Learning to Recommend with Location and Context

CHENG, Chen

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
September 2015

Thesis Assessment Committee

Professor CHAN Lai Wan (Chair)

Professor LYU Rung Tsong Michael (Thesis Supervisor)

Professor KING Kuo Chin Irwin (Thesis Co-supervisor)

Professor LAM Wai (Committee Member)

Professor Li Qing (External Examiner)

Abstract of thesis entitled:

Learning to Recommend with Location and Context

Submitted by CHENG, Chen

for the degree of Doctor of Philosophy

at The Chinese University of Hong Kong in September 2015

In the past decade, location-based social networks (LBSNs), such as Gowalla and Foursquare, have attracted millions of users to share their locations via check-in behaviors. Point-of-interest (POI) recommendation has been a significant task in LBSNs since it can help targeted users explore their surroundings as well as help third-party developers provide personalized services. Different from traditional recommender systems in music and movie, POI recommendation needs to consider the geographical influence, which has a great impact on users' choices. Apart from geo-information from locations, plenty of other auxiliary information is also available. All these auxiliary information are referred to as context. Recommendation with location and context brings new challenges for recommender systems and new methods need to be tailored for it. In this thesis, the challenges from three perspectives motivated by real life problems will be addressed.

Firstly, we consider the point-of-interest recommendation task in LBSNs. In this task, we try to recommend the potentially attractive while unvisited POIs to users. By carefully studying the users' moving patterns, we find that users tend to check in around several centers and different users have different number of centers. Based on this finding, we propose a novel Multi-center Gaussian Model to capture this pattern. Moreover, we consider the fact that users are usually more interested in the top 10 or 20 recommendations, which makes personalized ranking important in this task. To consider users' preferences, geographical influence and personalized ranking, we propose a unified POI recommendation framework which fuses them together.

Secondly, we consider the task of successive point-of-interest recommendation in LBSNs. Different from the first task, we would like to provide recommendations for users in the next few hours based on their current locations and previous check-in histories. To solve this task, we develop two novel matrix factorization models based on two prominent properties observed in the check-in sequence: personalized Markov chain and region localization.

Lastly, we explore the problem of context-aware recommendation in recommender systems. Most context-aware recommendation methods model pairwise interactions between all features, while in practice, not all the pairwise feature interactions are useful. Thus, it is challenging to select "good" interacting features effectively. To address this challenge, we propose a novel Gradient Boosting Factorization Machines (GBFM) model to incorporate the feature selection algorithm with Factorization Machines into a unified framework.

In summary, we propose several tailored methods for recommendation with location and context in this thesis. Extensive experiments on real life large-scale datasets confirm the effectiveness and efficiency of proposed methods.

論文題目 ：基於地點和文本的推薦學習

作者 ：程陳

學校 ：香港中文大學

學習 ：計算機科學與工程學習

修讀學位 ：哲學博士

摘要 ：

在過去十年裡，基於地點的社交網絡（LBSNs）比如 Gowalla，Foursquare 這些，已經吸引了上百萬的用戶通過簽到來分享他們的社交朋友圈和地理位置信息。興趣點(POI)推薦是 LBSNs 裡面的一個重要的任務，因為它可以幫助特定用戶來拓展他們的周邊，同時也可以幫助第三方開發者提供特定的個性化服務。和在音樂，電影上的傳統推薦系統不同，興趣點推薦需要考慮地點的地理信息，地理對用戶的選擇有很大的影響。除了地點里的地理信息，大量其他的輔助信息也可以得到。我們稱所有這些輔助信息為文本。基於地點和文本的推薦給推薦系統帶來了新的挑戰，新的方法需要針對它特別設計。在這篇論文中，激發與真實的現實問題，我們從三個方面來強調這些挑戰。

首先，我們考慮在 LBSNs 裡面的興趣點推薦這個任務。在這個任務中，我們嘗試推薦給用戶他們可能感興趣但還沒有去過的地點。通過仔細的研究用戶的移動模式，我們發現用戶傾向於在幾個中心周圍簽到，不同的用戶的簽到中心數通常不同。基於這個發現，我們提出一個新穎的多中心高斯模型來抓住這個模式。我們進一步考慮到用戶通常對前 10 或者 20 的推薦更感興趣，直接優化個性化的

個性化排序會得到更好的結果。為了考慮用戶的個人偏好，地理的影響和個性化推薦，我們提出了一個統一的興趣點推薦的框架把它們都融合在一起。

其次，我們考慮 LBSNs 裡面下個興趣點推薦的任務。在第一個任務中，我們把簽到作為整體來考慮，忽略了它們的時序的關係和影響。不同於第一個任務，這裡我們希望基於用戶當前的位置和之前的簽到歷史，對用戶接下來幾個小時可能去的地方進行推薦。為了解決這個任務，我們發現了簽到序列裡面兩個突出的性質：個性化的馬爾科夫鏈和區域的局部性。我們基於這兩個性質提出了兩個新的矩陣分解的方法。

最後，我們探索了推薦系統裡面文本敏感的推薦問題。大多數文本敏感推薦模型對所有的特征進行兩兩交互建模。實際上並不是所有的特征交互都有用。因而，選出有效的好的特征非常具有挑戰性。基於這個挑戰，我們提出了一個新穎的智能因子分解機（GBFM）的模型來把特征選擇算法和矩陣分解機融合進一個統一的框架下。

綜上所述，我們在這篇論文中對基於地點和文本的推薦提出了幾個精心設計的方法。大量在實際生活中大數據上面的實驗驗證了我們提出方法的正確性和有效性。

Acknowledgement

I would like to express my sincere gratitude and appreciation to the people who assisted me in the research presented in this thesis. First and foremost, I would like to thank my supervisors, Prof. Irwin King and Prof. Michael R. Lyu, for their great guidance and encouragement during my Ph.D. study at CUHK. I have learned so much from their guidance not only on knowledge, but also the right attitude to conduct high quality research. They also help me improve presentation skills as well as English writing skills. When I needed help, they always point out the right directions, which mean a lot to me. It is a great honor for me to have them as my superviors.

I am grateful to my thesis committee members, Prof. Wai Lam and Prof. Lai Wan Chan, for their helpful comments and suggestions about this thesis. My special thanks to Prof. Qing Li who kindly served as the external committee for this thesis.

I would like to thank my mentors, Dr. Fen Xia and Prof. Tong Zhang for their guidance, support and valuable suggestions during my internship at Baidu.

My four years studying at CUHK has been a remarkable experience for me. I would like to thank Dr. Haiqin Yang, Dr. Xin Xin, Dr. Yangfan Zhou and Dr. Chao Zhou for their efforts and constructive discussions in conducting the research work in this thesis. I enjoyed my staying with all my talented colleagues, Yu Kang, Qirun Zhang, Shouyuan Chen, Guang Ling, Tong Zhao, Hongyi Zhang, Jieming Zhu, Shenglin Zhao, Xixian Chen, Pinjia He and Negin Rahimi. Life would not be nearly as wonderful without these fantastic officemates. I also would like to thank Xin Zhuang, Si Shen and Jingdong Chen for their discussions during my internship at Baidu.

Last but not least, I would like to thank my parents and my grandma for their deep love and tireless support for me. Without them, this thesis would never have been completed.

To my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the rapid development of online e-commerce, music and movie websites, *Recommender Systems* are becoming more and more important since they focus on solving the information overload problem [91]. People often have trouble making decisions with too much information around. For example, we often need to decide which music to listen to, which movie to watch or which clothes to buy. Recommender systems can help alleviate this problem by making personalized suggestions that are likely to attract specific users. Typically, recommender systems are based on collaborative filtering (CF) techniques, which provide recommendations from similar users or items. CF is based on the assumption that similar users may have similar taste on the same item, and similar items may receive similar ratings from the same user [83]. It has been widely deployed and received great success in commercial websites such as Amazon[1] [118] as well as recommendation competitions such as Netflix prize[2] [62].

In the past decade, recommendation techniques have received much attention in both industry and academic communities [3, 112, 61, 114, 64] due to the importance of recommender systems to people's life as well as their potential commercial value. However, in traditional recommender systems, most collaborative filtering techniques focus on the user-item rating matrix only. Although they have

[1] http://www.amazon.com
[2] http://www.netflixprize.com

been widely adopted, the recommendation results may be inaccurate due to the data sparsity of the user-item matrix. On the other hand, due to the rapid development of mobile devices and ubiquitous Internet access, location information and other context information can be easily gathered, which can be utilized to improve recommendation performance.

Location and context information bring new challenges to recommendation techniques. In terms of location, location-based social networks (LBSNs), such as Gowalla[3] and Foursquare[4], have attracted millions of users with billions of check-ins. Point-of-interest recommendation is one of the important tasks in LBSNs since it can help users explore new POIs as well as bring the potential commercial value to third-party developers. Geographical influence has a great impact on users' choices to visit new POIs, while traditional recommender systems fail to take it into account. Besides location information, other context information also influences users' decisions. One example is that we need to consider the important factor of "Double 11 Bachelor Day" in China or the "Black Friday" after Thanksgiving in the States when providing recommendations. Merchants usually run big promotions on these dates, which will heavily affect customers' shopping behaviors. Another example is that a male user would often watch action movies with other male users, but would watch romantic movies with his girlfriend. As a result, new methods need to be tailored to effectively utilize these location and context information to alleviate the sparsity problem and improve the recommendation performance. In this thesis, we present several methods to address these new challenges when recommending with location and context.

In the following, we first present a brief introduction to recommender systems in Section 1.1. Then we introduce our contributions in Section 1.2 and present the overall structure of this thesis in Sec-

[3] http://www.gowalla.com
[4] http://www.foursquare.com

Table 1.1: Data type

Type	Examples
Explicit feedback data	• Rating data • "Thumb up/down" data
Implicit feedback data	• The purchase/shopping data of Amazon • The click data of displaying advertisements • The check-in data of LBSNs

tion 1.3.

1.1 Overview of Recommender Systems

Recommender Systems are information filtering systems that focus on solving the information overload problem. They can provide proactive and personalized suggestions based on users' past behaviors on the systems. At the same time, merchants can provide promotions to users according to the predictions of recommender systems. Good recommender systems in online websites are beneficial for both merchants and customers. In the following, we first discuss about the data types used in recommender systems. Then we introduce the POI recommendation in LBSNs and context-aware recommendation.

1.1.1 Data Type

In recommender systems, there are mainly two types of data: *explicit feedback* data and *implicit feedback* data. The details are shown in Table 1.1. Recommendation techniques vary depending on the specific data type that is being used in the system.

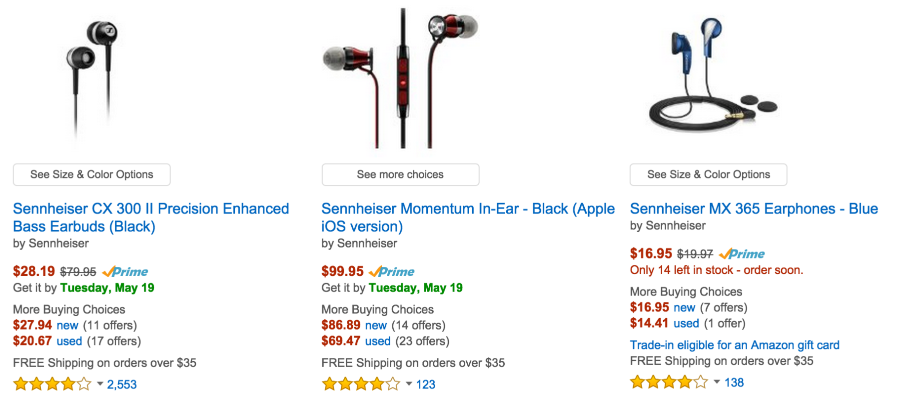


Figure 1.1: Rating example in Amazon

Explicit Feedback Data

The explicit feedback data are very popular and widely used in recommender systems. From the data, we can know both liked and disliked items of a user, i.e., we obtain both explicit positive and negative feedback from users.

Rating data is one of the explicit feedback data types, which is favored by many online websites such as Netflix[5], Douban[6], Amazon, Movielens[7], Yahoo! music[8], Yelp[9], etc. Users can give ratings to items in these websites in the 1-5 or 1-10 rating scale. Figure 1.1 shows an example of users' rating on Sennheiser earphones on the Amazon website. The higher rating means the user likes the item more. Taking the 1-5 scale rating system for example, a 5 rating means that a user likes the item very much; a 4 rating may mean the user likes the item. The rating of 3 may indicate the user thinks the item is just OK, while a rating of 1 or 2 may show that the user dislikes the item.

Another explicit feedback data type is the "Thumb up/down" data

[5] http://www.netflix.com
[6] http://www.douban
[7] https://movielens.org
[8] https://www.yahoo.com/music
[9] http://www.yelp.com

Figure 1.2: Thumb up/down example in YouTube

that is popular in Facebook[10], Google plus[11], YouTube[12], etc. Different from the rating data, users can only give one of the two ratings: like or dislike. The choice of like and dislike is often represented as the icon of "Thumb up" and "Thumb down" respectively. Figure 1.2 shows an example of YouTube videos. The users explicitly show their attitude towards the item, either like or dislike.

In the rating data type, different users may have different rating styles. Some users tend to give ratings generously while others give ratings in the opposite way. Although the same user is often consistent with his/her ratings, it is difficult to understand the attitudes towards items across different users. The "Thumb up/down" data can avoid this problem since there are only two choices for users.

Implicit Feedback Data

Implicit feedback data are also very common in many recommender systems. The purchase/shopping data of Amazon, the click data of

[10] http://www.facebook.com
[11] https://plus.google.com/
[12] https://www.youtube.com/

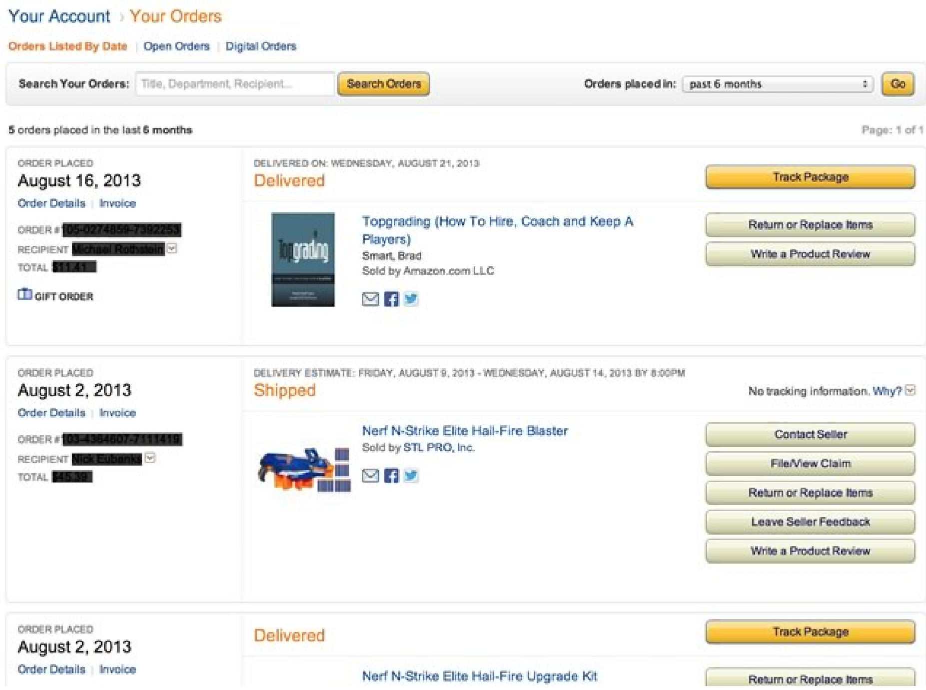


Figure 1.3: Purchase log in Amazon

displaying advertisements, the check-in data of LBSNs, etc., are all forms of implicit feedback data. Figure 1.3 shows an example of a user's purchase log on Amazon. The main difference between explicit feedback data and implicit feedback data is that we can only observe the positive feedback from users in implicit feedback data. For example, if we know a user has bought an item, we might infer that the user might like it. However, for all the items the user has not purchased yet, we cannot draw the conclusion that the user dislikes them. A user has not purchased the item might simply because the user does not know the item. Although some researchers argued that the explicit feedback data are biased, i.e., users tend to rate the items they like [93, 92, 74, 46], we have both positive and negative feedback in the explicit feedback data. However, we do not have any negative samples in the implicit feedback data. We refer to the recommendation with implicit feedback data as *One-class Recommendation* [96, 71, 51, 98].

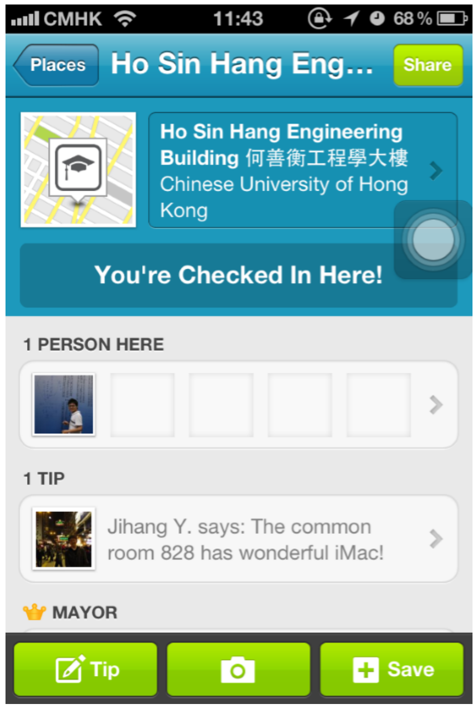


Figure 1.4: Foursquare check-in example

1.1.2 Point-of-interest Recommendation in LBSNs

In recent years, the rapid development of mobile devices and ubiquitous Internet access has led to the popularization of location-based social networks (LBSNs). The check-in behavior becomes a new life-style for millions of users to share their locations, tips and experience about POIs in LBSNs. It has been reported that there were over 20 million registered users corresponding to two billion check-ins by April, 2012[13]. Huge amount of information of users' physical movements in daily life and their preferences about the POIs are accumulated via check-in behavior. Figure 1.4 is an example of the check-in behavior in Foursquare.

POI recommendation has attracted much attention from both the research and industry communities [143, 117, 75, 72] as it can help users explore their nearby interesting POIs as well as enable third-

[13]http://statspotting.com/2012/04/foursquare-statistics-20-million-users-2-billion-check-ins/

party services, e.g., launching advertisements. Directly applying the traditional recommendation technique in POI recommendation will produce poor performance since the user-location matrix is too sparse. Fortunately, geographical influence can help alleviate this problem and improve the recommendation performance. In [143], the authors proposed to use the power-law distribution to model the geographical influence and unified it with the memory-based collaborative filtering methods. However, the power-law distribution was learned globally and memory-based collaborative filtering methods had to compute all the pairwise distances of users' whole visiting history, which was very time consuming. Moreover, users are more interested in the top 10 or 20 recommendation results, which makes personalized ranking important in POI recommendation. Based on this, in this thesis, we propose a unified POI recommendation framework that incorporates them together. Specifically, we first propose a Multi-center Gaussian Model (MGM) to capture users' moving patterns. Different from the power-law distribution, MGM is a personalized model that captures each user's moving behavior. Then we fuse MGM with model-based collaborative filtering methods in different ways to fuse the user preference, geographical influence and personalized ranking together.

Successive POI recommendation is another challenging task in LBSNs, which focuses on providing POI recommendation for users in the next few hours based on users' current locations and their previous check-in histories. In the general POI recommendation task, temporal relation and sequential effect are not considered. Successive POI recommendation is a much harder task, since we need to provide recommendations that may be attractive to a user while he/she does not visit them before at the successive time stamp. In traditional recommender systems, Koren et al. [63] considered employing temporal effect to improve the recommendation results. The authors assumed that users' preferences might change across time instead of following the basic assumption that users' preferences

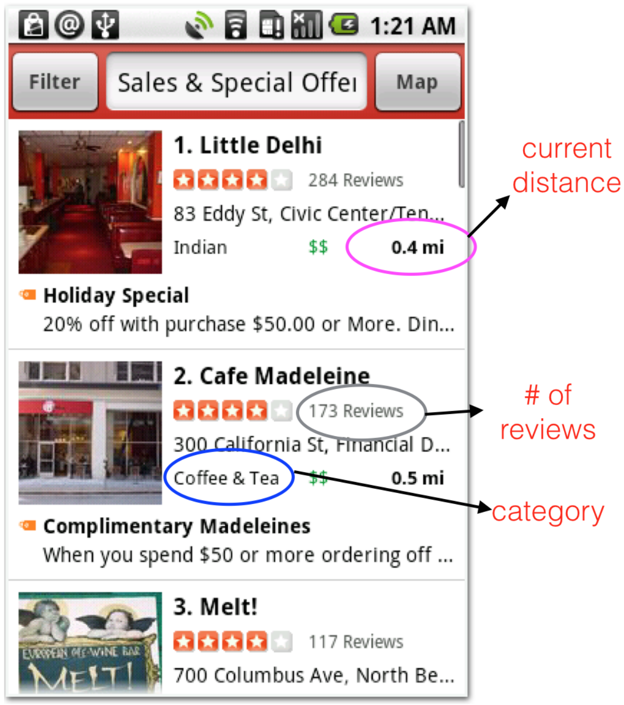


Figure 1.5: An example of context information in Yelp

would remain the same in most recommender systems. The Factorized Personalized Markov Chain (FPMC) was developed in Rendle et al. [105] to model the transition between items and FPMC provided recommendations for users to buy next items. However, in FPMC, geographical influence does not need to be considered. In LBSNs, we find that users' check-ins at the successive time stamp are constrained by personalized Markov chain and region localization. Based on this, in this thesis, we propose two novel matrix factorization methods to take these two factors into account.

1.1.3 Context-aware Recommendation

Most of traditional recommender systems are based on context-unaware recommendations since their techniques mainly analyze the user-item rating matrix and do not consider context information. In the real world, plenty of context information is available and is proven to be useful in recommendation to alleviate the data sparsity problem [53, 106, 57, 9, 102, 27]. We refer to recommendations with context information as context-aware recommendations.

Figure 1.5 gives an example of some context information in Yelp. The current distance to the restaurant, the number of reviews of a restaurant, the category of a restaurant, etc., are all context information that can be employed to generate restaurant recommendations to users. Meta-data, which is attached to the user or item itself, is the most common context information. For example, the age and gender from a user's profile, the genre of a movie, the average rating of an item, etc., are all meta-data. In addition to meta-data, context information also includes information attached to the whole recommendation event, such as a user's mood, the date, etc.

There are several existing methods that are being used in context-aware recommender systems. The most basic approach for context-aware recommendation is to conduct pre-filtering or post-filtering where a standard context-unaware method is applied [97, 22, 131, 2]. In order to consider the context information in the training phrase, several methods have been studied to incorporate meta-data into the matrix factorization method [68, 130]. In order to deal with all context features, several methods were explored in [135, 4, 57, 101, 106]. In these methods, context information was encoded as features; together with user and item, they were mapped from a feature space into a latent space. The Factorization Machines (FM) model [106] is currently a general and widely used method, which subsumes many methods such as SVD++. In FM, all features are assumed to be interacting with all other features. It is not always the case that all the feature interactions are useful. Thus, it is challenging to select automatically useful interacting features to reduce noise. To tackle this problem, we propose a novel Gradient Boosting Factorization Machines (GBFM) model to incorporate feature selection algorithm with Factorization Machines into a unified framework.

1.2 Thesis Contributions

The main contributions of this thesis could be described as follows:

1. **A Unified Point-of-interest Recommendation Framework in Location-based Social Networks**

   We propose a unified POI recommendation framework to fuse users' preferences, geographical influence and personalized ranking together to alleviate the data sparsity in LBSNs. Matrix factorization methods in traditional recommender systems cannot produce good POI recommendation performance due to the sparsity of the user-location matrix in LBSNs. Geographical influence plays an important role in users' check-ins and can help alleviate the data sparsity problem. Specifically, we first propose a Multi-center Gaussian Model (MGM) to capture the geographical influence based on the finding that users tend to check in around several centers and different users have different number of centers. Then we propose a method to incorporate matrix factorization with MGM together. Moreover, users in recommender systems are more interested in the top 10 or 20 recommendation results. Directly optimizing the pairwise ranking like Bayesian Personalized Ranking (BPR) produces better performance in top-k recommendation than directly using matrix factorization. To address the top-k ranking as well as the geographical influence, we propose two methods based on BPR, a state-of-the-art personalized ranking method, with different integration approaches. The experimental results on two large-scale real world LBSNs datasets illustrate the effectiveness of our proposed methods.

2. **Successive Point-of-interest Recommendation in Location-based Social Networks**

   In order to provide POI recommendation for users at the successive time stamp, we propose two novel matrix factorization methods called Factorized Personalized Markov Chain with Localized Region (FPMC-LR) and Factorized Personalized Markov Chain with Latent Topic Transition (FPMC-LTT) based on two prominent properties in the check-in sequence: personalized Markov chain and region localization. Both FPMC-LR and FPMC-LTT embed the two prominent properties in the models. The two models not only exploit the personalized Markov chain in the check-in sequence, but also take into account users' movement constraint, i.e., moving around a localized region. More importantly, by utilizing the information of localized regions, we not only reduce the computation cost, but also discard the noisy information to boost recommendation. The difference between FPMC-LR and FPMC-LTT is that the personalized Markov chain in FPMC-LR is built on location-wise level while FPMC-LTT is built on the latent topic transition. The number of observations on location-wise transition is very sparse in LBSNs, which makes it difficult to learn the latent location transition vector well. We observe that there is high transition probabilities between topics such as the transition from "Shopping" to "Food". FPMC-LTT models the latent topic transition probability, which can avoid the sparsity problem in FPMC-LR. We conduct thorough experiments on two real world LBSNs datasets. The experimental results demonstrate the merits of our proposed FPMC-LR and FPMC-LTT model.

3. **Context Feature Selection Algorithm in Context-aware Recommendation**

   In order to employ context features and select "good" context features effectively for recommender systems with sparse data, we propose a novel Gradient Boosting Factorization Machines (GBFM) model to incorporate the feature selection algorithm with Factorization Machines into a unified framework. Traditional matrix factorization methods focus on the user-item matrix only, while context information can be employed to improve recommendation performance and overcome the data sparsity problem. However, there are tens of context features available, which makes it challenging to select “good” context features among them. Most of existing state-of-the-art methods consider all of them, but not all of them are useful. We first propose a greedy interacting feature selection algorithm based on gradient boosting. Then, we fuse it with the Factorization Machines into a unified framework. The experimental results on both synthetic and real datasets demonstrate the efficiency and effectiveness of our algorithm compared with other state-of-the-art methods.

1.3 Thesis Organization

The rest of this thesis is organized as follows:

- **Chapter 2**

  In this chapter, we review the background knowledge and related work in the field of traditional recommender systems, ranking-oriented collaborative filtering, POI recommendation and context-aware recommendation. More specifically, we review the content-based filtering and collaborative filtering techniques. We focus more on collaborative filtering techniques as they are the most basic and applied techniques in recommender systems nowadays. We briefly review ranking-oriented collaborative filtering techniques, which aim to improve the ranking performance in recommender systems. Then we review existing methods in POI recommendation and context-aware recommendation.

- **Chapter 3**

  In this chapter, we focus on the task of point-of-interest recommendation, which attempts to recommend the potentially attractive while unvisited POIs to users. Firstly, we extract the characteristics from two large-scale LBSNs datasets: Gowalla and Foursquare, and we observe the multi-center check-in behavior of users in the datasets. Based on this, we build the Multi-center Gaussian Model (MGM), which captures the geographical influence in LBSNs. Secondly, we propose a unified framework to fuse the matrix factorization and MGM to capture both users' preferences and geographical influence. Thirdly, we further consider the importance of personalized ranking. We propose two methods based on BPR with different integration approaches with MGM. Lastly, we conduct thorough experiments on Gowalla and Foursquare to show the effectiveness of our proposed methods.

- **Chapter 4**

  In this chapter, we take the temporal relation and sequential effect into account and focus on the problem of how to provide successive POI recommendation in LBSNs. Firstly, we analyze the spatial-temporal properties in two large-scale real-world LBSNs datasets: Foursquare and Gowalla. After analyzing the dynamics of new POIs and inter check-ins, we observe two important properties: personalized Markov chain and localized region constraint. Secondly, based on the two properties, we propose two novel matrix factorization models: FPMC-LR and FPMC-LLT. The difference between FPMC-LR and FPMC-LLT is that the personalized Markov chain in FPMC-LR is modeled on the location-wise level, while it is modeled on the topic level in FPMC-LLT. Compared with other state-of-the-art methods, both FPMC-LR and FPMC-LLT are more efficient and effective. Finally, thorough experiments on Gowalla and Foursquare are conducted. The experimental results illustrate the merits of the two models.

- **Chapter 5**

  In this chapter, we explore the problem of how to select "good" context interacting features from tens of context features to improve context-aware recommendation performance. Firstly, we review the state-of-the-art method, the Factorization Machines (FM) model. We find that FM models pairwise interactions between all features. In this way, a certain feature latent vector is shared to compute the factorized parameters it involves. In practice, not all the pairwise feature interactions are useful. Secondly, in order to select "good" interacting features, we propose a novel interacting feature selection algorithm based on gradient boosting. Thirdly, we propose a novel Gradient Boosting Factorization Machines (GBFM) model to incorporate the feature selection algorithm with Factorization Machines. Finally, we conduct thorough experiments on a synthetic dataset and a real dataset to show the effectiveness of our algorithm compared with other state-of-the-art methods.

- **Chapter 6**

  The last chapter summarizes this thesis and addresses some potential directions that can be explored in the future.

In order to make each of these chapters self-contained, some critical contents such as model definitions or motivations may be briefly reiterated in some chapters.

□ **End of chapter.**

Chapter 2

Background Study

In this chapter, we investigate the background knowledge on traditional recommender systems, ranking-oriented collaborative filtering, POI recommendation and context-aware recommendation. The basic knowledge of traditional recommender systems is closely related to the whole thesis. Ranking-oriented collaborative filtering and POI recommendation are closely related to Chapter 3 and Chapter 4. The content of Chapter 5 is related to context-aware recommendation.

This chapter is organized as follows. In Section 2.1, we briefly go over the techniques in recommender systems including content-based filtering and collaborative filtering. We focus more on collaborative filtering as it is the dominating technique in today's recommender systems. We briefly review the two types of collaborative filtering methods: memory-based collaborative filtering and model-based collaborative filtering. In Section 2.2, we present a detailed survey on ranking-oriented collaborative filtering, which aims to improve ranking performance in recommender systems. In Section 2.3, we briefly discuss about the existing work in POI recommendation. Lastly, in Section 2.4, we review several state-of-the-art methods in context-aware recommendation.

