^P, \boldsymbol{\theta}_i^L$ and $\boldsymbol{\theta}_i^S$. Three of the four components are individual-specific (the individual $\boldsymbol{\theta}_i^I$, location $\boldsymbol{\theta}_i^L$, and social $\boldsymbol{\theta}_i^S$ components) and the other (population $\boldsymbol{\theta}^P$) is

shared across all individuals.

In the second step, the validation data D_v is used to estimate the mixture weights $\boldsymbol{\pi}$, given the known components estimated from D_{tr}. We use the Expectation-Maximization (EM) algorithm for this purpose, which iteratively estimates the mixing weights $\boldsymbol{\pi}$ that maximize the likelihood of D_v times a prior. We used a Dirichlet prior with parameters $\beta \cdot m_0$ over the weights, where $m_0 = \frac{1}{4}$ is a flat uniform prior and β is a scalar hyperparameter that controls the prior strength. We learned β by using simple gradient descent algorithm at the end of each iteration in the EM algorithm. EM typically converges very quickly in this context since the component distributions are known and fixed. By learning the mixture weights on validation data we are in effect learning the weights that are best in a predictive sense, i.e., combining the different smoothing components in a manner that best generalizes to new data. It is important that the mixture weights are estimated using a data set D_v that is separate from the data D_{tr} used to estimate the components—otherwise EM will overfit and put all the weight on the individual component $\boldsymbol{\theta}_i^I$ that "memorizes" D_{tr} and will not learn to smooth appropriately (e.g., see [64]).

In Figure 3.4 we plot the top 100 locations with the highest probability according to the learned adaptive mixture model for the same user that was presented in Figure 3.3. The figure shows that the locations with the highest probability are a combination of spatial locations at and near the individual's data, as well as generalizing to spatial locations that are either popular in the population (e.g Disneyland) or popular in the individual's social network of friends.

3.6 Experimental Setup

We describe two experimental frameworks for model evaluation in the remainder of the chapter: (1) evaluation with a single training and test data set, i.e., a batch approach (in this section below) and (2) sequential evaluation in an online framework (described later in the chapter). The batch framework has the advantage of simplicity, while the online approach mimics more closely how these algorithms would be used in real-world applications.

In the batch framework, each individual's data is partitioned chronologically into three disjoint data sets as described earlier in Section 3.5: *Training*, *Validation* and *Test*, with 60% of the data used for training, 20% for validation and the other 20% for test. The training data set is used to learn the model components while the validation data set is used to optimize the mixing weights and the hyperparameters. We then perform out-of-sample evaluation using the test data.

3.6.1 Evaluation Metrics

We evaluate our model using three different evaluation metrics based on item rankings and probability assigned to observed outcomes.

Average Percentile Rank on Test Data

For each individual, we can produce a ranking of the likelihood of visiting different locations in the future. Specifically, for each individual i, once a model is fit for that individual using the training and validation data, the model yields a predictive probability distribution $\boldsymbol{\theta}_i$ over all $j = 1, \ldots, M$ locations. The M location probabilities for individual i are then ordered and each new unseen event (location) in the test data for that individual yields a rank. To

make it easier to interpret these scores we linearly scale each rank to generate a percentile ranking (PR) between 0% and 100% such that if an event is ranked highest among all M locations it gets a score of 100% and if it is ranked lowest it gets a score of 0%.

We compute an average percentile rank (APR) by first computing the average of percentile ranks over the events for each individual and then averaging across individuals. This gives each individual equal weight in the evaluation experiments. An alternative is to simply average the percentile ranks across all events, which effectively gives more weight to individuals with more data. We evaluated both approaches to computing APR in our experiments and found that they yielded qualitatively similar results—so for simplicity we only report results below for APR computed by averaging over individuals.

Recall@T

This metric evaluates the ability of the model to assign high rank to observed locations in the test data set for each individual. For a given model, for each individual i, the predicted probabilities over the M locations are sorted in descending order, and *Recall@T* measures what fraction of locations in the test data for the ith individual were ranked at the top T locations (in all of our evaluations we set $T = 50$). For an individual i with n_i events (each with location j) in the test data, the *Recall@T* is evaluated as

$$Recall@T = \frac{1}{n_i} \sum_{k=1}^{n_i} \mathbf{I}(rank(i, j_k) \leq T) \tag{3.3}$$

where $\mathbf{I}$ denotes the indicator binary function $I(\cdot) \in \{0, 1\}$.

Note that *Recall@T* is less fine-grained than the APR method: it only focuses on the top-T ranked locations and only analyzes the order of these locations (and not if their probabilities are calibrated correctly).

Rank-based evaluation is frequently used in evaluation of recommender systems in which the goal is to be able to generate a ranked list of items (locations). Ranking using probabilities is common, but models without probabilistic semantics (such as non-negative matrix factorization methods) can also be evaluated as long as they produce scalar values that (broadly speaking) vary monotonically with the likelihood of occurrence.

Log-Probability on Test Data

Many applications require the probability of an event rather than just its rank. A typical example is in mobile advertising where decisions on which ads to show to an individual are often based on expected value calculations that are based on having available a probability distribution over different possible outcomes. A natural and widely-used evaluation metric for probabilistic models is to compute the average of the log-probabilities of events that occur in the test data. For example, if individual i has a predictive probability model $\boldsymbol{\theta}_i = [\theta_{i1}, \ldots, \theta_{iM}]$ (learned from the training and validation data), and n_i events (each with location j) in the test data, then that individual's average log-probability score is defined as

$$\log P = \frac{1}{n_i} \sum_{k=1}^{n_i} \log \theta_{ij_k}$$

Higher log-probability scores are better in the sense that intuitively model A is generalizing better than model B on new test data if model A places more probability mass than model B on events that actually occur in the test data (and has a higher log-probability score as a result). This intuition can be made rigorous in the sense that asymptotically (as the number of test data points grows) the model with the highest test log-probability score among a set of competing models is in fact the model that is closest (statistically) to the true data-generating process [63]. Since all models have a total probability mass of 1 that they must each distribute over the M possible outcomes, the log probability metric is fair in the sense

that more complex models do not necessarily have any advantage over simple ones.

Note also that because each θ_{ij} is a probability then the log probability scores for an individual's events $\log \theta_{ij}$ will always be negative, as will their average. In fact, for categorical outcomes, the best possible log probability score is a score of 0, which is only attainable if a model is perfectly accurate and it perfectly predicts with probability 1 every event in the test data set.

This evaluation metric was also used in Chapter 2 with the distinction of using a categorical probability distribution in this chapter for all models instead of a continuous density $f(\cdot)$.

As with the APR metric, we found that averaging over individuals or over events yielded very similar results, so in the interest of clarity we just present results for averaging over individuals below and will denote the resulting average log-probability score as $\log P$.

3.6.2 Adaptive Mixture with Individual vs. Global Mixing Weights

We first repeat the experiment from Section 2.5.3 and compare two versions of the four-component Adaptive Mixture model: one which uses globally-shared mixing weights $\boldsymbol{\pi}$ and another which uses individually learned mixing weights $\boldsymbol{\pi}_i$. For both models, we learn the individual multinomial components on the training set and optimize the mixing weights on the validation data set. We train both versions of the model and compare the APR, $Recall@50$ and $\log P$ results across all four data sets.

The results presented in Table 3.2 are consistent with the ones in Table 2.2. The Adaptive Mixture model with globally-learned mixture weights outperforms the model with individually learned mixing weights on average. Further analysis of the results showed that users for whom the global mixing weights approach performed better had relatively small validation

	APR		*Recall*@50		$\log P$	
Data set	Indiv	Global	Indiv	Global	Indiv	Global
TwOC	0.954	**0.955**	0.712	**0.715**	**-4.448**	-4.451
TwNY	0.931	**0.934**	0.577	**0.580**	-5.855	**-5.802**
GoSF	0.876	**0.882**	**0.440**	0.436	**-6.670**	-6.680
GoNY	0.832	**0.841**	**0.395**	0.392	**-7.016**	-7.022
Avg	0.898	**0.903**	**0.531**	**0.531**	-5.997	**-5.989**

Table 3.2: APR, Recall@50 and $\log P$ on the test data for the Adaptive Mixture model with individually and globally learned mixing weights across the different data sets and on average (last row). Higher results are better. Best performing methods indicated in bold font.

data sets at optimization time. As a result, the quality of their individually-learned mixing weights was low which led them to underperform the globally-learned mixing weights model. This echoes our results from Chapter 2 that suggested that it is possible that a hybrid approach will outperform the globally-learned mixing-weights Adaptive Mixture approach in real life applications.

In addition, similarly to the mixture-KDE model from the previous chapter, the model complexity (i.e., number of learned parameters) of the global mixing-weights Adaptive Mixture model is much lower than that of the individual-learned mixing-weights version as the number of mixing weights grows linearly with the number of users. For this reasons we focus on the global mixing-weights Adaptive Mixture model throughout the remainder of this chapter.

3.6.3 Adaptive Mixture with Varied number of Components

To justify the use of our individually-based mixture components, we evaluate the performance of the Adaptive Mixture model using different sets of components. Namely, we compare a two-component Adaptive Mixture model which includes individual and population components (**AM2**), and a pair of three-component models with either the Location-smoothing or the Social-smoothing components removed from the four-component model, notated as **Soc-AM3** and **Loc-AM3** respectively. We omit from this experiment any configuration of

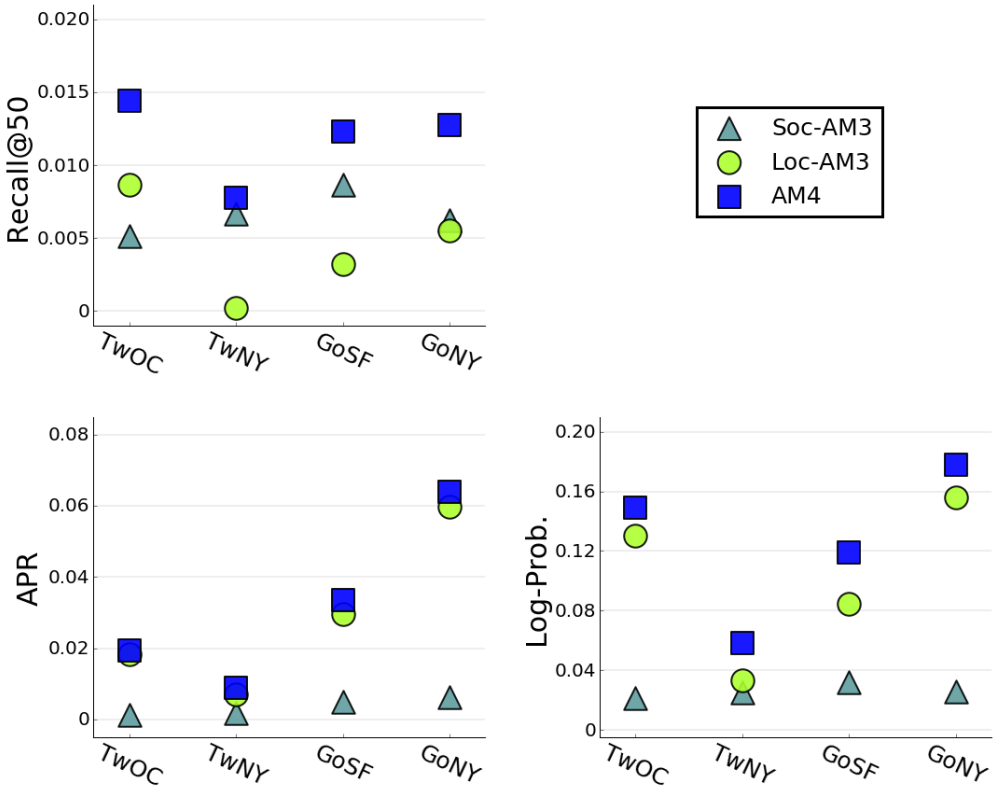


Figure 3.5: Recall@50 (top left) APR (bottom left) and log P (bottom right) results for the AM4, Soc-AM3 and Loc-AM3 models compared to the AM2 model that is used as baseline. Higher results are better. Best viewed in color.

the model that does not include the population components as it led to probability estimates of 0.0 for some test locations and to a computational error (log of 0).

We use the results of the AM2 model as baseline comparison and plot in Figure 3.5 the differences between the evaluated Adaptive Mixture configuration and the baseline. The plot shows that adding each component resulted in a systematic improvement of the model's predictive power across all evaluation metrics and data sets.

3.6.4 Comparing Adaptive Mixture and Existing Approaches

In our evaluations, we also compared the performance of our adaptive mixtures with well-known general-purpose baselines as well as with several recently-published models for loca-

tion recommendation that are considered to be state-of-the-art.

For the general-purpose baselines (i.e., models that are not tailored to location prediction) we used the following:

Baseline Multinomials: We evaluated two variants of a simple multinomial model for each individual. As a strawman baseline we used a maximum likelihood estimate (MLE) with $\hat{\theta}_{ij} = \frac{r_{ij}}{r_i}$, i.e., simple frequency ratios using each individual's own data, with no smoothing. We expect this to generalize poorly since it assumes that an individual will never visit a new location they have not visited in the past. A more realistic (and surprisingly competitive) baseline is to use a mean posterior estimate (MPE): $\hat{\theta}_{ij} = \frac{r_{ij}+\gamma \cdot \alpha_j}{\sum_j r_{ij}+\gamma \sum_j \alpha_j}$ where the α's are proportional to the global counts for each location (plus a small constant c to provide prior mass for locations with no counts in the historical data) and γ controls the strength of the prior. Both c and γ are estimated via grid search on the validation data.

NMF: Non-negative matrix factorization. In this approach the original $N \times M$ matrix of training data is approximated by the product of two matrices of size $N \times d$ (row weights) and $d \times M$ (column bases), where d is a lower-dimensional representation of the data (typically $d << M, N$). NMF is similar in spirit to classical matrix factorization methods such as singular value decomposition, but is better suited to count data since it adds the constraint that the basis functions and weighting coefficients are non-negative. We use the *scikit-learn* [4] implementation of NMF in our experiments and in the learning step we optimized (using grid search) the dimensionality d of the reduced space and the regularization term on the validation data.

PMF: Bayesian probabilistic Poisson matrix factorization [27]. This method can be viewed as a state-of-the-art probabilistic variant of NMF. It assumes that the counts are generated from a Poisson distribution with hierarchical Gamma priors on the levels of both individual

[4] http://scikit-learn.org/

activity and location activity. The priors allow the model to take into account individual and location effects. PMF produces estimates of Poisson rates of events λ_{ij} for each individual i and location j: we convert these rates into event probabilities by normalizing the rates for each individual, i.e., $\theta_{ij} = \lambda_{ij} / \sum_j \lambda_{ij}$. The hyper parameters used in this method and the value of d were determined via grid search on the validation data.