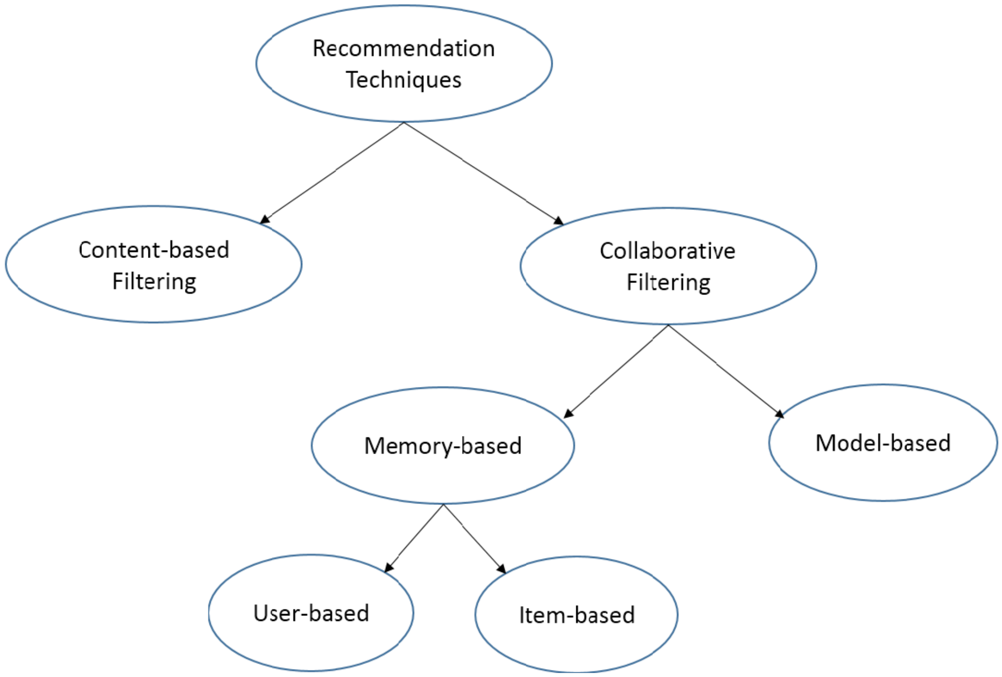


Figure 2.1: Classfication of recommendation techniques

2.1 Traditional Recommender Systems

With the rapid development of online e-commerce, music and movie websites and online social networks, we are facing exploding choices with much information around. *Recommender Systems* are becoming more and more important as they focus on solving the information overload problem [91, 47, 109, 122]. Recommender systems are a subclass of information filtering systems that attempt to provide recommendations on items (movies, music, POIs, books, etc.) that are likely of interest to the user. Many problems can be classified as recommendation problems, including search ranking [13, 15, 43, 55], query suggestion [18, 20, 87, 23], click-though rate prediction [28, 35, 6, 132], tag recommendation [154, 107, 70, 127, 65, 42] and item recommendation [3, 112, 61, 114, 64]. The techniques of recommender systems have been extensively studied in the past few decades.

We summarize the classification of recommendation techniques in traditional recommender systems in Figure 2.1 according to the way recommender systems work. Generally, recommender systems can be classified into two types: *content-based filtering* and *collaborative filtering*. Collaborative filtering attracts more attention since it is the most popular and effective method. Collaborative filtering is grouped into two general categories: memory-based collaborative filtering and model-based collaborative filtering [17]. Furthermore, memory-based collaborative filtering has two main classes: user-based collaborative filtering and item-based collaborative filtering. Model-based collaborative filtering includes the clustering model, the aspect model, the latent factor model, the Bayesian hierarchical model and the matrix factorization model. We do not show the categories of model-based collaborative filtering in the figure due to space limitation. In the following, we briefly review these methods.

2.1.1 Content-based Filtering

In content-based filtering systems [99, 25], a user is recommended items that are similar to the items the user preferred in the past. These systems adopt the ideas and employ the techniques from information retrieval [115, 7] and information filtering research [11]. Content-based filtering techniques try to learn the user's preference or match up the features of items by analyzing the user's content and profile. It has been widely adopted in news recommendation [60, 77, 1], music recommendation [25, 16] and movie recommendation [5, 34].

In general, content-based filtering systems operate in three steps. First, the systems maintain the information about the user's taste either from the initial user profile from the registration process or by the rating documents after registration. Then the systems analyze the content of documents and user profile. Finally, the systems recommend the ones that better match the user's preference and profile. For example, in the news recommendation [77], the authors

	i_1	i_2	i_3	i_4	i_5	i_6
u_1	?	3	4	?	1	?
u_2	1	2	?	5	?	1
u_3	2	?	?	3	1	1
u_4	5	1	3	?	?	2
u_5	4	5	?	?	?	3

Table 2.1: User-item rating matrix

first used topic modeling to classify news articles into different categories, then the user profile was built using a Bayesian model of click probability given the news category. In the movie recommendation application [5], the system tries to understand the content of the movies that the user rated highly in the past. Then, highly related movies will be recommended to the user.

2.1.2 Collaborative Filtering

Collaborative filtering (CF) methods provide recommendations to users by analyzing the co-occurrence patterns of user-item pairs. There are two major categories of collaborative filtering techniques: memory-based collaborative filtering and model-based collaborative filtering.

Memory-based Collaborative Filtering

Memory-based (neighborhood-based) collaborative filtering is widely adopted in commercial collaborative systems [73, 17]. These methods usually attempt to find similar users or items by assuming that similar users may have similar taste or similar items may have similar ratings. The most studied memory-based approaches are user-based collaborative filtering [17, 54, 45] and item-based collaborative filtering [73, 36, 118]. We introduce them in detail in the following.

In collaborative filtering methods, the user-item rating matrix is usually constructed. Suppose that we have M users and N items, then we can construct an $M \times N$ rating matrix. Each entry $r_{u,i}$ in the matrix denotes the rating given by a user u to an item i, where $r_{u,i} \in \{1, 2, ..., r_{\max}\}$. If the user u does not rate the item i, we leave it as blank or denote it as "?". Table 2.1 shows an example of the user-item rating matrix, in which we have 5 users and 6 items with $r_{\max} = 5$.

In user-based collaborative filtering methods, the prediction for a missing rating value $\hat{r}_{ui}$ is calculated as the weighted average of the ratings assigned by a set of similar users who rated the item before. The weight is determined by the similarity between the user and the user's neighbor users. In memory-based collaborative filtering, Vector Space Similarity (VSS) [17] and Pearson Correlation Coefficient (PCC) [109] are often employed to calculate the similarity between two users, which is calculated based on their co-rated items.

The similarity between two users u and a calculated by VSS is defined as:

$$Sim(u,a) = \frac{\sum\limits_{j \in I(u) \bigcap I(a)} r_{uj} \cdot r_{aj}}{\sqrt{\sum\limits_{j \in I(u) \bigcap I(a)} r_{uj}^2} \cdot \sqrt{\sum\limits_{j \in I(u) \bigcap I(a)} r_{aj}^2}}, \tag{2.1}$$

where item j belongs to the set where both user u and user a have rated. $I(u)$ is the set of items rated by user u and r_{uj} is the rating user u gave to item j. From the definition, we can observe that $Sim(u,a) \in [0,1]$, and a larger value means user u and user a are more similar.

However, one disadvantage of employing VSS to calculate the similarity is that VSS does not consider the factor that different users might have different rating styles. Some users are more generous and tend to give high ratings, while some users might be more critical and tend to give low ratings. PCC can solve this rating bias problem by adjusting ratings with users' average rating. The similarity calculated by PCC is defined as:

$$Sim(u,a) = \frac{\sum\limits_{j \in I(u) \bigcap I(a)} (r_{uj} - \overline{r}_u) \cdot (r_{aj} - \overline{r}_a)}{\sqrt{\sum\limits_{j \in I(u) \bigcap I(a)} (r_{uj} - \overline{r}_u)^2} \cdot \sqrt{\sum\limits_{j \in I(u) \bigcap I(a)} (r_{aj} - \overline{r}_u)^2}}, \tag{2.2}$$

where item j belongs to the set where both user u and user a have rated, and $\overline{r}_u$ represents the average rating of user u. Now $Sim(u,a) \in [-1,1]$, and also a larger value means the two users u and a are more similar.

After we obtain the similarity score between two users, we can predict the unknown rating $\hat{r}_{ui}$ by:

$$\hat{r}_{ui} = \frac{\sum\limits_{j \in S(u)} s_{uj} \cdot r_{ji}}{\sum\limits_{j \in S(u)} s_{uj}}, \tag{2.3}$$

where $S(u)$ is the set of user u's similar users, and s_{uj} is the similarity between user u and user j, which is calculated by VSS or PCC discussed above.

In practice, we often consider the k most similar users of user u instead of the whole similar user set $S(u)$. One possible problem with the method in Eq. (2.3) is the same as VSS. Some users are generous and give high ratings to items. On the other hand, some users are critical and give low ratings to items. Thus, the users' ratings are biased. In [17, 109], a simple solution is proposed to solve this problem by adjusting ratings with their mean:

$$\hat{r}_{ui} = \overline{r}_u + \frac{\sum\limits_{j \in S(u)} s_{uj} \cdot (r_{ji} - \overline{r}_j)}{\sum\limits_{j \in S(u)} s_{uj}}, \tag{2.4}$$

where $\overline{r}_u$ denotes the average rating given by user u.

Item-based collaborative filtering shares a very similar idea with user-based collaborative filtering. In item-based collaborative filtering, we calculate the missing rating value $\hat{r}_{ui}$ based on similar items instead of similar users. In many recommender systems, the number of items is much less than the number of users, which makes the item similarity more reliable than the user similarity. The prediction for the missing value $\hat{r}_{ui}$ is calculated as:

$$\hat{r}_{ui} = \frac{\sum\limits_{j \in S(i)} s_{ij} \cdot r_{uj}}{\sum\limits_{j \in S(i)} s_{ij}}, \tag{2.5}$$

where $S(i)$ stands for the set of item i's similar items. Similarly, the similarity between items i and j calculated by PCC is as follows:

$$Sim(i,j) = \frac{\sum\limits_{u \in U(i) \bigcap U(j)} (r_{ui} - \overline{r}_i) \cdot (r_{uj} - \overline{r}_j)}{\sqrt{\sum\limits_{u \in U(i) \bigcap U(j)} (r_{ui} - \overline{r}_i)^2} \cdot \sqrt{\sum\limits_{u \in U(i) \bigcap U(j)} (r_{uj} - \overline{r}_j)^2}}, \tag{2.6}$$

where $U(i)$ is the set of users who rated item i, and $\overline{r}_i$ is the average rating score of item i.

Due to the simplicity of memory-based CF methods, they are very popular and widely applied in commercial websites. However, there are some drawbacks of memory-based CF methods. Firstly, the similarity computed by PCC or VSS is not always accurate, especially when the data are very sparse. If a user only has a few ratings, it is difficult to obtain reliable similarity between the user and other users. Furthermore, the time and space complexity is relatively high since these methods need to obtain all the user/item similarity to compute predictions. Besides, these methods are based on heuristics without any objective function to optimize [67], so there is no global optimal in memory-based CF methods.

Model-based Collaborative Filtering

Different from memory-based collaborative filtering, model-based collaborative filtering trains a predefined model by employing the observed user-item ratings. Then the predefined model is employed to make further predictions [41, 113, 112, 49, 125, 100]. These models often outperform memory-based models and have received great success in many competitions such as Netflix competition [62] and KDD Cup 2007 [66].

Model-based algorithms include the clustering model [58], the aspect models [48, 49, 126], the latent factor model [21], the Bayesian hierarchical model [147] and matrix factorization models [113, 112, 63, 61]. Recently, several low-dimensional matrix approximation methods [14, 63, 108, 112, 113, 128] have been proposed due to their efficiency in dealing with large datasets. All of these methods focus on fitting the observed user-item matrix using low-rank approximation. They assume that only a small number of factors affect users' preferences. Finally, predictions can be made by computing the product of learned latent matrices.

Compared with memory-based collaborative filtering methods, model-based collaborative filtering methods are less prone to the data sparsity problem due to the low rank assumption. In memory-based collaborative filtering, the user/item similarity is not reliable when data are sparse, which makes the prediction inaccurate. Moreover, the model-based collaborative filtering methods are more efficient. The training time of model-based models is linear to the number of observations in the user-item rating matrix. When making predictions on unknown ratings, we just need to compute the inner product of the corresponding user and item latent vectors in constant time. However, in memory-based collaborative filtering, the prediction part is usually very time consuming since we need to compute all similarities between users or items.

In the following, we will discuss some of the most popular model-based methods.

Probabilistic Matrix Factorization

Probabilistic Matrix Factorization (PMF) [112] is one of the most popular model-based collaborative filtering methods. Assuming that we have an $M \times N$ partially observed user-item rating matrix R, PMF tries to approximate this matrix R by the multiplication of two low-rank matrices, i.e., $R \approx U^T V$. Here $U \in \mathbb{R}^{M \times D}$ and $V \in \mathbb{R}^{N \times D}$ are the latent user and item feature matrices respectively. D is the rank and $D \ll \min(M, N)$. U_i and V_j are the low-rank user-specific and item-specific latent feature vectors for user u_i and item v_j respectively.

Then the conditional distribution over the observed ratings is defined as:

$$p(R|U, V, \sigma^2) = \prod_{i=1}^{M} \prod_{j=1}^{N} \left[\mathcal{N}(R_{ij} | U_i^T V_j, \sigma^2 \right]^{I_{ij}^R}, \tag{2.7}$$

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2. I_{ij}^R is an indicator function whose value equals to 1 if user u_i rated item v_j and equals to 0 otherwise. We further place zero-mean spherical Gaussian priors on the user and item latent feature vectors:

$$p(U|\sigma_U^2) = \prod_{i=1}^{M} \mathcal{N}(U_i | 0, \sigma_U^2 \boldsymbol{I}), \tag{2.8}$$

$$p(V|\sigma_V^2) = \prod_{j=1}^{N} \mathcal{N}(V_j | 0, \sigma_V^2 \boldsymbol{I}). \tag{2.9}$$

Hence, through Bayesian inference, we have:

$$p(U, V | R, \sigma_R^2, \sigma_U^2, \sigma_V^2) \propto p(R|U, V, \sigma_R^2) p(U|\sigma_U^2) p(V|\sigma_V^2). \tag{2.10}$$

The log of posterior distribution over the user and item features is given by:

$$\begin{aligned}
& \ln p(U, V | R, \sigma_R^2, \sigma_U^2, \sigma_V^2) \\
= \; & -\frac{1}{2\sigma^2} \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij}^R (R_{ij} - U_i^T V_j)^2 - \frac{1}{2\sigma_U^2} \sum_{i=1}^{M} U_i^T U_i - \frac{1}{2\sigma_V^2} \sum_{j=1}^{N} V_j^T V_j \\
& -\frac{1}{2} \left(\left(\sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij}^R \right) \ln \sigma^2 + MD \ln \sigma_U^2 + ND \ln \sigma_V^2 \right) + C,
\end{aligned} \tag{2.11}$$

where C is a constant that does not depend on the parameters. Maximizing the log-posterior over user and item features with fixed hyperparameters (i.e., the observation noise variance and prior variances) is equivalent to minimizing the sum-of-squared-errors objective function with quadratic regularization terms:

$$\mathcal{L} = \frac{1}{2} \sum_{i=1}^{M} \sum_{j=1}^{N} I_{ij}^R (R_{ij} - U_i^T V_j)^2 + \frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_V}{2} \|V\|_F^2, \tag{2.12}$$

where $\lambda_U = \sigma^2/\sigma_U^2$, $\lambda_V = \sigma^2/\sigma_V^2$, and $\|\cdot\|_F$ denotes the Frobenius norm.

A local minimum of the objective function in Eq. (2.12) can be found by performing gradient descent on U and V. The partial derivatives of $\mathcal{L}$ with respect to U and V are:

$$\frac{\partial \mathcal{L}}{\partial U_i} = \sum_{j=1}^{N} I_{ij}^R (U_i^T V_j - R_{ij}) V_j + \lambda_U U_i, \tag{2.13}$$

$$\frac{\partial \mathcal{L}}{\partial V_j} = \sum_{i=1}^{M} I_{ij}^R (U_i^T V_j - R_{ij}) U_i + \lambda_U V_j. \tag{2.14}$$

Then at each iteration, we can update the parameters as follows:

$$U_i^{(t+1)} = U_i^{(t)} - \alpha \frac{\partial \mathcal{L}}{\partial U_i^{(t)}}, \tag{2.15}$$

$$V_j^{(t+1)} = V_j^{(t)} - \alpha \frac{\partial \mathcal{L}}{\partial V_j^{(t)}}. \tag{2.16}$$

Here, α is the step size. Usually, the training process will converge in a few hundred iterations. Note that, the time complexity at each iteration is linear with respect to the number of observations in the user-item rating matrix, which makes it possible to be scalable to large-scale datasets. PMF can be viewed as a probabilistic extension of the SVD model, since if all ratings have been observed, the objective in Eq. (2.12) reduces to the SVD objective when the limit of the prior variances goes to infinity.

After training the model, we obtain the parameters U and V. The prediction for an unknown rating $\hat{r}_{ij}$ is calculated as the inner product of U_i and V_j:

$$\hat{r}_{ui} = U_i^T V_j. \tag{2.17}$$

Extensions of PMF

The simple linear-Gaussian model in the original PMF makes predictions outside of the range of valid rating values (e.g., 1-5 ratings). A simple extension of the PMF model introduced above is that the dot product between user- and item-specific feature vectors is passed through the logistic function $g(x) = 1/(1 + \exp(-x))$, which bounds the range of predictions to $[0, 1]$. Now the objective function becomes:

$$p(R|U, V, \sigma^2) = \prod_{i=1}^{M} \prod_{j=1}^{N} \left[\mathcal{N}(R_{ij}|g(U_i^T V_j), \sigma^2\right]^{I_{ij}^R}. \tag{2.18}$$

We map the ratings $1, \ldots, K$ to the interval $[0, 1]$ using the function $t(x) = (x - 1)/(K - 1)$, so that the range of valid rating values matches the range of predictions made by PMF.

Bayesian Probabilistic Matrix Factorization (BPMF) [113] is another extension of PMF. In PMF, the hyper-parameters λ_U and λ_V need to be carefully tuned otherwise it is prone to overfitting. BPMF is a fully Bayesian treatment of the PMF model, in which model capacity is controlled automatically by integrating over all model parameters and hyper-parameters. In PMF, the priors of U and V are spherical Gaussian, while in BPMF, the Gaussian-Wishart priors are placed on the user and item hyper-parameters. Since the model in BPMF is intractable to solve, Markov Chain Monte Carlo is proposed to estimate the parameters.

One Class Collaborative Filtering

As discussed in Chapter 1, there are two data types in recommender systems: explicit feedback data and implicit feedback data. PMF is more suitable for explicit feedback data in practice. The techniques designed for implicit feedback data are called *one class collaborative filtering* (OCCF) [51, 96, 71, 98, 95].

In the method introduced in [51], a set of binary variables p_{ui} that indicate the preference of user u to item i are introduced. The p_{ui} values are derived by binarizing the r_{ui} values:

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases} . \tag{2.19}$$

In Eq. (2.19), r_{ui} is the frequency that user u purchased item i (here we use the purchase history data as the example). Here, if user u bought item i ($r_{ui} > 0$) before, then we have an indication that user u likes item i ($p_{ui} = 1$). On the other hand, if user u has never bought item i, we believe that there is no preference ($p_{ui} = 0$). However, different confidence levels should be considered according to the value of r_{ui}. If r_{ui} is large, we may have higher confidence to say that user u likes item i. Otherwise, we might have lower confidence to draw the conclusion. A set of variables c_{ui} are introduced to measure the confidence in observing p_{ui}. A plausible choice for c_{ui} would be:

$$c_{ui} = 1 + \alpha r_{ui}, \tag{2.20}$$

where α is used to control the rate increase.

Similar to PMF, we would like to learn two low rank matrices (i.e., user-specific and item-specific latent feature matrices $X \in \mathbb{R}^{m\times f}$ and $Y \in \mathbb{R}^{n\times f}$) to ensure $p_{ui} \approx x_u^T y_i$ with the corresponding confidence level c_{ui}. Here m and n are the number of users and items respectively, and f is the latent dimension. The objective function is as follows:

$$\mathcal{L} = \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda\left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2\right). \tag{2.21}$$

The term $\lambda\left(\sum\limits_u \|x_u\|^2 + \sum\limits_i \|y_i\|^2\right)$ is the regularization term to avoid overfitting.

Different from PMF, the objective function in Eq. (2.21) considers both observed and non-observed entries in the rating matrix. In other words, we have $m \times n$ terms and it is impossible to use the gradient descent method like in PMF to learn the parameters, since usually $m \times n$ can easily reach a few billion in most datasets. Alternative Least Square (ALS) method can be employed to solve this problem.

By differentiation we can find the analytic expression for x_u that minimizes the loss function in Eq. (2.21):

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u), \tag{2.22}$$

where C^u is a diagonal $n \times n$ matrix with $C^u_{ii} = c_{ui}$, the vector $p(u) \in \mathbb{R}^n$ contains all the preferences by u, and I is the identity matrix. Note that $Y^T C^u Y = Y^T Y + Y^T(C^u - I)Y$. Before looping through all users, we can precompute the matrix $Y^T Y$ in time $O(f^2 n)$. For $Y^T(C^u - I)Y$, notice that $C^u - I$ has only n_u non-zero elements, where n_u is the number of items for which $r_{ui} > 0$, i.e., the number of items user u purchased before, and usually $n_u \ll n$. Similarly, $C^u p(u)$ contains just n_u non-zero elements. Thus, the time complexity for computing x_u is $O(f^2 n_u + f^3)$. After updating x_u for all m users, the total time complexity is $O(f^2\mathcal{N} + f^3 m)$, where $\mathcal{N}$ is the number of observations in the user-item matrix. Since f is usually very small, we can see that the time complexity is still linear to the number of observed entries in the user-item matrix.

Similarly, the analytic expression for y_i is:

$$y_i = (X^T C^i X + \lambda I)^{-1} X^T C^i p(i), \tag{2.23}$$

where C^i is a diagonal $m \times m$ matrix with $C^i_{uu} = c_{ui}$, and the vector $p(i) \in \mathbb{R}^m$ contains all the preferences for item i. The total time for updating Y is $O(f^2\mathcal{N} + f^3 n)$.

There are several alternative methods proposed in OCCF. In [96], instead of considering all the missing entries like the method introduced above, the authors proposed three sampling schemes to sample negative user-item pairs from the missing entries. As a result, the training process is faster than the above method. Pan et al. [95] proposed a novel confidence weight scheme instead of using a heuristic function like Eq. (2.20) introduced above. The authors proposed to use two low rank matrices to learn the confidence weight on negative samples and forced the weight of observed entries to be 1.

Collaborative Filtering with Social Relationships

In PMF, one basic assumption is that users are independent and identically distributed, while it is not always true in the real world. In the real world, we often seek advice from our friends when making decisions. Social relationship plays an important role in recommender systems and many social information is available in recommender systems such as Douban, Epinion, etc. As a result, collaborative filtering with social relationships has been extensively studied in the past few years [84, 89, 88, 155, 10, 52].

In [84], the authors argued that both the user's personal preference and the preferences of the user's friends might affect the user's ratings to items. Suppose we have a directed social trust graph $\mathcal{G} = (\mathcal{U}, \mathcal{E})$, where the vertex set $\mathcal{U}$ denotes all the users in the social trust network and the edge set $\mathcal{E}$ denotes the trust relations between users. Let $S = \{S_{ij}\}$ denote the $m \times m$ matrix of $\mathcal{G}$, which is called the social trust matrix. For a pair of vertices, u_i and u_j, let $S_{ij} \in (0, 1]$ denotes the degree user i trusts user j. A larger value of S_{ij} means that user i trusts user j more. If there is no edge between user i and j, S_{ij} is 0. Then the log of the posterior distribution over the observed ratings is:

$$\begin{aligned}
&\ln p(U, V | R, S, \sigma^2, \sigma_U^2, \sigma_V^2) = \\
&-\frac{1}{2\sigma^2} \sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij}^R \left(R_{ij} - g\left(\alpha U_i^T V_j + (1-\alpha) \sum_{k \in \mathcal{T}(i)} S_{ik} U_k^T V_j \right)\right)^2 \\
&-\frac{1}{2\sigma_U^2} \sum_{i=1}^{m} U_i^T U_i - \frac{1}{2\sigma_V^2} \sum_{j=1}^{n} V_j^T V_j \\
&-\frac{1}{2} \left(\left(\sum_{i=1}^{m} \sum_{j=1}^{n} I_{ij}^R \right) \ln \sigma^2 + mD \ln \sigma_U^2 + nD \ln \sigma_V^2 \right) + C,
\end{aligned} \tag{2.24}$$

where $\mathcal{T}_i$ is the set of friends that user u_i trusts. The parameter α controls how much users trust themselves or their trusted friends.

Ma et al. [89] proposed a different model that imposed regularization terms to user-specific latent feature vectors based on the assumption that similar friends may have similar user latent feature vectors. In their work, in terms of two different regularization terms, they proposed two models: Average-based Regularization Model and Individual Regularization Model.

The objective function of Average-based Regularization Model is:

$$\begin{aligned}\min_{U,V} L(R,U,V) &= \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(R_{ij} - U_i^T V_j)^2 \\ &+ \frac{\alpha}{2}\sum_{i=1}^{m}\left\| U_i - \frac{\sum\limits_{f\in\mathcal{F}^+(i)} Sim(i,f)\times U_f}{\sum\limits_{f\in\mathcal{F}^+(i)} Sim(i,f)}\right\|_F^2 \\ &+ \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2,\end{aligned} \tag{2.25}$$

where $\alpha > 0$, $\mathcal{F}^+(i)$ is the set of friends of user u_i, and $Sim(i,f)$ denotes the similarity between user u_i and his/her friend u_f. In the above function, Ma et al. imposed a social regularization term $\|U_i - \frac{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)\times U_f}{\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)}\|_F^2$ to require that user u_i's latent vector should be similar to the weighted average latent vector of his/her friends.

In contrast to Average-based Regularization Model, Individual-based Regularization Model imposes a different social regularization term in the objective function, which constraints a user and the user's friends individually. The regularization term is:

$$\frac{\beta}{2}\sum_{i=1}^{m}\sum_{f\in\mathcal{F}^+(i)} Sim(i,f)\|U_i - U_f\|_F^2, \tag{2.26}$$

where $\beta > 0$, and $Sim(i,f)$ is the similarity between user u_i and u_f. The rest part of the objective function is the same in the two models.

2.2 Ranking-oriented Collaborative Filtering

In information retrieval field, the learning-to-rank (LTR) technique [80] has been well studied to address the importance of top-k ranking. Top-k ranking is also important in recommender systems, since users are more interested in top-k recommendation results. Most of ranking-oriented collaborative filtering methods borrow the idea from the LTR technique.

Liu et al. [80] classified LTR into three categories: point-wise, pairwise and list-wise. Point-wise approaches predict ranking scores for individual items, thus most of rating prediction models in collaborative filtering such as PMF can be regarded as point-wise approaches. A ranking list is created according to the predicted scores. But most rating prediction models use Mean Average Error (MAE) or Root Mean Square Error (RMSE) or as the metric, which makes it difficult to interpret as a measure of ranking quality. In terms of pairwise approaches, methods in [78, 79, 104] used the same idea in pairwise LTR to make predictions for every pair of user-item ratings concerning their relative ordering in the final list. List-wise LTR considered the difference between the reference list and the output list. List-wise collaborative filtering method was explored in [125].

Some other methods in ranking-oriented collaborative filtering are proposed to directly optimize the ranking metrics. CofiRank [134] proposed to directly optimize the ranking measure Normalized Discounted Cumulative Gain (NDCG). In [124], the model directly maximized the Mean Reciprocal Rank (MRR). Shi et al. [123] proposed a model that directly maximized Mean Average Precision (MAP) with the aim of creating an optimally ranked list of items for individual users under a given context.

In the following, we mainly review the pairwise method: Bayesian Personalized Ranking (BPR) [104], which is closely related to this thesis.

BPR is a method proposed to directly optimize for ranking in terms of implicit feedback data. Let U be the set of all users and I be the set of all items. The implicit feedback matrix is denoted as $S \subseteq U \times I$. The training set $D_S \subseteq U \times I \times I$ is defined as:

$$D_S = \{(u, i, j) | u \in U \wedge i \in I_u^+ \wedge j \in I \setminus I_u^+\}, \tag{2.27}$$

where I_u^+ is the set of items that user u purchased. The tuple (u, i, j) means that user u prefers item i over item j. The assumption of BPR is that if user u has purchased item i, we assume that user u prefers item i over all other non-observed items. Let $>_u \subset I \times I$ be a personalized total ranking, then for a sample $(u, i, j) \in D_S$, we have $i >_u j$.

The Bayesian formulation of finding the correct personalized ranking for all items is to maximize the following posterior probability:

$$p(\Theta| >_u) \propto p(>_u |\Theta)p(\Theta), \tag{2.28}$$

where Θ denotes the parameter vector. We further assume that all users are independent and the ordering of each pair of items (i, j) for a specific user is independent of the ordering of every other pair, then we have:

$$\prod_{u \in U} p(>_u |\Theta) = \prod_{(u,i,j) \in D_S} p(i >_u j|\Theta). \tag{2.29}$$

We define the individual probability that a user prefers item i over item j as:

$$p(i >_u j|\Theta) = \sigma(\hat{x}_{uij}(\Theta)), \tag{2.30}$$

where $\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj}$ is the difference of estimated values between (u, i) and (u, j). The value of $\hat{x}_{ui}$ can be estimated in many ways. For example, $\hat{x}_{ui} = U_u^T V_i$ in matrix factorization. And σ is the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.31}$$

Furthermore, by placing zero mean Gaussian prior on the parameter vector Θ, we can formulate the maximum posterior estimator to derive the optimization criterion for BPR:

$$\begin{aligned}\mathcal{L} &= \ln p(>_u |\Theta)p(\Theta) \\ &= \ln \prod_{(u,i,j)\in D_S} \sigma(\hat{x}_{uij})p(\Theta) \\ &= \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\ &= \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|_F^2,\end{aligned} \tag{2.32}$$

where λ_Θ is the regularization parameter.

To learn the parameters, stochastic gradient descent method can be applied. The gradient with respect to the model parameter is:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \Theta} &= \sum_{(u,i,j)\in D_S} \frac{\partial}{\partial \Theta} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \frac{\partial}{\partial \Theta}\|\Theta\|_F^2 \\ &\propto \sum_{(u,i,j)\in D_S} \frac{-e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta}\hat{x}_{uij} - \lambda_\Theta \Theta.\end{aligned} \tag{2.33}$$

Then for each tuple $(u, i, j) \in D_S$, the update rule is:

$$\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta}\hat{x}_{uij} + \lambda_\Theta \Theta \right), \tag{2.34}$$

where α is the step size.

2.3 Point-of-interest Recommendation

With the rapid growth of mobile devices and Internet access, it is easy to gather users' locations in location-based services. Location-based service (LBS) research became prevalent [82, 139, 140, 151] due to a wide range of potential applications, e.g., personalized marketing strategy analysis, personalized behavior study, context-aware analysis, etc. In particular, POI recommendation as one of the important tasks has attracted much research interest in recent years [56, 50, 152, 149, 69].

There are mainly two lines of work to solve POI recommendation: one line of research focuses on the GPS dataset [152, 149, 148, 69, 150, 24], while the other focuses on the LBSNs dataset [142, 143, 29]. In general, the GPS dataset is usually in small-scale with about one or two hundred users, but the data are very dense. The user's location is tracked by the system every few seconds, which makes it possible to know the real-time movement of a certain user. Contrarily, the LBSNs dataset is in large-scale with thousands of users, but the data are very sparse [94, 120]. The users in LBSNs voluntarily check in some POIs and the average check-in number of an active user in LBSNs is less than 10.

In the following, we mainly review two methods in the GPS dataset and one method in the LBSNs dataset.

2.3.1 Collaborative Location and Activity Recommendation

Zheng et al. [149] proposed a collaborative location and activity recommendation (CLAR) model to provide location and activity recommendation services for the Microsoft GeoLife Project[1]. CLAR was based on collective matrix factorization to propagate information among two additional information sources and the sparse location-activity matrix, so that the model can collaboratively predicted the missing entries in the location-activity matrix for recommendations.