PMF+: This is a simple ad-hoc extension of PMF where we use a 2 component weighted mixture with the PMF model as one component and the individual MLE estimate as the other component. The purpose of adding the individual MLE component is to provide additional personalization, i.e., individual-level detail. The mixture weights were optimized on the validation data set.

Our second set of comparisons is with methods that are considered state-of-the-art in location-based recommendation and prediction, specifically methods that incorporate the geographical preferences of an individual in order to increase the level of individual-level personalization:

Fused: A matrix factorization approach tailored for location data as described in [12]. For each individual, the geographical preference is learned via a multi-center Gaussian model. In addition, a general location preference is learned jointly for all individuals using probabilistic matrix factorization. The two components are learned independently and then combined as a product to produce a predictive probability distribution for each individual. In our evaluation we used the PMF as the matrix factorization component.

iGSLR: Individual geo-social location recommendation [70], an approach that combines collaborative filtering component, learned using social ties between individuals and individual geographic preference. The geographical preference is learned using a non-parametric kernel density approach (in contrast to the multi-centered Gaussian approach used in the Fused method).

	TwOC	TwNY	GoSF	GoNY	**Average**
MLE	0.799	0.701	0.669	0.651	0.705
MPE	0.936	0.925	0.848	0.777	0.871
AM4	**0.955**	**0.934**	**0.882**	**0.841**	**0.903**
NMF	0.871	0.869	0.821	0.759	0.830
PMF	0.914	0.904	0.830	0.759	0.852
PMF+	0.937	0.920	0.850	0.780	0.872
iGSLR	0.605	0.597	0.615	0.555	0.593
RGFM	0.914	0.912	0.841	0.757	0.856
Fused	0.919	0.904	0.830	0.759	0.853

Table 3.3: **Batch Average APR** on test data across individuals, for each data set and averaged across data sets. Higher scores are better. Best-performing methods indicated in bold font.

RGFM: Known as Ranked-GeoFM, this is a state-of-the-art method that has been shown to outperform a wide variety of other techniques [47]. The method has a geographical influence function that allows sharing of information between nearby locations. In addition, it uses a rank-based objective function, which in principle can lead to better ranking performance compared to methods trained using log-likelihood.

We compared all of these methods to our proposed 4-component adaptive mixture model (**AM4**), with individual, population, location, and social smoothing components.

3.7 Batch Experimental Results

For the batch evaluation experiments we computed the APR and *Recall*@50 test metrics using all of the models described earlier in Section 3.6.4 and the average log-probability test metric for models that produce probability distributions as predictions.

Tables 3.3 and 3.4 show the APR and *Recall*@50 results for the different models (rows) across the 4 different data sets (columns) as well as a 5th column that shows the average for

	TwOC	TwNY	GoSF	GoNY	**Average**
MLE	0.600	0.405	0.336	0.303	0.411
MPE	0.700	0.572	0.423	0.378	0.519
AM4	**0.715**	**0.580**	**0.436**	**0.392**	**0.531**
NMF	0.390	0.310	0.270	0.189	0.290
PMF	0.524	0.412	0.305	0.229	0.368
PMF+	0.712	0.569	0.426	0.376	0.521
iGSLR	0.156	0.138	0.146	0.080	0.130
RGFM	0.539	0.443	0.281	0.203	0.367
Fused	0.528	0.412	0.305	0.229	0.369

Table 3.4: **Batch Average *Recall@50*** on test data across individuals, for each data set and averaged across data sets. Higher scores are better. Best-performing methods indicated in bold font.

	TwOC	TwNY	GoSF	GoNY	**Average**
MPE	-4.50	-5.93	-7.21	-7.43	-6.27
AM4	**-4.45**	**-5.80**	**-6.68**	**-7.02**	**-5.99**
PMF	-6.33	-7.10	-7.53	-8.02	-7.25
PMF+	-4.56	-5.94	-6.77	-7.15	-6.11
Fused	-6.31	-7.10	-7.53	-8.02	-7.24

Table 3.5: **Batch Average Log-Probability** on test data across individuals, for each data set and averaged across data sets. Higher scores are better. Best-performing methods indicated in bold font.

each model across the data sets. The result for the best-performing model for each data set, and for the average across data sets, is highlighted in bold in each table.

The models have been divided into 3 different groups of 3: the first group consists of probability distributions and smoothed mixture variants (the MLE and MPE estimate, as well as our proposed adaptive mixture model with 4 components, AM4); the second group consists of matrix factorization methods (NMF, PMF, and PMF+); and the third group consists of methods customized for location prediction (iGSLR, RGFM, and Fused).

A number of trends emerge from the data in the tables. The MLE method is relatively poor as one might expect since it incorporates no spatial smoothing. The MPE estimate improves

significantly over the MLE estimate—and in fact is better than many of the more complex methods. The proposed AM4 method has the best APR, *Recall*@50 and $\log P$ of any of the methods evaluated, providing a systematic improvement over MPE.

The "generic" matrix factorization methods (NMF and PMF) do not perform as well as the simple MPE estimate—this confirms our conjecture earlier in the chapter that matrix factorization methods tend to oversmooth and do not retain enough detail about the individual's behavior to be able to predict well.

Among the location-prediction methods that we tried, RGFM and Fused are competitive with PMF but again are outperformed by the simpler MPE estimate (which in turn is outperformed by our proposed AM4 approach). The iGSLR method did not perform well on our data sets. We surmise that two factors impacted its accuracy on these data sets. In the original work, the collaborative filtering component was learned using a similarity between individuals' home locations [70]. In our data set we did not have that information and we used the social ties information instead, which is likely much less informative than knowing home locations. In addition, the data set used in our work is much sparser than that used in the original paper, and as a result the kernel density method may have been somewhat limited in its ability to learn meaningful geographical preferences for individuals.

To further analyze the predictive performance of our model, we compare in Figure 3.6 the batch APR for the AM4 approach (x-axis) and the best performing baseline (y-axis) from each of the three groups when looking at each individual separately. We performed a Wilcoxon signed rank test on the individual results and found that the AM4 approach dominates all of the other methods with a p-value of of $p < 0.001$ across all four data sets.

Table 3.5 shows the $\log P$ results for the 5 models that produce probability distributions. Note that although the MLE model produces probabilistic predictions, because it produces probability estimates of zero for new locations that an individual has not been to in the past

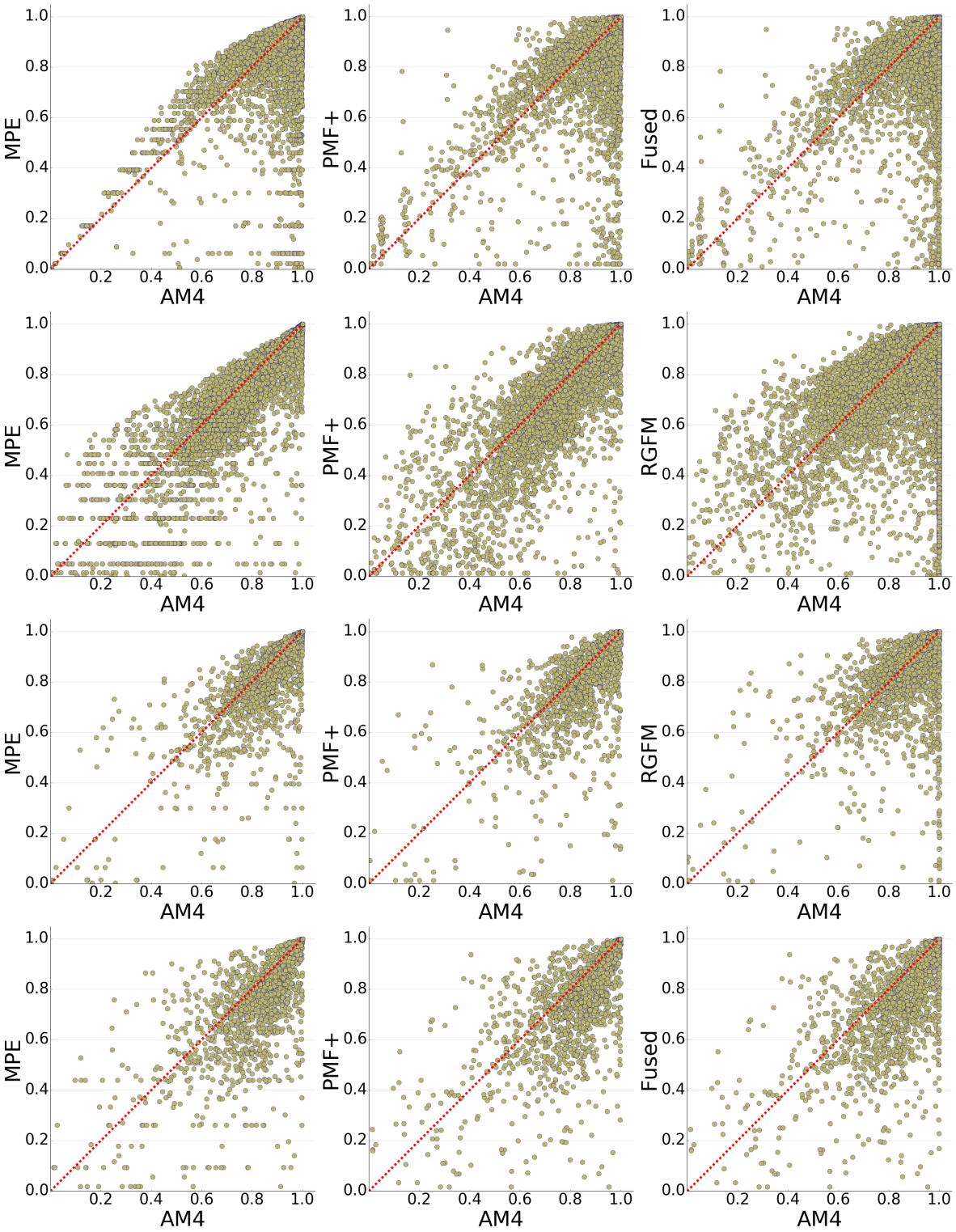


Figure 3.6: Scatter plots comparing the AM4 (x-axis) batch APR results for each individual with the best performing baseline from each group (y-axis) in the `TwOC` (top row), `TwNY` (second row), `GoSF` (third row) and `GoNY` (bottom row) data sets.

(i.e., no smoothing), the log-probability for such events are $-\infty$. For this reason we did not include MLE in Table 3.5.

The results for average log-probability in Table 3.5 reinforces the trends in Tables 3.3 and 3.4. MPE outperforms both PMF and Fused, and adding MPE to PMF improves the PMF score significantly which again confirms that including individual-level detail, via the MPE distribution, can help improve the quality of predictions from matrix factorization methods such as PMF. Once again, the proposed AM4 method is better than all other methods across all 4 data sets.

Batch Prediction for New Unseen Locations

In addition to the standard out-of-sample evaluation above, we also evaluated how well our AM4 method compares to the other baseline models in making predictions only for locations that are new for each individual, i.e., computing ranks and log-probabilities for each individual only for locations that are new (in the test data) for that individual. This type of evaluation is motivated by point-of-interest recommendation systems in which the system only recommends locations (or items) that the individual has not visited before.

Our initial assumption was that because of the *Individual* components in our model, the AM4 approach would be significantly outperformed by other approaches such as matrix factorization. However, the results in tables 3.7, 3.6 and 3.8 show that AM4 is still competitive to the most accurate methods. In terms of APR, Table 3.6, the AM4 method was the most accurate method on average. It significantly outperformed the matrix factorization methods NMF and PMF as well as the Fused method. The RGFM method was competitive with AM4 in general, having better APR on one data set and being close to AM4 on the rest. The log-probability results, in Table 3.8, tell a somewhat different story: here the PMF and Fused approaches performed better than the AM4 approach across all 4 data sets. This

	TwOC	TwNY	GoSF	GoNY	**Average**
MPE	0.830	0.870	0.764	0.675	0.784
AM4	**0.890**	0.887	**0.820**	**0.774**	**0.843**
NMF	0.839	0.829	0.767	0.693	0.782
PMF	0.833	0.862	0.768	0.681	0.786
PMF+	0.832	0.862	0.767	0.681	0.785
iGSLR	0.557	0.574	0.587	0.536	0.563
RGFM	0.885	**0.893**	**0.820**	0.727	0.831
Fused	0.845	0.862	0.768	0.682	0.789

Table 3.6: **Batch Average APR** evaluated on **new locations** across individuals, for each data set and averaged across data sets. Higher scores are better. Best-performing methods indicated in bold font.

	TwOC	TwNY	GoSF	GoNY	**Average**
MPE	0.235	0.272	0.121	0.106	0.183
AM4	0.272	0.285	0.140	0.124	0.205
NMF	0.200	0.191	0.116	0.085	0.148
PMF	0.269	0.274	0.154	0.126	0.206
PMF+	0.258	0.264	0.129	0.103	0.188
iGSLR	0.105	0.107	0.090	0.051	0.088
RGFM	**0.339**	**0.323**	**0.179**	**0.145**	**0.246**
Fused	0.272	0.274	0.152	0.126	0.206

Table 3.7: **Batch Average *Recall@50*** evaluated on **new locations** across individuals, for each data set and averaged across data sets. Higher scores are better. Best-performing methods indicated in bold font.

may indicate that the AM4 method could be improved by better calibrating its probability estimates of unseen locations.

	TwOC	TwNY	GoSF	GoNY	**Average**
MPE	-9.23	-8.83	-9.05	-9.08	-9.05
AM4	-8.96	-8.40	-8.53	-8.69	-8.64
PMF	-8.03	**-7.92**	**-8.26**	**-8.53**	-8.19
PMF+	-9.27	-8.62	-8.67	-8.87	-8.86
Fused	**-8.01**	**-7.92**	**-8.26**	**-8.52**	**-8.18**

Table 3.8: **Batch Average Log-Probability** evaluated on **new locations** across individuals, for each data set and averaged across data sets. Higher scores are better. Best-performing methods indicated in bold font.

3.8 Online Experimental Results

Online Learning Framework

While experiments based on batch training and test data sets are commonly used in research evaluations, for location data it is natural to learn the models in a sequential online fashion given that (a) location data typically contains time-stamps (allowing for sequential evaluation), and (b) in real-world applications it is likely that individual models would be built online (i.e., incrementally as new data arrives). For these reasons we include below additional experiments evaluating the different models in an online context where we learn models using data up to day t and make predictions for day $t+1$, then move one day forward and update the model with the new data from day $t+1$ and predict for day $t+2$, and so on. We use the same three metrics as earlier, APR, *Recall*@50 and $\log P$, where now each metric is computed by averaging over individuals on each prediction day, and then averaged over all prediction days. When making predictions for day $t+1$ we use data from the most recent w days (up to and including day t) for validation and the data prior to $t-w$ for training. Using $w=2$ provided a validation data set with sufficient amount of data for optimization. However, in real-world applications in which the amount of data is significantly larger the value of w could in principle also be optimized. The complete online evaluation scheme is illustrated in Figure 3.7.

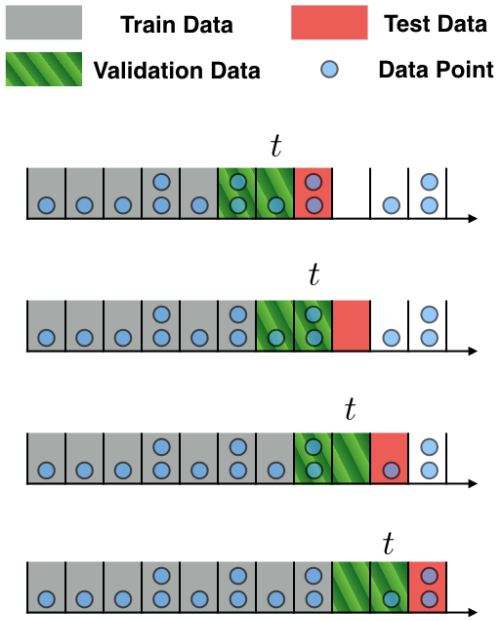


Figure 3.7: The online evaluation scheme used to simulate more realistic applications. Best viewed in color.

In our experiments we also used an initialization window for each individual before making any test predictions. The reason for doing this is primarily to assist in evaluation of the baseline methods: techniques based on smoothing such as MPE and AM4 do not need an initial window, but techniques such as NMF and MLE need an initialization period when used in an online mode. For each of the data sets, we defined the initial period to be 4 days, divided into 2 days of training data and 2 days of validation data. The first evaluation (test prediction) for each individual was performed on their 5th day of data.

For models with mixture components (AM4, BPF+), for each day t the components are learned on the training set and the mixture weights are learned via EM on the validation data, and where instead of using a flat prior as in the batch method here we use a prior on the weights that is proportional to the estimated weights for previous days.

We found that our experiments with the RGFM method did not complete in a reasonable

	TwOC	TwNY	GoSF	GoNY	**Average**
MLE	0.849	0.764	0.710	0.704	0.757
MPE	0.933	0.927	0.830	0.785	0.869
AM4	**0.960**	**0.937**	**0.869**	**0.853**	**0.905**
NMF	0.627	0.605	0.513	0.544	0.572
PMF	0.917	0.911	0.798	0.731	0.839
PMF+	0.933	0.924	0.824	0.777	0.865
iGSLR	0.596	0.595	0.601	0.547	0.585
Fused	0.924	0.911	0.800	0.732	0.842

Table 3.9: **Online Average Percentile Rank (APR)** evaluated on test data across individuals, for each data set and averaged across data sets. Higher scores are better. Best-performing methods indicated in bold font.

	TwOC	TwNY	GoSF	GoNY	**Average**
MLE	0.697	0.529	0.398	0.399	0.506
MPE	0.747	0.637	0.436	0.435	0.564
AM4	**0.757**	**0.639**	**0.442**	**0.441**	**0.570**
NMF	0.137	0.115	0.057	0.084	0.098
PMF	0.567	0.476	0.298	0.266	0.402
PMF+	0.751	0.628	0.432	0.432	0.561
iGSLR	0.099	0.100	0.000	0.000	0.050
Fused	0.572	0.476	0.298	0.266	0.403

Table 3.10: **Online Average Recall@50** evaluated on test data across individuals, for each data set and averaged across data sets. Higher scores are better. Best-performing methods indicated in bold font.

time on a standard Amazon server. This does not necessarily mean that it cannot perform in an online fashion, but indicates that a more scalable implementation would be required for online operation[5].

	TwOC	TwNY	GoSF	GoNY	**Average**
MPE	-4.01	-5.15	-6.96	-6.91	-5.76
AM4	**-3.92**	**-5.05**	**-6.68**	**-6.68**	**-5.58**
PMF	-6.38	-6.88	-7.79	-8.06	-7.28
PMF+	-4.02	-5.21	-6.85	-6.86	-5.73
Fused	-6.34	-6.88	-7.79	-8.06	-7.27

Table 3.11: **Online Average Log-Probability** evaluated on test data across individuals, for each data set and averaged across data sets. Higher scores are better. Best-performing methods indicated in bold font.

Online Prediction Results

The results presented in tables 3.9, 3.10 and 3.11 show the average score across all days for each data set as well as the average across all of the data sets. Not surprisingly the results are completely consistent with the batch results: the simple MPE method is typically more accurate than any of the more complex matrix factorization or location-based baselines, and our proposed AM4 method is more accurate again (often by relatively large margins) across all data sets.

To further explore why our model outperforms the other approaches we compared the day by day performance of the AM4 approach with three of the better-performing baselines: MPE, PMF+ and Fused. In Figure 3.8 we view the daily APR scores from the three models as a function of time. In order to reduce the day-to-day variability we smoothed the results using a moving average filter and plotted the smoothed lines. In general we can see that the models become more accurate over time (higher APR) as more data is available with the exception of the TwNY (top right) data set which is due to users with high variability in the test data that enter the data set during the first 50 days—this is typical of the "cold-start" problem where data is too sparse early on to learn accurate models. However the proposed AM4 model appears to have significantly less of an issue with "cold-start" than the other

[5]To learn this model, we used the Matlab code that was used in this publication for this model that uses a non-sequential batch learning algorithm. The code was provided by the author of the paper upon request via email.

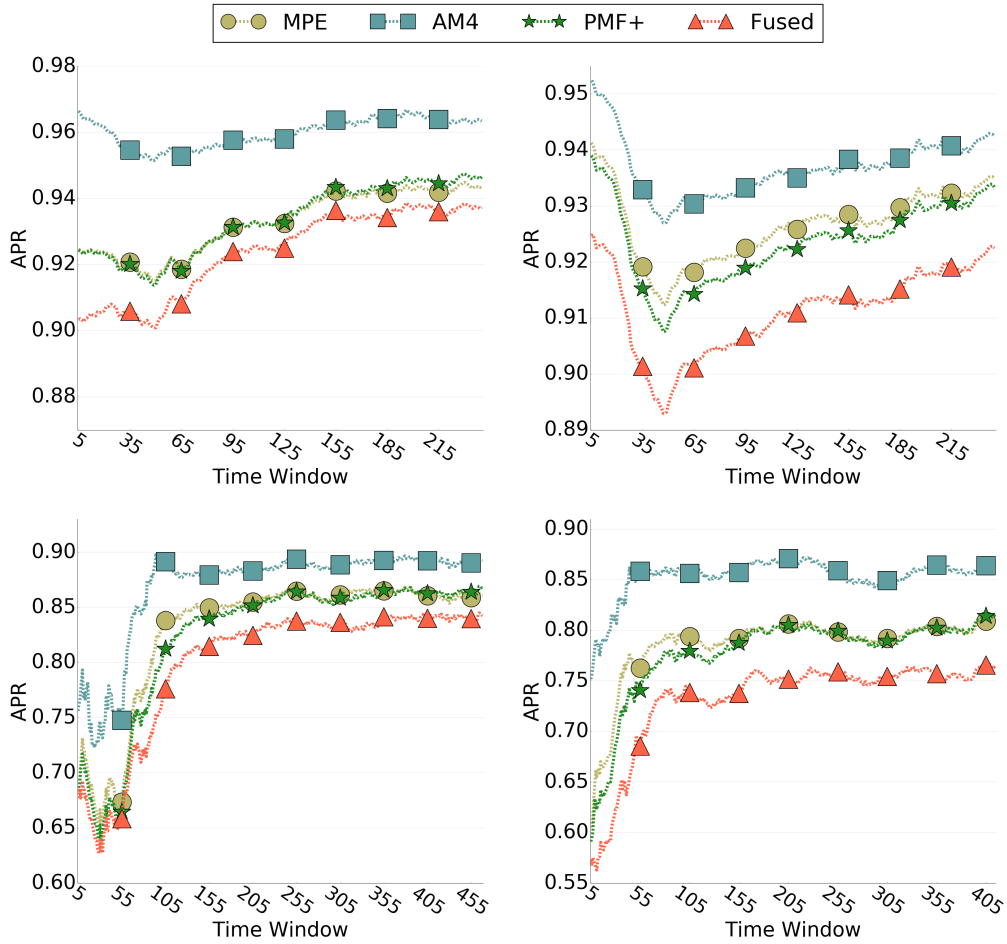


Figure 3.8: Moving average of the daily APR score for the top three ranked models for the `TwOC` (top left), `TwNY` (top right), `GoSF` (bottom left) and `GoNY` (bottom right) data sets. Best viewed in color.

approaches given that it is making relatively accurate predictions early on. Another way to look at this is that the benefit of the AM4 approach appears to be greatest, compared to the other methods, when there is relatively little data available for each individual (a common situation in real-world applications).

3.9 Scalability

Here we consider the complexity and scalability of our method in terms of the number of users N and number of locations M. At each time t it takes $\mathcal{O}(NM)$ time to update the

training counts and to compute the Individual and Population multinomial components. In addition, the time complexity of computing the Location smoothing and Social smoothing mlutinomial components is $\mathcal{O}(N^2M)$ and $\mathcal{O}(NM^2)$ respectively. The latter two complexities stem from the fact that we can think of the process as matrix multiplications CW_L and W_SC respectively where C is the individual-by-location observation matrix of size $N \times M$ and W_S and W_L are the diffusion weight matrices size $N \times N$ and $M \times M$ respectively and used for Social (rows) smoothing and Location (column) smoothing (both matrices are described earlier in Section 3.4).

In addition to the component computations, at each time t our model runs an EM algorithm to optimize the mixing weights. With k components, the time complexity of a single iteration of the EM algorithm is $\mathcal{O}(Nk)$. However, since k is fixed to $k = 4$ in our algorithm the time complexity in practice is $\mathcal{O}(N)$. The time taken for hyperparameter optimization is not significant.

Overall it is clear that the time complexity of our method is dominated by the matrix multiplications that are required to compute the Location and Social smoothing components $\boldsymbol{\theta}^L$ and $\boldsymbol{\theta}^S$: $\mathcal{O}(N^2M + NM^2)$. The space complexity of our approach is $\mathcal{O}(N^2 + M^2)$, the sizes of the Social and Location diffusion weight matrices. However, in practice both these matrices, as well as as the observation matrix C are sparse and require much less space to represent in a sparse representation.

In order to scale our algorithm for large data sets, we took couple of steps to reduce both space and time requirements. First, the time consuming process of computing the smoothing components $\boldsymbol{\theta}^L$ and $\boldsymbol{\theta}^S$ is independent for each user. By implementing this process using a multicore architecture, we can reduce this process (in principle) by a factor of P (number of processors) to $\mathcal{O}(\frac{N}{P}NM + \frac{N}{P}M^2)$. While this speedup is just a constant factor reduction in complexity, it is very noticeable in practice. Second, by using sparse matrix multiplication techniques we can take advantage of the sparse structure of C and W_S. The time complexity

of computing the Social smoothing components is reduced from $\mathcal{O}(N^2M)$ to $\mathcal{O}(wM)$ where w is the number of non-zero values in W_S and $w << N^2$, the total number of elements in W_S. For the Location smoothing component the time complexity using sparse matrix operations is $\mathcal{O}(cM)$ where c is the number of non-zero values in the observations matrix C and $c << NM$, the total number of elements in C.

Our implementation of the algorithm used in the experimental results part of this chapter utilized both techniques for time complexity optimization, resulting in a highly scalable implementation. We implemented the algorithm in Python and ran it on a 16 core machine, where each core is a Intel Xeon E5620 with 2.4 GHz processor, and measured the clock-time using the largest data set of geotagged tweets from New York, NY. At each time t in the evaluation, the algorithm process includes: estimation of the individual components, EM for the mixture weights, and optimization of the hyperparameters, and took approximately $100ms$ on average per user. The time for test predictions and accumulating the training counts was negligible (less than 1ms).

3.10 Conclusions

The primary contributions of this chapter include the following:

- We introduced a general framework for smoothing of individual-level categorical distributions using mixture models.
- Based on this approach we developed an Adaptive Mixture model that is relatively simple to implement, computationally efficient, and can easily operate in an online manner.
- We explored continuous space geolocaion data discretization based on publicly available geo-parcel information.

- We applied this framework to the problem of learning location models from historical matrices of individual-location counts.
- We evaluated our model with extensive batch and online schemes using multiple evaluation metrics showing results on Twitter and Gowalla data sets. The results showed that our model outperforms existing approaches for individual-level location modeling in both the batch and online prediction scenarios.

Chapter 4

Prediction of Sparse User-Item Consumption Rates with Zero-Inflated Poisson Regression

The previous two chapters presented predictive models for individual-level event data where the focus is on predicting the identity of future items that a user will interact with. In this chapter we broaden the range of data sets under investigation beyond location data and address the problem of developing user models that can predict the rate at which individuals consume both old and new items. We use zero-inflated Poisson (ZIP) regression models as the basis for our modeling approach, leading to a general framework for modeling user-item consumption rates over time. We show that these models are more flexible in capturing user behavior than alternatives such as well-known latent factor models based on matrix factorization. We compare the performance of ZIP regression and latent factor models on three different data sets involving online music streaming, restaurant reviews, and social media interactions. The ZIP regression models are systematically more accurate across all three data sets across different prediction metrics.

4.1 Introduction

In many aspects of our daily lives the way we consume products and items has evolved from physical consumption to virtual shops and services. We purchase books online instead of shopping at brick-and-mortar stores, stream music and movies online instead of purchasing physical copies, and so on. The digital nature of our consumption provides the opportunity for tailoring of individual user experiences that can benefit both the consumer and the provider. As a consequence, the ability to develop predictive individual-level models for user-item consumption from past observations is increasingly important.

Building accurate models of consumption in a typical digital environment is challenging for multiple reasons. In particular, as an individual moves forward through time, their future behavior can be broadly categorized as consisting of a combination of (a) consuming items that they have interacted with in the past (i.e., repeat consumption), and (b) seeking novel items that they have not consumed in the past (i.e., new consumption). User modeling in this context requires that a model must learn to balance these two aspects of behavior. Individual heterogeneity (i.e., the fact that there is significant variability in behavior across users) further complicates the modeling process. Another significant challenge is data sparsity given that the number of items a user typically consumes is a very small fraction of the total number of available items.

In this chapter we focus on the problem of predicting *rates* of item consumption per unit time (days, weeks, months) for individual users. The prediction of rates is broadly useful in a variety of applications since it allows us to predict not only which items a user will consume, but also *how often* those items will be consumed. For example, prediction of rates of consumption for specific items and specific sets of users is important in the design and engineering of proxy-caching systems for online streaming media content [56]. For contexts where items have different costs associated with them, predictions of the rates at which a

user will consume specific items can be used as the basis for estimating the expected value of a customer from the provider perspective. Rates also can be used to help evaluate the expected benefit of interventions such as providing incentives to a user. For example, if some users have a high rate of usage for a particular app on their mobile phones and other users have low rates of usage for the same app, the latter group is likely to be a better target for incentivization than the former [31].