The CLAR model extracted three matrices: location-activity matrix, location-feature matrix and activity-activity matrix. For the location-activity matrix, the entry X_{ij} is the count of activity j performed at location i. The count information is obtained from the

[1] http://research.microsoft.com/en-us/projects/geolife/

user's comments. For locations that don't have comments, X_{ij} is 0. The location-feature matrix Y reflects the type of a POI. The entry of matrix Y is :

$$Y_{ij} = \frac{q_{ij}}{\sum_{j=1}^{l} q_{ij}} \times \log \frac{|\boldsymbol{q}_i|}{|\boldsymbol{q}_i : q_{ij} > 0|}, \tag{2.35}$$

where $|\boldsymbol{q}_i|$ is the number of all the count vectors (i.e., number of locations), and $|\boldsymbol{q}_i : q_{ij} > 0|$ is the number of count vectors having non-zero j-th type POIs. This way, we reasonably increase the weights for those important types that are fewer but unique and decrease the weight for those extensively distributed POIs. The activity-activity matrix reflects the correlations between different activities. To get this correlation information, the authors put each pair of activities a_i and a_j together as a query and submitted it to Bing to get the web-page hit counts. The entry of the activity-activity matrix Z is defined as:

$$Z_{ij} = h_{ij}/h^*, \forall i = 1, \ldots, n; j = 1, \ldots, n, \tag{2.36}$$

where h_{ij} is the hit count for activity i and activity j based on some search engines. h^* is the maximal hit counts among all the hits counts for each pair of activities.

After obtaining the above three matrices, the objective function is defined as:

$$\begin{aligned} L(U, V, W) &= \frac{1}{2}\|I \circ (X - U^T V)\|_F^2 + \frac{\lambda_1}{2}\|Y - U^T W\|_F^2 \\ &+ \frac{\lambda_2}{2}\|Z - V^T V\|_F^2 + \frac{\lambda_3}{2}(\|U\|_F^2 + \|V\|_F^2 + \|W\|_F^2), \end{aligned} \tag{2.37}$$

where I is an indicator matrix with its entry $I_{ij} = 0$ if X_{ij} is missing, otherwise is 1. The operator "$\circ$" denotes the entry-wise product.

After having the complete location-activity matrix X, when a user queries some locations, we can look up the rows of X and rank the row's value in a descending order and return a list of corresponding activities for activity commendation.

2.3.2 Community Location Model

Since the user-location matrix created based on the GPS dataset is huge, the number of similar locations that need to be considered in computing the recommendations can be numerous. As a result, the identification of truly relevant locations from numerous candidates is challenging. Leung et al. [69] proposed a memory-based co-clustering method, Community Location Model (CLM) framework, to address this. The main idea is that instead of considering the user-location matrix only, we further consider the activity as well and compute pairwise similarity based on the other two factors. For example, when computing user similarity, we use the information of locations that the user visited and activities that the user performed.

The instance of CLM is a user-activity-location tripartite graph (i.e., CLM graph). The similarity can be computed as follows:

$$\begin{aligned} sim(u_i, u_j) &= \frac{\sum_{k=1}^{n} \frac{LCS(a(k,u_i),a(k,u_j))}{max(|a(k,u_i)|,|a(k,u_j)|)}}{n}\alpha_1 + \frac{L_{u_i} \cdot L_{u_j}}{\|L_{u_i}\|\|L_{u_j}\|}(1-\alpha_1), \\ sim(a_i, a_j) &= \frac{LCS(a_i, a_j)}{\max(|a_i|, |a_j|)}\alpha_2 + \frac{U_{a_i} \cdot U_{a_j}}{\|U_{a_i}\|\|U_{a_j}\|}(1-\alpha_2), \\ sim(l_i, l_j) &= \frac{U_{l_i} \cdot U_{l_j}}{\|U_{l_i}\|\|U_{l_j}\|}\alpha_3 + \frac{A_{l_i} \cdot A_{l_j}}{\|A_{l_i}\|\|A_{l_j}\|}(1-\alpha_3). \end{aligned} \tag{2.38}$$

Here $LCS(\cdot, \cdot)$ is the *longest common subsequence*, $a(k, u_i)$ denotes the activities performed by u_i on day k. L_{u_i} is a weight vector for the set of neighbor location nodes of the user node u_i. The weight of a location neighbor node in L_{u_i} is the weight of the link connecting u_i and the location in the CLM graph. Similarly, U_{l_i} is a weight vector for the set of neighbor user nodes of the location node l_i. A_{l_i} is a weight vector for the set of neighbor activity nodes of the location node l_i.

Then Leung et al. proposed a Community-based Agglomerative-Divisive Clustering (CADC) algorithm to iteratively cluster different types of entities (i.e., users, activities, locations) simultaneously based on CLM.

Finally, assuming that the active user u_a is visiting location l_c when he/she is performing activity a_b, then from the cluster, we can get the sub-cluster CADC(U,A,L). The locations in this sub-cluster are the intersection of three sets: the set of locations similar to l_c, the set of locations visited by users similar to u_a and the set of locations where activities similar to a_b are performed. Then we recommend the top-k locations to the user.

2.3.3 Collaborative Point-of-interest Recommendation with Geographical Influence

Both of the above two methods in the GPS dataset do not consider the geographical influence when performing POI recommendation. However, in LBSNs, due to the sparsity of the user-location matrix, utilizing matrix factorization or memory-based CF alone will yield poor performance. Ye et al. [143] explored geographical influence for POI recommendation in LBSNs. The authors proposed to use the power-law distribution to model the geographical influence and then further fused it with memory-based collaborative filtering methods.

The authors first calculated the pairwise distance between each user's check-ins and plotted the check-in probability as the function of physical distance. Then they proposed to use the power-law distribution to model the check-in probability to the distance between two POIs visited by the same user as follows:

$$y = a \times x^b, \tag{2.39}$$

where a and b are parameters of the power-law distribution, x is the distance between two POIs visited by the same user, and y is the probability of distance x.

For a given user u_i and his/her visited POI set L_i, the probability that u_i has check-in activities at all locations in L_i is defined by considering the pairwise distance of POIs in L_i:

$$Pr[L_i] = \prod_{l_m, l_n \in L_i \bigwedge m \neq n} Pr[d(l_m, l_n)], \tag{2.40}$$

where $d(l_m, l_n)$ denotes the distance between POIs l_m and l_n, and $Pr[d(l_m, l_n)] = a \times d(l_m, l_n)^b$, which follows the power-law distribution.

Then for a new POI l_j, user u_i and his visited POI history set L_i, the probability for user u_i to check in location l_j is defined as follows:

$$\begin{aligned} Pr[l_j|L_i] &= \frac{Pr[l_j \bigcup L_i]}{Pr[L_i]} \\ &= \frac{Pr[L_i] \times \prod_{l_y \in L_i} Pr[d(l_j, l_y)]}{Pr[L_i]} \\ &= \prod_{l_y \in L_i} Pr[d[l_j, l_y]]. \end{aligned} \tag{2.41}$$

To make POI recommendations, we can sort all the POIs in $L - L_i$ according to their probabilities according to Eq. (2.41) and return the top-k POIs.

At last, a linear fusion framework is proposed to integrate ranked lists provided by geographical influence and two common methods, the user-based method and the friend-based method, into the final ranked list.

Let $S_{i,j}$ denote the check-in probability score of user u_i at POI l_j, i.e., a larger value of $S_{i,j}$ means u_i will more likely check in at l_j. Let $S^u_{i,j}$, $S^s_{i,j}$ and $S^g_{i,j}$ denote the check-in probability scores of user u_i at POI l_j, corresponding to the methods based on user preference, social influence and geographical influence, respectively. Then the final $S_{i,j}$ is:

$$S_{i,j} = (1 - \alpha - \beta) S^u_{i,j} + \alpha S^s_{i,j} + \beta S^g_{i,j}, \tag{2.42}$$

where the two weighting parameters α and β $(0 \leq \alpha + \beta \leq 1)$ denote the relative importance of social influence and geographical influence compared with user preference. Here $\alpha = 1$ states that $S_{i,j}$ depends completely on the prediction based on social influence; $\beta = 1$ states that $S_{i,j}$ depends completely on the prediction based on geographical influence, while $\alpha = \beta = 0$ states that $S_{i,j}$ counts only on user preference.

Next we estimate the check-in probability $p^u_{i,j}$, $p^s_{i,j}$ and $p^g_{i,j}$ for a user u_i to visit a POI l_j in order to obtain $S^u_{i,j}$, $S^s_{i,j}$ and $S^g_{i,j}$, respectively.

The prediction of $p^u_{i,j}$ can be estimated based on the idea of user-based collaborative filtering:

$$p^u_{i,j} = \frac{\sum\limits_{u_k} w_{i,k} \cdot c_{k,j}}{\sum\limits_{u_k} w_{i,k}}, \tag{2.43}$$

where $w_{i,k}$ is the similarity between user i and k, and $c_{k,j} = 1$ if user k visited j, otherwise $c_{k,j} = 0$. Similarly, the prediction of $p^s_{i,j}$ can be estimated based on the idea of friend-based collaborative filtering:

$$p^s_{i,j} = \frac{\sum\limits_{u_k \in F_i} SI_{i,k} \cdot c_{k,j}}{\sum\limits_{u_k \in F_i} SI_{i,k}}, \tag{2.44}$$

where F_i is the set of u_i's friends, and $SI_{k,i}$ is the weight measuring social influence from u_k to u_i.

Finally, $p^g_{i,j}$ can be estimated from Eq. (2.41):

$$p^g_{i,j} = Pr[l_j|L_i] = \prod_{l_y \in L_i} Pr[d[l_j, l_y]]. \tag{2.45}$$

After we get the check-in probability estimation, we obtain the corresponding scores as follows:

$$\begin{aligned} S^u_{i,j} &= \frac{p^u_{i,j}}{Z^u_i}, \text{ where } Z^u_i = \max_{l_j \in L - L_i} \{p^u_{i,j}\}, \\ S^s_{i,j} &= \frac{p^s_{i,j}}{Z^s_i}, \text{ where } Z^s_i = \max_{l_j \in L - L_i} \{p^s_{i,j}\}, \\ S^g_{i,j} &= \frac{p^g_{i,j}}{Z^g_i}, \text{ where } Z^g_i = \max_{l_j \in L - L_i} \{p^g_{i,j}\}. \end{aligned} \tag{2.46}$$

Here Z^u_i, Z^s_i and Z^g_i are normalization terms.

2.4 Context-aware Recommendation

Contextual information is proven to be useful in recommender systems. The recent work in KDDCup 2012 [102, 27] show the effectiveness of utilizing context information for recommendations. In terms of employing context information, Baltrunas et al. [9] proposed a simple model that introduced a basis term for each context feature or item context interacting feature. More complicate methods like matrix factorization were also explored. Karatzoglou et al. [57] proposed a Multiverse recommendation model by modeling the data as a user-item-context N-dimension tensor. However, the computation complexity of this model is very high, which makes it impossible to be applied in large-scale datasets. Rendle et al. [106] proposed to apply the Factorization Machines (FM) model [101] to overcome the problem in Multiverse recommendation. The authors transformed the recommendation problem into a prediction problem and FM modeled all interactions between pairs of features.

In the following, we review two context-aware recommendation models: Multiverse recommendation model and Factorization Machines.

2.4.1 Multiverse Recommendation Model

In Multiverse recommendation model [57], besides user and item, the contextual variable C is considered as a new dimension. Then the user-item rating matrix is extended to a user-item-context tensor $Y \in \mathcal{Y}^{n\times m\times c}$, where n, m, c are the number of users, items and contextual variables respectively. $D \in \{0; 1\}^{n\times m\times c}$ is a binary tensor, where $D_{ijk} = 1$ if Y_{ijk} is observed. We denote U_{i*} as the entries of the i-th row of matrix U.

High Order Singular Value Decomposition (HOSVD) [59] is applied to decompose the user-item-context tensor. The 3-dimensional tensor is factorized into three matrices ($U \in \mathbb{R}^{n\times d_U}$, $M \in \mathbb{R}^{m\times d_M}$ and $C \in \mathbb{R}^{c\times d_C}$) and one central tensor ($S \in \mathbb{R}^{d_U\times d_M\times d_C}$). In this case, the prediction function for a single user i, item j and context k combination becomes:

$$\hat{Y}_{ijk} = S \times_U U_{i*} \times_M M_{j*} \times_C C_{k*}. \tag{2.47}$$

Here $\times_U$ is a tensor-matrix multiplication operator, where the subscript shows the direction on the tensor on which to multiply the matrix. For example, $T = Y \times_U U$ is $T_{ijk} = \sum_{i=1}^{n} Y_{ijk} U_{ij}$. The objective function for the model is defined as:

$$\mathcal{L} = L(\hat{Y}, Y) + \Omega(U, M, C) + \Omega(S). \tag{2.48}$$

Here, $L(\hat{Y}, Y)$ is the loss function:

$$L(\hat{Y}, Y) = \frac{1}{\|S\|_1} \sum_{i,j,k} D_{ijk} l(\hat{Y}_{ijk}, Y_{ijk}), \tag{2.49}$$

where $l : \mathbb{R} \times \mathcal{Y} \to \mathbb{R}$ is a point-wise loss function. $\Omega(U, M, C) + \Omega(S)$ is the regularization term that restricts the complexity of U, M, C and S:

$$\Omega(U, M, C) = \frac{1}{2}\lambda \left(\|U\|_F^2 + \|M\|_F^2 + \|C\|_F^2\right), \tag{2.50}$$

$$\Omega(S) = \frac{1}{2}\lambda_S \|S\|_F^2. \tag{2.51}$$

Stochastic gradient descent (SGD) can be applied to learn the parameters U, M, C and S. However, assuming that the latent dimension is k for all context features and there are total m context features, the time complexity to compute the prediction in Eq. (2.47) is $O(k^m)$, which is not very efficient in practice.

2.4.2 Factorization Machines

The Factorization Machines (FM) model [101] is a general predictor working with any real valued feature vector. FM models all interactions between variables using factorized parameters, which allows it to estimate interactions even in problems with huge sparsity. Since context information can be encoded as the features into the feature vector, FM can be applied in context-aware recommendation [106].

FM learns a rating prediction function $y : \mathbb{R}^n \to T$ from a real valued feature vector $x \in \mathbb{R}^n$ to a target domain T (e.g., $T = \mathbb{R}$ for regression or $T = \{+, -\}$ for classification). For rating data in recommender systems, T can be regarded as a subset of $\mathbb{R}$. For the implicit feedback data, the task can be regarded as classification. All the observations are treated as the training samples denoted as $D = \{(\boldsymbol{x}^{(1)}, y^{(1)}), (\boldsymbol{x}^{(2)}, y^{(2)}), \ldots\}$. The model equation for a factorization machine of degree $d = 2$ is defined as:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j, \tag{2.52}$$

where the model parameters that have to be estimated are: $w_0 \in \mathbb{R}$, $\boldsymbol{w} \in \mathbb{R}^n$ and $\boldsymbol{V} \in \mathbb{R}^{n \times k}$. $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size k:

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}. \tag{2.53}$$

A row $\boldsymbol{v}_i$ within $\boldsymbol{V}$ describes the i-th variable, and k is the latent dimension.

A 2-way FM (degree $d = 2$) captures all single and pairwise interactions between variables: w_0 is the global bias, w_i models the strength of the i-th variable, and $\hat{w}_{i,j} := \langle \mathbf{v}_i, \mathbf{v}_j \rangle$ models the interaction between the i-th and j-th variable.

The rating prediction in Eq. (2.52) can be computed in linear time, which is an appealing property. The last term in Eq. (2.52) can be reformulated as:

$$\begin{aligned}
&\sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j \\
&= \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j - \frac{1}{2} \sum_{i=1}^{n} \langle \mathbf{v}_i, \mathbf{v}_i \rangle x_i x_i \\
&= \frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} x_i \right) \left(\sum_{j=1}^{n} v_{j,f} x_j \right) - \sum_{i=1}^{n} v_{i,f}^2 x_i^2 \right) \\
&= \frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{n} v_{i,f} x_i \right)^2 - \sum_{i=1}^{n} v_{i,f}^2 x_i^2 \right).
\end{aligned} \tag{2.54}$$

We can see that the time complexity for the rating prediction function in Eq. (2.54) is $O(kn)$, which is in linear time.

Then similar to other matrix factorization methods, the final objective function for FM is:

$$\arg\min_{\Theta} \sum_{i=1}^{N} l(\hat{y}(\mathbf{x}_i), y) + \sum_{\theta \in \Theta} \lambda_{(\theta)} \theta^2, \tag{2.55}$$

where N is the number of total training samples and $\lambda_{(\theta)}$ is the regularization parameter. In practice, l can be the logit loss for binary classification or the least square loss for regression.

Note that, FM models all the pairwise interactions of context features. In practice, not all of the features are useful. We need to find out useful features from tens of contextual features, which will be detailed in Chapter 5.

□ **End of chapter.**

Chapter 3

A Unified Point-of-interest Recommendation Framework in Location-based Social Networks

In the past few years, Location-based social networks (LBSNs) become popular as millions of users would like to share their social friendship and their locations on them. Plenty of valuable information is accumulated based on the check-in behaviors, which makes it possible to learn users' moving patterns as well as their preferences. In LBSNs, point-of-interest (POI) recommendation is one of the most significant tasks since it can help targeted users explore their surroundings as well as help third-party developers provide personalized services. Matrix factorization is a promising method for this task since it can capture users' preferences to locations and is widely adopted in traditional recommender systems such as movie recommendation. However, the sparsity of the check-in data makes it difficult to capture users' preferences accurately. Geographical influence can help alleviate this problem and have a large impact on the final recommendation result. By studying users' moving patterns, we find that users tend to check in around several centers and different users have different numbers of centers. Based on this, we propose a Multi-center Gaussian Model (MGM) to capture this pattern via modeling the probability of a user's check-in on a location.

Moreover, users are usually more interested in the top 20 or even top 10 recommended POIs, which makes personalized ranking important in this task. From previous work, directly optimizing for pairwise ranking like Bayesian Personalized Ranking (BPR) achieves better performance in the top-k recommendation than directly using matrix matrix factorization that aims to minimize the point-wise rating error. To consider users' preferences, geographical influence and personalized ranking, we propose a unified POI recommendation framework, which unifies all of them together. Specifically, we first fuse MGM with matrix factorization methods and further with BPR using two different approaches. We conduct experiments on Gowalla and Foursquare datasets, which are two large-scale real world LBSNs datasets publicly available online. The results on both datasets show that our unified POI recommendation framework can produce better performance.

3.1 Introduction

Recently, with the rapid development of mobile devices and ubiquitous Internet access, location-based social services become prevalent. Online LBSNs such as Gowalla, Foursquare, etc., have attracted millions of users to share their social friendship, experiences and tips of POIs via check-in behaviors. These information pieces embed abundant hints of users' preferences on locations. The information not only can be utilized to help a specific user explore new places of the city, but also can facilitate third-parties such as advertisers to provide specific advertisements for the recommended positions. Hence, POI recommendation becomes a significant task in LBSNs.

To solve the POI recommendation task in LBSNs, matrix factorization is a promising tool since it is a widely adopted method in traditional recommender systems such as movie recommendation [112]. We first construct the user-location matrix, whose entry is the visiting frequency of a user to a location. Then we can obtain the user's preference on locations by performing matrix factorization on the user-location matrix. However, the extreme sparsity of the user-location matrix makes it difficult to capture the user's preference accurately. In our crawled Gowalla dataset, for example, the density of the user-location matrix is only 2.08×10^{-4}.

Fortunately, due to the availability of geographical information (i.e., latitude and longitude) of POIs, researchers can study users' moving patterns and leverage this geographical influence to help improve POI recommendation. In Ye et al. [143], geographical influence is considered by assuming a power-law distribution between the check-in probability and the distance along the whole check-in history. The parameters of the power-law distribution are learned based on all users' histories, thus they are not personalized. In this chapter, we carefully study each user's movement and find that users tend to check in around several centers and different users have different number of centers. We refer to this as multi-center check-in behavior. Based on this finding, we propose a Multi-center Gaussian Model (MGM) to capture this movement pattern. For each user, we will extract the centers based on his/her check-ins. Then for a new location to the user, we define the probability based on the user's centers.

Moreover, in real mobile app recommendation scenarios, users are usually more interested in the top 20 or even top 10 recommended POIs, which makes personalized ranking important in this task. Most of previous work on POI recommendation was mainly based on matrix factorization that minimized the point-wise prediction error for each entry in the user-location matrix. From previous work [104], directly optimizing for pairwise ranking like Bayesian Personalized Ranking (BPR) produces better performance in the top-k recommendation than directly using matrix factorization. To address the top-k ranking as well as the geographical influence, we propose two methods based on BPR, a state-of-the-art personalized ranking method, with different integration approaches.

To our best knowledge, this is the first piece of work to combine the Multi-center Gaussian Model with matrix factorization and BPR into a unified framework in LBSNs, which explores users' preferences, geographical influence and personalized ranking in POI recommendation. Our contributions are threefold. First, we mine a large-scale dataset crawled from Gowalla and extract the characteristics to find out the multi-center check-in behavior. Second, based on the data properties, we model the probability of a user's check-in on a location as a Multi-center Gaussian Model (MGM). This is different from the early POI recommendation work in LBSNs [143], which assumed a power-law distribution of the check-in probability with respect to the distance within the whole check-in history. Third, we propose a unified POI recommendation framework to fuse users' preferences, geographical influence and personalized ranking together. Our experimental results on two large-scale real-world online LBSNs datasets show that the unified POI recommendation framework presented in this chapter can achieve significantly better performance than other state-of-the-art methods.

3.2 Related Work

The work in this chapter is closely related to POI recommendation and ranking-oriented collaborative filtering (CF). In the following, we briefly review the related work.

3.2.1 Point-of-interest Recommendation

Location-based service (LBS) research became prevalent due to a wide range of potential applications, e.g., personalized marketing strategy analysis [139], personalized behavior study [82], POI recommendation [151], etc. In particular, POI recommendation has attracted much research interest in recent years [56, 50, 152, 149, 69].

In the following, we review several main approaches in collaborative filtering community.

One line of research is to solve POI recommendation based on the extracted stay points from GPS trajectory logs of several hundred monitored users [152, 149, 148, 69, 150, 24]. In [149], three matrices, i.e., location-activity matrix, location-feature matrix and activity-activity matrix, were constructed. Based on the three matrices, a collective matrix factorization method was proposed to mine POIs and activities. Zheng et al. [69] explored a tensor factorization on the user-location-activity tensor to provide POI recommendation. In [69], a memory-based method called Collaborative Location Model (CLM) was proposed to incorporate activity to facilitate the recommendation.

The other line of work centers on POI recommendation based on the LBSNs data [142, 143]. A pioneer task of POI recommendation in LBSNs debuted in [142]. The work has been extended and further studied in [143]. More specifically, geographical influence is considered by assuming a power-law distribution between the check-in probability and the distance along the whole check-in history [143]. However, the paper ignored the user's multi-center check-in behavior. Moreover, the proposed method had to compute all pairwise distances of the whole visiting history, which was very time consuming. Temporal information has also been considered to improve POI recommendation. In [39], temporal non-uniformness and temporal consecutiveness were addressed to model temporal cyclic patterns of check-ins. Geographical and temporal information were incorporated together in [145]. Apart from temporal information, content information has been studied as well. Liu et al. [76] employed an aggregated LDA model to study the effect of POI related tags. In [40], three types of content information are investigated and they were modeled into a unified POI recommendation framework.

3.2.2 Ranking-oriented Collaborative Filtering

Top-k recommendation has been studied in collaborative filtering in the past few years. CofiRank [134] was the first proposed ranking-oriented CF approach, which introduced structured ranking loss into the collaborative filtering framework. Bayesian personalized ranking (BPR) [104] was proposed as a state-of-the-art recommendation algorithm for situations with binary relevance data. The optimization criterion of BPR was essentially based on pairwise comparisons between relevant and a sample of irrelevant items. Several methods were explored to optimize directly the ranking metrics. In [124], the CF model directly maximized the Mean Reciprocal Rank (MRR) and [123] proposed a model that directly maximized Mean Average Precision (MAP) with the aim of creating an optimally ranked list of items for individual users under a given context. Learning to rank techniques have also been applied in ranking-oriented CF. In [8], the authors proposed to use user and item latent vectors as the feature vector in a learning-to-rank framework. Volkovs et al. [133] further proposed an efficient method to extract a good feature vector, which was used by the learning-to-rank framework later with only 17 parameters.

In summary, the GPS dataset is usually in small-scale with about one or two hundred users, but the data are very dense. Contrarily, the LBSNs dataset is in large-scale with thousands of users, but the data are very sparse [94, 120]. To solve large-scale recommendation problems, matrix factorization is a promising tool due to its success in Netflix competition [14, 63]. However the data sparsity of LBSNs data makes the results of matrix factorization inaccurate. Moreover, traditional matrix factorization approaches do not consider the geographical influence, which has a great effect on POI recommendation. Besides, the final purpose of POI applications is to recommend a few top locations, where the ranking performance is important in this task, while previous work does not emphasize personalized ranking in POI recommendation. In this chapter, we propose a unified POI recommendation framework that incorporates user preference, geographical influence as well as personalized ranking together.

3.3 Check-in Data Characteristics

In this chapter, we conduct experiments on two publicly available online LBSNs datasets: Gowalla and Foursquare. Gowalla is an LBSNs website created in 2009 for users to check in to various locations through mobile devices. We collect a complete snapshot, including users' profile, users' check-in locations, check-in time stamps, users' friend lists and location details, from Gowalla during the period from February 2009 to September 2011 via the provided public API. To reduce noise in data, we remove users with less than 10 check-ins and locations with less than 20 visits. Foursquare is another LBSNs website similar to Gowalla. We use the four month Foursquare dataset which spans from May 2010 to August 2010 provided by [32]. Similarly, in order to remove noise, we require that all users should have at least 10 check-ins. But we do not have the social information in the provided Foursquare dataset. The basic statistics of the datasets are summarized in Table 3.1. In the table, we use tilde to denote the average count.

Details of the data are depicted in the following:

- The Gowalla dataset has $4{,}128{,}714$ check-ins from $53{,}944$ users on $367{,}149$ locations and totally $306{,}958$ edges are in the whole users' social graph. The density of the user-location matrix in the Gowalla dataset is about 2.08×10^{-4}. Table 3.2 is an illustration of the user-location matrix. On the other hand, the Foursquare dataset consists of $6{,}084$ users, $37{,}976$ locations with $218{,}935$ check-ins. The density of the Foursquare dataset is about 9.48×10^{-4}.

Table 3.1: Basic statistics of the Gowalla and Foursquare dataset for POI recommendation

$\#U$	$\#L$	$\#E$
$53,944$	$367,149$	$306,958$
$\#\widetilde{U}$	$\#\widetilde{L}$	$\#\widetilde{E}$
51.33	7.54	11.38
$\#$max. U	$\#$max. L	$\#$max. E
$2,145$	$3,581$	$2,366$

(a) Gowalla

$\#U$	$\#L$
$6,084$	$37,976$
$\#\widetilde{U}$	$\#\widetilde{L}$
35.98	5.76
$\#$max. U	$\#$max. L
182	985

(b) Foursquare

Table 3.2: User-location check-in frequency matrix

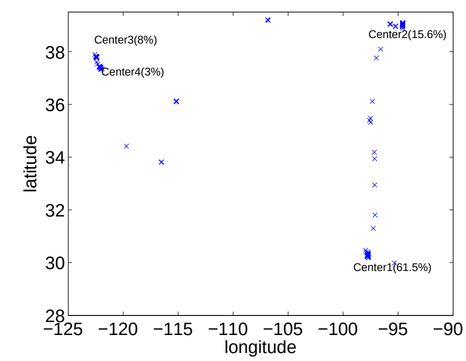
	l_1	l_2	l_3	l_4	l_5	l_6	$\cdots$	$l_{\lvert\mathcal{L}\rvert-1}$	$l_{\lvert\mathcal{L}\rvert}$
u_1	?	?	164	?	1	?	$\cdots$	?	1
u_2	40	2	?	?	?	1	$\cdots$	?	?
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$
$u_{\lvert\mathcal{U}\rvert-1}$	?	?	1	1	?	?	$\cdots$	2	?
$u_{\lvert\mathcal{U}\rvert}$	?	2	?	?	1	?	$\cdots$	?	10

- The average number of visited locations of a user is 51.33 and 35.98 for the Gowalla and Foursquare dataset, respectively. The average number of visiting users for a location is 7.54 in the Gowalla dataset and 5.76 in the Foursquare dataset. The average number of friends of a user is 11.38 in the Gowalla dataset.
- In the Gowalla dataset, the maximum number of locations for a user is $2{,}145$; in the Foursquare dataset, the maximum number is 182. The maximum number of visiting users for a location is $3{,}581$ for Gowalla and 985 for Foursquare. The maximum number of friends of a user is $2{,}366$.

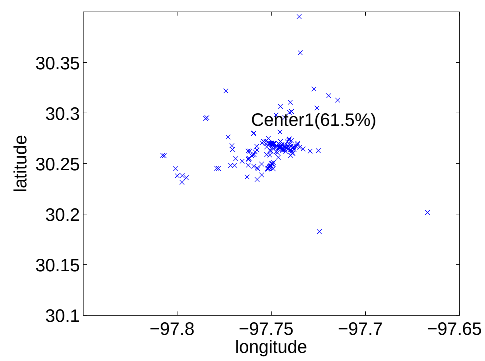
In the following, we further study the location distribution, frequency distribution and the social relationship among users' check-ins. Since Gowalla and Foursquare share similar characteristics, we only show the results from Gowalla.

3.3.1 Location Distribution

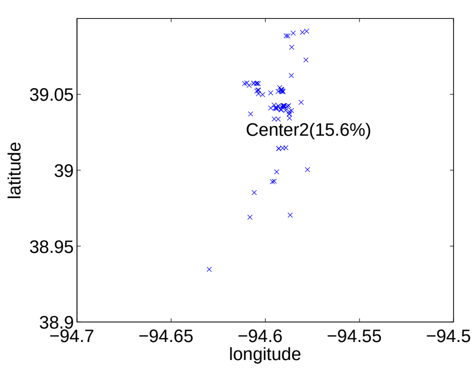
Figure 3.1(a) shows the longitude and latitude of a typical user's check-in locations, where the locations form four centers. The details of each center are further shown in Figure 3.1(b)-3.1(d). This observation reaches our assumption differently from the power-law distribution on users' check-in histories in [143]. In addition, our statistics are also a little different from the two states ("home" and "office") check-in behavior mentioned in [33]. After examining the comments of locations, we find that other than the centers of "home" and "office" (counting above half of a user's check-ins), other centers count at least 10% of the check-ins. These centers may be a user's usual business travel places, e.g., an office of a branch of a large company or vocation places, which provide abundant information that needs to be differentiated. This means for each user, there may exist several centers around which the user would like to conduct activities. Note that the POIs near these centers have a higher chance to be checked in than the POIs which are far away. It reflects



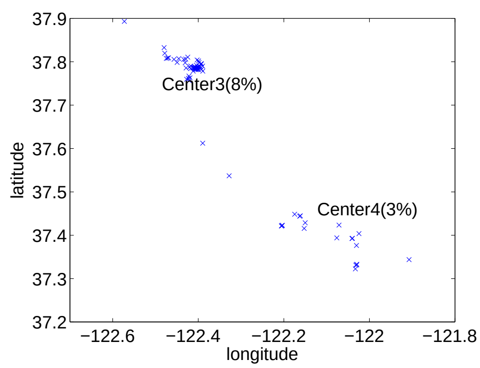
(a) Multi-center overview



(b) Center 1

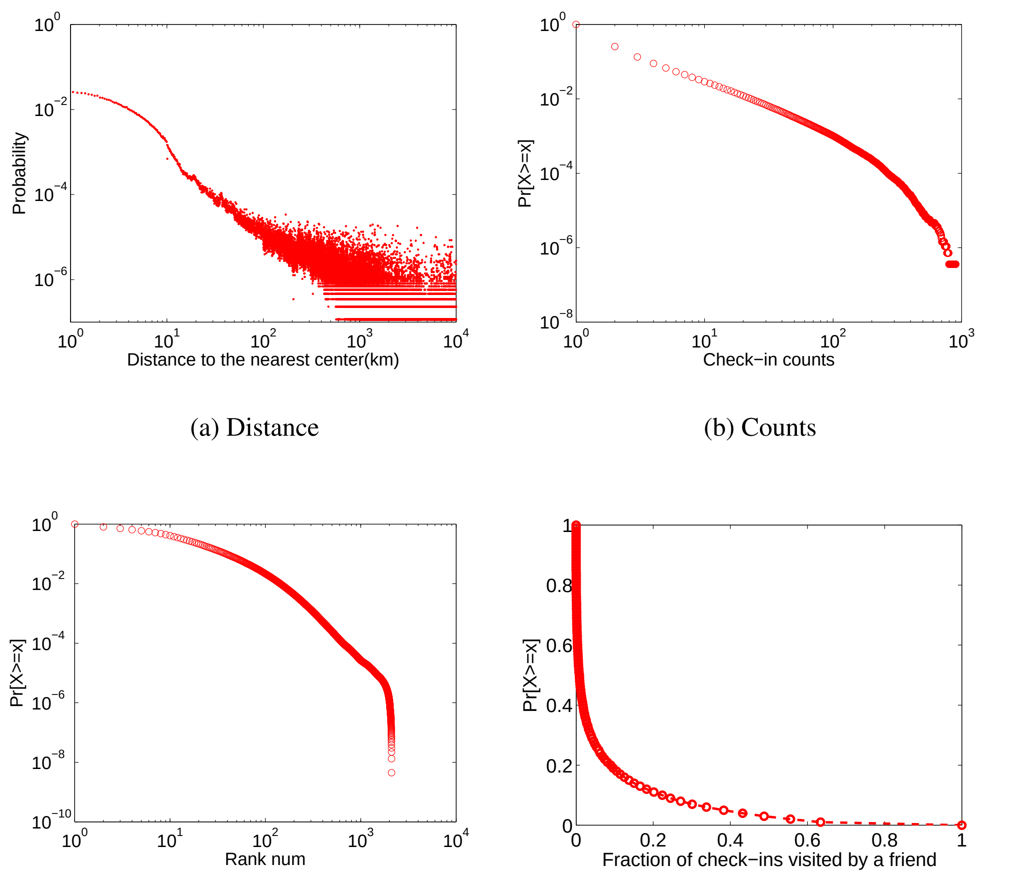


(c) Center 2



(d) Center 3 and 4

Figure 3.1: A typical user's multi-center check-in behavior



(a) Distance

(b) Counts

(c) Top-k locations

(d) Friends

Figure 3.2: Check-ins probability vs. distance, counts, top-k locations, common check-ins of friends

the fact that most of the time human beings hang out around several familiar areas.

3.3.2 Frequency Distribution

Figure 3.2(b) plots the Complementary Cumulative Distribution Function (CCDF) for each user's check-in numbers at each location. It is shown that about 74% of locations are only visited once and only about 3% of locations are visited more than 10 times. This means that users usually visit several important places, e.g., home, office and some stores, with very high frequency, while most of other places are seldom visited. Overall, these places are around several centers. Figure 3.2(c) further shows the CCDF function of top-k frequently visited locations. The most visited location accounts for about 18.8% of all users' check-ins. The top 10 most visited locations account for 68% of all check-ins and the ratio increases to 80.5% for the top-20 most visited locations, following the *Pareto principle* (a.k.a. 80-20 rule) [116].

3.3.3 Social Influence

In the dataset, we find that the average overlap of a user's check-ins to his/her friends' check-ins is only 9.6%. This indicates that less than 10% of a user's check-ins are also visited by the user's friends, which is similar to the statistics reported in [33]. Figure 3.2(d) plots the CCDF of the fraction of a user's check-ins that are visited by his/her friends. It is known that for about 38% of users, their check-in locations are not checked in by their friends, while almost 90% of users contain less than 20% of common check-ins with their friends. The statistics are a little different from that in [33], but the overall trend is similar. These observations imply that social relationship has a limited effect on users' check-ins, but it still cannot be ignored.

3.4 Unified Point-of-interest Recommendation Framework

The problem of personalized POI recommendation is defined as follows: given a partially observed user-location check-in frequency matrix (users in $\mathcal{U}$ and locations in $\mathcal{L}$) and users' social relationship $\mathcal{F}$, the task is to recommend top-k locations to a user that the user does not visit before. To solve this problem, we first propose a personalized Multi-center Gaussian Model (MGM) to capture the geographical influence on a user's check-ins. Then we depict the matrix factorization, consider the social information, and propose a fused MF framework to include geographical influence. Finally, we introduce the unified framework, which incorporates geographical influence and matrix factorization to directly optimize the ranking loss for POI recommendation.

3.4.1 Multi-center Gaussian Model (MGM)

A significant characteristic of check-in locations is that they are usually located around several centers as shown in Figure 3.1. The second characteristic of check-in locations is that the probability of a user visiting a location is inversely proportional to the distance from its nearest center; see Figure 3.2(a).

These two characteristics indicate that geographical information plays a strong influence on the user's check-in behavior. Based on the statistics from Figure 3.1 and Figure 3.2(a), we adopt Gaussian distribution to model the user's check-in behavior and propose the Multi-center Gaussian Model (MGM). That is, the probability of a user u, visiting a POI l, given the multi-center set C_u, is defined by:

$$P(l|C_u) = \sum_{c_u=1}^{|C_u|} P(l \in c_u) \frac{f_{c_u}^{\alpha}}{\sum_{i \in C_u} f_i^{\alpha}} \frac{\mathcal{N}(l|\mu_{c_u}, \Sigma_{c_u})}{\sum_{i \in C_u} \mathcal{N}(l|\mu_i, \Sigma_i)}. \qquad (3.1)$$

Here, l denotes the longitude and latitude of a position, C_u is the set of centers for the user u, and i is one center in the set C_u. For each center, calculating Eq. (3.1) consists of the multiplication of three terms:

- $P(l \in c_u) \propto 1/dist(l, c_u)$ determines the probability of the location l which belongs to the center c_u, which is inversely proportional to the distance between the location l and the center c_u.

- The second term denotes the normalized effect of check-in frequency f_{c_u}, on the center c_u. The parameter $\alpha \in (0, 1]$ is introduced to maintain the frequency aversion property, where very high check-in frequency does not play too significant effect.

- The third term denotes the normalized probability of a location belonging to the center c_u, where $\mathcal{N}(l|\mu_{c_u}, \Sigma_{c_u})$ is the probability density function of the Gaussian distribution, and μ_{c_u} and Σ_{c_u} correspond to the mean and covariance matrices of regions around the center c_u.

Next we introduce how to find the centers for each individual user. We propose a greedy clustering algorithm among the check-ins due to the Pareto principle [116], which is very efficient. The computational complexity is linear to the number of observations in the user-location matrix. This property can be observed from Figure 3.1 and Figure 3.2(c). There are several more advanced techniques to calculate data similarity, which can be referred to [141]. But most of them are not efficient for large-scale datasets.

In our greedy algorithm, we first scan from the most visited POIs and combine all other visited check-in locations, whose distance is less than d kilometers from the selected POI, into a region. If the ratio of the total check-in number of this region to the user's total check-in amount is greater than a threshold θ, we set these check-in positions as a region and determine its center. Algorithm 1 shows the procedure of discovering multiple centers. In our experiments,

Algorithm 1 Multi-center Discovering Algorithm

for all user i in the user set $\mathcal{U}$ **do**
  Rank all check-in locations in $|\mathcal{L}|$ according to visiting frequency
  $\forall l_k \in L$, set $l_k.center = -1$;
  Center_list $= \emptyset$; center_no = 0;
  for $i = 1 \to |L|$ **do**
    if $l_i.center == -1$ **then**
      center_no++; Center $= \emptyset$; Center.total_freq = 0;
      Center.add(l_i); Center.total_freq += l_i.freq;
      for $j = i + 1 \to |L|$ **do**
        if $l_j.center == -1 \; and \; dist(l_i, l_j) \leq d$ **then**
          $l_j.center =$ center_no; Center.add(l_j);
          Center.total_freq += l_j.freq;
        end if
      end for
      if Center.total_freq $\geq |u_i|$.total_freq * θ **then**
        Center_list.add(Center);
      end if
    end if
  end for
  RETURN Center_list for user i;
end for

by trial on the training dataset, we set θ to 0.02, the the distance threshold d to 15 and the frequency control parameter α to 0.2.

3.4.2 Matrix Factorization

Matrix Factorization (MF) is one of the most popular methods for recommender systems [112, 113, 14, 63]. It has been shown to be particularly effective in recommender systems as well as in the well-known Netflix prize competitions. Given the partially observed entries in a $|\mathcal{U}| \times |\mathcal{L}|$ frequency matrix F, the goal of MF is to find two low-rank matrices $U \in \mathbb{R}^{K \times |\mathcal{U}|}$ and $L \in \mathbb{R}^{K \times |\mathcal{L}|}$ such that $F \approx U^T L$. The predicted probability of a user u who is likely to visit a location l is determined by

$$P(F_{ul}) \propto U_u^T L_l. \tag{3.2}$$

Probabilistic Matrix Factorization (PMF)

PMF is one of the most famous MF models in collaborative filtering, which is proposed in [112]. It assumes that the conditional distribution over the observed rating is:

$$p(F|U, L, \sigma_R^2) = \prod_{i=1}^{|\mathcal{U}|} \prod_{j=1}^{|\mathcal{L}|} [\mathcal{N}(F_{ij}|U_i^T L_j, \sigma_R^2)]^{I_{ij}^R}, \tag{3.3}$$

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2. I_{ij}^R is the indicator function that equals to 1 if user u_i has visited location l_j and equals to 0 otherwise. The zero-mean spherical Gaussian priors are also placed on user and location latent feature vectors:

$$p(U|\sigma_U^2) = \prod_{i=1}^{|\mathcal{U}|} \mathcal{N}(U_i|0, \sigma_U^2 \boldsymbol{I}), p(L|\sigma_V^2) = \prod_{j=1}^{|\mathcal{L}|} \mathcal{N}(L_j|0, \sigma_V^2 \boldsymbol{I}). \tag{3.4}$$

Through Bayesian inference, we have the following objective function:

$$\min_{U,L} \frac{1}{2} \sum_{i=1}^{|\mathcal{U}|} \sum_{j=1}^{|\mathcal{L}|} I_{ij}^R (F_{ij} - U_i^T L_j)^2 + \lambda_1 \|U\|_F^2 + \lambda_2 \|L\|_F^2, \tag{3.5}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. In practice, we can use the sigmoid function $g(x) = 1/(1 + \exp(-x))$ to convert the rating into $(0, 1)$. Now the objective functions becomes:

$$\min_{U,L} \sum_{i=1}^{|\mathcal{U}|} \sum_{j=1}^{|\mathcal{L}|} I_{ij} (g(F_{ij}) - g(U_i^T L_j))^2 + \lambda_1 \|U\|_F^2 + \lambda_2 \|L\|_F^2. \tag{3.6}$$

Note: The observed frequency data are all positive, which makes the data biased. Consequently, it is a standard one-class collaborative filtering problem [96, 95, 51]. We sample the same number of unobserved data from the rest matrix and deem their frequency as 0.

PMF with Social Regularization (PMFSR)

In both real world and online world, we usually turn to our friends for suggestions. This gives us a hint that social information can be beneficial to recommender systems. It has been shown to be useful in recommender systems [88, 154, 90, 89]. The main idea is that users and their friends are assumed to have similar taste in some degree according to their similarity. We adopt the PMF with Social Regularization (PMFSR) [89], where the Individual-based Regularization Model proposed to impose constraints between one user and his/her friends individually. The objective function is defined as follows:

$$\begin{aligned}\min_{U,L} \Omega(U, L) &= \sum_{i=1}^{|\mathcal{U}|}\sum_{j=1}^{|\mathcal{L}|} I_{ij}(g(F_{ij}) - g(U_i^T L_j))^2 \\ &+ \beta \sum_{i=1}^{|\mathcal{U}|}\sum_{f\in\mathcal{F}(i)} Sim(i,f)\|U_i - U_f\|_F^2 \\ &+ \lambda_1\|U\|_F^2 + \lambda_2\|L\|_F^2, \end{aligned} \tag{3.7}$$

where $\mathcal{F}(i)$ is the set of friends for user u_i, and $Sim(i, f)$ is the similarity between user u_i and his/her friend u_f. The similarity between a user and the user's friends can be computed by measuring the check-ins of them. There are two very popular methods we can borrow from the literature namely Vector Space Similarity (VSS) and Pearson Correlation Coefficient (PCC) [17].

VSS can be used to define the similarity between a user i and his/her friend f based on their common check-ins:

$$Sim(i, f) = \frac{\sum_{j\in I(i)\cap I(f)} F_{ij} \cdot F_{fj}}{\sqrt{\sum_{j\in I(i)\cap I(f)} F_{ij}^2} \cdot \sqrt{\sum_{j\in I(i)\cap I(f)} F_{fj}^2}}, \tag{3.8}$$

where j is the location where user i and his/her friend f both checked in. A larger value of VSS means that the two users are more common.

The drawback of VSS calculation is that it does not consider the different check-in styles of different users. Some users could be much more active and produce lots of check-ins. Hence PCC is proposed to solve this problem by adjusting check-in frequency with users' mean check-in frequency:

$$Sim(i,f) = \frac{\sum_{j \in I(i) \cap I(f)} (F_{ij} - \bar{F}_i) \cdot (F_{fj} - \bar{F}_f)}{\sqrt{\sum_{j \in I(i) \cap I(f)} (F_{ij} - \bar{F}_i)^2} \cdot \sqrt{\sum_{j \in I(i) \cap I(f)} (F_{fj} - \bar{F}_f)^2}}, \tag{3.9}$$

where $\bar{F}_i$ denotes the average check-in frequency of user i. We can see that since the PCC value is ranged from $[-1, 1]$, we can use a mapping function $f(x) = (x + 1)/2$ to map the value to $[0, 1]$. In this chapter, we use PCC to calculate the user similarity.

Probabilistic Factor Models (PFM)

The PMF model makes assumption on the Gaussian distribution, which may not be appropriate when applied to the frequency data. This is demonstrated in our later experiment results. Since the check-in data in LBSNs are naturally frequency, we turn to Probabilistic Factor Models (PFM) [28, 86], which can model the frequency data directly.

PFM places Gamma distributions as priors on the latent matrices U and L, while it defines a Poisson distribution on the frequency. Gamma distribution is suitable for modeling nonnegative values, while Gaussian distribution can model both negative and non-negative values. If we use Gaussian distribution, the model will generate negative frequency values, which is unreasonable in the real world.

The generative process of the check-in frequecy f_{ij} is as follows:

1. $\forall k$, Generate $u_{ik} \sim \text{Gamma}(\alpha_k, \beta_k)$.

2. $\forall k$, Generate $l_{jk} \sim \text{Gamma}(\alpha_k, \beta_k)$.

3. Generate the check-in frequency $f_{ij} \sim \text{Possion}(y_{ij})$, where $y_{ij} = \sum_{k=1}^{K} u_{ik} l_{jk}$.

Since the latent vectors of U and L follow the Gamma distribution, we have:

$$p(U|\boldsymbol{\alpha},\boldsymbol{\beta}) = \prod_{i=1}^{|\mathcal{U}|}\prod_{k=1}^{K} \frac{u_{ik}^{\alpha_k-1} \exp -u_{ik}/\beta_k}{\beta_k^{\alpha_k}\Gamma(\alpha_k)}, \tag{3.10}$$

$$p(L|\boldsymbol{\alpha},\boldsymbol{\beta}) = \prod_{j=1}^{|\mathcal{L}|}\prod_{k=1}^{K} \frac{v_{jk}^{\alpha_k-1} \exp -v_{jk}/\beta_k}{\beta_k^{\alpha_k}\Gamma(\alpha_k)}, \tag{3.11}$$

where $\boldsymbol{\alpha} = \{\alpha_1, \ldots, \alpha_K\} > \mathbf{0}_K$ and $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_K\} > \mathbf{0}_K$ are parameters for Gamma distributions. K is the latent dimension, and $\Gamma(\cdot)$ is the Gamma function. The Poisson distribution of F given Y can then be defined as:

$$p(F|Y) = \prod_{i=1}^{|\mathcal{U}|}\prod_{j=1}^{|\mathcal{L}|} \frac{y_{ij}^{f_{ij}} \exp(-y_{ij})}{f_{ij}!}. \tag{3.12}$$

Since $Y = U^T L$, the posterior distribution of U and L given F can be modeled as:

$$p(U, L|F, \boldsymbol{\alpha}, \boldsymbol{\beta}) \propto p(F|Y)p(U|\boldsymbol{\alpha},\boldsymbol{\beta})p(L|\boldsymbol{\alpha},\boldsymbol{\beta}). \tag{3.13}$$

Taking the log of posterior distribution, which leads to seeking U and L by minimizing $\Psi(U, L; F)$:

$$\begin{aligned}\Psi(\cdot,\cdot;\cdot) = &\sum_{i=1}^{|\mathcal{U}|}\sum_{k=1}^{K}((\alpha_k - 1)\ln(U_{ik}/\beta_k) - U_{ik}/\beta_k) \\ &+ \sum_{j=1}^{|\mathcal{L}|}\sum_{k=1}^{K}((\alpha_k - 1)\ln(L_{jk}/\beta_k) - L_{jk}/\beta_k) \\ &+ \sum_{i=1}^{|\mathcal{U}|}\sum_{j=1}^{|\mathcal{L}|}(F_{ij}\ln(U^T L)_{ij} - (U^T L)_{ij}) + c, \end{aligned} \tag{3.14}$$

where c is a constant term derived from the posterior distribution.

3.4.3 A Fusion Framework with User Preference and Geographical Influence

We can observe that either PMF, PMFSR or PFM only models users' preferences on locations. They do not explore the geographical influence. As observed from Figure 3.2(a), users tend to check in locations around their centers. It can be very helpful for POI recommendation, especially when we have very few check-ins, where matrix factorization does not perform very well. Hence, we fuse users' preferences on a POI and the probability from MGM together to determine the probability of a user u visiting a location l, which is defined as follows:

$$P_{ul} = P(F_{ul}) \cdot P(l|C_u), \tag{3.15}$$

where $P(l|C_u)$ is calculated by Eq. (3.1) via MGM and $P(F_{ul})$ encodes users' preferences on a location determined by Eq. (3.2). After we get the final predicted value P_{ul}, we can obtain a ranked list of recommended POIs for user u. Finally, we recommend the top k locations to the user.

3.4.4 A Final Fusion Framework

Since our final goal is to recommend a ranking POI list to users, directly optimizing the ranking loss is desirable. Bayesian Personalized Ranking (BPR) [104] is a state-of-the-art method that tries to minimize the pairwise ranking loss over user rated items and unrated items. On the other hand, geographical influence has a great effect on POI recommendation; therefore, we propose two methods to incorporate MGM with BPR, which combine pairwise ranking with geographical effect together. In the following we describe BPR model first, then we detail the two combined location ranking methods.

Bayesian Personalized Ranking (BPR)

In LBSNs, all the check-ins are implicit feedback data, which means we only observe the positive data. The unobserved data, i.e., the missing user-location pairs, are a mixture of real negative feedback (the user is not interested in visiting the location) and missing values (the user might want to check in the location but has not visited there).

In BPR, the task is to derive a personalized ranking $>_u$ over locations for each user u. The basic assumption is that if user u checks in location i while not checking in location j, we say the user prefers location i over location j, denoted as $i >_u j$. We assume that there is an estimator $\hat{x} : U \times L \to \mathbb{R}$, which is used to define the ranking:

$$i >_u j \Leftrightarrow \hat{x}_{ui} > \hat{x}_{uj}. \tag{3.16}$$

The estimator $\hat{x}$ is usually calculated through matrix factorization:

$$\hat{x}_{ui} = U_u^T L_i. \tag{3.17}$$

The Bayesian formulation of finding the correct personalized ranking for all locations in $\mathcal{L}$ is to maximize the following posterior probability:

$$p(\Theta| >_u) \propto p(>_u |\Theta) p(\Theta), \tag{3.18}$$

where Θ represents the parameters.

We further assume that all users are independent and the ordering of each location pairs (i, j) for a specific user is also independent. Thus, the likelihood function for all users can be defined as:

$$\prod_{u \in \mathcal{U}} p(>_u |\Theta) = \prod_{(u,i,j) \in S} p(i >_u j|\Theta), \tag{3.19}$$

where $S = \{(u, i, j) | u \in \mathcal{U}, i \in \mathcal{L}_u^+ \wedge j \in \mathcal{L} \setminus \mathcal{L}_u^+\}$, and $\mathcal{L}_u^+$ is the set of locations visited by user u.

The individual probability of user u preferring location i to location j is defined as:

$$p(i >_u j|\Theta) = \sigma(\hat{x}_{uij}(\Theta)), \tag{3.20}$$

where σ is the logistic sigmoid function $\sigma(x) = 1/(1 + \exp(-x))$, and

$$\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj}. \tag{3.21}$$

We further place a Gaussian prior over the parameters:

$$p(\Theta) \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}). \tag{3.22}$$

We use maximum a posterior (MAP) to estimate the parameters:

$$\arg\max_{\Theta} \ln \prod_{u \in \mathcal{U}} p(>_u |\Theta) p(\Theta). \tag{3.23}$$

Substituting Eq. (3.20) and Eq. (3.21) into Eq. (3.23), we have the final objective function:

$$\arg\max_{\Theta} \sum_{(u,i,j) \in S} \ln(\sigma(\hat{x}_{ui} - \hat{x}_{uj})) - \lambda_{\Theta} \|\Theta\|^2. \tag{3.24}$$

Stochastic gradient descent (SGD) can be applied to learn the model parameters Θ. We denote $\mathcal{F}$ as the objective function in Eq. (3.24). The gradient of $\mathcal{F}$ with respect to the model parameters is:

$$\begin{aligned}
\frac{\partial \mathcal{F}}{\partial \Theta} &= \sum_{(u,i,j) \in S} \frac{\partial}{\partial \Theta} \ln(\hat{x}_{ui} - \hat{x}_{uj}) - \lambda_{\Theta} \frac{\partial}{\partial \Theta} \|\Theta\|^2 && (3.25) \\
&\propto \sum_{(u,i,j) \in S} (1 - \sigma(\hat{x}_{uij})) \cdot \frac{\partial}{\partial \Theta} (\hat{x}_{ui} - \hat{x}_{uj}) - \lambda_{\Theta} \Theta && (3.26) \\
&= \sum_{(u,i,j) \in S} (1 - \sigma(\hat{x}_{uij})) \cdot \frac{\partial}{\partial \Theta} (U_u^T L_i - U_u^T L_j) - \lambda_{\Theta} \Theta. && (3.27)
\end{aligned}$$

Here $\Theta = \{U, L\}$. Note that

$$\begin{aligned}
\frac{\partial}{\partial U_u} (U_u^T L_i - U_u^T L_j) &= L_i - L_j, && (3.28) \\
\frac{\partial}{\partial L_i} (U_u^T L_i - U_u^T L_j) &= U_u, && (3.29) \\
\frac{\partial}{\partial L_j} (U_u^T L_i - U_u^T L_j) &= -U_u. && (3.30)
\end{aligned}$$

Algorithm 2 Learning Algorithm for BPRLR2

draw U,L from $\mathcal{N}(0, \sigma^2)$
repeat
  draw (u, i, j) uniformly from S'
  Calculate $\sigma(\hat{x}_{uij})$
  Update U_u, L_i, L_j according to:
  $U_u = U_u + \alpha\,((1 - \sigma(\hat{x}_{uij}) \cdot (L_i - L_j) - \lambda_\Theta U_u)$
  $L_i = L_i + \alpha\,((1 - \sigma(\hat{x}_{uij}) \cdot (U_u) - \lambda_\Theta L_i)$
  $L_j = L_j + \alpha\,((1 - \sigma(\hat{x}_{uij}) \cdot (-U_u) - \lambda_\Theta L_j)$
until convergence
return U,L

For each triple (u, i, j) we draw from S, the update rule is:

$$\Theta \leftarrow \Theta + \alpha \left((1 - \sigma(\hat{x}_{uij})) \cdot \frac{\partial}{\partial \Theta}(\hat{x}_{ui} - \hat{x}_{uj}) - \lambda_\Theta \Theta \right), \quad (3.31)$$

where α is the step size.

Ranking in POI Recommendation

We propose two methods to incorporate Bayesian Personalized Ranking with geographical influence. The first method is the same as the fuse framework in Section 3.4.3. The final probability that user u visits a location l is consequently defined as

$$P_{ul} = \hat{x}_{ul} \cdot P(l|C_u), \quad (3.32)$$

where $\hat{x}_{ul}$ is estimated from BPR. We refer to this method as BPR Location Recommendation 1 (BPRLR1).

In the second method we borrow the idea from [30]. Instead of maximizing the difference between visited locations and all unvisited locations, we focus on maximizing the difference between visited locations and unvisited locations that are near users' centers. This idea is very intuitive, since users tend to check in locations near their activity centers, we do not consider the far away locations, which may introduce noise otherwise.

We denote N_u as the set of locations in the nearby activity area for user u. We define $N_u = \{l|P(l|C_u) > 0\}$, which requires that location l has a chance to be checked in by the MGM model. Then we define the trained pairwise location set $S' = \{(u,i,j)|u \in \mathcal{U}, i \in \mathcal{L}_u^+ \wedge j \in N_u \setminus \mathcal{L}_u^+\}$. Now the objective function is:

$$\arg\max_{\Theta} \sum_{(u,i,j)\in S'} \ln(\sigma(\hat{x}_{ui} - \hat{x}_{uj})) - \lambda_\Theta \|\Theta\|^2. \tag{3.33}$$

After we get the learned parameters, we employ the estimator $\hat{x}_{ui}$ to obtain the ranking list. We refer to this method as BPR Location Recommendation 2 (BPRLR2). The learning algorithm is shown in Algorithm 2.

3.4.5 Complexity Analysis

The computation cost consists of the calculation of matrix factorization models and calculating the probability of a user visiting a POI. The training time for the matrix factorization models scales linearly with respect to the number of observations [112, 89]. For the probability computation, the cost is to calculate the centers. This also scales linearly with respect to the number of observations. Hence, the proposed fused framework in Section 3.4.3 is linear with respect to the number of observations. We use SGD to learn parameters in BPRLR1 and BPRLR2. In each iteration, we update the parameters U_u, L_i and L_j. The cost of the iteration is $O(K)$, where K is the latent dimension and is usually very small. In practice, the convergence iteration number is a few times of the observations. So both BPRLR1 and BPRLR2 are efficient and can scale up to very large-scale datasets.

3.5 Experiments

The experiments address the following three questions:

1. How do our approaches compare with the baseline and the state-of-the-art algorithms?
2. How do the geographical influence and ranking loss affect the performance?
3. What is the performance on users with different check-in frequency? This is a scenario for cold-start users whose check-ins are few.

3.5.1 Setup and Metrics

The experimental data include user-location check-in records, users' friendship list and geographical information (longitude and latitude of check-in locations). We split the crawled Gowalla dataset and Foursquare dataset into two non-overlapping sets: a training set and a test set, where the proportion of training data and test data is 70% and 80%, respectively. Here, 70%, for example, means we randomly select 70% of the observed data for each user as the training data to predict the remaining 30% data. The random selection was carried out 5 times independently and we report the average result. The hyper-parameters are tuned by cross validation. For all experiments, we set the regularization term λ to 0.1 and the step size α to 0.2.

POI recommendation is to recommend the top-N highest ranked positions to a targeted user based on a ranking score from a recommendation algorithm. To evaluate the model performance, we are interested in finding out how many locations in the test set are recovered in the returned POI recommendation. Hence, we use the Precision@N and Recall@N as the metrics to evaluate the returned ranking list against the check-in locations where users actually visit. These two metrics are standard metrics to measure the performance of POI recommendation [143]. Precision@N defines the ratio of recovered POIs to the N recommended POIs, while Recall@N defines the ratio of recovered POIs to the size of the test set. In the experiments, N is set to 5 and 10, respectively.