Consumption behavior is often a combination of repeat and new consumption. For example, some users' behaviors may be highly repetitive in nature, i.e., they tend to visit the same restaurants, listen to the same music, etc., and rarely try new items. Other users may have behavior at the other extreme, continuously exploring new items and rarely returning to old items. The explore-exploit tradeoff, well known in computer science within the context of reinforcement learning, is also well-established in cognitive science as a basic trait of how humans interact with the world around them (e.g., [17, 69]).

These observations suggest that in addition to handling significant heterogeneity in terms of individual behavior, the notion that there is a steady-state behavior for many users may be a fallacy in the sense that users are continuing over time to both exploit and explore the choice of items available to them. Rather than having user models that are represented as fixed distributions over items, individual behavior can be thought as a dynamic process over time that is driven by feedback from past item consumption, both positive and negative. To capture these ideas we develop individual-level Poisson-based regression models where the predicted rate that a user will consume an item in the next time period is modeled as a function of an individual's past behavior. In addition, our model uses global contextual information (such as item popularity) in order to better generalize to prediction of new items.

The primary contribution of this chapter is the development of a systematic approach for modeling user-item consumption rates over time using Poisson-based regression models with zero-inflation. We show that our proposed approach is scalable to large-scale data sets.

Through a systematic investigation of several user-item consumption data sets from multiple domains, we demonstrate that this model can capture individual-level user preferences for both old and new items as a function of past behavior and contextual information. We compare our model to state-of-the-art alternatives both empirically and qualitatively.

On the surface the problem we address looks very similar to that of the classic recommender system problem. However, it is important to note that the modeling goals and evaluation criteria in our work are significantly different. Recommender systems focus only on prediction and ranking of new items that a user has not consumed in the past, e.g., for items such as movies or books, where typically an item is only consumed once by a user. In contrast we specifically focus on problems where consumption is a mix of repeated and novel item consumption (such as music listening and online websites visits), where the natural approach is to model the *rates* at which items are consumed and where evaluation considers how well these rates are predicted (rather than just evaluating the likelihood of whether a user will consume an item or not).

The rest of this chapter proceeds as follows. In Section 4.2 we explore different user-item consumption sequence data sets, and provide motivation for our modeling approach in Section 4.3. In Section 4.4 we describe our proposed ZIP model for understanding and predicting user-item consumption rates and we show how this model is learned using user-item consumption observations in Section 4.5. Section 4.6 provides an overview of the existing approaches for modeling user-item consumption data. In Section 4.7 we compare our proposed model to a variety of state-of-the-art models and interpret the results. Section 4.8 discusses the scalability of the approach and we conclude with a brief summary in Section 4.9.

Dataset	N	M	t	T	% **non-zero**
reddit	1000	1000	week	52	2.5
lastfm	931	19997	month	50	0.5
Yelp	2836	203	2 months	12	1.7

Table 4.1: Summary of the three data sets used in this chapter: number of unique users N, unique items M, time-window t, number of windows T, and the percentage of data points that are non-zero.

4.2 User-Item Consumption Data

In this chapter we consider user-item consumption counts in discrete time (by day, week, month, etc.). We define $y_{ij}^t \in \{0, 1, 2, \ldots\}$ as the number of consumptions of item $j \in \{1, 2, \ldots, M\}$ by user $i \in \{1, 2, \ldots, N\}$ in time window $t \in \{1, 2, \ldots, T\}$.

Our work is motivated by the challenge of creating a general framework for consumer behavior data across different domains. To that end, we investigate multiple publicly available data sets that represent different types of items and consumption activities. The three data sets are summarized and compared in Table 4.1.

reddit: reddit is a popular social network with over 15M unique active monthly users[1]. It contains on the order of 1 million topic-focused subgroups (known as subreddits) where users can post, comment, vote on content and more. In our work we considered data from a sample of $N = 1000$ users with high activity and $M = 1000$ highly active subreddits throughout 2015. The value of y_{ij}^t is defined as the number of times user i posted (or commented) in a subreddit j during a given week t.

lastfm: lastfm is an online music streaming service that allows users to listen to a selected song or playlist. The particular dataset we use contains the listening actions over time of nearly $N = 1000$ users[2]. We consider artists as items and retain the top $M = 20K$ artists that are most frequently listened to during the period of time of February 2005 to June 2009.

[1]https://www.reddit.com/r/AskReddit/about/traffic/

[2]http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html

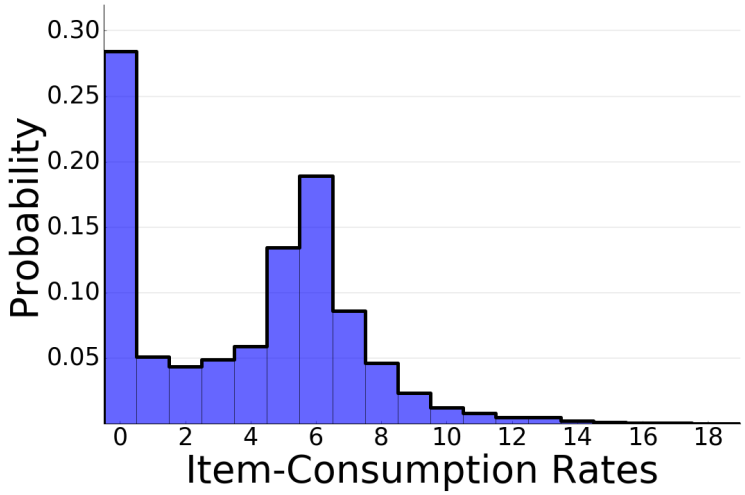


Figure 4.1: The distribution of item-consumption rates for a sample of user-item pairs with similar *average* rates.

In the lastfm dataset, the value of y_{ij}^t represents the number of times that user i listened to a song performed by artist j during a month t.

Yelp: Yelp is a popular review platform that allows users to share their experience with different service providers such as restaurants. The dataset that we use has been widely used as a benchmark across recommender system studies[3]. For our experiments we focused on the histories of $N = 2836$ unique users and their reviews of $M = 203$ types of restaurants (e.g. fast food, Mexican, sushi, etc.) in the Scottsdale and Phoenix (Arizona, USA) metropolitan areas between June 2014 and June 2016. y_{ij}^t is the number of times user i reviewed a restaurant of type j during every two months t.

4.3 Excess of Zeros and Heterogeneity

Two typical characteristics of sparse user-item data sets are (1) an excess of zeros and (2) heterogeneity across both users and items. We discuss both characteristics below in turn and describe our approach to handling each from a modeling perspective.

[3]http://www.yelp.com/dataset_challenge

4.3.1 Excess Zeros

A common feature of user-item consumption data sets, particularly when the number of items is large, is a very high rate of zeros, i.e., most users have zero consumptions of the vast majority of items. This is certainly true of the three data sets we analyze in this chapter where 98% to 99% of the entries are zero across the data sets. This is not surprising. For high-dimensional data sets each user will only be exposed to or be aware of a relatively small fraction of the potential items that they could interact with. In addition, there are practical limits (e.g., from cognitive and economic perspectives) in terms of how many items a user can realistically interact with.

In statistical modeling an overabundance of zeros is often referred to as "zero-inflation" [57, 44], to reflect the phenomenon that the frequency of zeros in the data is significantly higher than what a typical parametric model for count data (such as a Poisson model) can handle. This observation has been made in application contexts as diverse as epidemiology, economics, and manufacturing (e.g., see [8]), but has seen relatively little application to the type of high-dimensional user-item consumption data that we investigate in this chapter. To illustrate the phenomenon of zero-inflation Figure 4.1 shows a histogram of the y_{ij}^t values for a sample of user-item pairs from all datasets with an average consumption rate between 5 and 6 across time (values of $y_{ij}^t = 0$ were excluded in computing the average). The histogram illustrates the variability of y_{ij}^t values across all time windows. We can see that the consumption rate has a bimodal distribution with one mode at $y_{ij}^t = 0$ and additional mode at $y_{ij}^t = 6$. We selected the average rate of 5 to 6 for illustration; however, similar bimodal patterns occur for different values of average number of user-item consumption. The bimodal nature of this data suggests that user-item rate can be represented as a mixture of two processes: an *exposure process* and a *rate process*.

Exposure Process: The exposure process describes whether or not a user i has been

exposed to item j at time t. The concept of exposure captures the idea that for large item sets a typical user is likely to be unaware of (or unexposed to) most items in the "item vocabulary" (see also [51, 33]), e.g., in music-listening many artists are unknown to many users. We define $z_{ij}^t \in \{0, 1\}$ as an indicator variable to indicate if user i was exposed to item j at time t. We can model $P(z_{ij}^t = 1)$ via a Bernoulli distribution with parameter π_{ij}^t, where the Bernoulli parameter will be a function of the past history of user i and item j.

Rate Process: Conditioned on exposure, i.e., $z_{ij}^t = 1$, the rate process accounts for the number of times user i consumes item j at time t. A natural and simple distribution for the rate process is the Poisson model, parameterized by the expected consumption rate λ_{ij}^t:

$$P(y_{ij}^t = k|\lambda_{ij}^t) = \frac{{\lambda_{ij}^t}^k e^{-\lambda_{ij}^t}}{k!} \tag{4.1}$$

where $k = 0, 1, 2, \ldots$ is the number of consumptions. There are a number of other alternatives for defining probability distributions over count data that we could have used in our modeling approach, such as the Non-Negative Binomial distribution. We chose to use the Poisson distribution in this chapter for a couple of reasons: (a) it is simple to implement and reproduce and (b) it is straightforward to interpret the model parameters.

4.3.2 Data Heterogeneity

Another common feature of high-dimensional user-item data sets is heterogeneity. Figure 4.2 shows boxplots of the average number of unique items each user consumes (left panel) and the average number of total consumed items for each user (right panel), per week, for each of the three data sets. The figure clearly indicates (a) significant variability across users, as well as (b) significant variability across the different data sets. A plausible explanation for user variability is that different users can have significantly different *budgets*, either monetary or non-monetary (e.g. time), for consuming items. In addition there can be significant

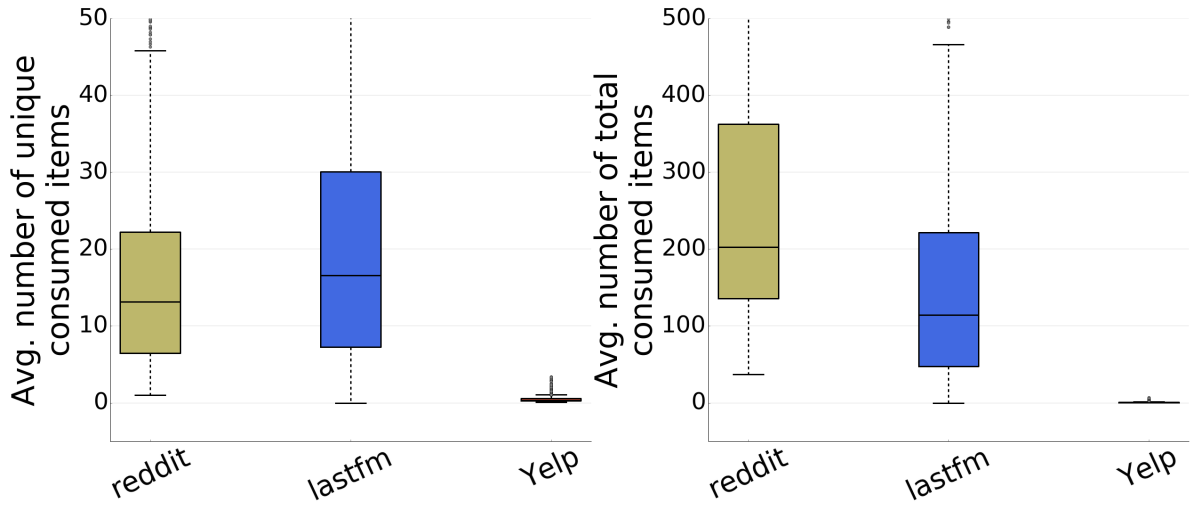


Figure 4.2: Average number of unique consumed items (left) and total number of consumed items (right) per week for each user across each of the three data sets.

variation in the *domain-specific cost* for the consumption of a typical item across different data sets, leading to significant differences in the scale of user-item consumption across different domains, i.e., *domain-specific cost offsets*. For example, the "cost" to a user of listening to a song (lastfm) is significantly less than the cost of eating at a restaurant (Yelp).

Another contribution to data heterogeneity is the natural variation across users (and data sets) of some users to *explore* new items compared to their tendency to *exploit* known items. For example, a user who has a low tendency for exploration will naturally tend to repeat their behavior and, therefore, the number and identity of unique items this user consumes is likely to remain relatively small and static over time. On the other hand a different user could have a tendency to be easily bored with items (a state that could be detected from recent activity [38]) and a corresponding tendency to often explore new items, where the new items are perhaps strongly influenced by global popularity and trends in the data.

4.4 Poisson Regression with Zero-Inflation

Given the prevalence of zero-inflation and heterogeneity in user-item data sets we propose to model the observations y_{ij}^t for user i, item j, at time t, as

1. a mixture of an exposure process and a rate process, where π_{ij}^t is the mixture weight and λ_{ij}^t is the expected rate that user i will consume item j at time t (conditioned on being exposed to the item), and
2. regression models for each of π_{ij}^t and λ_{ij}^t, conditioned on features $\boldsymbol{x}_{ij}^t$.

4.4.1 Zero-Inflated Poisson Models

The *exposure* and *rate* processes are modeled via a mixture of two components: (a) a delta function at zero and (b) a Poisson distribution. The mixture model weights and Poisson rate parameters, π_{ij}^t and λ_{ij}^t respectively, are user and item-dependent and are implicit functions of the features $\boldsymbol{x}_{ij}^t$—we provide more details on the conditional models for these parameters later in this section.

We can write the probability of $P_{zip}(y_{ij}^t = k|\pi_{ij}^t, \lambda_{ij}^t)$ as:

$$P_{zip}(y_{ij}^t = k) = \begin{cases} (1 - \pi_{ij}^t) + \pi_{ij}^t P_\lambda(k|\lambda_{ij}^t), & k = 0 \\ \pi_{ij}^t P_\lambda(k|\lambda_{ij}^t), & k = 1, 2, \ldots \end{cases} \tag{4.2}$$

where $P_\lambda(k|\lambda_{ij}^t)$ is the Poisson probability defined in Equation 4.1. The model above is known as the zero-inflated Poisson (ZIP) regression model, where the regression aspect arises through the conditioning of π_{ij}^t and λ_{ij}^t on the features [44]. In the ZIP model, zeros can be

generated either by (a) the Bernoulli random variable π_{ij}^t taking value 0 or (b) π_{ij}^t taking value 1 and the Poisson model generating a value $k = 0$. From a generative perspective these two "routes" for generating zeros can be interpreted as either (a) the user i not being exposed to item j, or (b) the user being exposed but deciding not to consume the item (by drawing a zero from the Poisson distribution).

An alternative to the ZIP model is to use a shifted Poisson process for the rate that has a minimum value of $k = 1$ (rather than $k = 0$). In the statistical literature this is known as a *hurdle model* in the sense that the Poisson model is invoked if the count is greater than the "hurdle" (where here the hurdle value is 0). We empirically compared the hurdle and the ZIP model (results not shown) and found that the ZIP model systematically outperformed the hurdle variant in terms of modeling and predicting user-item consumption rates for our three data sets. For this reason we focus on the ZIP model in the rest of the chapter.

4.4.2 Regression Features for the Mixture Parameters

We model heterogeneity across users and items via generalized linear regression models for both π_{ij}^t and λ_{ij}^t, where the regression models depend on feature vectors $\boldsymbol{x}_{ij}^t$ that vary by user i, item j and time t. The regression models use two constant intercepts, globally-shared and individual-specific, that capture the effect of *global domain costs* and heterogeneity in *individual-specific budgets* respectively. In addition, to capture the personal balance between *explore* and *exploit* for each individual, we use four data-driven features from an individual user's historical data and contextual information:

- **Past user-item preference** ($x_{ij}^{\bar{t}}$): Represents the average rate that user i consumes item j over time. This feature is motivated by repetitive users who have a high probability of *exploitation*, i.e., they consume similar items at the same rates in each time period. For each user, we compute the past item consumption preference as

Covariate	Notation	Value
Global domain costs	x_0	1
User-specific Budget	x_{i0}	1
Past user-item preference	$x_{ij}^{\bar{t}}$	$\log\left(1+\frac{\sum_{\tau=1}^{t} y_{ij}^{\tau}}{t}\right)$
Current user-item activity	x_{ij}^{t}	$\log(1+y_{ij}^{t})$
Historical item popularity	$x_{j}^{\bar{t}}$	$\log(1+\frac{\sum_i \sum_{\tau=1}^{t} y_{ij}^{\tau}}{tN})$
Current item popularity	x_{j}^{t}	$\log(1+\frac{\sum_i y_{ij}^{t}}{N})$

Table 4.2: Definition of features used in our regression models, based on user and item historical data.

$x_{ij}^{\bar{t}} = \log\left(1+\frac{\sum_{\tau=1}^{t} y_{ij}^{\tau}}{t}\right)$. We add one to prevent numerical errors in the log function.

- **Current user-item activity** (x_{ij}^{t}): For each user i and item j we compute the recent activity as the log of (plus one) the number of times the item was consumed in the current time-window $x_{ij}^{t} = \log(1+y_{ij}^{t})$. This feature is motivated by recent studies on the effect of recency in user-item consumption. For example, in [11] and [2] it was shown that users with preference for exploitation are more likely to consume items from their recent past, whereas users with high preference for *exploration* are likely to avoid recently consumed items as a result of boredom [37], [38], [34].

- **Historical item popularity** ($x_{j}^{\bar{t}}$): This feature models the overall popularity of an item. It is estimated as the average past consumption rate of item j across all users: $x_{j}^{\bar{t}} = \log(1+\frac{\sum_j \sum_{\tau=1}^{t} y_{ij}^{\tau}}{tN})$. This features is expected to capture the behavior of users whose *exploration* preferences are affected by conformity [68].

- **Current item popularity** (x_{j}^{t}): In addition to the overall item popularity, this feature captures immediate trends. It models users who are more affected by trends such as hype as a result of a sale, or "death" of an item. The value of this feature is computed as $x_{j}^{t} = \log(1+\frac{\sum_j y_{ij}^{t}}{N})$.

A complete summary of the features and their values is presented in Table 4.2.

The use of features based on a user's past observations to predict the future behavior of the individual is an instance of an observation-driven time-series modeling approach (which we discuss further in the related-work section below). In particular, this allows for an individual's behavior to change over time in a non-stationary fashion. For example some individuals could be permanently in exploration mode to the extent that their future behavior is always different to their past (in terms of specific item consumption). More typical is the case where future behavior is a combination of repeat and novel item consumption, to varying degrees across different individuals.

The particular set of features we use in this chapter (in Table 4.2) are quite general. More specific features could also be incorporated (e.g., depending on the specific semantics of the domain), including (potentially) static features that provide side-information about users and items, or exogenous time-varying features such as seasonality or calendar effects [60].

4.4.3 Regression Modeling of Mixture Parameters

Using the regression feature from Table 4.2 we model the parameters of the exposure and rate processes in the following way:

Exposure Process: The value of π_{ij}^t is estimated using *logistic regression*, conditioned on the globally-shared and the individual-specific intercept coefficients η_0 and η_{i0} respectively, as well as the individual-based feature coefficient vector $\boldsymbol{\eta}_u = \{\eta_{i1}, \eta_{i2}, \eta_{i3}, \eta_{i4}\}$. We denote the data-driven feature vector as $\boldsymbol{x}_{ij}^t = \{x_{ij}^{\bar{t}}, x_{ij}^t, x_j^{\bar{t}}, x_j^t\}$ and write the logistic function as:

$$\pi_{ij}^t = \frac{1}{1 + e^{-(\eta_0 x_0 + \eta_{i0} x_{i0} + \boldsymbol{\eta}_i \boldsymbol{x}_{ij}^t)}} \tag{4.3}$$

Rate Process: Similarly, the value of λ_{ij}^t is modeled via *Poisson regression* with a globally-shared and individual-specific coefficient β_0 and β_{i0}, as well as an individual-specific coeffi-

cient vector $\boldsymbol{\beta}_u = \{\beta_{i1}, \beta_{i2}, \beta_{i3}, \beta_{i4}\}$. In addition, as proposed in [27] and [42], we added an additional intercept ($x_{j0} = 1$) with an item-specific offset β_{j0} to accommodate heterogeneity across items.

The resulting Poisson regression model can be written as

$$\log \lambda_{ij}^t = \beta_0 x_0 + \beta_{i0} x_{i0} + \beta_{j0} x_{j0} + \boldsymbol{\beta}_i \boldsymbol{x}_{ij}^t \tag{4.4}$$

where the feature vector $\boldsymbol{x}_{ij}^t$ is defined in the same way as in Equation 4.3.

4.5 Learning Algorithms

Since the ZIP regression model can be expressed as a two-component mixture model, with $P_\lambda(y_{ij}^t|\lambda = 0)$ as the zero-inflation component, the model parameters can be estimated via a standard application of the Expectation-Maximization (EM) algorithm. EM is a general procedure for iterative optimization of a likelihood function with missing information. For mixture models the missing information for each data point y_{ij}^t is the identity of which component generated that data point. In particular, for ZIP mixtures this information is missing for all the zeros in the data set, $y_{ij}^t = 0$, since these data points could have been generated by either component. For values $y_{ij}^t > 0$ the data are unambiguously assigned to the rate component $P_\lambda(y_{ij}^t|\lambda_{ij}^t)$.

The E-step computes the membership probability (equivalent to the expected value of the binary membership indicator) for each data point $y_{ij}^t = 0$, conditioned on current estimates of the model parameters. The M-step generates maximum likelihood estimates of the parameters conditioned on the membership probabilities provided by the E-Step. Under fairly broad conditions, repeated application of E and M steps is guaranteed to converge to a (local) maximum of the likelihood function.

E-step: In the E-Step, for each of the zero-valued data points $y_{ij}^t = 0$, we compute the membership probability w_{ij}^t, namely the probability that this zero was generated by the rate component. These membership probabilities can be computed by applying Bayes rule to the definition of the mixture model above, $P_{zip}(y_{ij}^t = k|\pi_{ij}^t, \lambda_{ij}^t)$, where the parameters $\pi_{ij}^t, \lambda_{ij}^t$ are the current parameter estimates (from the most recent M-step or their initial values at the first iteration).

$$w_{ij}^t = \frac{\pi_{ij}^t P_\lambda(y_{ij}^t|\lambda_{ij}^t)}{(1-\pi_{ij}^t)P_\lambda(y_{ij}^t|\lambda=0) + \pi_{ij}^t P_\lambda(y_{ij}^t|\lambda_{ij}^t)} \tag{4.5}$$

Data points with membership weights closer to 1 are more likely (according to the current parameters) to have been generated by the rate component and, conversely, data with weights closer to 0 are more likely to have been generated by the zero-inflated component.

M-step: The M-step optimizes the parameters of the model conditioned on the current estimates of the w_{ij}^t membership values. Our ZIP model has two sets of parameters, the logistic regression parameters for the mixture weights $\boldsymbol{\eta} = \{\eta_0, \boldsymbol{\eta}_i\}$, and the rate parameters for the Poisson rate component in the mixture model, $\boldsymbol{\beta} = \{\beta_0, \boldsymbol{\beta}_j, \boldsymbol{\beta}_i\}$. The logistic regression uses the membership weights as targets and the Poisson regression uses weighted regression with the weights being the membership weights.

Neither the logistic or Poisson regression can be performed in closed-form, so we use gradient descent within each M-step to estimate the coefficients for each model. The gradients in both cases (logistic and Poisson) involve dense sums over all $N \times M \times T$ data values, where N, M and T are the number of users, items and time-windows respectively. This is in contrast to sparse estimation methods such as Poisson matrix factorization that can ignore the zeros in the data, effectively working with only a tiny fraction of the full data matrix for highly sparse data. Thus, in order to achieve a scalable algorithm, we use *stochastic gradient descent* (SGD) instead of full gradient methods, inspired by the success of SGD in training of large-

scale deep neural networks on large data sets. SGD approximates the exact gradient at each gradient update by estimating the gradient in a stochastic manner using a small randomly-selected subset of rows ("mini-batches") from the data matrix. We discuss the convergence of our EM + SGD method in more detail in Section 4.8 later in the chapter—at this point it is sufficient to note that our implementation is as fast (or faster) in wall-clock time when compared to publicly-available implementations of other competing approaches.

The step size in each SGD step was determined via the ADAM algorithm [39] which provides a systematic way of conditioning the step size on the level of confidence in the gradient. We found empirically that ADAM worked well for our SGD-based optimization problems (and that convergence could be difficult to attain without it) in agreement with work in deep learning literature where the combination of SGD and adaptive step-size (such as ADAM) is essential to the success of training models on large data sets.

One final note is that rather than maximizing the likelihood we maximized the likelihood times a prior, i.e., maximum a posteriori EM estimation. In log-space this corresponds to maximizing (in the M-step) the log-likelihood plus a regularization term corresponding to the log prior. In our experiments we found that empirically-determined MAP priors were particularly effective. To compute the empirical prior we trained the model using global coefficients (assuming all data belong to a single user) with L2 regularization. The learned coefficients were then used as a common prior for all users.

4.6 Related Work

The conceptual basis of our work builds from a rich literature in statistics on modeling count data [8]. For example, within the framework of generalized linear models the Poisson mean is modeled as $\exp(\sum \beta_k x_k)$ where the β_k's are regression coefficients and the x_k's are the

inputs to the model. In the context of longitudinal data (data across multiple individuals), it is common to use fixed and random effects to account for individual-level heterogeneity, e.g., by allowing for individual-specific intercept terms in the mean such as $\exp(\beta_i + \sum \beta_k x_k)$ where β_i is the offset for individual i (e.g., [18], ch.7). The incorporation of time-dependence into such models can typically be categorized into one of two general categories ([8], ch 7.2): *observation-driven* models where the counts are modeled directly as functions of past counts (such as autoregressive models for count data), or *parameter-driven* models where the counts depend on a latent state-space process (such as a hidden Markov model or a linear-Gaussian filter). The models we propose in this chapter are in the *observation-driven* category. The dynamic matrix factorization methods (discussed below) that we compare to in our experiments are in the *parameter-driven* category. The use of zero-inflation is also well-known in statistical modeling of count data (e.g., [44, 29]) and can be combined with other modeling components (as we do in our proposed approach) such as temporal dependence and fixed effects.

While our work builds on much of the above, a significant difference is that we model *high-dimensional count vectors* (i.e., a large number of items), orders of magnitude larger in dimensionality than the low-dimensional (often scalar) count data that is usually the primary focus in the statistical literature for modeling of count time-series. To handle the optimization challenges of parameter estimation for high-dimensional counts we use techniques from stochastic gradient optimization, which have not (to date at least) been used in statistics for count modeling.

Another significant line of related work is in matrix factorization of user-item consumption data. The most well-known approach in this context over the past decade has been the bilinear Gaussian model based on an SVD decomposition (e.g., [42]). The expected target

value y_{ij} is usually represented in such models as

$$E[y_{ij}] = \theta_i' \phi_j + \beta_0 + \beta_i + \beta_j, \tag{4.6}$$

where $\theta_i' \phi_j$ is the inner product of low-dimensional latent vector representations for user i and item j, and β_0, β_i, and β_j are constant, user, and item offsets respectively. In this framework the latent vectors θ_i and ϕ_j and parameters $\beta_0, \beta_i, \beta_j$ are typically estimated from the data using least-squares. This is equivalent to maximizing the likelihood of a Gaussian model for the y_{ij}'s (e.g., [62]). This is a useful approach for data that can be approximated by a symmetric distribution but is not ideal for the types of highly skewed count data we are focusing on in this chapter.

More recent work in matrix factorization has built on ideas from non-negative matrix factorization to develop models that are more appropriate for count data, e.g., where the expectation in Equation 4.6 above represents the mean of a Poisson model for the y_{ij}'s—known as Poisson matrix factorization (PMF). A typical approach is to estimate the parameters within a Bayesian framework (such as variational inference) and to place priors (such as Gamma priors) on the parameters θ_i and ϕ_j [27, 52].

Of particular relevance to this chapter is the recently-introduced dynamic Poisson matrix factorization model (DPMF) [10] which models the expected counts as a function of time t as:

$$E[y_{ij}^t] = \theta_{it}' \phi_{jt} + \dots$$

where t is a discrete time index (such as days, weeks, etc). Here the latent user and item vectors are allowed to evolve dynamically over time, such that predictions for time $t+1$ are functions of the latent vectors estimated at time t. This DPMF approach (and matrix factorization in general) can be viewed as an instantiation of a *parameter-driven* latent-space

model, in contrast to the *observation-driven* model that we pursue here.

Another recent strand of related work (in the non-dynamic PMF context) is the use of zero-inflated models in probabilistic matrix factorization. Liang et al. [51] proposed the framework of *exposure matrix factorization* (ExpoMF) which uses zero-inflation to explicitly account for exposure effects in matrix factorization of large binary user-item data sets. They found that ExpoMF systematically outperformed traditional PMF methods that did not account for exposure. In a similar vein, Jain et al. [33] developed a probabilistic matrix factorization framework with zero-inflation to handle exposure effects in multi-label classification with very large numbers of labels

Our work differs from all of the above work in matrix factorization, in terms of (a) prediction of consumption rates rather than ratings or binary data, (b) modeling both repeat and novel consumption over time, and (c) the use of user and item-specific regression models rather than matrix factorization.