Table 3.3: Performance comparisons on the Gowalla dataset with $K = 20$

Ratio	Metrics	Dimension = 20							
		MGM	PMF	PMFSR	PFM	FMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0317	0.0140	0.0153	0.0173	0.0643	0.0645	**0.0791**	0.0500
	Improve	149.53%	465.00%	416.99%	357.23%	23.02%	22.64%		58.20%
	R@5	0.0113	0.0032	0.0035	0.0040	0.0202	0.0187	**0.0264**	0.0168
	Improve	133.63%	725.00%	654.29%	560.00%	30.69%	41.18%		57.14%
	P@10	0.0273	0.0166	0.0166	0.0172	0.0635	0.0615	**0.0682**	0.0615
	Improve	149.82%	310.84%	310.84%	296.51%	7.40%	10.89%		10.89%
	R@10	0.0194	0.0079	0.0078	0.0084	0.0395	0.0355	**0.0445**	0.0396
	Improve	129.38%	463.29%	470.51%	429.76%	12.66%	25.35%		12.37%
80%	P@5	0.0263	0.0106	0.0107	0.0114	0.0464	0.0462	**0.0544**	0.0334
	Improve	106.84%	413.21%	408.41%	377.19%	17.24%	17.75%		62.87%
	R@5	0.0141	0.0034	0.0034	0.0039	0.0207	0.0194	**0.0258**	0.0160
	Improve	82.98%	658.82%	658.82%	561.54%	24.64%	32.99%		61.25%
	P@10	0.0226	0.0120	0.0121	0.0117	0.0452	0.0427	**0.0468**	0.0412
	Improve	107.08%	290.00%	286.78%	300.00%	3.54%	9.60%		13.59%
	R@10	0.0244	0.0082	0.0084	0.0083	0.0404	0.0358	**0.0442**	0.0382
	Improve	81.15%	439.02%	426.19%	432.53%	9.41%	23.46%		15.71%

Table 3.4: Performance comparisons on the Gowalla dataset with $K = 30$

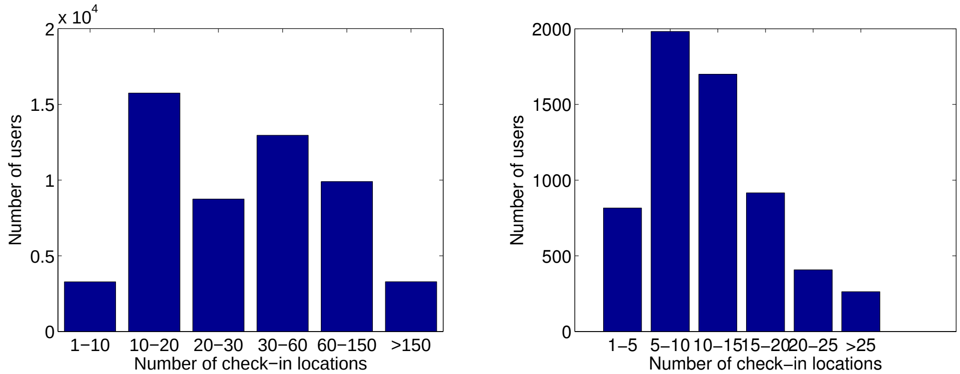
Ratio	Metrics	Dimension = 30							
		MGM	PMF	PMFSR	PFM	FMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0317	0.0148	0.0158	0.0173	0.0672	0.0674	**0.0802**	0.0517
	Improve	153.00%	441.89%	407.59%	363.58%	19.35%	18.99%		55.13%
	R@5	0.0113	0.0033	0.0035	0.0040	0.0212	0.0199	**0.0270**	0.0175
	Improve	138.94%	718.18%	671.43%	575.00%	27.36%	35.68%		54.29%
	P@10	0.0273	0.0162	0.0174	0.0173	0.0656	0.0643	**0.0700**	0.0628
	Improve	156.41%	332.10%	302.30%	304.62%	6.71%	8.86%		11.46%
	R@10	0.0194	0.0075	0.0080	0.0084	0.0408	0.0382	**0.0465**	0.0408
	Improve	260.82%	833.33%	775.00%	733.33%	71.57%	83.25%		71.57%
80%	P@5	0.0263	0.0106	0.011	0.0114	0.0486	0.0488	**0.0551**	0.0348
	Improve	109.51%	419.81%	400.91%	383.33%	13.37%	12.91%		58.33%
	R@5	0.0141	0.0035	0.0037	0.0039	0.0218	0.0210	**0.0263**	0.0172
	Improve	86.52%	651.43%	610.81%	574.36%	20.64%	25.24%		52.91%
	P@10	0.0226	0.0115	0.0117	0.0117	0.0472	0.0450	**0.0479**	0.0432
	Improve	111.95%	316.52%	309.40%	309.40%	1.48%	6.44%		10.88%
	R@10	0.0244	0.0079	0.0081	0.0085	0.0424	0.0386	**0.0456**	0.0407
	Improve	86.89%	477.22%	462.96%	436.47%	7.55%	18.13%		12.04%

Table 3.5: Performance comparisons on the Foursquare dataset with $K = 20$

Ratio	Metrics	Dimension = 20						
		MGM	PMF	PFM	FMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0409	0.0591	0.0706	0.1190	0.1074	0.1447	**0.1734**
	Improve	323.96%	193.40%	145.61%	45.71%	61.45%	19.83%	
	R@5	0.0306	0.0258	0.0308	0.0588	0.0513	0.0749	**0.0878**
	Improve	186.93%	240.31%	185.06%	49.32%	71.15%	17.22%	
	P@10	0.0373	0.0610	0.0652	0.1157	0.1078	0.1501	**0.1671**
	Improve	347.99%	173.93%	156.29%	44.43%	55.01%	11.33%	
	R@10	0.0531	0.0550	0.0608	0.1152	0.1032	0.1545	**0.1699**
	Improve	219.96%	208.91%	179.44%	47.48%	64.63%	9.97%	
80%	P@5	0.0288	0.0448	0.0486	0.0830	0.0771	0.1031	**0.1273**
	Improve	342.01%	184.15%	161.93%	53.37%	65.11%	23.47%	
	R@5	0.0332	0.0311	0.0362	0.0645	0.0572	0.0826	**0.0969**
	Improve	191.87%	211.58%	167.68%	50.23%	69.41%	17.31%	
	P@10	0.0265	0.0466	0.0504	0.0812	0.0766	0.1042	**0.1207**
	Improve	355.47%	159.01%	139.48%	48.65%	57.57%	15.83%	
	R@10	0.0586	0.0647	0.0671	0.1245	0.1138	0.1648	**0.1859**
	Improve	217.24%	187.33%	177.05%	49.32%	63.36%	12.80%	

Table 3.6: Performance comparisons on the Foursquare dataset with $K = 30$

Ratio	Metrics	Dimension = 30						
		MGM	PMF	PFM	FMFMGM	BPR	BPRLR1	BPRLR2
70%	P@5	0.0409	0.0621	0.0718	0.1201	0.1086	0.1484	**0.1783**
	Improve	335.94%	187.12%	148.33%	48.46%	64.18%	20.15%	
	R@5	0.0306	0.0277	0.0312	0.0594	0.0528	0.0763	**0.0901**
	Improve	194.44%	225.27%	188.78%	51.68%	70.64%	18.09%	
	P@10	0.0373	0.0638	0.0663	0.1166	0.1107	0.1522	**0.1698**
	Improve	355.23%	166.14%	156.11%	45.63%	53.39%	11.56%	
	R@10	0.0531	0.0574	0.0622	0.1166	0.1070	0.1568	**0.1728**
	Improve	225.42%	201.05%	177.81%	48.20%	61.50%	10.20%	
80%	P@5	0.0288	0.0450	0.0482	0.0833	0.0820	0.1050	**0.1287**
	Improve	346.88%	186.00%	167.01%	54.50%	56.95%	22.57%	
	R@5	0.0332	0.0306	0.0364	0.0640	0.0606	0.0834	**0.0998**
	Improve	200.60%	226.14%	174.18%	55.94%	64.69%	19.66%	
	P@10	0.0265	0.0478	0.0512	0.0811	0.0796	0.1053	**0.1227**
	Improve	363.02%	156.69%	139.65%	51.29%	54.15%	16.52%	
	R@10	0.0586	0.0657	0.0677	0.1242	0.1176	0.1658	**0.1898**
	Improve	223.89%	188.89%	180.35%	52.82%	61.39%	14.48%	



(a) Gowalla (b) Foursquare

Figure 3.3: Distribution of user groups

3.5.2 Comparison

In the experiments, the compared approaches include:

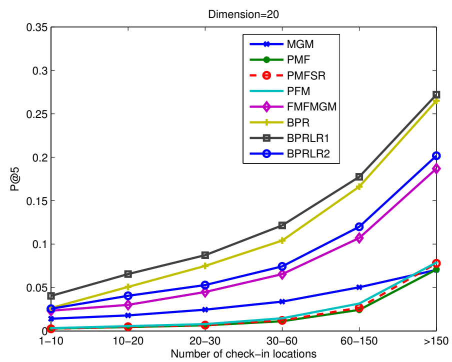
1. **Multi-center Gaussian Model (MGM)**: this method recommends a position based on the probability calculated by Eq. (3.1).

2. **PMF**: this is a well-known method in matrix factorization [112]. We describe the details in Section 3.4.2. Its objective function is shown in Eq. (3.6).

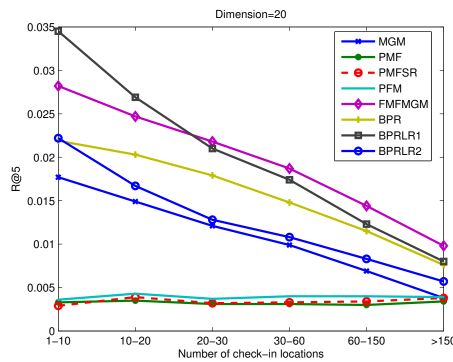
3. **PMF with Social Regularization (PMFSR)**: this method is proposed to include the social friendship under the PMF framework [89], which is introduced in Section 3.4.2. Its objective function is shown in Eq. (3.7).

4. **Probabilistic Factor Models (PFM)**: this method is a promising method to model frequency data [86]. Its objective function is shown in Eq. (3.14) and the details are in Section 3.4.2.

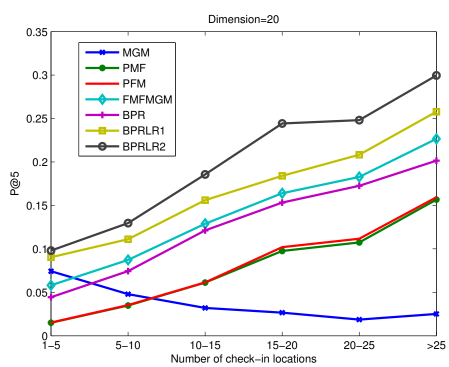
5. **FMF with MGM (FMFMGM)**: this is the Fused Matrix Factorization framework with the Multi-center Gaussian Model (FMFMGM). The user's preference on locations is calculated



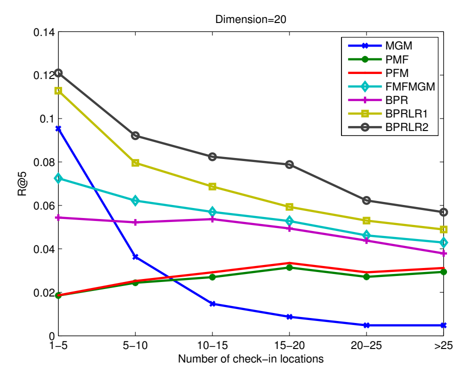
(a) Precision@5 on different user groups in Gowalla



(b) Recall@5 on different user groups in Gowalla



(c) Precision@5 on different user groups in Foursquare



(d) Recall@5 on different user groups in Foursquare

Figure 3.4: Performance comparison on different user groups

by the PFM model. Here, we select PFM because PFM can model the frequency data better than PMF.

6. **BPR**: this method is a ranking-oriented method for implicit data [104]. We introduced the details in Section 3.4.4.

7. **BRPLR1**: this method is the first scheme we proposed to incorporate BPR and geographical influence.

8. **BPRLR2**: this method is the second scheme we proposed to incorporate BPR and geographical influence.

Tables 3.3-3.6 report the average of five-run results on the top 5 and top 10 recommendation by the competing models using 20 and 30 as the number of latent feature dimensions, respectively. The results show that:

- FMFMGM outperforms PMF, PMFSR and PFM significantly in all metrics. For example, in Gowalla, FMFMGM attains 0.0643 in terms of P@5 when the latent dimension is 20 and 70% of data are used for training, while PFM, the best current model without considering location information, achieves 0.0173 for the counter part. This implies that geographical influence plays a significant role in POI recommendation. By utilizing the geographical influence, we can provide much more accurate POI recommendations to targeted users.

- FMFMGM achieves significantly better performance than MGM in both Gowalla and Foursquare datasets. That is, for the case of the latent dimension being 30 and 80% of data for training, the performance increases from 0.0141 for MGM to 0.0218 for FMFMGM. This verifies that the probability of a user visiting a POI is controlled by both the user's personal preference and the personal check-in location constraints. By utilizing users' personalized tastes captured by MF models, we can attain more accurate predictions.

- PMFSR attains a little better results than those of PMF. This shows that social influence is not so important in POI recommendation and it also coincides the fact that friends share very low, only 9.6%, common POIs.

- BPR almost achieves comparable performance with FMFMGM, which verifies our assumption that ranking loss affects the final recommendation. An interesting result is that in Gowalla, BPRLR1 performs the best, while in Foursquare, BPRLR2 performs the best. The reason might be that the data in our Gowalla dataset are sparser than the Foursquare dataset. Note when we use the second scheme, i.e., focusing on nearby POIs, it may not work well on Gowalla. Since the Gowalla data are sparser we may not have enough samples to learn the parameters properly.

3.5.3 Performance on Different Users

One challenge of the POI recommendation is that it is difficult to recommend POIs to those users who have very few check-ins. In order to compare our methods with the other methods thoroughly, we first group all the users based on the frequency of observed check-ins in the training set. Then we evaluate the model performances within different user groups. Here, users are grouped into 6 types: “1-10”, “10-20”, “20-30”, “30-60”,“60-150” and “>150” for Gowalla; “1-5”, “5-10”, “10-15”, “15-20”,“20-25” and “>25” for Foursquare. The number denotes the frequency range of users’ check-ins in the training data.

Figure 3.3 summarizes the distribution on different ranges of users’ check-in frequency in 70% of the training data. From Figure 3.4, we observe that when users’ check-in frequency is small, MGM outperforms PMF, PMFSR and PFM. But when users’ check-in frequency becomes larger, PMF, PMFSR and PFM performs better than MGM. It is reasonable since when users’ check-in frequency is small, especially for cold-start users, it is difficult to learn users' preferences. Thus, geographical information plays more influence on the prediction. When more check-in information is available, both users' preferences and geographical influence can be learned more accurately, but users' preferences dominate the geographical influence. When taking the ranking loss into account, we achieve the best performance, especially when the dataset is denser, both BPRLR1 and BPRLR2 consistently outperform other competing methods.

3.6 Conclusion

In this chapter, we have investigated the characteristics of the large-scale check-in data from two popular LBSNs websites, Gowalla and Foursquare. Based on the extracted properties of the data, we proposed a novel Multi-center Gaussian Model to model the geographical influence of users' check-in behavior. We then considered users' social information and propose a fused matrix factorization method to include the geographical influence of users' check-in locations. Furthermore, we proposed to incorporate ranking-oriented CF with all the information together into a unified framework. Results from extensive experiments showed that our proposed methods outperformed other state-of-the-art approaches significantly.

There are several directions worthy of consideration for future study: 1) how to model extremely sparse frequency data, e.g., by designing more subtle sampling techniques, to improve MF methods; 2) how to include other information, e.g., location category and activity, into our fused framework; 3) how to incorporate temporal effect on POI recommendation to capture the change of users' preferences. We will continue to explore these future directions.

□ **End of chapter.**

Chapter 4

Successive Point-of-interest Recommendation in Location-based Social Networks

In the past few years, millions of users are getting used to check in point-of-interests (POIs) on Location-based social networks (LBSNs). POI recommendation is one of the most important services in LBSNs, as it can help provide better user experience as well as enable third-party services, e.g., launching advertisements. Several methods have been proposed in the research community for the POI recommendation service. However, most of the previous efforts mainly consider the "check-ins" as a whole, ignoring their temporal relation or sequential effect. They can only recommend POIs globally but cannot know where a user would like to go in the near future. In this chapter, we consider the task of successive POI recommendation in LBSNs, which is a much harder task than the standard POI recommendation . To solve this task, we develop two matrix factorization models called Factorized Personalized Markov Chain with Localized Region (FPMC-LR) and Factorized Personalized Markov Chain with Latent Topic Transition (FPMC-LLT) based on two prominent properties observed in the check-in sequence: personalized Markov chain and region localization. Both FPMC-LR and FPMC-LTT embed the personalized Markov chain and the region localization. They not only exploit the personalized Markov chain in the check-in sequence, but also take into account the user's movement constraint, i.e., moving around a localized region. More importantly, by utilizing the information of localized regions, we not only reduce the computation cost, but also discard the noisy information to boost recommendations. However, the personalized Markov chain in FPMC-LR is built on the location-wise level. The number of observations on location-wise transitions is very small in LBSNs, which makes it difficult to learn the latent location transition vector well. We observe that there are high transition probabilities between topics such as the transition from "Shopping" to "Food". FPMC-LTT models the latent topic transition probability, which can avoid the sparsity problem in FPMC-LR. Results on two real-world LBSNs datasets demonstrate the merits of our proposed FPMC-LR and FPMC-LTT model.

4.1 Introduction

The check-in behavior becomes a new life-style for millions of users who share their locations, tips and experiences about POIs with their friends in location-based social networks (LBSNs). The online check-ins embed abundant information of users' physical movements in daily life, users' connections to others as well as their preferences on the POIs. Among various services in LBSNs, POI recommendation is especially important as it allows users to know new POIs and explore their locations while facilitating advertisers to launch advertisements to targeted users.

Recently, POI recommendation in LBSNs has attracted much attention in both the research and industry communities [143, 117]. Collaborative filtering (CF) is a mainstream technique to solve this task. Both memory-based and model-based CF methods have been proposed and investigated to learn users' preferences on the POIs from the user-location check-in data [29, 143, 72]. However, pre-

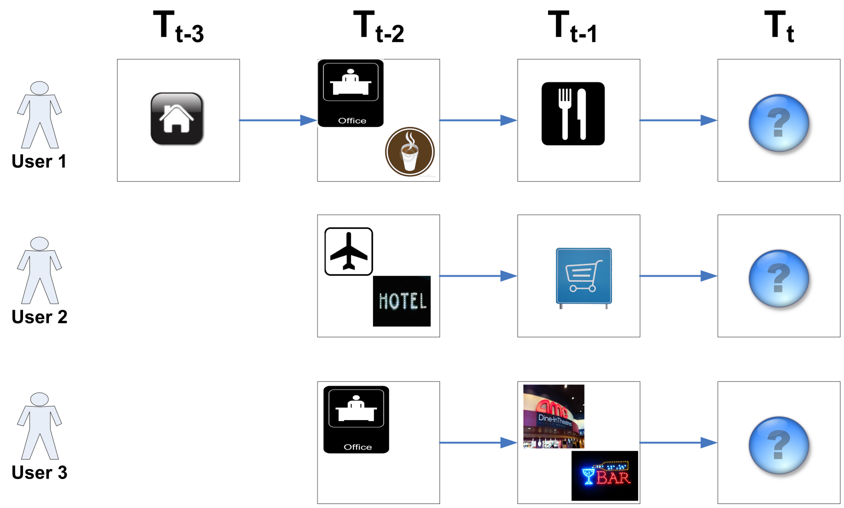
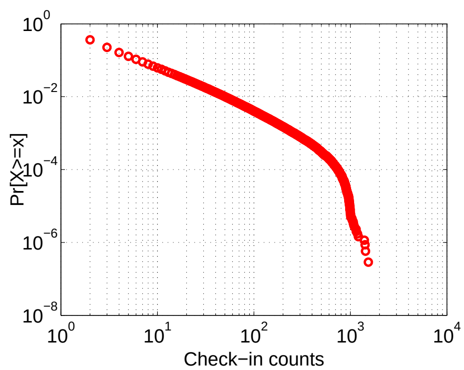


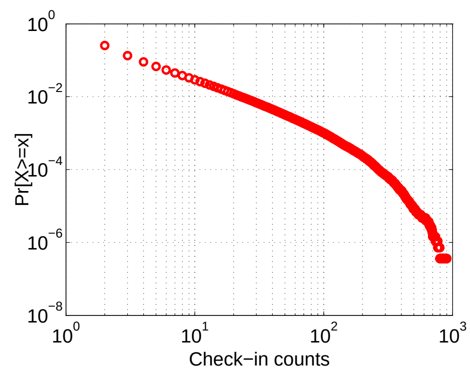
Figure 4.1: An example of three users' check-in sequences

viously proposed methods consider all check-ins as a whole while overlook the temporal relation. Temporal effects have been explored to improve POI recommendation performance in [39, 145], but the task is still to provide POI recommendation based on the overall history, not providing successive POI recommendation based on the previous state. The statistics in Figure 4.2 show that apart from a few routinely visiting POIs such as office and home, most POIs are visited less than 10 times, which accounts for 90% of total visited POIs. It indicates that most POIs are visited occasionally and they are related to the user's current location. Hence, POI recommendation is very time-critical. A good POI recommendation service should be able to provide good recommendations promptly based on the user's current status.

Hence, in this chapter, different from previous work, we consider the task of successive POI recommendation in LBSNs. This task is much harder than the standard POI recommendation because it recommends locations that a user does not visit frequently or has not visited before, but may like to visit at the successive time stamp based on the current status. This task is more significant since it can provide various personalized favorite services in LBSNs. For



(a) Foursquare



(b) Gowalla

Figure 4.2: Check-ins probability vs. counts

example, it may tell a user where to have fun after dinner or suggest the discount information of some products in nearby shops when the user is shopping. Although this task is very difficult, we believe that the collaborative information shared in users' check-in histories can be further utilized to boost the recommendation. Figure 4.1 gives an intuitive example of this. User 3 visited a cinema and then a bar after work. It may also be good to suggest user 1 to go there after the dinner. The significance of successive POI recommendation in LBSNs and the promising benefit of utilizing the collaborative information trigger our in-depth study in the check-in data.

There are two main properties, i.e., personalized Markov chain and localized region constraint, in the LBSNs datasets, see Section 4.3 for more details. Based on these two observations, we propose two matrix factorization models called FPMC-LR and FPMC-LLT. FPMC-LR and FPMC-LTT include the information of the personalized Markov chain and the localized region constraint. Although our models borrow the idea of Factoring Personalized Markov Chain (FPMC) for solving the task of next-basket recommendation [105], we emphasize the user's movement constraint, i.e., moving around a local region, and focus on a different problem. More specifically, we only consider the locations around the user's previous check-in history, which yields a much smaller set, accounting for about 0.7% and 0.3% of the set on all locations for Foursquare and Gowalla, respectively. More importantly, we not only reduce the computation cost, but also discard possible noisy information.

One disadvantage of the FPMC-LR model may be that the sparsity of location transition data in LBSNs makes it difficult to learn the latent location transition vector well. The average number of different check-in locations of each user is less than 100, while there are around $30{,}000$ and $60{,}000$ locations in Foursquare and Gowalla, respectively. We further develop the FPMC-LTT model, which models the transition probability on the latent topic level together with the two properties to overcome the data sparsity problem. The motivation is straightforward since there are high correlations between different location topics. As illustrated in the previous example in Figure 4.1, there is a high probability to check in locations of entertainment after the locations whose topics are work related. More details are discussed in Section 4.3.5. In terms of the latent topic level, we have enough data to train the model well.

We summarize our contributions in the following:

- We formally define the problem of successive POI recommendation in LBSNs and analyze the spatial-temporal properties in two large-scale real-world LBSN datasets: Foursquare and Gowalla. After analyzing the dynamics of new POIs and inter check-ins, we observe two important properties: personalized Markov chain and localized region constraint.

- We propose two novel matrix factorization methods, namely FPMC-LR and FPMC-LTT, to incorporate these two properties. More importantly, we not only reduce the computation cost, but also discard noisy information.

- We conduct detailed experimental evaluation on the analyzed large-scale LBSN datasets and show that our models consistently outperform other state-of-the-art methods.

4.2 Related Work

The work presented in this chapter is closely related to four different categories: matrix factorization, POI recommendation, POI prediction and successive POI recommendation. In the following, we briefly review the related work.

4.2.1 Matrix Factorization

Matrix factorization techniques have been widely adopted in recommender systems [61, 112]. The basic idea behind these models was using two low rank latent vectors to approximate the user-item rating matrix and then employing the matrix to make further predictions. These methods were very efficient since only a small number of latent factors influenced preferences and the time complexity was linear to the number of observations. Several methods were explored to incorporate the temporal effect into the matrix factorization. Koren et al. [63] proposed the timeSVD++ model to incorporate time factors into the matrix factorization models. Xiong et al. [137] further split the user-item rating matrix into pieces according to the time slot, which turned the rating matrix into rating tensor. The Factorized Personalized Markov Chain (FPMC) was developed in [105] to model the item transitions.

4.2.2 Point-of-interest Recommendation

Location-based social networks have received much attention in recent years due to the new characteristics of spatial-temporal-social information embedded in the check-in data and the prevalence of various interesting real-world applications [144, 151, 31]. Research topics covered in this area include user behavior study [121], movement pattern analysis [119], community detection [136] and POI recommendation [152]. Among all of these topics, POI recommendation is one of the most important topics due to its high value in both the research and industry communities.

Currently, there are two lines of work to solve the task of POI recommendation. One line of research is conducted based on the GPS trajectory logs [152, 149, 148, 69]. The GPS trajectory data usually consist of a small number of users, but with dense records [150, 24]. Many collaborative filtering algorithms, e.g., collective matrix factorization [149], tensor factorization [148], memory-based collaborative location model (CLM) [69], etc., have been proposed and deemed the locations as items in traditional recommender systems.

The other line of work focuses on LBSNs data, which are very sparse and on a large-scale [142, 143, 29]. The related work consists of three sub-categories. The first sub-category explores the geographical influence. Ye et al. [143] modeled the check-in probability with the distance of the whole check-in history by the power-law distribution and incorporated it with memory-based collaborative filtering methods. Multi-center Gaussian Model (MGM) was proposed in [29] to model users' multi-center check-in behaviors via a multi-center Gaussian model. The authors then fused the MGM model with model-based collaborative filtering methods. Liu et al. [75] investigated a novel geographical probabilistic factor analysis framework, which can take various factors, such as geographical influences, POI popularity, etc., into consideration. GeoMF [72] explored to augment POIs' latent factors by the influence area of POIs.

The second sub-category related work attempts to make use of content information to boost POI recommendation. Liu et al. [76] developed a Topic and Location-aware Probabilistic Matrix Factorization (TL-PMF) method to combine the LDA model and the matrix factorization model, in which content information was embedded through the LDA model part. Most recently, Gao et al. [40] integrated three types of content information into a unified framework.

The third sub-category is to study the effect of temporal information. Yuan et al. [145] explored a user-based collaborative filtering method to incorporate the temporal cyclic information and geographical information. Gao et al. [39] suggested a model-based method to leverage the non-uniformness and consecutiveness of users' check-in behavior.

4.2.3 Point-of-interest Prediction

The POI prediction aims to model users' movement patterns and predict the location the user might visit at a certain time [110, 111, 119]. Usually, the predicted location is visited by the user before, not the new location like in POI recommendation. Scellato et al. [119] described a novel approach to location prediction based on nonlinear time series analysis of the arrival and residence time of users in relevant places. Sadilek et al. [110] employed a dynamic Bayesian network to predict the location a user would visit in the next time slot. Long-term location prediction was explored in [111].

4.2.4 Successive Point-of-interest Recommendation

There are a few existing work focusing on successive POI recommendation, which is addressed in this chapter. Sang et al. [117] proposed a probabilistic approach that estimated the transition probability from one POI to another, conditioned on the current context and check-in history in a Markov chain. Zhang et al. [146] explored a similar method while considering the additive Markov chain to estimate the transition probability. The transition probabilities of their models were all estimated with the check-in counts from observed data, which were very sparse.

Our work is different from the existing work. We focus on the successive POI recommendation task and propose two matrix factorization methods FPMC-LR and FPMC-LTT to embed the two aforementioned properties, i.e., personalized Markov chain and region localization, into them.

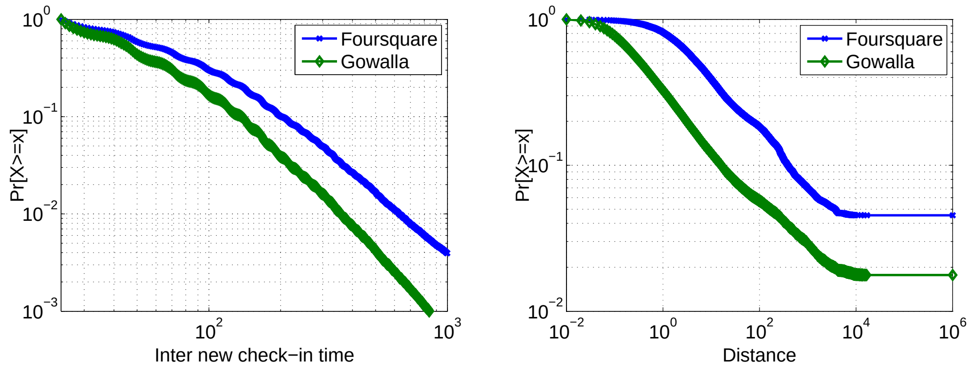
4.3 Successive Point-of-interest Recommendation in LBSNs

4.3.1 Problem of Successive Point-of-interest Recommendation

Let $\mathcal{U}$ be a set of users and $\mathcal{L}$ be a set of locations. $\mathcal{L}_u$ denotes the check-in history of user u. Due to the low density of the LBSNs data, we merge consecutive check-ins in T hours as a slide window to construct a set of check-ins. As a result, we construct a slide window set $\mathcal{T}$ to denote the user's visiting time stamp. The check-in set of user u at time t is denoted by $\mathcal{L}_u^t$, where $t \in \mathcal{T}$. Given a sequence of check-ins (i.e., $\mathcal{L}_u^1, \ldots, \mathcal{L}_u^t$) and the position of each location (i.e., the latitude and longitude), the problem of successive POI recommendation is to provide the most suitable recommendation for user u at time $t+1$.

4.3.2 New Point-of-interests Dynamics

New POIs are locations that a user has not visited before and will be recommended in the next time stamp. The inter check-in time and location distance on new POIs are defined as the temporal interval and distance between a new POI and the previous POI, respectively. Figure 4.3 shows the properties of new POIs dynamics on the time and location distance. Figure 4.3(a) reports how often a user would like to explore new POIs by calculating the Complementary Cumulative Distribution Function (CCDF) on the inter check-in time of new POIs. It shows that almost 70% of Foursquare users and 80% of Gowalla users would like to check-in a new POI after about 100 hours. The ratio increases to 90% for Foursquare and 95% for Gowalla after 200 hours. It is noted that although users would like



(a) The inter new check-in time in hours

(b) The minimum distance of new POI to check-in history

Figure 4.3: The new POI dynamics

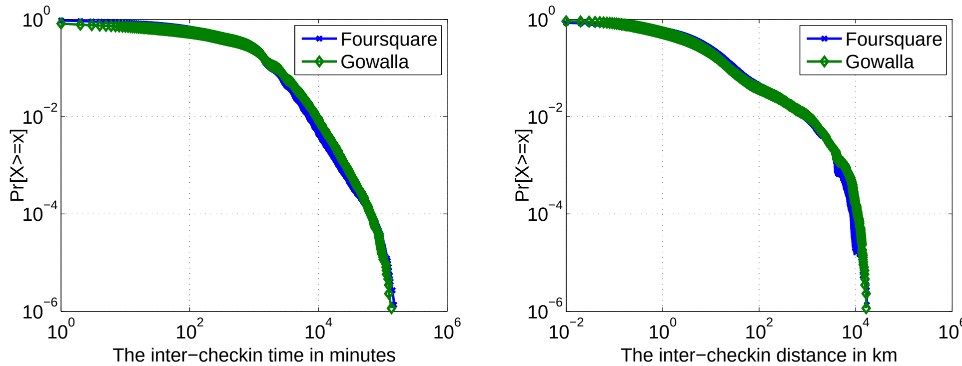
to explore new POIs, as shown in Figure 4.2, most of their check-ins are distributed among a few frequently visited places, e.g., home and office.

Figure 4.3(b) shows the spatial property of a new POI versus previously successively visited POIs. Obviously, users' exploration on new POIs is restricted by the geographical influence. More specifically, about 60% of Foursquare new POIs and about 88% of Gowalla new POIs are within 10 km of users' previous check-in locations. When the distance increases to 100 km, the number of new POIs account to about 80% for Foursquare and about 95% for Gowalla, respectively. This observation implies that users in Foursquare prefer to explore new farther POIs than Gowalla users.

4.3.3 Inter Check-in Dynamics

The property of inter check-in dynamics is another key factor in revealing the temporal relation of the LBSNs data. We obtain similar results in [94] and observe two significant properties on the LBSNs data: personalized Markov chain and localized region constraint.

Figure 4.4(a) shows that almost 40% and 48% successive check-



(a) The inter check-in time in minutes

(b) The inter check-in distance in km

Figure 4.4: The inter check-in time in minutes.

ins occur in Foursquare and Gowalla, respectively, within two hours. The ratio is raised to about 70% for both Foursquare and Gowalla when the inter check-in time is larger than 12 hours. After further studying the categories of two successive check-ins for a user in a short period, we find that there is a strong connection between the two check-ins. For example, cinemas or bars may always be visited after restaurants, as users would like to relax after dinner. This is exactly a personalized Markov chain property, which motivates us to utilize the transition probability for solving the task of successive POI recommendation.

Figure 4.4(b) shows the CCDF of inter check-in distance. It is observed that more than 75% of inter check-ins in Foursquare and more than 80% of inter check-ins in Gowalla occur within 10 km. Only less than 5% of inter check-in distance is more than 100 km in both datasets. This observation is reasonable since most users' inter check-ins occur within a specific area they live or the long distance inter check-ins imply an occasional journey. Overall, users' movements are constrained by their geographical influence within a short time. Hence, when we provide successive POI recommendation, we mainly consider the new POIs near a user's previous check-ins.

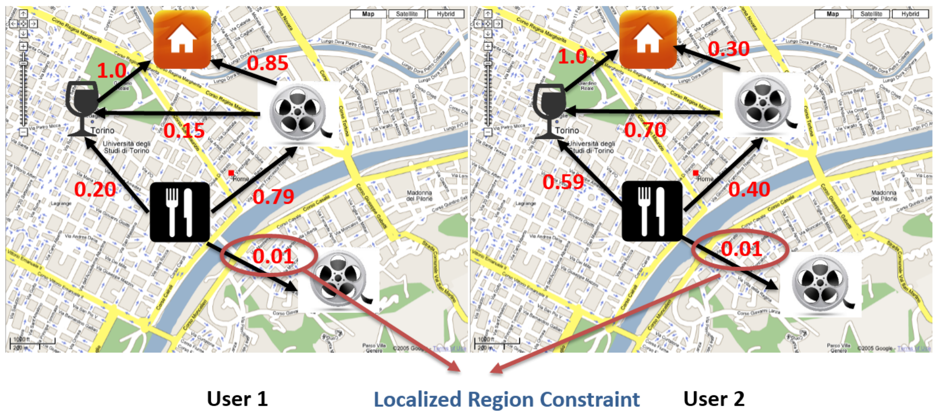


Figure 4.5: A toy example illustrating the two main property in LBSNs

4.3.4 A Toy Example

Figure 4.5 shows a toy example of two users' check-ins to illustrate the two main properties: personalized Markov chain and localized region constraint. The numbers on the line denote the users' transition probabilities. For example, for user 1 on the left side, after having dinner, the probability that the user will go to the bar is 0.20. User 2 has a different transition probability because the Markov chain is personalized. The big difference in probability from visiting the two movie theaters reflects the localized region constraint. Since the movie theater is near both users' homes, the probability of visiting the upper one is much larger than that of visiting the one on the bottom right corner.

4.3.5 Topic Transition

The check-in data in LBSNs are very sparse for each user. As a result, it might not be easy to learn the location transition latent vector well. Table 4.1 gives the top 20 topic transitions in the Gowalla dataset. In Gowalla, each POI is attached with several category tags by the system. We use the category as the topic and calculate the overall transition probability between each topic. Since we do not

Table 4.1: Top 20 topic transitions in Gowalla

Topic(from)	**Topic**(to)
Conference	Home
Tram	Library
Sports	Coffee Shop
Hotel	Mall
Outdoors	Food
Entertainment	Starbucks
Pub	Subway
Golf Shop	Coffee Shop
Hotel	Food
School	Apartment
Movie	Art & Culture
Apparel	Food
Four Seasons	Train Station
Museum	Food
Bears Sports	Mall
Aquatics	Bakery
Rental Car	Coffee Shop
Apparel	Gas & Automotive
Lab	Burgers
Cave	Breakfast

have the category information of the Foursquare dataset, we only report the results for the Gowalla dataset. A similar result on the Foursquare dataset is reported in [94]. From the table, we can observe that many of the transitions make sense in real life. If we know that a user is at a location with topic "School", for example, the user may go to a POI with topic "Apartment" next. This motivates us to propose the FPMC-LLT model, which models the latent topic transition probability.

4.4 Our Models

In this section, we first introduce the FPMC-LR model and then discuss the FPMC-LTT model.

4.4.1 Factorized Personalized Markov Chain with Localized Region (FPMC-LR)

Our FPMC-LR is to recommend a successive POI via the probability that user u will visit location l at time t, which is calculated by

$$x_{u,i,l} = p(l \in \mathcal{L}_u^t | i \in \mathcal{L}_u^{t-1}). \tag{4.1}$$

Based on the first-order Markov chain property, the probability can be calculated by

$$p(l \in \mathcal{L}_u^t | \mathcal{L}_u^{t-1}) = \frac{1}{|\mathcal{L}_u^{t-1}|} \sum_{i \in \mathcal{L}_u^{t-1}} p(l \in \mathcal{L}_u^t | i \in \mathcal{L}_u^{t-1}), \tag{4.2}$$

where $p(l \in \mathcal{L}_u^t | i \in \mathcal{L}_u^{t-1})$ is the probability of user u moving from location i to location l.

In FPMC, all locations are considered for each user, which yields a transition tensor $\mathcal{X} \in [0,1]^{|\mathcal{U}| \times |\mathcal{L}| \times |\mathcal{L}|}$. From a different approach, our FPMC-LR considers only the neighborhood locations. More specifically, we divide the whole earth into different square grids, each with side length of d km. Then, for each location l, its neighbor locations will be those fallen in one of the nine adjacent square grids:

$$N_d(\mathcal{L}_u^t) = \{l \in \mathcal{L} \setminus \mathcal{L}_u^{t-1} : D(l, l_0) \leq d, \forall l_0 \in \mathcal{L}_u^{t-1}\},$$

where $D(l, l_0)$ is the distance between l and l_0 calculated by the Haversine formula.

Let $N(\mathcal{L}_u^t)$ be the neighbor location set of the check-in history of user u at time t. Our FPMC-LR yields a transition tensor $\mathcal{X} \in [0,1]^{|\mathcal{U}|\times|\mathcal{L}|\times|N_d(\mathcal{L})|}$. It is noted that $|N_d(\mathcal{L})|$ is reduced largely, e.g., around one hundred when $d = 40$, which accounts for less than 0.7% and 0.3% of the total locations in Foursquare and Gowalla datasets, respectively. Hence, our FPMC-LR can save the time cost when compared with FPMC. Since FPMC provides a good framework for successive POI recommendation, we adopt it in our method, but focus on the localized region constraint. This motivates the name of our model.

Low-rank approximation is a promising tool to recover the partially observed transition tensor $\mathcal{X}$ when it is sparse. Here, we adopt a special case of Canonical Decomposition, which models the pairwise interaction among the three modes of the tensor (i.e., user $\mathcal{U}$, last location $\mathcal{I}$ and next location $\mathcal{L}$):

$$\hat{x}_{u,i,l} = \boldsymbol{v}_u^{\mathcal{U},\mathcal{L}} \cdot \boldsymbol{v}_l^{\mathcal{L},\mathcal{U}} + \boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}} + \boldsymbol{v}_u^{\mathcal{U},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{U}}, \tag{4.3}$$

where $\boldsymbol{v}_u^{\mathcal{U},\mathcal{L}}$ and $\boldsymbol{v}_l^{\mathcal{L},\mathcal{U}}$ model the latent features for users and the next locations, respectively. Other notations are similarly defined. This gives a set of model parameters:

$$\Theta = \{\boldsymbol{V}^{U,L}, \boldsymbol{V}^{L,U}, \boldsymbol{V}^{U,I}, \boldsymbol{V}^{I,U}, \boldsymbol{V}^{L,I}, \boldsymbol{V}^{I,L}\}. \tag{4.4}$$

Combining Eq. (4.2) and Eq. (4.3), we obtain

$$\begin{aligned}\hat{p}(l \in \mathcal{L}_u^t | \mathcal{L}_u^{t-1}) &= \frac{1}{|\mathcal{L}_u^{t-1}|} \sum_{i \in \mathcal{L}_u^{t-1}} \hat{x}_{u,i,l} \\ &= \frac{1}{|\mathcal{L}_u^{t-1}|} \sum_{i \in \mathcal{L}_u^{t-1}} (\boldsymbol{v}_u^{\mathcal{U},\mathcal{L}} \cdot \boldsymbol{v}_l^{\mathcal{L},\mathcal{U}} + \boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}} + \boldsymbol{v}_u^{\mathcal{U},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{U}}) \\ &= \boldsymbol{v}_u^{\mathcal{U},\mathcal{L}} \cdot \boldsymbol{v}_l^{\mathcal{L},\mathcal{U}} + \frac{1}{|\mathcal{L}_u^{t-1}|} \sum_{i \in \mathcal{L}_u^{t-1}} (\boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}} + \boldsymbol{v}_u^{\mathcal{U},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{U}}). \end{aligned} \tag{4.5}$$

Notice that the last step holds as the interaction between $\mathcal{U}$ and $\mathcal{L}$ are independent of the last location i.

Our goal of successive POI recommendation is to recommend top-k new POIs to users. Thus we can model it as a ranking $>_{u,t}$ over locations:

$$i >_{u,t} j :\Leftrightarrow \hat{x}_{u,t,i} > \hat{x}_{u,t,j}. \tag{4.6}$$

A sequential BPR optimization criterion can be derived similar to the general BPR approach [104]. Then for user u at time t, the best ranking can be modeled as:

$$p(\Theta| >_{u,t}) \propto p(>_{u,t} |\Theta)p(\Theta). \tag{4.7}$$

Following the FPMC model, we assume different users check in locations independently. In practice, most of users are not influenced by others when checking in and social influence has little effect as well [29]. We can estimate the model using maximum a posterior (MAP):

$$\arg\max_{\Theta} \prod_{u \in \mathcal{U}} \prod_{\mathcal{L}_u^t \in \mathcal{L}_u} \prod_{i \in \mathcal{L}_u^t} \prod_{j \in N_d(\mathcal{L}_u^{t-1}) \setminus \mathcal{L}_u^t} p(>_{u,t} |\Theta)p(\Theta). \tag{4.8}$$

The ranking probability can be further expressed by:

$$\begin{aligned} p(>_{u,t}) = p(i >_{u,t} j) &= p(\hat{x}_{u,t,i} > \hat{x}_{u,t,j} \mid \Theta) \\ &= p(\hat{x}_{u,t,i} - \hat{x}_{u,t,j} > 0 \mid \Theta). \end{aligned} \tag{4.9}$$

Using the logistic function σ defined by $p(z > 0) = \sigma(z) = \frac{1}{1+e^{-z}}$, we can reformulate Eq. (4.9) as:

$$p(i >_{u,t} |\Theta) = \sigma(\hat{x}_{u,t,i} - \hat{x}_{u,t,j}). \tag{4.10}$$

Furthermore, by placing Gaussian priors on the model parameters $\Theta \sim \mathcal{N}(0, \frac{1}{\lambda_\Theta})$, we can seek the optimal solution of our FPMC-LR by:

$$\begin{aligned}
&\arg\max_{\Theta} \ln p(>_{u,t} |\Theta) p(\Theta) \\
=&\arg\max_{\Theta} \ln \prod_{u\in\mathcal{U}} \prod_{\mathcal{L}_u^t\in\mathcal{L}_u} \prod_{i\in\mathcal{L}_u^t} \prod_{j\in N(\mathcal{L}_u^{t-1})\setminus\mathcal{L}_u^t} \sigma(\hat{x}_{u,t,i} - \hat{x}_{u,t,j}) p(\Theta) \\
=&\arg\max_{\Theta} \sum_{u\in\mathcal{U}} \sum_{\mathcal{L}_u^t\in\mathcal{L}_u} \sum_{i\in\mathcal{L}_u^t} \sum_{j\in N(\mathcal{L}_u^{t-1})\setminus\mathcal{L}_u^t} \ln \sigma(\hat{x}_{u,t,i} - \hat{x}_{u,t,j}) \\
&- \lambda_\Theta \|\Theta\|_F^2.
\end{aligned} \tag{4.11}$$

To recommend a new location, we rank candidate locations based on the probability of $\hat{x}_{u,t,l}$:

$$\hat{x}_{u,t,l} = \boldsymbol{v}_u^{\mathcal{U},\mathcal{L}} \cdot \boldsymbol{v}_l^{\mathcal{L},\mathcal{U}} + \frac{1}{|\mathcal{L}_u^{t-1}|} \sum_{i\in\mathcal{L}_u^{t-1}} (\boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}} + \boldsymbol{v}_u^{\mathcal{U},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{U}}). \tag{4.12}$$

As shown in [105], the term $\boldsymbol{V}^{U,I} \cdot \boldsymbol{V}^{I,U}$ vanishes since it does not affect the final ranking. This yields a more compact expression for $\hat{x}_{u,t,l}$:

$$\hat{x}_{u,t,l} = \boldsymbol{v}_u^{\mathcal{U},\mathcal{L}} \cdot \boldsymbol{v}_l^{\mathcal{L},\mathcal{U}} + \frac{1}{|\mathcal{L}_u^{t-1}|} \sum_{i\in\mathcal{L}_u^{t-1}} \boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}}. \tag{4.13}$$

Learning Algorithm

Directly optimizing the objective function in Eq. (4.11) is very time consuming. Even though we only consider neighbor location pairs, the number of quadruples is still huge, i.e., $O(|S||\bar{N}|)$, where $S = \{(u,t,i)|u \in \mathcal{N}, t \in \mathcal{T}, i \in \mathcal{L}_u^t, \mathcal{L}_u^t \in \mathcal{L}_u\}$ and $|\bar{N}|$ is the average number of neighbor locations. We follow the strategy used in [104] to draw the quadruples independently and apply the stochastic gradient descent on the bootstrap samples. The detailed algorithm is shown in Algorithm 3.

Algorithm 3 Learning Algorithm for FPMC-LR

draw $\boldsymbol{V}^{U,I}, \boldsymbol{V}^{I,U}, \boldsymbol{V}^{I,L}, \boldsymbol{V}_{L,I}$ from $\mathcal{N}(0, \sigma^2)$
repeat
    draw (u,t,i) uniformly from S
    draw location j uniformly from $N(\mathcal{L}_u^{t-1})\backslash\mathcal{L}_u^t$
    for $f = 1 \to k_{U,I}$ **do**
        update $v_{u,f}^{U,I}, v_{i,f}^{I,U}, v_{j,f}^{I,U}$
    end for
    for $f = 1 \to k_{I,L}$ **do**
        update $v_{i,f}^{I,L}, v_{j,f}^{I,L}$
        for $l \in \mathcal{L}_u^{t-1}$ **do**
            update $v_{l,f}^{L,I}$
        end for
    end for
until convergence
return $\boldsymbol{V}^{U,I}, \boldsymbol{V}^{I,U}, \boldsymbol{V}^{I,L}, \boldsymbol{V}_{L,I}$

For each parameter θ, the update procedure is performed as:

$$\begin{aligned}\theta &= \theta + \alpha(\frac{\partial}{\partial\theta}(\ln\sigma(\hat{x}_{u,t,i} - \hat{x}_{u,t,j}) - \lambda_\theta\theta^2)) \\ &= \theta + \alpha((1 - \sigma(\hat{x}_{u,t,i} - \hat{x}_{u,t,j}))\frac{\partial}{\partial\theta}(\hat{x}_{u,t,i} - \hat{x}_{u,t,j}) - 2\lambda_\theta\theta)),\end{aligned} \tag{4.14}$$

where α is the step size.

4.4.2 Factorized Personalized Markov Chain with Latent Topic Transition (FPMC-LLT)

We define a global latent topic transition matrix $\boldsymbol{A} \in \mathbb{R}^{k \times k}$ for all users. The user latent vector and location latent vector are denoted as $\boldsymbol{U} \in \mathbb{R}^{|\mathcal{U}|\times k}$ and $\boldsymbol{L} \in \mathbb{R}^{|\mathcal{L}|\times k}$, respectively. Assuming that at time t the check-in location is l for a user, then at time $t+1$, the location latent vector should be similar to $\boldsymbol{A}^T\boldsymbol{L}_l$, which is the expected location latent vector after transition. Instead of evaluating the probability that user u will visit location l at time t given the previous check-in locations at time $t-1$ using Eq. (4.5), we estimate the probability as:

$$\hat{p}(l \in \mathcal{L}_u^t|\mathcal{L}_u^{t-1}) = \eta\boldsymbol{U}_u \cdot \boldsymbol{L}_l + (1-\eta)Sim(\boldsymbol{L}_l, \frac{1}{|\mathcal{L}_u^{t-1}|}\boldsymbol{A}^T \sum_{i\in\mathcal{L}_u^{t-1}} \boldsymbol{L}_i). \tag{4.15}$$

The probability is determined by two components: one is the user preference for the location l, while the other one is based on the location latent space similarity between latent vector of location l and the expected average location latent vector after transition. We use a parameter η to smooth these two factors. The similarity can be measured by many methods. Here, we use the cosine similarity for convenience.

Compared with the FPMC-LR model, we replace the location transition part $\frac{1}{|\mathcal{L}_u^{t-1}|}\sum_{i\in\mathcal{L}_u^{t-1}} \boldsymbol{v}_l^{\mathcal{L},\mathcal{I}} \cdot \boldsymbol{v}_i^{\mathcal{I},\mathcal{L}}$ with the location latent space similarity. We introduce a global latent topic transition matrix $\boldsymbol{A}$ to capture the latent topic transition, instead of modeling the pairwise location transition.

Then, following the same inductions of the FPMC-LR model, we have the same final objective function as that in Eq. (4.11) with parameter set $\Theta = \{\boldsymbol{U}, \boldsymbol{L}, \boldsymbol{A}\}$. The learning algorithm for FPMC-LTT is shown in Algorithm 4.

For each parameter $\theta \in \Theta = \{\boldsymbol{U}, \boldsymbol{L}, \boldsymbol{A}\}$, the update rule is the same as in Eq. (4.14).

Relation to POI recommendation

Successive POI recommendation is closely related to the problem of POI recommendation introduced in Chapter 3. In POI recommen-

Algorithm 4 Learning Algorithm for FPMC-LTT

draw $\boldsymbol{U}, \boldsymbol{L}, \boldsymbol{A}$, from $\mathcal{N}(0, \sigma^2)$
repeat
  draw (u, t, i) uniformly from S
  draw location j uniformly from $N(\mathcal{L}_u^{t-1}) \backslash \mathcal{L}_u^t$
  for $f = 1 \rightarrow K$ **do**
    update $\boldsymbol{U}_{u,f}, \boldsymbol{L}_{i,f}, \boldsymbol{L}_{j,f}$
  end for
  for $f = 1 \rightarrow K$ **do**
    for $l \in \mathcal{L}_u^{t-1}$ **do**
      update $\boldsymbol{L}_{l,f}$
    end for
  end for
  update the transition matrix $\boldsymbol{A}$
until convergence
return $\boldsymbol{U}, \boldsymbol{L}, \boldsymbol{A}$

dation, we do not consider the time effect. We treat all users' POIs at different time as the same, which makes the models in Chapter 3 unable to capture the change of users' tastes. On the other hand, the successive POI recommendation in this chapter is more time sensitive. We would like to provide recommendations to users in the near future. The recommendations is closely related to a user's previous check-ins, which is reflected on the two main properties: personalized Markov chain and region localization. Personalized Markov chain embeds the transition of users' preference. Region localization reflects the geographical influence since users would like to check in POIs near to previous check-ins. And the two main properties motivate us to propose the two models in this chapter. Besides, the problem of successive POI recommendation is more significant because it is more close to users' demand in the real life.

4.4.3 Complexity Analysis

Since we use SGD to learn FPMC-LR and FPMC-LTT, we analyze the cost for each iteration. For FPMC-LR, in each iteration, we update the related parameters and the cost is $O(nK)$, where n is the average set size of users' previous check-ins (the average size of $|\mathcal{L}_u^{t-1}|$) and K is the latent dimension. Usually n is very small. Thus, the computational cost at each iteration is linear to K. For FPMC-LTT, the cost of each iteration is $O(nK + K^2)$. Since we need to update the transition matrix $\boldsymbol{A}$, whose cost is $O(K^2)$. The $O(nK)$ part is the same as FPMC-LR. In general, the algorithms can converge in several hours. We discuss the efficiency in detail in the experimental part.

4.5 Experiments

In the experiments, we address the following questions: 1) How do our approaches compare with the baseline model and other state-of-the-art methods? 2) How does the smooth parameter η affect the model performance? 3) How does the parameter of the geographic grid side length d, which determines the neighbor locations, affect the model performance? 4) What is the convergence and efficiency property of our models?

4.5.1 Datasets

We evaluate the models on the two publicly available LBSNs datasets: Foursquare and Gowalla. Gowalla provides public APIs, which allow us to crawl all users' information including all check-in histories with the time stamp and location details. Although it is not possible to crawl Foursquare data using their APIs directly, part of Foursquare users link their accounts with Twitter and their check-in information can be crawled from Twitter. In this chapter, we use the Foursquare dataset provided by [32] and the Gowalla dataset from [29]. For both datasets, we use four month check-in history of users from May 2010 to August 2010. To remove outliers and clean up the data, we require that every user should have check-ins at least

Table 4.2: Basic statistics of the Foursquare and Gowalla dataset for successive POI recommendation

	$\#U$	$\#L$	# check-in	# avg. check-in
Foursquare	3571	28754	744055	208.36
Gowalla	4510	59355	873071	193.58

120 times and each location should be visited at least five times. The basic statistics are summarized in Table 4.2.

4.5.2 Evaluation Metrics

The experiments are tested as follows: check-ins in the last time slot is used as the test data, while the previous check-in history is used as the training data. Since recommending infrequently visited POIs is more meaningful, we only keep POIs which are visited less than five times by the user before the test period and remove them from the training set. Note that, this setting makes it much harder to recommend new POIs to a user compared with recommending POIs the user has visited before. This can also explain why we can only get very low precision and recall values in the results. In our experiments, we use Precision@k , Recall@k and Mean Average Precision (MAP)@k to evaluate the performance. The precision and recall are defined as:

$$\text{P@}k := \frac{|S|}{k}, \ \text{R@}k := \frac{|S|}{|\mathcal{L}_u^{t+1}|}, \tag{4.16}$$

where $|S|$ is the number of top-k recommended POIs visited by user u at last time $t+1$.

MAP is a well-known metric used to evaluate the top-k performance. The definition is:

$$\text{MAP@}k = \sum_{i=1}^{N} \text{ap@}k_i/N, \tag{4.17}$$

Table 4.3: Performance comparison on Foursquare

Metrics	PMF	PTF	FPMC	FPMC-LR	FPMC-LLT
P@10	0.0185	0.0170	0.0275	0.0360	0.0370
Improve	100.00%	117.65%	34.55%	2.78%	
R@10	0.1542	0.1417	0.2325	0.3033	0.3093
Improve	100.58%	118.28%	33.03%	1.98%	
MAP@10	0.0784	0.0712	0.1265	0.1583	0.1612
Improve	105.61%	126.40%	27.43%	1.83%	

where N is the number of users, and ap@k is the average precision at k for the user:

$$\text{ap@}k = \sum_{i=1}^{k} P(i)/(\text{\# of POIs checked in } k \text{ locations}), \tag{4.18}$$

where $P(i)$ is the precision at cut-off i in the location list.

4.5.3 Comparison

In this section, we compare our models with the following state-of-the-art methods:

1. **PMF**: probabilistic matrix factorization is a well-known method in matrix factorization [112]. It is widely used in traditional recommender systems.

2. **PTF**: probability tensor factorization is introduced in [137] for modeling time evolving relation data.

3. **FPMC**: this method is proposed in [105], which is a strong baseline model embedding users' preference and their personalized Markov chain to provide next-basket item recommendation.

The experimental results on Foursquare and Gowalla datasets are shown in Table 4.3 and Table 4.4. We set the number of latent dimension to 60 for all the compared models and let the time window

Table 4.4: Performance comparison on Gowalla

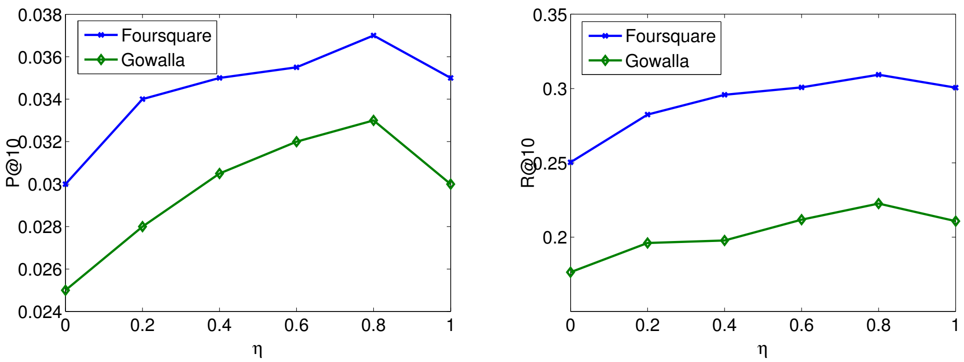
Metrics	PMF	PTF	FPMC	FPMC-LR	FPMC-LLT
P@10	0.0130	0.0110	0.0220	0.0310	0.0330
Improve	153.85%	200.00%	50.00%	6.45%	
R@10	0.1040	0.0785	0.1575	0.2116	0.2226
Improve	114.04%	183.57%	41.33%	5.20%	
MAP@10	0.0575	0.0473	0.0853	0.1072	0.1126
Improve	95.83%	138.05%	32.00%	5.04%	

size be six hours. For our FPMC-LR and FPMC-LTT, we set the geographic grid side length d to be 40 km. We set λ_θ to be 0.03 through setting the last visits in the training as the validation set. The results show that:

- FPMC , FPMC-LR and FPMC-LLT all outperform PMF and PTF significantly. More specifically, FPMC-LR outperforms PMF and PTF by at least 90% and 110% respectively, while FPMC also beats PMF and PTF by 50% and 60% respectively. This implies that personalized Markov chain plays an important role when performing successive POI recommendation. The location transition in short time provides valuable information on where the user would like to go in the next.

- It may be surprising that PMF performs slightly better than PTF. One possible reason may be that PTF assumes that the latent features in successive time periods are similar. However, this assumption is not always valid for the LBSNs data since the features may be periodic. For example, most users have similar preference patterns on every morning or every Sunday. The poor results of PTF imply the assumption of PTF does not fit for the LBSNs data.

- FPMC-LTT and FPMC-LR perform much better than FPMC. For example, FPMC-LR outperforms FPMC by around 30% and 40% for precision and recall, respectively. This verifies



(a) P@10

(b) R@10

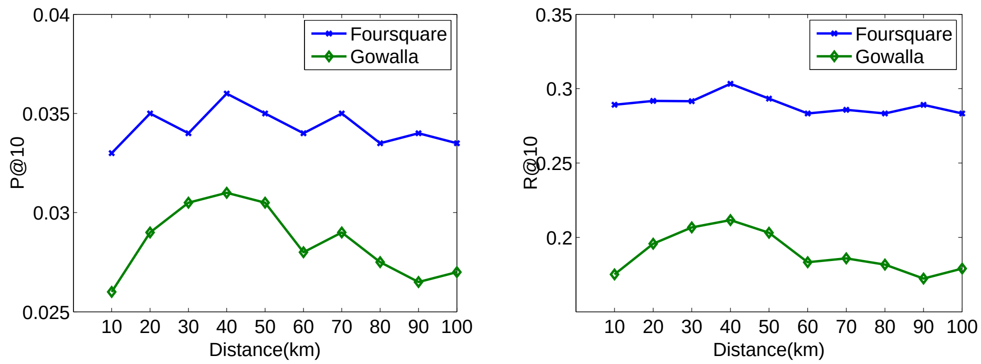
Figure 4.6: Impact of parameter η

that restricting the comparing set to a localized region can reduce noisy information and achieve better performance compared with considering all the locations. As a user's movement is constrained locally in short time, it is enough to only consider the rank-pairs of current check-in and nearby previously visited locations.

- FPMC-LLT further boosts the recommendation performance by around 5% on Gowalla and 2% on Foursquare compared with FPMC-LR. This indicates that modeling the topic transition can alleviate the location-wise transition sparsity problem. Since the Foursquare data are denser than the Gowalla data, we can see that the improvement percentage is less on the Foursquare dataset.

4.5.4 Impact of Parameter η

In FPMC-LLT, the parameter η balances the user's preference on the location and the similarity between the recommended location latent space and expected location latent space after transition. Figure 4.6 shows the impact of η on both Foursquare and Gowalla datasets in



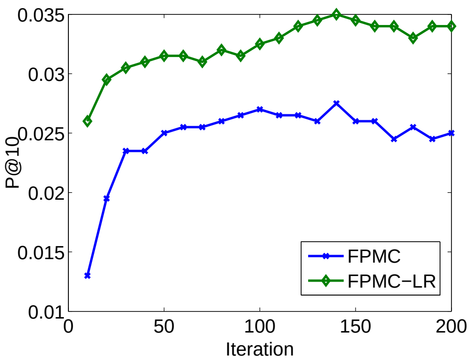
(a) P@10 (b) R@10

Figure 4.7: Impact of parameter d

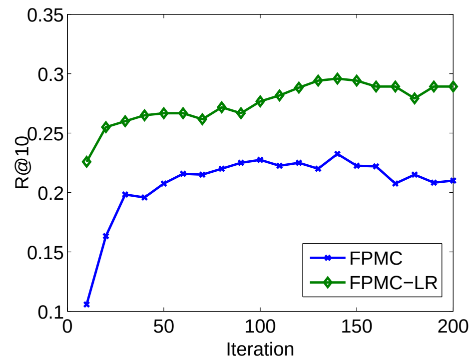
terms of P@10 and R@10. When $\eta = 1$, the FPMC-LLT relies only on the user preference and when $\eta = 0$, the PPMC-LLT relies only on the latent topic space similarity. From the figure, we can observe that the performance improves when η grows from 0 and reaches the best performance when η reaches 0.8, and then the performance decreases again. This indicates that we mainly rely on the user's preference to provide a recommendation, while the latent topic space similarity can help boost the recommendation performance.

4.5.5 Impact of Parameter d

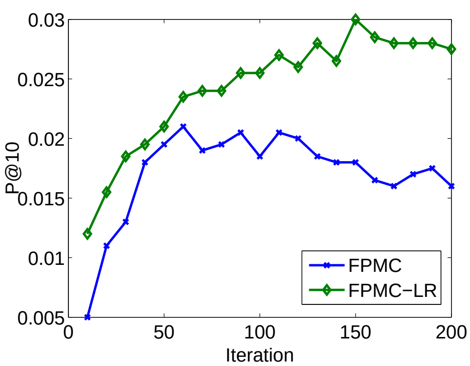
In FPMC-LR and FPMC-LLT, the parameter d is an important factor in controlling the size of the neighborhood check-in history of a user at time t. This parameter affects the number of locations as well as the model performance. We show the results on FPMC-LR here as FPMC-LLT has similar results. Figure 4.7 shows the impact of d in both Foursquare and Gowalla datasets on P@10 and R@10. From the figure, we can see that on both Foursquare and Gowalla, the model performs best when d is 40 km. When d is small, we only consider a very small set of nearby locations, which does not include enough information and yields suboptimal performance. When d is



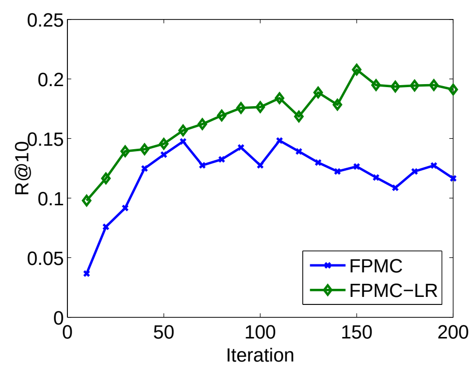
(a) P@10(Foursquare)



(b) R@10(Foursquare)



(c) P@10(Gowalla)



(d) R@10(Gowalla)

Figure 4.8: Convergence analysis

large, e.g., 100 km, the model has to consider much more rank-pairs and may introduce more noisy information, which yields poor performance. An extreme case is to set d large enough to cover all neighborhood areas in the whole earth and consider all locations, which is equivalent to the case of FPMC model. The obtained results confirm the intuition that the localization constraint plays an important role in successive POI recommendation.

4.5.6 Convergence and Efficiency Analysis

Figure 4.8 shows the performance change of our FPMC-LR and FPMC with respect to the number of iterations. We do not show FPMC-LTT as its results are similar to FPMC-LR. Here, at each iteration, we draw 2×10^5 quadruples to calculate the stochastic gradient descent based on the BPR criterion. Our experiments are conducted on a PC with an Intel Pentium D CPU (3.0 GHz, Dual Core) and 2G memory. An average time for an iteration is about 30 seconds. From the figure, we can see that at each iteration, FPMC-LR always performs better than FPMC and attains its best performance at around 150 iterations.

Here, we also claim another advantage of PFMC-LR in the recommendation procedure: its efficiency. When recommending potential POIs, FPMC needs to consider all the locations, while FPMC-LR only considers the neighborhood locations of a user's current location (usually less than 1% of the whole location).

4.6 Conclusion

In this chapter, we considered the task of successive POI recommendation in LBSNs. We first investigated the spatial-temporal properties of the two LBSNs datasets. We then proposed two novel matrix factorization models, i.e., FPMC-LR and FPMC-LLT, to include both personalized Markov chain and localized regions for solving the recommendation task. Our experimental results on two large-scale LBSNs datasets showed the effectiveness and efficiency of our models compared with several state-of-the-art methods.

There are still several other aspects worthy of consideration in the future: 1) how can we utilize the contextual information of POIs, e.g., the location category and the activities conducted there; 2) how to incorporate the users' periodic check-in behaviors to capture users' periodic preferences; 3) how to find more useful check-in sequences, e.g., higher-order Markov chain; 4) How to incorporate social information to strengthen successive POI recommendation. Progress in these directions would advance POI recommendation field for our community.

□ **End of chapter.**

Chapter 5

Gradient Boosting Factorization Machines

Recommendation techniques have been well developed in the past decade. Most of them build models only based on the user-item rating matrix. However, in the real world, there is plenty of auxiliary information available in recommendation systems. We can utilize these information as additional features to improve recommendation performance. We refer to recommendation with auxiliary information as context-aware recommendation. Context-aware Factorization Machines (FM) is one of the most successful context-aware recommendation models. FM models pairwise interactions between all features, in this way, a certain feature latent vector is shared to compute the factorized parameters it involves. In practice, there are tens of context features but not all the pairwise feature interactions are useful. Thus, one important challenge for context-aware recommendation is how to effectively select "good" interacting features. In this chapter, we focus on solving this problem and propose a greedy interacting feature selection algorithm based on gradient boosting. Then we propose a novel Gradient Boosting Factorization Machines (GBFM) model to incorporate the feature selection algorithm with Factorization Machines into a unified framework. The experimental results on both synthetic and real datasets demonstrate the efficiency and effectiveness of our algorithm compared with other state-of-the-art methods.