There has also been recent work on *continuous-time modeling* of time-stamped user-item data, using Markov approaches [37, 38, 2], Poisson point processes [31], and neural networks [34]. While these papers share a common motivation with our work in terms of analyzing explore/exploit aspects of user consumption, the focus and methodologies are significantly different to what we pursue in this chapter, with less emphasis on user-item rate prediction and without the use of zero-inflation or regression models.

4.7 Experiments and Results

Below we describe the results of comparing the ZIP model to baselines and to a number of well-known approaches from the literature for modeling sparse user-item count data. For all of the experiments described below the model parameters were estimated using data from

times $t = 1$ to time $t = T - 2$, with hyperparameter tuning via grid search using data at time $t = T - 1$. The models were then evaluated on holdout test data at time $t = T$.

4.7.1 Performance Metrics

We evaluated our models using four different metrics.

Log-Loss: The log-loss is the sum of the negative log-probabilities of each user-item consumption rate in the test data:

$$-\log P = -\frac{1}{N_{te}} \sum_i \sum_j \log P(y_{ij}^T)$$

where $P(y_{ij}^T)$ is the probability of the observed count y_{ij}^T, under the model being evaluated, and where N_{te} is the total number of test points. The log-loss metric is widely used in the evaluation of machine learning algorithms that produce probabilistic predictions and is bounded below by zero. A model that assigns higher probability, or lower negative log-probability, to the observed test data is preferred over a model that assigns lower probability (or a higher negative log-probability).

Precision, Recall, and F1: Let $\hat{y}_{ij}^T$ denote the expected number of times (according to a particular model) that user i will consume item i during time-window T. For the ZIP model, by the linearity of expectation we have that $\hat{y}_{ij}^T = \hat{\pi}_{ij}^T \hat{\lambda}_{ij}^T$ where $\hat{\pi}_{ij}^T$ and $\hat{\lambda}_{ij}^T$ are point (MAP) parameter estimates learned by the model on the training data.

For each pair i, j we can compute the precision and recall of a prediction $\hat{y}_{ij}^T$, relative to the observed value y_{ij}^T, as follows. Precision can be defined in the context of count data as $\frac{\min\{y_{ij}^T, \hat{y}_{ij}^T\}}{\hat{y}_{ij}^T}$, i.e., it is the fraction of user-item consumptions that the model predicted would occur that actually did occur. Similarly, recall can be defined as $\frac{\min\{y_{ij}^T, \hat{y}_{ij}^T\}}{y_{ij}^T}$, which is the fraction of observed user-item consumptions that did occur that the model predicted

would occur. These pairwise user-item precision and recall values can be averaged over all user-item pairs to obtain overall precision and recall numbers. Models that systematically underestimate y_{ij}^T (e.g., that predict all zeros) will have high precision but low recall, and vice-versa for models that systematically overestimate y_{ij}^T. For our experimental results below we report the *F1* score, which combines both precision and recall, in the standard fashion as:

$$F1 = 2 \times \frac{\text{Prec} \times \text{Rec}}{\text{Prec} + \text{Rec}}$$

MSE: Mean squared error between the expected number of times that a user i will consume item i during time-window T and the observed value y_{ij}^T:

$$MSE = \frac{1}{N_{te}} \sum_i \sum_j \left(y_{ij}^T - \hat{y}_{ij}^T\right)^2$$

MAE: Mean absolute error between the expected number of times that a user i will consume item i during time-window T and the observed value y_{ij}^T:

$$MAE = \frac{1}{N_{te}} \sum_i \sum_j \left|y_{ij}^T - \hat{y}_{ij}^T\right|$$

4.7.2 Poisson Regression with and without Zero-Inflation

We first compare the ZIP Poisson regression model (**ZIP**) to a Poisson regression model (**PR**) without a zero-inflation component. We use the same features for both models. Table 4.3 shows that the ZIP model systematically outperforms over the PR model on holdout data, for all four metrics across all 3 data sets.

To further analyze the contribution of the zero-inflated component we evaluated each of

	Log-Loss		*F1*		*MSE*		*MAE*	
Dataset	PR	ZIP	PR	ZIP	PR	ZIP	PR	ZIP
reddit	0.287	**0.118**	0.73	**0.84**	167.56	**135.78**	0.38	**0.22**
lastfm	0.115	**0.045**	0.12	**0.34**	2.46	**1.68**	0.06	**0.03**
Yelp	0.092	**0.071**	0.09	**0.16**	**0.03**	**0.03**	0.06	**0.04**

Table 4.3: Log-Loss, F1 measure, MSE and MAE on the test data for the PR and ZIP models across different data sets. Lower values are better for Log-Loss, MSE and MAE and higher values better for F1. Best performing methods indicated in bold font.

the models in terms of their ability to predict the zeros. We focused on user-item pairs with $y_{ij}^T = 0$ and computed the Log-Loss for those pairs under each model. The Log-Loss values for the PR model are 0.064, 0.035 and 0.042 in the reddit, lastfm and Yelp data sets respectively. The corresponding values for the ZIP model are an order of magnitude lower: 0.004, 0.004 and 0.017. This significant improvement is directly attributable to the presence of the zero-inflation component in the ZIP model.

4.7.3 Comparing ZIP to Baselines and Matrix Factorization

Below we describe results obtained from comparing the ZIP model to a set of simple baselines and to several well-known approaches in the literature based on matrix factorization for count data.

GR (Global Rate): This is defined as the global rate at which each item is consumed in the training data, computed by averaging across all users and time-stamps:

$$\hat{\lambda}_{ij}^{GR} = \hat{\lambda}_j = \frac{1}{N}\frac{1}{T}\sum_{i=1}^{N}\sum_{\tau=1}^{T} y_{ij}^{\tau} \qquad 1 \leq j \leq M$$

MPE (Mean Posterior Estimate): This is the mean posterior estimate (MPE) of each user-item rate, with a conjugate Bayesian $Gamma(\gamma_0, \gamma_1)$ prior, based on the counts in the

training data:

$$\hat{\lambda}_{ij}^{MPE} = \frac{\sum_{\tau=1}^{T} y_{ij}^{\tau} + \gamma_0}{T + \gamma_1}$$

where γ_0 and γ_1 are determined via grid search to optimize the log-loss of the validation data for the MPE model.

PMF (Poisson Matrix Factorization): This is a latent factor matrix factorization model with a Poisson distribution for the observed counts. In our results we used a state-of-the-art Bayesian PMF version implementation by Liang et al. [52] The model was fit to the aggregated data across all time windows. At prediction time the predicted rates from the model were divided by the number of time-windows in the training data set to scale the rate for predictions for a single time-window.

$$\hat{\lambda}_{ij} = \frac{\theta_i' \phi_j}{T}$$

DPMF (Dynamic Poisson Matrix Factorization): This is an extension of the PMF model that learns latent-space decomposition from a sequence of user-item consumption counts [10]. The latent-space vectors for users and items are estimated for each time-window jointly by modeling the change in θ_i^t and ϕ_j^t between different time steps t using a Kalman filter.

Hyperparameters for both PMF and DPMF were determined via grid search on the validation data. One hyperparameter of particular interest is the number of factors used by both models. In Figure 4.3 we plot the *Log-Loss* computed on the validation data set using the PMF algorithm with different numbers of factors, fixing all other hyperparameters. For all data sets, the number of factors that resulted in the best predictive results on the validation data set was larger than $min(N, M)$ (normally, a value of $d << min(N, M)$ is selected).

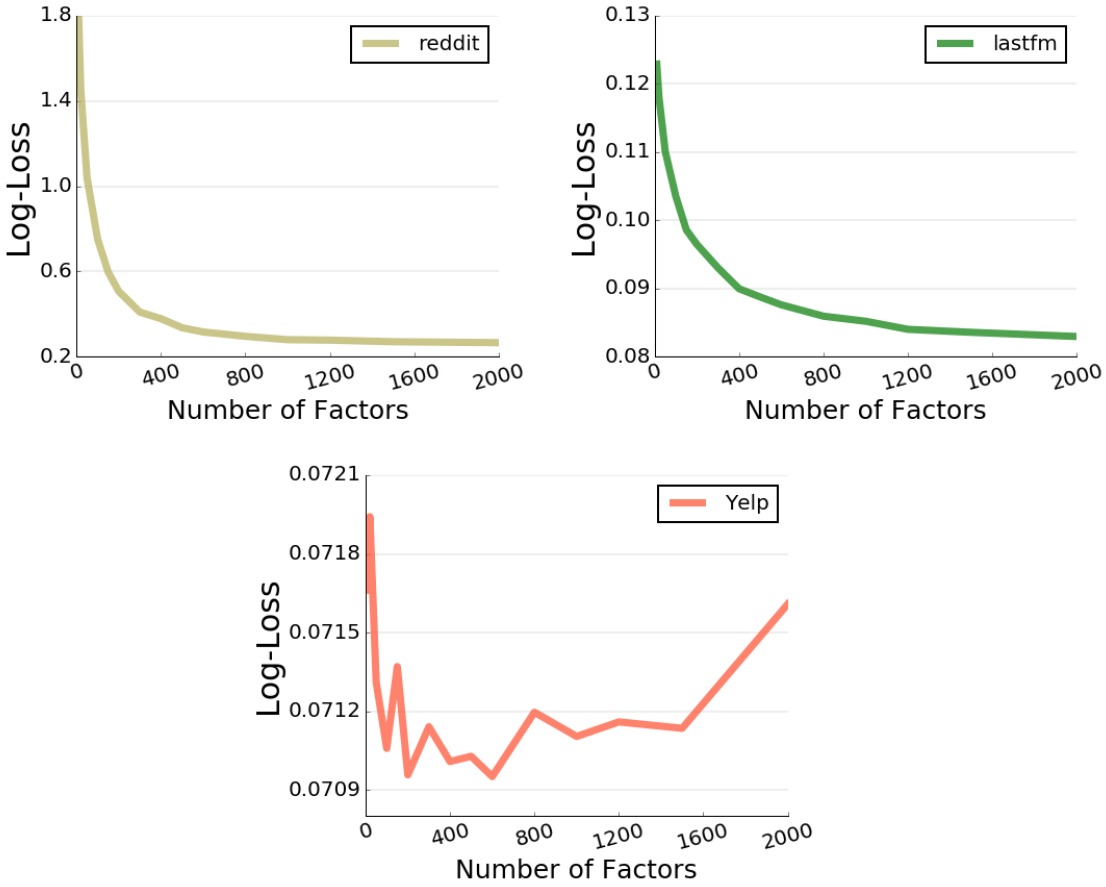


Figure 4.3: Log-Loss value on the validation data set for different number of factors across all data sets.

A possible explanation for the high number of factors is the fact that matrix factorization approximates a lower dimensional representation of users and items in order to generalize from past observation. This means that for PMF (and similarity for DPMF), using a low number of factors does not offer sufficient modeling power to capture the details of the target historical data. For data sets with a significant amount of repeat consumption, this is a considerable setback since a user's history is indicative of future behavior.

In keeping with standard matrix factorization methodology, we limit the number of factors we are willing to consider to a moderate range. For all data sets the number of factors used was in the range of 200 to 500.

Tables 4.4, 4.5 4.6 and 4.7 show the *Log-Loss*, *F1*, *MSE* and *MAE* scores on the test data,

	GR	MPE	PMF	DPMF	ZIP
reddit	2.978	0.233	0.332	0.812	**0.118**
lastfm	0.147	0.085	0.097	0.100	**0.045**
Yelp	0.075	0.090	0.077	0.077	**0.071**

Table 4.4: Log-Loss on the test data for different algorithms across different data sets. Lower scores are better. Best-performing methods indicated in bold font.

	GR	MPE	PMF	DPMF	ZIP
reddit	0.07	0.66	0.61	0.56	**0.84**
lastfm	0.03	0.27	0.15	0.21	**0.34**
Yelp	0.11	0.12	0.14	0.11	**0.16**

Table 4.5: F1-scores on the test data for different algorithms across different data sets. Higher scores are better. Best-performing methods indicated in bold font.

	GR	MPE	PMF	DPMF	ZIP
reddit	544.782	142.191	144.656	464.711	**135.782**
lastfm	1.825	2.809	2.152	1.831	**1.676**
Yelp	0.031	0.029	0.029	0.029	**0.028**

Table 4.6: MSE on the test data for different algorithms across different data sets. Lower scores are better. Best-performing methods indicated in bold font.

	GR	MPE	PMF	DMF	ZIP
reddit	0.975	0.339	0.339	0.462	**0.217**
lastfm	0.055	0.045	0.051	0.042	**0.034**
Yelp	0.040	**0.033**	0.037	0.046	0.036

Table 4.7: MAE on the test data for different algorithms across different data sets. Lower scores are better. Best-performing methods indicated in bold font.

for each of the baselines and PMF and DPMF models, compared to the ZIP model. The ZIP model is significantly more accurate than the other methods for all metrics for all data sets, except for the MAE score for the MPE model on the Yelp data set.

We further analyze the predictive performance of our model and compare in Figure 4.4 the ZIP model *Log-Loss* and the best performing alternative approaches when looking at each individual separately. We performed a Wilcoxon signed rank test on the results and found that the ZIP model dominates all other methods with a p-value of $p < 0.001$ across all data sets.