5.1 Introduction

Recommendation systems have been well studied in the past decade. Matrix factorization methods [61, 112] have become popular due to their good performance and efficiency in dealing with large-scale datasets. These methods focus on approximating the user-item rating matrix using low rank representations. Most of them only consider user and item interactions and ignore context information. However, in real world scenarios, plenty of auxiliary information is available and is proven to be useful especially in large-scale industry datasets. For example, in the Weibo celebrity recommendation scenario, the user's and celebrity's age and gender, the popularity of a celebrity, the recent following behavior of the user, etc., can help make better recommendations. Recent work in KDDCup 2012 [102, 27] show the effectiveness of utilizing auxiliary information for recommendations.

In terms of utilizing auxiliary information, several methods have been studied to incorporate meta-data (e.g., user profile, movie genre, etc.) [68, 130] and more general auxiliary information [135, 4, 57, 101, 106]. In these methods, auxiliary information was encoded as features; together with user and item, they were mapped from a feature space into a latent space. The Factorization Machines (FM) model [106] is currently a general and widely used method that can easily incorporate any context feature. In FM, all features are assumed to be interacting with all other features. For example, assuming there are n features, then for a certain feature i, the latent vector $\boldsymbol{v}_i$ is shared with $n-1$ interacting features. It is not always the case that all the feature interactions are useful. Useless feature interactions can introduce noise in learning latent feature vector $\boldsymbol{v}_i$. Thus it is challenging to select useful interacting features automatically to reduce noise.

The most recent work in [26] introduced an automatic feature construction method in matrix factorization using gradient boosting. In their method, feature functions were constructed using a greedy gradient boosting method and then incorporated into the matrix factorization framework. Different from their method, in this chapter, we focus on selecting useful interacting features under the Factorization Machines framework. At each step, we propose a greedy gradient boosting method to select interacting features efficiently. Then we additively optimize the selected latent vector by optimizing the residual loss. Another difference is that our method is more efficient in selecting categorical interacting features compared with the binary decision tree construction algorithm in their paper. In this chapter, the main contributions are summarized as follows:

- We propose an efficient interacting feature selection algorithm using gradient boosting, which can reduce noise compared with the Factorization Machines method.
- We propose a novel Gradient Boosting Factorization Machines (GBFM) model by incorporating the feature selection algorithm with Factorization Machines into a unified framework.
- The experimental results on both synthetic and real datasets show the effectiveness of our proposed method compared with Factorization Machines and other state-of-the-art methods.

The rest of this chapter is organized as follows. Section 5.2 introduces the related work. Section 5.3 gives the details of our method. Section 5.4 presents the experimental results and discussions. The chapter is concluded in Section 5.5.

5.2 Related Work

The work presented in this chapter is closely related to matrix factorization, context-aware recommendation and gradient boosting. In the following, we briefly review the related work.

5.2.1 Matrix Factorization

Matrix factorization techniques [129, 12, 61, 112, 4] have been shown to be particularly effective in recommender systems as well as the well-known Netflix prize competition. They usually outperform the item-based method [118]. The main idea behind matrix factorization is to learn two low rank latent matrices $U \in \mathbb{R}^{k\times m}$ and $V \in \mathbb{R}^{k\times n}$ to approximate the observed user-item rating matrix $R \in \mathbb{R}^{m\times n}$ so that

$$R \approx U^T V, \tag{5.1}$$

where m, n are the number of users and items respectively, and k is the latent dimension.

The objective function for matrix factorization is that:

$$\min_{U,V} \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}^{R}(R_{ij} - U_i^T V_j)^2 + \frac{\lambda_1}{2}\|U\|_F^2 + \frac{\lambda_2}{2}\|V\|_F^2, \tag{5.2}$$

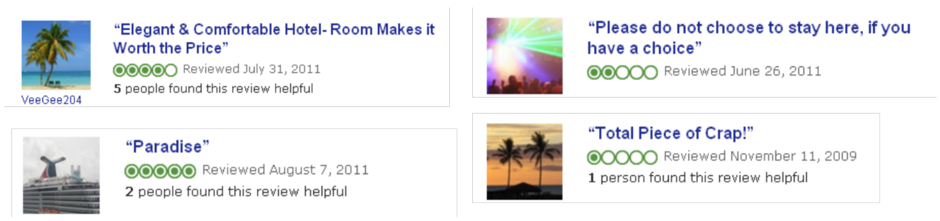
where I_{ij}^R is the indicator function that equals to 1 if user i rated item j and equals to 0 otherwise. The above equation can be easily solved either by stochastic gradient descent (SGD) or alternating least squares (ALS).

5.2.2 Context-aware recommendation

Contextual information has been proven to be useful in recommender systems and context-aware recommendation has already been widely studied. In order to incorporate auxiliary information, several variants of matrix factorization have been proposed. In [63, 137], temporal features were explored to help capture the user's preference more precisely. Social [85, 52] and location information [143, 29, 30] have also been explored. The meta-data, such as the user's age and the item's genre, was incorporated into the matrix factorization model [130, 68]. In addition to the meta-data, which is attached to user or item itself, context features also include the information attached to the whole recommendation event, such as a user's mood, day of the week, etc.

There are several methods explored to deal with all of those context features. The most basic approach for context-aware recommendation is to conduct pre-filtering or post-filtering where a standard context-unaware method is applied [97, 2]. More complicate methods like matrix factorization were also explored. Karatzoglou et al. [57] proposed a Multiverse recommendation model by modeling the data as a user-item-context N-dimension tensor. However, the computational cost of the model is high, which is intolerable in practice. Rendle et al. [106] proposed to apply Factorization Machines (FM) [101] to overcome the problem in Multiverse recommendation. The authors transformed the recommendation problem into a prediction problem and FM modeled all interactions between pairs of features with the target. They further proposed to deal with relational data in [103] through the block structure within a feature.

The idea of tree based random partition has been explored in [153, 81]. Zhong et al. [153] assumed that contextual information was reflected by user and item latent vectors. In their method, the random tree partition was conducted to split the user-item matrix by grouping users and items with similar contexts. Then matrix factorization was applied on the sub-matrices. Liu et al. [81] employed the similar idea but explicitly used context information to split the user-item matrix into sub-matrices according to specific context values. The prediction was the average value of each prediction from T generated decision trees. However, they failed to discuss how to select useful features especially when there are tens of features.



(a) Online rating example

	v_1	v_2	v_3	v_4	v_5	v_6
u_1		5	2		3	
u_2	4			3		4
u_3			2			2
u_4	5			3		
u_5		5	5			3

(b) User-item rating matrix

	User			Movie				Mood			…	R
$\mathbf{x}^{(1)}$	1	0	0	1	0	0	0	1	0	0	…	4
$\mathbf{x}^{(2)}$	0	1	0	0	1	0	0	0	0	1	…	2
$\mathbf{x}^{(3)}$	1	0	0	0	1	0	0	0	1	0	…	5
$\mathbf{x}^{(4)}$	0	0	1	0	0	1	0	0	0	1	…	1

(c) Context-aware prediction

Figure 5.1: Context-aware recommendation

5.2.3 Gradient Boosting

Gradient Boosting has been successfully used in classification [44] and learning-to-rank [138, 19]. In each step, gradient boosting greedily conducts coordinate descent in the function space to select a feature function. Chen et al. [26] proposed to use gradient boosting method to construct a feature function for each user/item latent vector at each step automatically. Different from their work, in this chapter, we employ the gradient boosting algorithm to find the best interacting feature at each step. Then similar to gradient boosting, we additively optimize latent feature vectors. Besides, in the real world, there are many categorical features with large size, however, the binary tree splitting algorithm in their work is not efficient in dealing with such features.

5.3 Gradient Boosting Factorization Machines

In this section, we first describe the context-aware recommendation problem we study in this chapter. Then we briefly review context-aware FM, which is closely related to our work. Lastly, we present the details of our proposed Gradient Boosting Factorization Machines model with complexity analysis and discussions.

5.3.1 Preliminaries and Problem Definition

Traditional recommendation systems only consider the user-item rating matrix as in Figure 5.1(b) to make recommendations. However, rich context information is available in the real world. Figure 5.1(a) shows an example of the context-aware online rating system. In the example, we can easily get the rating time and the rating comments. These information provides a new information dimension for recommendations. We can encode the context information, together with user and item, as either real value or categorical features. The corresponding rating is encoded as the target value. This way, we can transform the context-aware recommendation problem into a prediction problem as shown in Figure 5.1(c). The figure shows an example about users $\mathcal{U}$ watch movies $\mathcal{I}$ in mood $\mathcal{M}$:

$$\begin{aligned}\mathcal{U} &= \{u_1, u_2, u_3\},\\ \mathcal{I} &= \{i_1, i_2, i_3, i_4\},\\ \mathcal{M} &= \{Happy, Normal, Sad\}.\end{aligned}$$

We first give the definition of *context*. *Context* is the additional information available which can influence the rating behavior. For example, the mood of the user when he/she rated the item, the user's gender, the quality of the item, etc., are all context. We define a context as a variable $c \in \mathcal{C}$. For example, the first tuple in Figure 5.1(c) states that user u_1 gave movie i_1 4 stars in a *Happy* mood. Next we give the formal definition of the context-aware recommendation problem. We denote the user set as $\mathcal{U}$ and the item set as $\mathcal{V}$. Assume there are $m-2$ additional context features, we further denote the context features as $\mathcal{C}_3, \ldots, \mathcal{C}_m$. In fact, user and item can be regarded as the first and second "context" feature, respectively. For simplicity, we denote the user set as $\mathcal{C}_1$ and the item set as $\mathcal{C}_2$. In this chapter, we only consider categorical features for simplicity, since in practice most features are categorical and real value features can also be segmented into categorical features [102].

The training data can be encoded as feature vectors as shown in Figure 5.1(c) by transforming categorical features to indicator variables. We denote n_i as the number of different feature values for context feature $\mathcal{C}_i$. Each context feature set is $\mathcal{C}_i = \{c_{i,0}, \ldots, c_{i,n_i}\}$. We further denote the length of the feature vector as d, which equals to $n_1 + \ldots + n_m$. The training data are denoted as $\mathcal{S} = \sum_{i=1}^{N}(\mathbf{x}_i, y_i)$, where N is the number of total training instances. $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \mathbb{R}$ are the feature vector and target value for instance i, respectively. Our problem is to estimate the following rating function

$$\hat{y} : \mathbb{R}^d \to \mathbb{R}, \tag{5.3}$$

which minimizes the following objective function:

$$\arg\min_{\Theta} \sum_{i=1}^{N} l(\hat{y}_i, y_i) + \Omega(\hat{y}). \tag{5.4}$$

Here l is a differentiable convex loss function that measures the difference between the prediction rating $\hat{y}_i$ and the target rating y_i, and Θ is the parameter set to be estimated. The second term Ω measures the complexity of the model to avoid overfitting.

5.3.2 Context-aware Factorization Machines

The Factorization Machines model [101] is a generic model class that subsumes many well-known recommendation methods including SVD++ [61], matrix factorization [129] and PITF [107]. Rendle et al. [106] proposed to apply FM to solve the context-aware recommendation problem and it is proven to be effective in KDDCup 2012 [102] as well.

In [106], FM was restricted to be 2-way FM. In this setting, the FM models all interactions between pairs of variables with the target including nested ones, by using factorized interaction parameters. The rating prediction function is:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} \hat{w}_{i,j} x_i x_j. \tag{5.5}$$

From the perspective of classification problem, w_0 is the global bias, w_i is the weight for feature x_i, and $\hat{w}_{i,j}$ is the weight for feature $x_i x_j$. We refer to the feature $x_i x_j$ as the interacting feature, which indicates the instance have both the feature value x_i and x_j. For example, in the first tuple in Figure 5.1(c), one interacting feature is $u_1 v_1$ since we have user u_1 and item v_1 in the tuple.

The factorized parameter $\hat{w}_{i,j}$ is defined as:

$$\hat{w}_{i,j} := \langle \mathbf{v}_i, \mathbf{v}_j \rangle = \sum_{f=1}^{k} v_{i,f} \cdot v_{j,f}. \tag{5.6}$$

The model parameters Θ that need to be estimated are:

$$w_0 \in \mathbb{R},\ \mathbf{w} \in \mathbb{R}^d \text{ and } \mathbf{V} \in \mathbb{R}^{d\times k}. \tag{5.7}$$

Note that the latent matrix $\mathbf{V}$ can be regarded as the concatenation of all m latent feature matrices $\mathbf{V}_i \in \mathbb{R}^{n_i \times k}, i = 1, \ldots, m$. The final objective function is:

$$\arg\min_{\Theta} \sum_{i=1}^{N} l(\hat{y}(\mathbf{x}_i), y) + \sum_{\theta\in\Theta} \lambda_{(\theta)}\theta^2, \tag{5.8}$$

where $\lambda_{(\theta)}$ is the regularization parameter. In practice, l can be the logit loss for binary classification:

$$l(\hat{y}, y) = \log(1 + \exp(-\hat{y}y)), \tag{5.9}$$

or the least square loss for regression:

$$l(\hat{y}, y) = (\hat{y} - y)^2. \tag{5.10}$$

In the real world, it is not surprising that we can have tens of context features[1] and not all interacting features are useful for rating predictions. Note that FM models all pairwise interactions between context features and the weight of the interacting feature is defined in Eq. (5.6). The latent vector $\mathbf{v}_i$ is shared by every feature vector $\mathbf{v}_j$ in order to estimate the feature weight $\hat{w}_{i,j}$. If the interacting feature $x_i x_j$ is not useful (i.e., the estimate function $\hat{y}$ does not have the item $\hat{w}_{i,j}x_i x_j$), in such case, the estimation for parameter $\mathbf{v}_j$ and $\mathbf{v}_j$ will be affected. In order to select "good" interacting features effectively, we propose our Gradient Boosting Factorization Machines model.

5.3.3 Gradient Boosting Factorization Machines

GBFM Training

In this section, we present our proposed GBFM training algorithm. We borrow the idea of the boosting method [38] to select one inter-

[1] In KDDCup 2012 Task 1 we can easily extract around 20 features and there are more features in the real industry world.

acting feature at each step and additively optimize the target function. The rating prediction function in our model is defined as:

$$\hat{y}_s(\mathbf{x}) := \hat{y}_{s-1}(\mathbf{x}) + \sum_{i \in \mathcal{C}_p} \sum_{j \in \mathcal{C}_q} \mathbb{I}[i, j \in \mathbf{x}] \langle \mathbf{V}_p^i, \mathbf{V}_q^j \rangle, \tag{5.11}$$

where s is the iteration step of the learning algorithm. At step s, we greedily select two interacting features $\mathcal{C}_p$ and $\mathcal{C}_q$, where $\mathbb{I}$ is the indicator function whose value is 1 if the condition holds otherwise is 0. The feature selection algorithm will be introduced later. $\mathbf{V}_p \in \mathbb{R}^{n_p \times k}$ and $\mathbf{V}_q \in \mathbb{R}^{n_q \times k}$ are the low rank matrices for feature $\mathcal{C}_p$ and feature $\mathcal{C}_q$, respectively, where k is the low rank dimension. After interacting features $\mathcal{C}_p$ and $\mathcal{C}_q$ are selected, we estimate the parameters $\mathbf{V}_p$ and $\mathbf{V}_q$ at step s. For example, at step s we select the interaction between user and item, we need to learn the low rank latent matrices $\mathbf{U}$ and $\mathbf{V}$. The objective function for estimating $\mathbf{V}_p$ and $\mathbf{V}_q$ is:

$$\operatorname*{arg\,min}_{\mathbf{V}_p, \mathbf{V}_q} \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \|\mathbf{V}_p\|_F^2 + \|\mathbf{V}_q\|_F^2. \tag{5.12}$$

Assume we totally have S steps and denote the interacting features selected at step s as $\mathcal{C}_{sp}$ and $\mathcal{C}_{sq}$, then the final prediction function is:

$$\hat{y}_S(\mathbf{x}) = \hat{y}_0(\mathbf{x}) + \sum_{s=1}^{S} \sum_{i \in \mathcal{C}_{sp}} \sum_{j \in \mathcal{C}_{sq}} \mathbb{I}[i, j \in \mathbf{x}] \langle \mathbf{V}_{sp}^i, \mathbf{V}_{sq}^j \rangle, \tag{5.13}$$

where $\hat{y}_0(x)$ is the initialized prediction function.

The details of the algorithm are shown in Algorithm 5.

Algorithm 5 Gradient Boosting Factorization Machines Model

```
Input: Training Data 𝒮 = {x_i, y_i}_{i=1}^N
Output: ŷ_S(x) = ŷ_0(x) + Σ_{s=1}^S ⟨v_si, v_sj⟩
Initialize rating prediction function as ŷ_0(x)
for s = 1 → S do
    Select interacting features C_p and C_q from Greedy Feature Selection Algorithm
    Estimate latent feature matrices V_p and V_q
    Update ŷ_s(x) := ŷ_{s−1}(x) + Σ_{i∈𝒞_p} Σ_{j∈𝒞_q} 𝕀[i, j ∈ x]⟨V_p^i, V_q^j⟩
end for
```

Input: Training Data $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^N$
Output: $\hat{y}_S(x) = \hat{y}_0(x) + \sum_{s=1}^S \langle \mathbf{v}_{si}, \mathbf{v}_{sj} \rangle$
Initialize rating prediction function as $\hat{y}_0(x)$
for $s = 1 \to S$ **do**
    Select interacting features C_p and C_q from *Greedy Feature Selection Algorithm*
    Estimate latent feature matrices $\mathbf{V}_p$ and $\mathbf{V}_q$
    Update $\hat{y}_s(\mathbf{x}) := \hat{y}_{s-1}(\mathbf{x}) + \sum_{i \in \mathcal{C}_p} \sum_{j \in \mathcal{C}_q} \mathbb{I}[i, j \in \mathbf{x}] \langle \mathbf{V}_p^i, \mathbf{V}_q^j \rangle$
end for

Greedy Feature Selection Algorithm

In this section, we show how to effectively select the "good" interacting feature at each step, which is the core part of our model. From the view of gradient boosting, at each step s, we would like to search a function f in the function space $\mathcal{F}$ that minimizes the objective function:

$$\mathcal{L} = \sum_{i=1}^{N} l(\hat{y}_s(\mathbf{x}_i), y_i) + \Omega(f), \tag{5.14}$$

where $\hat{y}_s(\mathbf{x}) = \hat{y}_{s-1}(\mathbf{x}) + \alpha_s f_s(\mathbf{x})$. In our GBFM, the function f is set to factorized feature interactions like in FM. However, it is impossible to search all feature interactions to find the best one (i.e., decrease the objective function most) due to the high computational cost. In order to find the desirable interacting feature, we propose a greedy layer-wised algorithm to find the n-way interacting feature. Our idea is as follows: suppose that there are n layers in total, at each layer, we greedily select the feature $\mathcal{C}_i$ that makes the objective function decrease fastest. At the end, we will get the n-way interacting feature. We assume that the function f has the following form:

$$f_l(\mathbf{x}) = \prod_{t=1}^{l} q_{\mathcal{C}_{i(t)}}(\mathbf{x}), \tag{5.15}$$

where $q_{\mathcal{C}_{i(t)}}(\mathbf{x})$ is the function learned at layer t with $i(t)$-th context feature selected. The q function maps the latent feature vector $\mathbf{x}$ to the real value domain. There are d elements in the feature set and for each element we assign a weight w_{tj} to it. The function $q_{\mathcal{C}_{i(t)}}(\mathbf{x})$ is defined as:

$$q_{\mathcal{C}_{i(t)}}(\mathbf{x}) = \sum_{j \in \mathcal{C}_{i(t)}} \mathbb{I}[j \in \mathbf{x}] \cdot w_{tj}, \tag{5.16}$$

where $\mathbb{I}$ is the indicator function. Although the q function looks very complex, in fact, in each instance there is only one non-zero element corresponding to feature $\mathcal{C}_{i(t)}$. The value of the q function just takes the weight corresponding to the non-zero element in feature $C_{i(t)}$. Take Figure 5.1(c) as an example again, suppose we select feature $\mathcal{C}_1$ at layer t, then the q function for first tuple is w_{t1}.

Searching the function f to optimize the objective function in Eq. (5.14) can be hard for a general convex loss function l. The most common way is to approximate it by least-square minimization [37]. We denote the negative value of the first derivative and the second derivative at instance i as g_i and h_i, respectively:

$$g_i = -\frac{\partial l(\hat{y}_s(\mathbf{x}_i), y_i)}{\partial \hat{y}_s(\mathbf{x}_i)}|_{\hat{y}_s(\mathbf{x}_i)=\hat{y}_{s-1}(\mathbf{x}_i)}, \tag{5.17}$$

$$h_i = \frac{\partial^2 l(\hat{y}_s(\mathbf{x}_i), y_i)}{\partial \hat{y}_s(\mathbf{x}_i)^2}|_{\hat{y}_s(\mathbf{x}_i)=\hat{y}_{s-1}(\mathbf{x}_i)}. \tag{5.18}$$

The first part of Eq. (5.14) can be approximated as:

$$\begin{aligned} \mathcal{L} &= \sum_{i=1}^{N} l(\hat{y}_{s-1}(\mathbf{x}_i) + \alpha_s f_s(\mathbf{x}_i), y_i) \\ &\approx \sum_{i=1}^{N} l(\hat{y}_{s-1}(\mathbf{x}_i), y_i) - g_i(\alpha_s f_s(\mathbf{x}_i)) + \frac{1}{2} h_i(\alpha_s f_s(\mathbf{x}_i))^2. \end{aligned} \tag{5.19}$$

Then Eq. (5.14) is equivalent to:

$$\mathcal{L}(f) = \sum_{i=1}^{N} h_i(g_i/h_i - f_s(\mathbf{x}_i))^2 + \Omega(f_s). \tag{5.20}$$

By replacing the f_s function with the function f defined in Eq. (5.15), we get the objective function for selecting n-way interacting features. Even using heuristic functions, finding the best interacting feature is still impossible. Instead, we learn the function f layer by layer. At layer t, we assume that the functions $q_{\mathcal{C}_{i(1)}}, \ldots, q_{\mathcal{C}_{i(t-1)}}$ have been learned, i.e., f_{t-1} has been learned, and we select the $i(t)$-th feature, then we have:

$$f_t(\mathbf{x}) = f_{t-1}(\mathbf{x}) \cdot q_{\mathcal{C}_{i(t)}}(\mathbf{x}). \tag{5.21}$$

Our problem is finalized to find the $i(t)$-th feature:

$$\operatorname*{arg\,min}_{i(t)\in\{1,\ldots,m\}} \sum_{i=1}^{N} h_i(\frac{g_i}{h_i} - f_{t-1}(\mathbf{x}_i) \cdot q_{\mathcal{C}_{i(t)}}(\mathbf{x}_i))^2 + \lambda \sum_{\theta\in\Theta} \theta^2. \tag{5.22}$$

Here we use L2-regularization to control the model complexity. To obtain the feature $i(t)$ that minimizes Eq. (5.22), we calculate the q function for all features. Without loss of generality, we assume the selected feature at layer t is $\mathcal{C}_{i(t)}$. The problem is actually transformed to estimate the weight for each $n_{i(t)}$ item in the feature $\mathcal{C}_{i(t)}$. For a certain element j in $\mathcal{C}_{i(t)}$, we denote its corresponding weight as w_{ij}. The solution for w_{ij} is:

$$w_{ij} = \operatorname*{arg\,min}_{w} \sum_{i=1}^{N} h_i(g_i/h_i - f_{t-1}(\mathbf{x}_i) \cdot \mathbb{I}(j \in \mathbf{x}_i) \cdot w)^2 + \lambda w^2. \tag{5.23}$$

We denote $z_i = g_i/h_i$ and let

$$\begin{aligned} a &= \sum_{i=1}^{N} \mathbb{I}(j \in \mathbf{x}_i) z_i h_i f_{t-1}(\mathbf{x}_i), \\ b &= \sum_{i=1}^{N} \mathbb{I}(j \in \mathbf{x}_i) h_i (f_{t-1}(\mathbf{x}_i))^2. \end{aligned} \tag{5.24}$$

Then the analytic solution for w_{ij} is:

$$w_{ij} = \frac{a}{b+\lambda}. \tag{5.25}$$

Note that although we need to calculate the q function for all features, we can compute a and b for all features at the same time by scanning the training data just once. After we get the q function for all features, it is easy to select the best feature that satisfies Eq. (5.22). We repeat this process at each layer, at the end, we can obtain the heuristic n-way interacting feature. Like FM, in our method, we consider 2-way interacting feature only. The details of the algorithm are shown in Algorithm 6.

Algorithm 6 Greedy Feature Selection Algorithm

Input: Training Data $\mathcal{S} = \{\mathbf{x}_i, y_i\}_{i=1}^{N}$, context feature set $\mathcal{C}$
Output: n-way interacting feature $\mathcal{C}_{i(1)}, \ldots, \mathcal{C}_{i(n)}$.
for $l = 1 \to n$ **do**
  $\mathcal{A} = \emptyset$ // $\mathcal{A}$ is the set of context features already selected
  Maintain two vectors $\mathbf{a}$ and $\mathbf{b}$ for all categorical values in $\mathcal{C}$, both initialized to $\mathbf{0}$
  for $(\mathbf{x}_i, y_i)$ in $\mathcal{S}$ **do**
    compute $tempa = z_i h_i f_{t-1}(\mathbf{x}_i)$ and $tempb = h_i(f_{t-1}(\mathbf{x}_i))^2$
    for $j = 1 \to d$ **do**
      if $\mathbf{x}_{ij}$ is non-zero and not in $\mathcal{A}$ **then**
        add $tempa$ to $\mathbf{a}_j$ and $tempb$ to $\mathbf{b}_j$
      end if
    end for
  end for
  Compute weights for all categorical features in $\mathcal{C} - \mathcal{A}$ according to Eq. (5.23).
  Select the feature $\mathcal{C}_{i(l)}$ according to Eq. (5.22).
  Add feature $\mathcal{C}_{i(l)}$ into $\mathcal{A}$
end for

Complexity Analysis

The computational complexity for *Greedy Feature Selection Algorithm* is $O(nN)$, where N is the training data size, and n is the number of layers. At each layer, the best feature can be selected by scanning the training dataset once as described in Algorithm 6. Usually, $n \ll N$ and in 2-way FM, $n = 2$. So the computational cost for Algorithm 6 is $O(N)$.

In Algorithm 5, the estimation for $\mathbf{V}_p$ and $\mathbf{V}_q$ is usually carried out by stochastic gradient descent (SGD). The complexity for this part is $O(kN)$, where k is the number of iterations. In total, the complexity for GBFM is $O(SN + kSN)$, where S is the number of boosting steps as stated in the algorithm. The computational complexity is still linear to the training dataset size.

In addition, GBFM can be speedup by multi-threading and parallelization. The computation of first and second derivatives can be decoupled, thus the derivatives can be easily computed by multi-threading or by a cluster of computers. The gradient of Eq. (5.12) also can be decoupled and parallelization is possible for Algorithm 5.

Discussions

In this section, we first discuss the insights of our heuristic function f in Eq. (5.15), which is the key part of Algorithm 6. Then we discuss the relationship between the proposed GBFM and other state-of-the-art methods. At last, we discuss some variants of our model.

Insights of heuristic function f: The main idea of our algorithm is that at each layer we greedily select a context feature $\mathcal{C}_i$ according to Eq. (5.22) and we compute the corresponding weight vector, i.e., $q_{\mathcal{C}_i}$. We can regard it as the low rank latent feature matrix for feature $\mathcal{C}_i$ like in FM with the latent dimension $k = 1$. Then the heuristic function f is an instance of CANDECOMP/PARAFAC (CP) decomposition [59] with $k = 1$. We greedily use this f function to choose interacting features. In practice, for large-scale datasets in the industry, we can additively use this function f as the "weak learner" instead of $\langle \mathbf{V}_p, \mathbf{V}_q \rangle$ to quickly find useful interacting features, since the computational cost is relatively low.

Relation to Factorization Machines: The Factorization Machines (FM) model is a strong baseline method for context-aware recommendation [106]. The main difference between our model and FM is that FM models all interactions between context features, while our method only considers part of them. For example, in Figure 5.1(c), the rating prediction function of FM is:

$$\begin{aligned} \hat{y}(\mathbf{x}(u,i,c_3)) &= w_0 + w_u + w_i + w_{c_3} + \langle \mathbf{v}_u, \mathbf{v}_i \rangle \\ &+ \langle \mathbf{v}_u, \mathbf{v}_{c_3} \rangle + \langle \mathbf{v}_i, \mathbf{v}_{c_3} \rangle. \end{aligned} \tag{5.26}$$

However in our GBFM, we may only consider the (user,item) and (user,mood) interaction pairs. If the interacting feature (item,mood) is actually not useful, then the term $\langle \mathbf{v}_i, \mathbf{v}_{c_3} \rangle$, which is the weight for feature $\mathbf{x}_i \mathbf{x}_{c_3}$, will introduce noise for the prediction function. Another difference is that in our algorithm, we additively learn the latent feature matrices, which are not shared to compute other factorization weights. For example, suppose that we select the (user,item) pair in the first step and the (user,mood) pair in the second step, the latent feature matrix $\mathbf{V}_u$ is not the same. It may lose the advantage of generalization compared with FM.

Relation to GBMF: Gradient Boosting Matrix Factorization [26] is the state-of-the-art model, which is a general functional matrix factorization using gradient boosting. GBMF is under the framework of matrix factorization [112]. The model assumes that the user/item latent low rank matrix is functional and each time a function f is added to the latent dimension $\mathbf{U}_k$. Different from GBMF, our model is under the framework of Factorization Machines and we use gradient boosting to greedily select "good" interacting features. Another difference is the construction method of high-order categorical features. In our algorithm, we can efficiently find the "best" features according to Algorithm 6, while their binary splitting tree algorithm may fail for categorical features since the cost for finding the best binary split is exponential.

Variants of our GBFM: There are several variants of our proposed GBFM. In this chapter, we only use the 2-way interacting feature like in FM. It can be easily extended to n-way FM by selecting the n-way interacting feature. Another variant is that we can fully optimize the selected interacting features instead of additively optimizing interacting features one by one. We refer to this variant as GBFM-Opt. The difference between GBFM-Opt and FM is that GBFM-Opt only considers some "good" selected 2-way interacting features, while FM considers all of them.

5.4 Experiments

In this section, we empirically investigate whether our proposed method can achieve better performance compared with other state-of-the-art methods with large number of context features. Furthermore, we would like to examine whether the interacting features selected by our algorithm are more effective compared with pairwise interactions in FM.

5.4.1 Datasets

We conduct our experiments on two datasets: a synthetic dataset and a real world dataset, i.e., the Tencent microblog[2].

Synthetic data: Since there are few public datasets that have many context features. We construct a synthetic dataset for comparison. The data generation process is as follows: assume that we have m context features and each context feature $\mathcal{C}_i$ has n_i values, we generate the latent context features from zero-mean spherical Gaussian

[2] http://kddcup2012.org/c/kddcup2012-track1/data

Table 5.1: Statistics of datasets

Dataset	# Users	#Items	#Observed Entries
Synethic data	1000	1000	16270
Tencent microblog	2.3 M	6095	73 M

as follows:

$$\mathbf{V}_i^j \sim \mathcal{N}(\mathbf{0}_K, \sigma^2 \mathbf{I}_K),$$

where $j = 1, \ldots, n_i$, $\mathbf{0}_K$ is a K-dimension vector with all elements set to 0, and $\mathbf{I}_K$ is the $K \times K$ identity matrix. We also generate the weight vector $\mathbf{w} \sim \mathcal{N}(\mathbf{0}_{d+1}, \sigma^2 \mathbf{I}_{d+1})$, where $d = \sum_{i=1}^m n_i$. We incorporate the global bias into the weight vector. Then we select several 2-way interacting features. We denote the set of interacting features as $\mathcal{F}$. Then the rating is obtained by rescaling the sigmoid function value from 1 to D by:

$$\begin{aligned} \hat{y}(\mathbf{x}) &= \sum_{i=0}^{d} w_i x_i + \sum_{(p_1,p_2)\in\mathcal{F}} \sum_{i\in\mathcal{C}_{p_1}} \sum_{j\in\mathcal{C}_{p_2}} \mathbb{I}[i, j \in \mathbf{x}] \langle \mathbf{V}_{p_1}^i, \mathbf{V}_{p_2}^j \rangle, \\ \hat{y}(\mathbf{x}) &= \lceil g(\hat{y}) \times D \rceil, \end{aligned}$$

where D is the rating scale, and $g(x) = 1/(1 + \exp(-x))$. In our experiments, we set the number of context features $m = 10$, latent dimension $K = 5$, rating scale $D = 5$ and feature value size $n_i = 1000$.

Tencent microblog dataset: Tencent microblog is one of the largest social media services in China like Sina Weibo and Twitter. The dataset is designed for KDDCup 2012 competition and it contains the celebrity recommendation records of about 2.3 million users over a time period of about two months. In this dataset, the celebrities are regarded as items for recommendations. The system recommends a celebrity to a user at a certain time and the user's response is either "accept" or "reject". The dataset contains rich context information such as the user's age and gender, the item's category, time information, etc. We can also extract the session information such as the number of recommendation records before current recommendation. The dataset is split into the training data and the test data by time. The test data are furthermore split into a public set and a private set for independent evaluations. The dataset is extremely sparse with only about two positive records (i.e., accept the recommendation) for each user. Besides, nearly 70% of users in the test data are never occurred in the training data.

Table 5.1 shows the statistics for both the synthetic dataset and the Tencent microblog dataset.

5.4.2 Setup and Metrics

We randomly remove 20% of dataset as the test data and the remaining 80% data as the training data for the synthetic data. We repeat the experiments five times and report the average result. For the Tencent microblog data, the dataset is already split into the training data and the test data. We further use the $1/5$ training data as the validation data to tune parameters. We conduct evaluations on the public test dataset. We extract 18 features from the data, including user, item, the number of tweets, the number of followers/followee numbers, etc. We treat all of the features as categorical features.

For the synthetic dataset, we use two metrics, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), to measure the prediction quality of different methods. MAE and RMSE are defined as follows:

$$\begin{aligned}\text{MAE} &= \frac{\sum_i |\hat{y}(i) - y_i|}{N}, && (5.27)\\ \text{RMSE} &= \sqrt{\frac{\sum_i (\hat{y}(i) - y_i)^2}{N}}, && (5.28)\end{aligned}$$

where N is the number of training instances.

For the Tencent microblog data, Mean Average Precision (MAP) is used as the metric:

$$\text{MAP@}k = \sum_{i=1}^{N} \text{ap@}k_i / N, \tag{5.29}$$

where N is the number of users, and ap@k is the average precision at k for the user:

$$\text{ap@}k = \sum_{i=1}^{k} P(i)/(\text{number of items clicked in m items}), \quad (5.30)$$

where $P(i)$ is the precision at cut-off i in the item list.

5.4.3 Performance Comparison

In our experiments, we compare the following methods:

- **PMF**: this method is well known in recommender systems and proposed in [112]. It only uses the user-item matrix for recommendations.
- **Context-aware FM**: this method is proposed in [106]. It is a strong baseline method introduced in Section 5.3.2.
- **GBFM**: this method is our newly proposed model, which is described in Algorithm 5.
- **GBFM-Opt**: this method is a variant of our GBFM, which is discussed in Section 5.3.3. When the training process stops after S steps , we can obtain S interacting features. We fully optimize these S interacting features.

For GBFM, we use 1-way feature linear model as the initialized prediction function. Grid search is applied to find the regularization parameter λ. We set it to 0.1 for the synthetic data and 0.8 for the Tencent microblog data. The latent dimension k is set to 5 and 10 for the synthetic data and the Tencent microblog data, respectively. We use square loss for the synthetic data and logit loss for the Tencent microblog data. The detailed comparison results are shown in Table 5.2 and Table 5.3.

From the tables, we can observe that:

Table 5.2: Results on the synthetic data in RMSE and MAE

Method	RMSE	MAE
PMF	1.9881	1.7650
FM	1.9216	1.6981
GBFM	1.8959	1.6354
GBFM-Opt	1.8611	1.5762

Table 5.3: Results on the Tencent microblog data in MAP

Method	MAP@1	MAP@3	MAP@5
PMF	22.88%	34.50%	37.95%
FM	24.36%	36.77%	40.32%
GBFM	24.62%	37.17%	40.90%
GBFM-Opt	24.66%	37.23%	40.98%

- Both our proposed GBFM and GBFM-Opt model achieve better performance on both the synthetic data and the Tencent microblog data in terms of all metrics compared with PMF and FM. On the synthetic data, FM gives 0.066 reduction over PMF in terms of RMSE, and GBFM further gives 0.026 reduction over FM. Since the synthetic data are generated from part of 2-way feature interactions, the results reveal that our proposed GBFM can learn "good" interacting features. While on the Tencent microblog data, FM improves 2.25% in terms of MAP@3 compared with PMF. GBFM can still be able to improve the performance by 0.4%. This result verifies our assumption that selecting "good" features is better than considering all of pairwise interacting features.

- It is not surprising that the performances of FM, GBFM and GBFM-Opt are much better than PMF. It reveals the importance of utilizing auxiliary information on context-aware recommendation. It is even more critical on the Tencent microblog data since most of users in the test data do not exist in the training data, which is "cold-start" users for PMF. But our models can deal with these "cold-start" users very well, since we can make use of the context features effectively.

- The performance of GBFM-Opt can illustrate whether the selected interacting features are useful for recommendations. We can observe that on both datasets, GBFM-Opt can achieve even better performance than GBFM. On the synthetic data, GBFM-Opt improves a lot (0.035) compared with GBFM in terms of RMSE, while on the Tencent microblog data GBFM-Opt is slightly better than GBFM. The results reveal that the features selected by our GBFM is quite useful compared with considering all the pairwise interactions. Recall the discussions we conducted in Section 5.3.3, compared with GBFM-Opt, GBFM loses the advantage of generalization, which may be the main reason why GBFM-Opt is better than GBFM. Compared with the synthetic data, the Tencent microblog data are much sparser thus it is not easy to benefit from generalization, which explains why GBFM-Opt only improves a little compared with GBFM on the Tencent microblog data.

5.5 Conclusion

In this chapter, we proposed a novel model called GBFM, which incorporated the interacting feature selection algorithm with Factorization Machines into a unified framework. Experiments on both synthetic and real datasets showed that our model can effectively select "good" interacting features and achieve better performance compared with other state-of-the-art methods.

There are several interesting directions worthy of consideration in the future: 1) we would like to explore how to find high order features; 2) we are interested to extend our GBFM with better high order feature selection algorithms; 3) it is also interesting to explore how to effectively deal with other features apart from categorical features.

□ **End of chapter.**

Chapter 6

Conclusion

In this chapter, we summarize the main contributions made in this thesis and discuss several potential directions.

6.1 Summary

Collaborative filtering techniques, which focus on the user-item rating matrix only, cannot provide accurate recommendations when data are very sparse. Fortunately, location information and other context information can be easily gathered due to the rapid development of mobile devices and ubiquitous Internet access, which can be employed to alleviate the data sparsity problem. In this thesis, we made contributions to solve problems in recommender systems, in which the data are very sparse while location and context information can help improve recommendation performance. To each of these problems, we identified the properties of the data and designed practical models and algorithms for them. Through our experiments, we demonstrated the merits of our algorithms compared with other state-of-the-art methods.

Specifically, in Chapter 3, we proposed a unified POI recommendation framework in LBSNs, which fused user preference, geographical influence and personalized ranking together to alleviate the data sparsity problem. In order to capture the geographical influence, we carefully studied the users' check-in data and proposed a novel Multi-center Gaussian Model (MGM). Then we proposed a framework to incorporate matrix factorization with MGM. In this way, we fused user preference and geographical influence. Besides, users are more interested in top 10 or 20 results in recommender systems, which makes personalized ranking important. To take personalized ranking into account, furthermore, we proposed two methods to incorporate MGM with BPR in two different approaches. Our methods outperformed other state-of-the-art methods since all of them do not consider the geographical influence, user preference and personalized ranking together. The experimental results also illustrated that our models can relieve the data sparsity problem compared with traditional matrix factorization models.

In Chapter 4, we considered another problem in LBSNs with sparse data, i.e., successive POI recommendation, which provided recommendations for users in the near future. The general POI recommendation problem in Chapter 3 is time unaware. We provide recommendations to users based on the whole check-in histories, which does not consider the difference between check-ins at different time. On the other hand, in successive POI recommendation, the recommendation results are heavily relied on the previous check-ins. In order to consider the temporal effect, specifically, we developed two novel matrix factorization models, i.e., FPMC-LR and FPMC-LTT, based on the two prominent properties in the check-in sequence: personalized Markov chain and region localization. Both FPMC-LR and FPMC-LTT embedded personalized Markov chain and region localization with the user preference. Since users would like to check in POIs near to previous check-ins, considering region localization can alleviate the data sparsity problem by filtering out far-away POIs. FPMC-LTT modeled personalized Markov chain on topic-wise level, while FPMC-LR modeled on location-wise level. FPMC-LTT outperformed FPMC-LR since the location transition data in location-wise level are too sparse in LBSNs while modeling on topic-wise level can alleviate the sparsity problem. As both our models considered the properties of the LBSNs data and embedded them in the models, our models performed much better than other state-of-the-art methods.

Finally, in Chapter 5, we explored to employ context information effectively in recommender systems with sparse data. Traditional matrix factorization methods focus on the user-item matrix only, while context information can be employed to improve recommendation performance and overcome the data sparsity problem. Thus, context-aware methods are proposed to consider context features. However, most of context-aware methods model pairwise interactions between all features. In practice, not all the pairwise feature interactions are useful. Thus, selecting “good” interacting features effectively is very challenging. To address this challenge, we proposed a novel Gradient Boosting Factorization Machines (GBFM) model to incorporate the feature selection algorithm with Factorization Machines into a unified framework. Compared with other state-of-the-art methods, our model demonstrated its effectiveness since our model can effectively select useful interacting features, especially in recommendation systems with tens of context features. Besides, since our model can select effective context features, our model can relieve the cold-start problem as well, which cannot be solved by traditional recommendation methods such as PMF.

6.2 Future Work

Although a substantial number of promising achievements on techniques and their applications have been presented in this thesis, there are still numerous open issues that need to be further explored in the future.

Firstly, we plan to investigate how to design more effective sampling strategies for negative samples in POI recommendation. POI recommendation belongs to one class collaborative filtering (OCCF) and most OCCF methods use simple sampling strategies for negative samples such as uniform sampling. There is no sampling strategy specially designed for POI recommendation in which geographical influence needs to be considered. This direction could be very interesting and can alleviate the data sparsity problem in POI recommendation.

Secondly, we would like to explore how to use techniques from topic modeling, such as LDA, to utilize the content information in LBSNs to improve both POI recommendation and successive POI recommendation. We have the users' comments for POIs. Applying topic models on these comments, we can obtain the latent topic space for POIs even when a POI receives few check-ins. Combining matrix factorization methods and topic modeling in POI recommendation can improve recommendation performance as well as alleviate the cold-start problem.

Thirdly, we plan to investigate how to explore high order Markov chain to improve recommendation performance in successive POI recommendation. In our method, for simplicity, we only consider the first order Markov chain. High order Markov chain would make the model more complicated but can potentially improve the performance. It is interesting to explore methods to balance the two factors.

Fourthly, for context-aware recommendation, we would like to explore other methods for finding effective high-order interacting features. In our method introduced in Chapter 5, we use a greedy algorithm to find 2-order interacting features, while in practice, some higher order feature is also useful. However, the space of higher order features grows exponentially with the number of orders, which makes it very challenging to figure out good ways to find effective high order features.

□ **End of chapter.**

List of Publications

[1] **Chen Cheng**, Haiqin Yang, Irwin King and Michael R. Lyu. A Unified Point-of-interest Recommendation Framework in Location-based Social Networks. Manuscript submitted to ACM Transactions on Intelligent Systems and Technology (TIST), 2015.

[2] **Chen Cheng**, Fen Xia, Tong Zhang, Irwin King and Michael R. Lyu. Gradient Boosting Factorization Machines. In *Proceedings of the 8th ACM Conference on Recommender Systems (Recsys)*, 2014.

[3] **Chen Cheng**, Haiqin Yang, Michael R. Lyu and Irwin King. Where You Like to Go Next: Successive Point-of-interest Recommendation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, 2013.

[4] **Chen Cheng**, Haiqin Yang, Irwin King and Michael 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 (AAAI)*, 2012.

Bibliography

[1] Fabian Abel, Qi Gao, Geert-Jan Houben, and Ke Tao. Twitter-based user modeling for news recommendations. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2962–2966. AAAI Press, 2013.

[2] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. pages 103–145, 2005.

[3] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[4] Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *KDD*, pages 19–28, 2009.

[5] Monireh Amini and Mahdi Nasiri. New approach in hybrid movie recommender systems using collaborative filtering and content-based filtering methods. *International Journal of Computer Science and Information Security*, 12(10):1, 2014.

[6] Tasos Anastasakos, Dustin Hillard, Sanjay Kshetramade, and Hema Raghavan. A collaborative filtering approach to ad recommendation using the query-ad click graph. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1927–1930. ACM, 2009.

[7] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[8] Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 143–152. ACM, 2012.

[9] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. Matrix factorization techniques for context aware recommendation. In *RecSys*, pages 301–304, 2011.

[10] Punam Bedi, Harmeet Kaur, and Sudeep Marwaha. Trust based recommender system for semantic web. In *IJCAI*, volume 7, pages 2677–2682, 2007.

[11] Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[12] Robert M. Bell and Yehuda Koren. Lessons from the Netflix prize challenge. *SIGKDD Explorations*, 9(2):75–79, 2007.

[13] Robert M. Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, pages 43–52, 2007.

[14] Robert M. Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD*, pages 95–104, 2007.

[15] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[16] Dmitry Bogdanov, MartíN Haro, Ferdinand Fuhrmann, Anna Xambó, Emilia Gómez, and Perfecto Herrera. Semantic audio

content-based music recommendation and visualization based on user preference examples. *Information Processing & Management*, 49(1):13–33, 2013.

[17] John S. Breese, David Heckerman, and Carl Myers Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.

[18] Chris Buckley and Ellen M. Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32. ACM, 2004.

[19] Christopher J.C. Burges. From RankNet to LambdaRank to LambdaMART: An overview. *Learning*, 11:23–581, 2010.

[20] Robin D. Burke, Kristian J. Hammond, Vladimir Kulyukin, Steven L. Lytinen, Noriko Tomuro, and Scott Schoenberg. Question answering from frequently asked question files: Experiences with the FAQ FINDER system. *AI magazine*, 18(2):57, 1997.

[21] John F. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR*, pages 238–245, 2002.

[22] Pedro Cano, Markus Koppenberger, and Nicolas Wack. Content-based music audio recommendation. In *Proceedings of the 13th annual ACM international conference on Multimedia*, pages 211–212. ACM, 2005.

[23] Xin Cao, Gao Cong, Bin Cui, Christian S. Jensen, and Quan Yuan. Approaches to exploring category information for question retrieval in community question-answer archives. *ACM Transactions on Information Systems (TOIS)*, 30(2):7, 2012.

[24] Xin Cao, Gao Cong, and Christian S. Jensen. Mining significant semantic locations from GPS data. *Proceedings of the VLDB Endowment*, 3(1-2):1009–1020, 2010.

[25] Chih-Ming Chen, Ming-Feng Tsai, Jen-Yu Liu, and Yi-Hsuan Yang. Using emotional context from article for contextual music recommendation. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 649–652. ACM, 2013.

[26] Tianqi Chen, Hang Li, Qiang Yang, and Yong Yu. General functional matrix factorization using gradient boosting. In *Proceedings of The 30th International Conference on Machine Learning*, pages 436–444, 2013.

[27] Tianqi Chen, Linpeng Tang, Qin Liu, Diyi Yang, Saining Xie, Xuezhi Cao, Chunyang Wu, Enpeng Yao, Zhengyang Liu, and Zhansheng Jiang. Combining factorization model and additive forest for collaborative followee recommendation. *KDD CUP*, 2012.

[28] Ye Chen, Michael Kapralov, Dmitry Pavlov, and John Canny. Factor modeling for advertisement targeting. In *NIPS*, pages 324–332, 2009.

[29] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI*, 2012.

[30] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 2605–2611. AAAI Press, 2013.

[31] Zhiyuan Cheng, James Caverlee, Krishna Yeswanth Kamath, and Kyumin Lee. Toward traffic-driven location-based web search. In *CIKM*, pages 805–814, 2011.

[32] Zhiyuan Cheng, James Caverlee, Kyumin Lee, and Daniel Z. Sui. Exploring millions of footprints in location sharing services. In *ICWSM*, 2011.

[33] Eunjoon Cho, Seth A. Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *KDD*, pages 1082–1090, 2011.

[34] Sang-Min Choi, Sang-Ki Ko, and Yo-Sub Han. A movie recommendation algorithm based on genre correlations. *Expert Systems with Applications*, 39(9):8079–8085, 2012.

[35] Kushal S. Dave and Vasudeva Varma. Learning the click-through rate for rare/new ads from similar ads. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 897–898. ACM, 2010.

[36] Mukund Deshpande and George Karypis. Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[37] Jerome Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[38] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Special invited paper. additive logistic regression: A statistical view of boosting. *Annals of statistics*, pages 337–374, 2000.

[39] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Exploring temporal effects for location recommendation on location-

based social networks. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 93–100. ACM, 2013.

[40] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Content-aware point of interest recommendation on location-based social networks. AAAI, 2015.

[41] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[42] Ziyu Guan, Jiajun Bu, Qiaozhu Mei, Chun Chen, and Can Wang. Personalized tag recommendation using graph-based ranking on multi-type interrelated objects. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 540–547. ACM, 2009.

[43] Venkat N. Gudivada, Vijay V. Raghavan, William I. Grosky, and Rajesh Kasanagottu. Information retrieval on the world wide web. *IEEE Internet Computing*, 1(5):58–68, 1997.

[44] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*, volume 2. Springer, 2009.

[45] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237, 1999.

[46] Jose M. Hernandez-lobato, Neil Houlsby, and Zoubin Ghahramani. Probabilistic matrix factorization with non-random missing data. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1512–1520, 2014.

[47] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual commu-

nity of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[48] Thomas Hofmann. Collaborative filtering via Gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 259–266. ACM, 2003.

[49] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

[50] Tzvetan Horozov, Nitya Narasimhan, and Venu Vasudevan. Using location for personalized POI recommendations in mobile environments. In *Proceedings of the International Symposium on Applications on Internet*, SAINT '06, pages 124–129, Washington, DC, USA, 2006. IEEE Computer Society.

[51] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. IEEE, 2008.

[52] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 135–142. ACM, 2010.

[53] Meng Jiang, Peng Cui, Rui Liu, Qiang Yang, Fei Wang, Wenwu Zhu, and Shiqiang Yang. Social contextual recommendation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 45–54. ACM, 2012.

[54] Rong Jin, Joyce Y. Chai, and Luo Si. An automatic weighting scheme for collaborative filtering. In *SIGIR*, pages 337–344, 2004.

[55] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.

[56] Eui-young Kang, Hanil Kim, and Jungwon Cho. Personalization method for tourist point of interest (POI) recommendation. In Bogdan Gabrys, Robert Howlett, and Lakhmi Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, volume 4251 of *Lecture Notes in Computer Science*, chapter 48, pages 392–400. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2006.

[57] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys*, pages 79–86, 2010.

[58] Arnd Kohrs and Bernard Merialdo. Clustering for collaborative filtering applications. In *In Computational Intelligence for Modelling, Control & Automation. IOS*. Citeseer, 1999.

[59] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[60] Michal Kompan and Mária Bieliková. Content-based news recommendation. In *E-commerce and web technologies*, pages 61–72. Springer, 2010.

[61] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434, 2008.

[62] Yehuda Koren. The BellKor solution to the Netflix grand prize. *Netflix prize documentation*, 81, 2009.

[63] Yehuda Koren. Collaborative filtering with temporal dynamics. In *KDD*, pages 447–456, 2009.

[64] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[65] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation. In *Proceedings of the third ACM conference on Recommender systems*, pages 61–68. ACM, 2009.

[66] Miklós Kurucz, András A. Benczúr, Tamás Kiss, István Nagy, Adrienn Szabó, and Balázs Torma. KDD cup 2007 task 1 winner report. *SIGKDD Explorations*, 9(2):53–56, 2007.

[67] Helge Langseth and Thomas Dyhre Nielsen. A latent model for collaborative filtering. *International Journal of Approximate Reasoning*, 53(4):447–466, 2012.

[68] Neil D. Lawrence and Raquel Urtasun. Non-linear matrix factorization with Gaussian processes. In *ICML*, page 76, 2009.

[69] Kenneth Wai-Ting Leung, Dik Lun Lee, and Wang-Chien Lee. CLR: a collaborative location recommendation framework based on co-clustering. In *SIGIR*, pages 305–314, 2011.

[70] Xin Li, Lei Guo, and Yihong Eric Zhao. Tag-based social interest discovery. In *Proceedings of the 17th international conference on World Wide Web*, pages 675–684. ACM, 2008.

[71] Yanen Li, Jia Hu, ChengXiang Zhai, and Ye Chen. Improving one-class collaborative filtering by incorporating rich user information. In *Proceedings of the 19th ACM international con-*

ference on Information and knowledge management, pages 959–968. ACM, 2010.

[72] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong 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*, pages 831–840. ACM, 2014.

[73] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative fitering. In *IEEE Internet Computing*, pages 76–80, 2003.

[74] Guang Ling, Haiqin Yang, Michael R. Lyu, and Irwin King. Response aware model-based collaborative filtering. *arXiv preprint arXiv:1210.4869*, 2012.

[75] Bin Liu, Yanjie Fu, Zijun Yao, and Hui 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.

[76] Bin Liu and Hui Xiong. Point-of-interest recommendation in location based social networks with topic and location awareness. In *SDM*, volume 13, pages 396–404, 2013.

[77] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010.

[78] Nathan N. Liu and Qiang Yang. EigenRank: a ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on*

Research and development in information retrieval, pages 83–90. ACM, 2008.

[79] Nathan N. Liu, Min Zhao, and Qiang Yang. Probabilistic latent preference analysis for collaborative filtering. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 759–766. ACM, 2009.

[80] Tie-Yan Liu. *Learning to rank for information retrieval*. Springer Science & Business Media, 2011.

[81] Xin Liu and Karl Aberer. SoCo: a social network aided context-aware recommender system. In *WWW*, pages 781–802, 2013.

[82] Eric Hsueh-Chan Lu, Vincent S. Tseng, and Philip S. Yu. Mining cluster-based temporal mobile sequential patterns in location-based service environments. *IEEE Trans. Knowl. Data Eng.*, 23(6):914–927, 2011.

[83] Hao Ma, Irwin King, and Michael R. Lyu. Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 39–46. ACM, 2007.

[84] Hao Ma, Irwin King, and Michael R. Lyu. Learning to recommend with social trust ensemble. In *SIGIR*, pages 203–210, 2009.

[85] Hao Ma, Irwin King, and Michael R. Lyu. Learning to recommend with explicit and implicit social relations. *ACM TIST*, 2(3):29, 2011.

[86] Hao Ma, Chao Liu, Irwin King, and Michael R. Lyu. Probabilistic factor models for web site recommendation. In *Proceedings of the 34th international ACM SIGIR conference on*

Research and development in Information, pages 265–274. ACM, 2011.

[87] Hao Ma, Michael R. Lyu, and Irwin King. Diversifying query suggestion results. In *AAAI*, volume 10, 2010.

[88] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. SoRec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940. ACM, 2008.

[89] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 287–296. ACM, 2011.

[90] Hao Ma, Tom Chao Zhou, Michael R. Lyu, and Irwin King. Improving recommender systems by incorporating social contextual information. *ACM Transactions on Information Systems (TOIS)*, 29(2):9, 2011.

[91] Pattie Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40, 1994.

[92] Benjamin M. Marlin and Richard S. Zemel. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*, pages 5–12. ACM, 2009.

[93] Benjamin M. Marlin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. Collaborative filtering and the missing at random assumption. *arXiv preprint arXiv:1206.5267*, 2012.

[94] Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil. An empirical study of geographic user activity patterns in foursquare. *ICWSM'11*, 2011.

[95] Rong Pan and Martin Scholz. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In *KDD*, pages 667–676, 2009.

[96] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *ICDM*, pages 502–511, 2008.

[97] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *RecSys*, pages 265–268, 2009.

[98] Ulrich Paquet and Noam Koenigstein. One-class collaborative filtering with random graphs. In *Proceedings of the 22nd international conference on World Wide Web*, pages 999–1008. International World Wide Web Conferences Steering Committee, 2013.

[99] Michael J. Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[100] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 473–480. Morgan Kaufmann Publishers Inc., 2000.

[101] Steffen Rendle. Factorization machines. In *ICDM*, pages 995–1000, 2010.

[102] Steffen Rendle. Social network and click-through prediction with factorization machines. In *KDD-Cup Workshop*, 2012.

[103] Steffen Rendle. Scaling factorization machines to relational data. *PVLDB*, 6(5):337–348, 2013.

[104] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[105] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. In *WWW*, pages 811–820, 2010.

[106] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644, 2011.

[107] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90, 2010.

[108] Jason D. M. Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719. ACM, 2005.

[109] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[110] Adam Sadilek, Henry Kautz, and Jeffrey P. Bigham. Finding your friends and following them to where you are. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 723–732. ACM, 2012.

[111] Adam Sadilek and John Krumm. Far out: Predicting long-term human mobility. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[112] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, 2007.

[113] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *ICML*, pages 880–887, 2008.

[114] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[115] Gerard Salton. Automatic text processing: the analysis, transformation and retrieval of information by computer, 1989.

[116] Robert Sanders. The pareto principle: its use and abuse. *Journal of Services Marketing*, 1(2):37–40, 1987.

[117] Jitao Sang, Tao Mei, Jian-Tao Sun, Changsheng Xu, and Shipeng Li. Probabilistic sequential POIs recommendation via check-in data. In *SIGSPATIAL/GIS*, pages 402–405, 2012.

[118] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295, 2001.

[119] Salvatore Scellato, Mirco Musolesi, Cecilia Mascolo, Vito Latora, and Andrew T Campbell. NextPlace: a spatio-temporal prediction framework for pervasive systems. In *Pervasive Computing*, pages 152–169. Springer, 2011.

[120] Salvatore Scellato, Anastasios Noulas, Renaud Lambiotte, and Cecilia Mascolo. Socio-spatial properties of online

location-based social networks. *Proceedings of ICWSM*, 11, 2011.

[121] Salvatore Scellato, Anastasios Noulas, and Cecilia Mascolo. Exploiting place features in link prediction on location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1046–1054. ACM, 2011.

[122] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[123] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Alan Hanjalic, and Nuria Oliver. TFMAP: Optimizing map for top-n context-aware recommendation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 155–164. ACM, 2012.

[124] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 139–146. ACM, 2012.

[125] Yue Shi, Martha Larson, and Alan Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 269–272. ACM, 2010.

[126] Luo Si and Rong Jin. Flexible mixture model for collaborative filtering. In *In Proc. of ICML*, 2003.

[127] Börkur Sigurbjörnsson and Roelof Van Zwol. Flickr tag recommendation based on collective knowledge. In *Proceedings of the 17th international conference on World Wide Web*, pages 327–336. ACM, 2008.

[128] Nathan Srebro and Tommi Jaakkola. Weighted low-rank approximations. In *ICML*, pages 720–727, 2003.

[129] Nathan Srebro, Jason D. M. Rennie, and Tommi Jaakkola. Maximum-Margin matrix factorization. In *NIPS*, 2004.

[130] David H. Stern, Ralf Herbrich, and Thore Graepel. Matchbox: large scale online bayesian recommendations. In *WWW*, pages 111–120, 2009.

[131] Ja-Hwung Su, Hsin-Ho Yeh, Philip S. Yu, and Vincent S. Tseng. Music recommendation using content and context information mining. *Intelligent Systems, IEEE*, 25(1):16–26, 2010.

[132] Yukihiro Tagami, Shingo Ono, Koji Yamamoto, Koji Tsukamoto, and Akira Tajima. CTR prediction for contextual advertising: learning-to-rank approach. In *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, page 4. ACM, 2013.

[133] Maksims Volkovs and Richard S. Zemel. Collaborative ranking with 17 parameters. In *Advances in Neural Information Processing Systems*, pages 2294–2302, 2012.

[134] Markus Weimer, Alexandros Karatzoglou, Quoc Viet Le, and Alex Smola. Maximum margin matrix factorization for collaborative ranking. *Advances in neural information processing systems*, 2007.

[135] Jason Weston, Chong Wang, Ron J. Weiss, and Adam Berenzweig. Latent collaborative retrieval. In *ICML*, 2012.

[136] Xiangye Xiao, Yu Zheng, Xing Xie, and Qiong Luo. Inferring social ties between users with human location history.

[137] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, pages 211–222, 2010.

[138] Jun Xu and Hang Li. AdaRank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

[139] Haiqin Yang, Shouyuan Chen, Michael R. Lyu, and Irwin King. Location-based topic evolution. In *Proceedings of the 1st international workshop on Mobile location-based service*, 2011.

[140] Haiqin Yang, Irwin King, and Michael R. Lyu. *Sparse Learning Under Regularization Framework*. LAP Lambert Academic Publishing, first edition, April 2011.

[141] Haiqin Yang, Zenglin Xu, Jieping Ye, Irwin King, and Michael R. Lyu. Efficient sparse generalized multiple kernel learning. *IEEE Transactions on Neural Networks*, 22(3):433–446, March 2011.

[142] Mao Ye, Peifeng Yin, and Wang-Chien Lee. Location recommendation for location-based social networks. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 458–461. ACM, 2010.

[143] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik Lun Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *SIGIR*, pages 325–334, 2011.

[144] Jing Yuan, Yu Zheng, and Xing Xie. Discovering regions of different functions in a city using human mobility and POIs. In *KDD*, pages 186–194, 2012.

[145] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. Time-aware point-of-interest recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 363–372. ACM, 2013.

[146] Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. LORE: Exploiting sequential influence for location recommendations. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas, TX*, 2014.

[147] Yi Zhang and Jonathan Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 47–54. ACM, 2007.

[148] Vincent W. Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In *Proceedings of the 24rd AAAI Conference on Artificial Intelligence*, 2010.

[149] Vincent Wenchen Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative location and activity recommendations with GPS history data. In *WWW*, pages 1029–1038, 2010.

[150] Yu Zheng and Xing Xie. Learning travel recommendations from user-generated GPS traces. *ACM TIST*, 2(1):2, 2011.

[151] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. *TWEB*, 5(1):5, 2011.

[152] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from GPS trajectories. In *WWW*, pages 791–800, 2009.

[153] ErHeng Zhong, Wei Fan, and Qiang Yang. Contextual collaborative filtering via hierarchical matrix factorization. In *SDM*, pages 744–755, 2012.

[154] Tom Chao Zhou, Hao Ma, Irwin King, and Michael R. Lyu. TagRec: Leveraging tagging wisdom for recommendation. In *Proceedings of CSE*, pages 194–199, 2009.

[155] Jianke Zhu, Hao Ma, Chun Chen, and Jiajun Bu. Social recommendation using low-rank semidefinite program. In *AAAI*, pages 158–163, 2011.