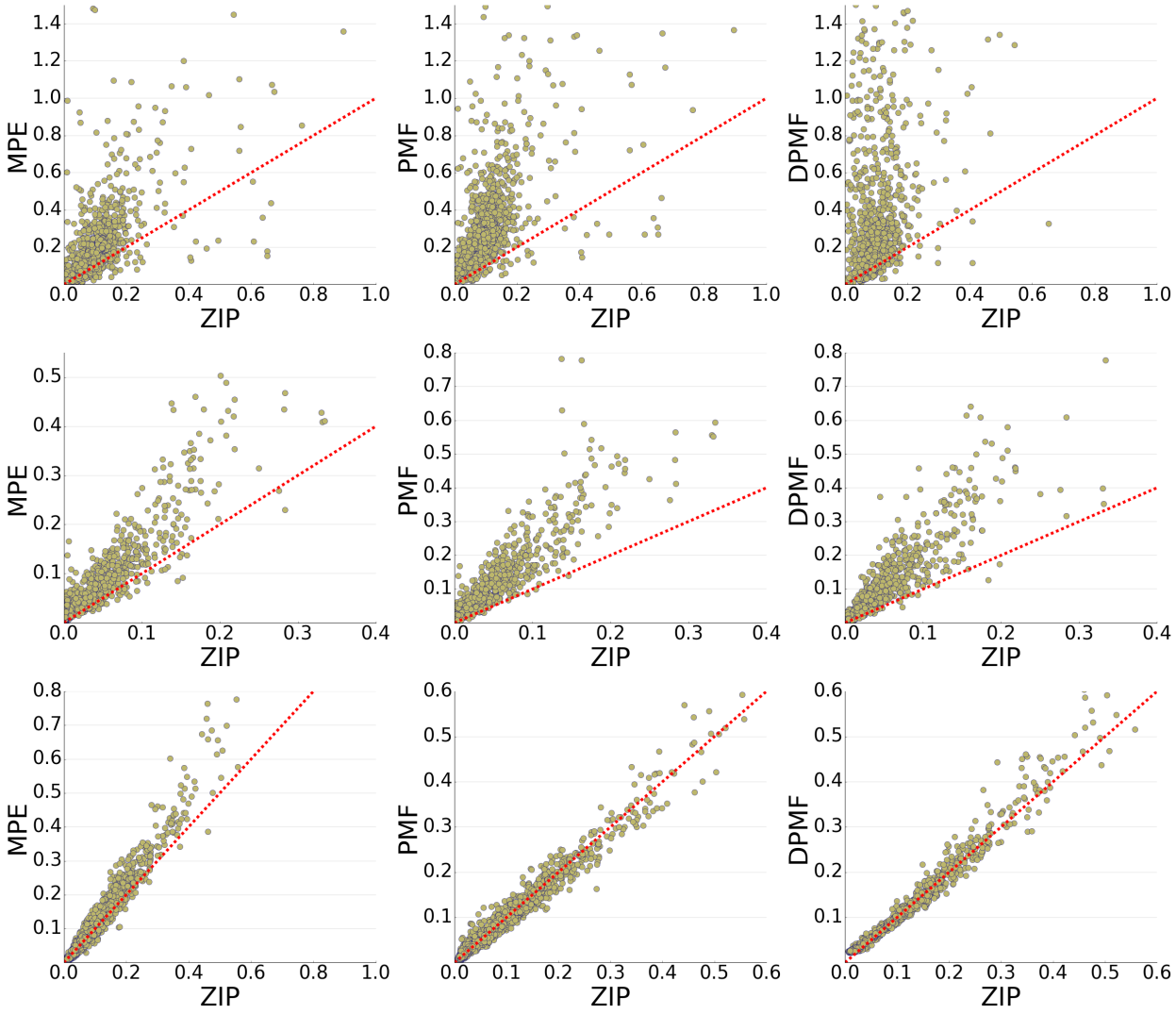


Figure 4.4: Scatter plots comparing ZIP with MPE (left column), PMF (middle column) and DPMF (right column) test *Log-Loss* scores for individuals in the reddit (top row) lastfm (middle row) and Yelp (bottom row) data sets.

For both the reddit and lastfm data sets the margins of improvement of the ZIP model over both the PMF and DPMF models are quite large (shown in Table 4.8). There are two likely reasons for this improvement. The first is that the zero-inflation component in the ZIP model provides a more flexible way to handle excess zeros than PMF or DPMF. The second reason is that the user-specific features in the regression approach (such as the history variable of what specific items a user consumed in the past) allows the regression model to more accurately model individual-level details than the matrix factorization (MF) approaches.

	ZIP / PMF				ZIP / DPMF			
Data Set	*Log-Loss*	*F1*	*MSE*	*MAE*	*Log-Loss*	*F1*	*MSE*	*MAE*
reddit	0.36	1.38	0.93	0.64	0.15	1.50	0.29	0.47
lastfm	0.48	2.27	0.78	0.67	0.45	1.62	0.92	0.81

Table 4.8: Relative Log-Loss, F1, MSE and MAE in percentage on the test data for the ZIP model compared to PMF and DPMF across different data sets. Lower values are better for Log-Loss, MSE and MAE and higher values better for F1.

The MF approaches are constrained by the dimensionality of their latent spaces, limiting the level of detail (e.g., specific combinations of items) available for modeling individual users. We explore both of these in more detail below.

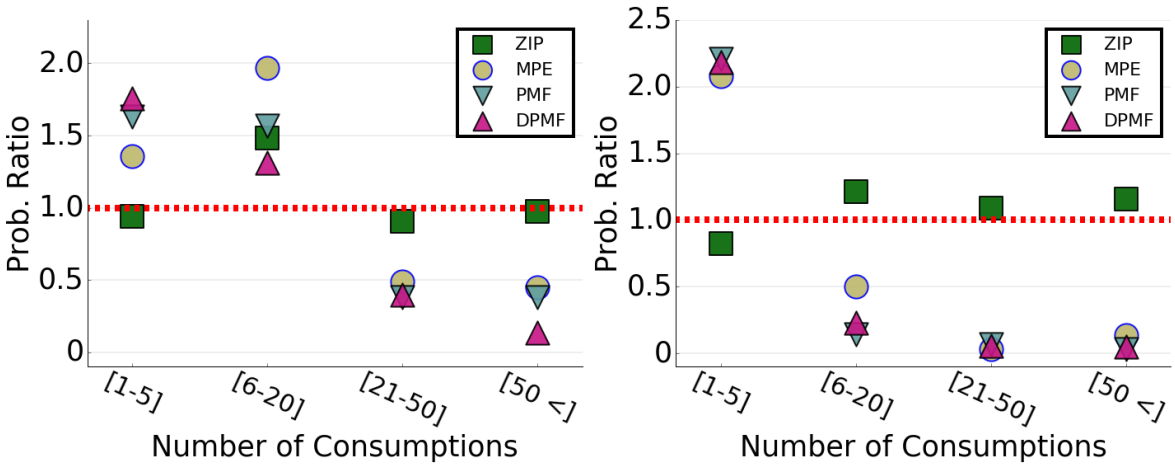


Figure 4.5: The ratio of probability for the number of consumptions for selected users from reddit (left) and lastfm (right) at time T assigned by the evaluated models compared to the ground truth (GT). Best viewed in color.

Modeling excess zeros: Modeling the zeros provides the ZIP model with a principled way of down-weighting the 0's in the process of learning the rate parameter. As a result the rate-process coefficients are free to fit a larger range of y_{ij}^T values (in particular, high numbers of consumptions). In Figure 4.5 we plot the ratio between (a) the predicted number of $\hat{y}_{ij}^T$ counts, across different ranges of y, for the different models, and (b) the ground truth number for those values in the test data. These plots are for users with high variance in their y_{ij}^T values, for both the reddit (left) and lastfm (right) data sets. We can see that in order to fit the excess of 0's, the baselines tend to systematically and significantly overestimate the

low rates and to underestimate the high rates, relative to ground truth—while the proposed ZIP regression model (green squares) tends to be much more accurate.

Balancing Explore-Exploit: By separately modeling the *exposure* and *rate* processes, the ZIP model is able to capture the heterogeneity across users in terms of their *explore/exploit* behavior. In particular, the exposure process coefficient η_{i0} corresponds to the estimated user-specific budget and captures the number of unique items a user will consume. As η_{i0} increases, the probability that a user will be exposed to an item and consume it is also predicted to increase, corresponding to a higher predicted tendency for *exploration*. To illustrate this, in Figure 4.6 we plot the correlation between the value of η_{i0} and the number of unique items the user consumed in the lastfm (left) and reddit (right) data sets. The clear positive correlation between the two (shown as the regression line in red) demonstrates the ability of our model to achieve an appropriate balance between *exploration* and *exploitation*, resulting in better individual-level predictive models. In addition, the individual-level coefficients provide interpretable detail about each specific user, quantifying their individual tendency for exploration within the context of a broad rate prediction model.

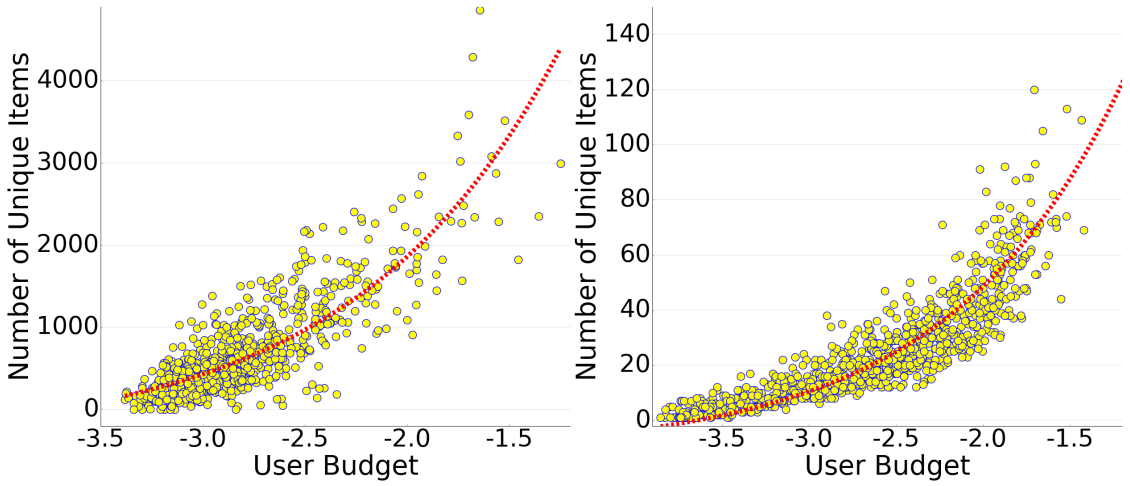


Figure 4.6: Number of unique items each user consumed as a function of the user-specific budget coefficient (η_{i0}) in the *lastfm* (left) and *reddit* (right) data sets. The red line indicates an exponential curve fitted to the scatter plot.

4.7.4 Online Predictions

One advantage of using *observation-driven* models is that it is natural to make predictions at multiple discrete time points t. For each new time t we simply use the learned parameters $\boldsymbol{\eta}$ and $\boldsymbol{\beta}$ to make predictions based on the data features at time $t-1$ (i.e: no re-training of the model is needed for time t). This is in contrast to steady-state *parameter-driven* models where without additional training, new data will not be taken into consideration at prediction time.

This is particularly useful in applications in which the behavior patterns observed from the data are especially dynamic but the user-specific preferences for *explore* and *exploit* are relatively fixed. As a simple example, consider a user that consistently explores new items without consuming an item more than once. For such a user, the explore preference is fixed (roughly speaking) whereas the identity of consumed items changes with each consumption. Approaches such as PMF and DPMF will require frequent retraining in order to reflect the constant change of consumed items, a process which can be prohibitively expensive for real-world applications.

For an alternative, more realistic example in the context of applications with repeated consumption, consider a user with a relatively high preference for exploiting known items. At time t' the user is introduced to a new item (e.g. new music artist) and starts to exploit it (i.e. engage in repeated consumption). *Observation-driven* models are capable of reflecting the interest in the new item quickly, as future predictions are derived directly from the data. However, it is possible that *parameter-driven* models will require multiple observations of the new item consumption in order to represent the new interest in the model's parameters.

To illustrate the advantage of using an *observation-driven* model for predicting user-item consumption, we train all models on the first $T-10$ time windows and plot the *Log-Loss* results for each time-window t in the last 10 time-windows without additional training in

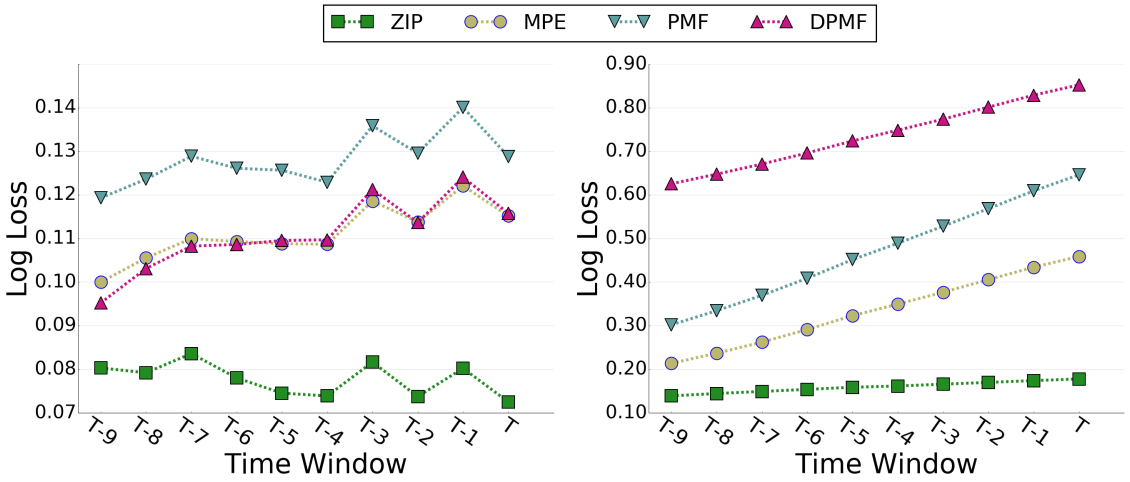


Figure 4.7: Log-Loss reported by the different models at each time-window in the lastfm (left) and reddit (right) test data sets. The Yelp data set only has 10 time-windows and was omitted from this experiment. Best viewed in color.

Figure 4.7. The plot clearly shows that the ZIP model predictive accuracy stays roughly the same over time, while the predictive accuracy of the baseline decreases over time. This indicates that the use of an *observation-driven* model can better represent dynamic user-consumption behavior. In practice, it would be possible to re-estimate the model parameter at pre-determined intervals or when accuracy falls below a certain threshold.

4.8 Scalability

In fitting our regression models our dense "data matrix" can be thought of as having $N \times M \times T$ rows and d columns (for the features), where d is the number of coefficients in the regression, N, M and T are the number of users, items and time-windows respectively. Thus, direct gradient optimization would be $\mathcal{O}(dNMT)$ per gradient computation. Using SGD, the time complexity of a single gradient step is $\mathcal{O}(d \times R)$ where R is the minibatch size. If we think of NMT as the effective total number of rows in the full data set, then to gain the benefits of SGD we need to select R such that $R << NMT$. In typical applications of SGD with dense data the minibatch size can be quite small, e.g., $R = 10, R = 100$. However,

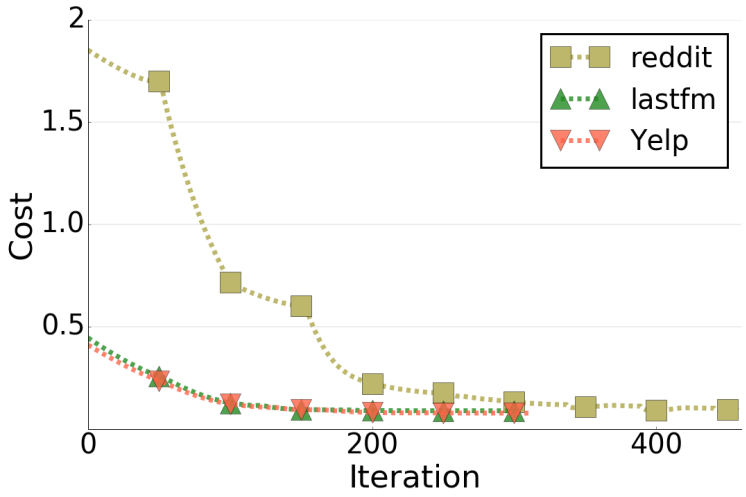


Figure 4.8: Optimization cost value (negative log-Likelihood) at each SGD iteration for each dataset. Markers denote the point in the iterative process where E-steps were performed. Best viewed in color.

with highly skewed data (as in the user-item data sets of interest here), the minibatch sizes need to be significantly larger to ensure that there are enough non-zeros in each minibatch. We found that a minibatch size of $R = 50,000$ worked well in terms of relatively fast and reliable convergence. The time complexity of a single E-step is $\mathcal{O}(d \times N \times M \times T)$, i.e., proportional to the number of rows, making it the most expensive part in the algorithm in terms of time complexity. It is possible that some efficiency could be gained here via an approximate E-step but we did not investigate this here given that we execute far more gradient steps (within the M-step) than E-steps.

Figure 4.8 shows convergence plots, where the y-axis is the cost function (Log-Loss) at each iteration. Each iteration on the x-axis marks a single stochastic gradient (minibatch) step and the markers indicate the point in the algorithm where E-steps occurred (each M-step consists of multiple gradient steps). We see from the convergence plots that the algorithm converges quickly for each of the three data sets in our experiments. We implemented our algorithm in Python (with Cython to speed-up)[4]. Each mini-batch iteration in our implementation ran in a matter of few milliseconds and the relatively expensive E-step took

[4] `https://github.com/MosheLichman/ZIP-Regression`

7, 62, and 1 seconds on average for the reddit, lastfm, and Yelp data sets respectively. Our implementation used a single core — it is relatively straightforward to distribute the computation of the gradients and membership weights by using multiple cores, rendering the algorithm scalable to much larger data sets than what we used here.

4.9 Conclusions

The primary contributions of this chapter include the following:

- We explored individual-level user-item consumption rate data sets where user consumption is a mix of repeat (exploit) and novel (explore) behavior over time.
- We introduced a framework using a zero-inflated Poisson regression for prediction of consumption rates in high-dimensional user-item consumption data sets.
- We evaluated our model with batch and online schemes using multiple evaluation metrics showing results on the reddit, lastfm and Yelp data sets. The results showed that our model outperforms existing approaches for individual-level user-item consumption modeling in both the batch and online prediction scenarios.

Bibliography

[1] J. R. Bellegarda. Statistical language model adaptation: review and perspectives. *Speech Communication*, 42(1):93–108, 2004.

[2] A. R. Benson, R. Kumar, and A. Tomkins. Modeling User Consumption Sequences. In *Proceedings of the 25th International Conference on World Wide Web*, pages 519–529. International World Wide Web Conferences Steering Committee, 2016.

[3] J. F. Bithell. An application of density estimation to geographical epidemiology. *Statistics in Medicine*, 9(6):691–701, 1990.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[5] J. E. Blumenstock, R. Chokkalingam, V. Gaikwad, and S. Kondepudi. Probabilistic Inference of Unknown Locations: Exploiting Collective Behavior when Individual Data is Scarce. In *Proceedings of the Fifth ACM Symposium on Computing for Development*, pages 103–112. ACM, 2014.

[6] L. Breiman, W. Meisel, and E. Purcell. Variable Kernel Estimates of Multivariate Densities. *Technometrics*, 19(2):135–144, 1977.

[7] D. Brockmann, L. Hufnagel, and T. Geisel. The scaling laws of human travel. *Nature*, 439:462–465, 2006.

[8] A. C. Cameron and P. K. Trivedi. *Regression Analysis of Count Data*. Cambridge University Press, 2013.

[9] J. Chang and E. Sun. Location3: How Users Share and Respond to Location-Based Data on Social Networking Sites. In *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, pages 74–80, 2011.

[10] L. Charlin, R. Ranganath, J. McInerney, and D. M. Blei. Dynamic Poisson Factorization. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 155–162. ACM, 2015.

[11] J. Chen, C. Wang, and J. Wang. Will You "Reconsume" the Near Past? Fast Prediction on Short-Term Reconsumption Behaviors. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 23–29, 2015.

[12] C. Cheng, H. Yang, I. King, and M. R. Lyu. Fused Matrix Factorization with Geographical and Social Influence in Location-Based Social Networks. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[13] E. Cho, S. A. Myers, and J. 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*, pages 1082–1090. ACM, 2011.

[14] J. Cranshaw, R. Schwartz, J. I. Hong, and N. Sadeh. The Livehoods Project: Utilizing Social Media to Understand the Dynamics of a City. In *Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media*, pages 58–65, 2012.

[15] J. Cranshaw and T. Yano. Seeing a home away from the home: Distilling proto-neighborhoods from incidental data with latent topic modeling. In *Proceedings of NIPS Workshop of Computational Social Science and the Wisdom of the Crowds*, 2010.

[16] A. Culotta. Estimating County Health Statistics with Twitter. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, pages 1335–1344. ACM, 2014.

[17] N. D. Daw, J. P. O'doherty, P. Dayan, B. Seymour, and R. J. Dolan. Cortical substrates for exploratory decisions in humans. *Nature*, 441:876–879, 2006.

[18] P. Diggle, P. Heagerty, K.-Y. Liang, and S. Zeger. *Analysis of Longitudinal Data*. Oxford University Press, 2002.

[19] N. Donthu and R. T. Rust. Estimating Geographic Customer Densities Using Kernel Density Estimation. *Marketing Science*, 8(2):191–203, 1989.

[20] N. Eagle and A. S. Pentland. Eigenbehaviors: identifying structure in routine. *Behavioral Ecology and Sociobiology*, 63(7):1057–1066, 2009.

[21] Federal Trade Commission. 2006 Identity Theft Survey Report. https://www.ftc.gov/os/2007/11/SynovateFinalReportIDTheft2006.pdf, 2007.

[22] J. Fieberg. Kernel density estimators of home range: smoothing and the autocorrelation red herring. *Ecology*, 88(4):1059–1066, 2007.

[23] A. Fowler, K. Partridge, C. Chelba, X. Bi, T. Ouyang, and S. Zhai. Effects of Language Modeling and Its Personalization on Touchscreen Typing Performance. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 649–658. ACM, 2015.

[24] V. Frias-Martinez, V. Soto, H. Hohwald, and E. Frias-Martinez. Characterizing Urban Landscapes Using Geolocated Tweets. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 239–248, 2012.

[25] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis*, volume 2. CRC press Boca Raton, FL, 2014.

[26] M. C. Gonzlez, C. A. Hidalgo, and A.-L. Barabsi. Understanding individual human mobility patterns. *Nature*, 453:779–782, 2008.

[27] P. Gopalan, J. M. Hofman, and D. M. Blei. Scalable recommendation with hierarchical Poisson factorization. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pages 326–335, 2015.

[28] A. G. Gray and A. W. Moore. Nonparametric Density Estimation: Toward Computational Tractability. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 203–211, 2003.

[29] D. B. Hall. Zero-inflated Poisson and binomial regression with random effects: a case study. *Biometrics*, 56(4):1030–1039, 2000.

[30] S. Hasan, X. Zhan, and S. V. Ukkusuri. Understanding Urban Human Activity and Mobility Patterns Using Large-scale Location-based Data from Online Social Media. In *Proceedings of the 2Nd ACM SIGKDD International Workshop on Urban Computing*, pages 6:1–6:8. ACM, 2013.

[31] S. A. Hosseini, K. Alizadeh, A. Khodadadi, A. Arabzadeh, M. Farajtabar, H. Zha, and H. R. Rabiee. Recurrent Poisson Factorization for Temporal Recommendation. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017.

[32] B. Hu and M. Ester. Spatial Topic Modeling in Online Social Media for Location Recommendation. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 25–32. ACM, 2013.

[33] V. Jain, N. Modhe, and P. Rai. Scalable Generative Models for Multi-label Learning with Missing Labels. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1636–1644, 2017.

[34] H. Jing and A. J. Smola. Neural Survival Recommender. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 515–524. ACM, 2017.

[35] Y. Kanza, E. Kravi, and U. Motchan. City Nexus: Discovering Pairs of Jointly-visited Locations Based on Geo-tagged Posts in Social Networks. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 597–600. ACM, 2014.

[36] B. Kapicioglu, D. S. Rosenberg, R. E. Schapire, and T. Jebara. Collaborative Place Models. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 3612–3618, 2015.

[37] K. Kapoor, V. Kumar, L. Terveen, J. A. Konstan, and P. Schrater. "I Like to Explore Sometime": Adapting to Dynamic User Novelty Preferences. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 19–26. ACM, 2015.

[38] K. Kapoor, K. Subbian, J. Srivastava, and P. Schrater. Just in Time Recommendations: Modeling the Dynamics of Boredom in Activity Streams. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 233–242. ACM, 2015.

[39] D. Kinga and J. B. Adam. A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[40] S. Kinsella, V. Murdock, and N. O'Hare. "I'm Eating a Sandwich in Glasgow": Modeling Locations with Tweets. In *Proceedings of the 3rd International Workshop on Search and Mining User-generated Contents*, pages 61–68. ACM, 2011.

[41] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th International Conference on Machine Learning*, pages 315–322, 2002.

[42] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, 2009.

[43] J. Krumm and E. Horvitz. Eyewitness: Identifying Local Events via Space-time Signals in Twitter feeds. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 20:1–20:10. ACM, 2015.

[44] D. Lambert. Zero-Inflated Poisson Regression, With an Application to Defects in Manufacturing. *Technometrics*, 34(1):1–14, 1992.

[45] T. K. Landauer and S. T. Dumais. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211–240, 1997.

[46] R. Lee, S. Wakamiya, and K. Sumiya. Urban area characterization based on crowd behavioral lifelogs over Twitter. *Personal and Ubiquitous Computing*, 17(4):605–620, 2013.

[47] X. Li, G. Cong, X.-L. Li, T.-A. N. Pham, and S. Krishnaswamy. Rank-GeoFM: A Ranking Based Geographical Factorization Method for Point of Interest Recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 433–442. ACM, 2015.

[48] X. Li, Y.-Y. Wang, and A. Acero. Learning Query Intent from Regularized Click Graphs. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 339–346. ACM, 2008.

[49] Z. Li, B. Ding, J. Han, R. Kays, and P. Nye. Mining Periodic Behaviors for Moving Objects. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1099–1108. ACM, 2010.

[50] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui. GeoMF: Joint Geographical Modeling and Matrix Factorization for Point-of-interest Recommendation. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 831–840. ACM, 2014.

[51] D. Liang, L. Charlin, J. McInerney, and D. M. Blei. Modeling User Exposure in Recommendation. In *Proceedings of the 25th International Conference on World Wide Web*, pages 951–961. International World Wide Web Conferences Steering Committee, 2016.

[52] D. Liang, J. W. Paisley, D. Ellis, et al. Codebook-based scalable music tagging with Poisson matrix factorization. In *Proceedings of the Fifteenth International Society for Music Information Retrieval Conference*, pages 167–172, 2014.

[53] B. Liu, Y. Fu, Z. Yao, and H. Xiong. Learning Geographical Preferences for Point-of-interest Recommendation. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1043–1051. ACM, 2013.

[54] B. Liu, H. Xiong, S. Papadimitriou, Y. Fu, and Z. Yao. A General Geographical Probabilistic Factor Model for Point of Interest Recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1167–1179, 2015.

[55] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT press, 2012.

[56] R. Rejaie, M. Handley, H. Yu, and D. Estrin. Proxy caching mechanism for multimedia playback streams in the internet. In *Proceedings of the 4th International Web Caching Workshop*, 1999.

[57] M. Ridout, C. G. Demétrio, and J. Hinde. Models for count data with many zeros. In *Proceedings of the XIXth International Biometric Conference*, pages 179–192, 1998.

[58] A. Sadilek, H. A. Kautz, and V. Silenzio. Modeling Spread of Disease from Social Interactions. In *Proceedings of the Sixth AAAI International Conference on Weblogs and Social Media*, pages 322–329, 2012.

[59] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell. NextPlace: A Spatio-temporal Prediction Framework for Pervasive Systems. In *Proceedings of the 9th International Conference on Pervasive Computing*, pages 152–169. Springer Berlin Heidelberg, 2011.

[60] M. W. Seeger, D. Salinas, and V. Flunkert. Bayesian Intermittent Demand Forecasting for Large Inventories. In *Advances in Neural Information Processing Systems 29*, pages 4646–4654. Curran Associates, Inc., 2016.

[61] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

[62] A. P. Singh and G. J. Gordon. A unified view of matrix factorization models. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 358–373. Springer Berlin Heidelberg, 2008.

[63] P. Smyth. Model selection for probabilistic clustering using cross-validated likelihood. *Statistics and Computing*, 10(1):63–72, 2000.

[64] P. Smyth and D. Wolpert. Linearly combining density estimators via stacking. *Machine Learning*, 36(1-2):59–83, 1999.

[65] L. Song, D. Kotz, R. Jain, and X. He. Evaluating Next-Cell Predictors with Extensive Wi-Fi Mobility Data. *IEEE Transactions on Mobile Computing*, 5(12):1633–1649, 2006.

[66] S. J. Vaughan-Nichols. Will Mobile Computing's Future Be Location, Location, Location? *Computer*, 42(2):14–17, 2009.

[67] H. Wang, M. Terrovitis, and N. Mamoulis. Location Recommendation in Location-based Social Networks Using User Check-in Data. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 374–383. ACM, 2013.

[68] Y. Wang, N. J. Yuan, D. Lian, L. Xu, X. Xie, E. Chen, and Y. Rui. Regularity and Conformity: Location Prediction Using Heterogeneous Mobility Data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1275–1284. ACM, 2015.

[69] R. C. Wilson, A. Geana, J. M. White, E. A. Ludvig, and J. D. Cohen. Humans use directed and random exploration to solve the explore–exploit dilemma. *Journal of Experimental Psychology: General*, 143(6):2074–2081, 2014.

[70] J.-D. Zhang and C.-Y. Chow. iGSLR: Personalized Geo-social Location Recommendation: A Kernel Density Estimation Approach. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 334–343. ACM, 2013.

[71] J.-D. Zhang and C.-Y. Chow. GeoSoCa: Exploiting Geographical, Social and Categorical Correlations for Point-of-Interest Recommendations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 443–452. ACM, 2015.

[72] J.-D. Zhang, C.-Y. Chow, and Y. Li. LORE: Exploiting Sequential Influence for Location Recommendations. In *Proceedings of the 22Nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 103–112. ACM, 2014.

[73] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining Interesting Locations and Travel Sequences from GPS Trajectories. In *Proceedings of the 18th International Conference on World Wide Web*, pages 791–800. ACM, 2009.

[74] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and P. B. Schölkopf. Learning with Local and Global Consistency. In *Advances in Neural Information Processing Systems 16*, pages 321–328. MIT Press, 2003.