Improving Recommender Systems with Rich Side Information

by

Chenyi Zhang

B.Sc. Zhejiang University, 2010

Dissertation Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy

in the
School of Computing Science
Faculty of Applied Science

SIMON FRASER UNIVERSITY

Summer 2015

Approval

Name:	Chenyi Zhang
Degree:	Doctor of Philosophy (Computer Science)
Title:	*Improving Recommender Systems with Rich Side Information*
Examining Committee:	**Dr. Ze-Nian Li** (chair) Professor, Computing Science Simon Fraser University

Dr. Ke Wang
Senior Supervisor
Professor, Computing Science
Simon Fraser University _____

Dr. Jianling Sun
Supervisor
Professor, Computer Science
Zhejiang University _____

Dr. Qianping Gu
Internal Examiner
Professor, Computing Science
Simon Fraser University _____

Dr. Huan Liu
External Examiner
Professor, Computer Science and Engineering
Arizona State University _____

Date Defended: 4 Aug 2015

Abstract

Recommender systems have become extremely popular in recent years since they can provide personalized information to user from a large amount of data, which is typically noisy and hard to exploit. Traditional approaches mainly leverage the user-item rating matrix for recommendation. Beyond the rating matrix, however, there exists rich side information in recommender systems, which is a good source to improve the performance of rating prediction. In this thesis, we studied three types of side information (i.e., content, temporal, spatial), pointed out some open issues that are unsolved by the existing models and proposed our solutions in these areas.

We incorporate side information with some domain knowledge to improve the recommender systems. In recommendation with content information, we proposed a feature centric model to analyze the feature-level preferences instead of the item-level preferences, thus, make prediction according to feature-level preferences. We further proposed a recommendation by blending content and attributes in heterogeneous networks. In recommendation with temporal information, we proposed temporal matrix factorization to model the user's interest shift over time; such changes are essential for developing accurate recommender systems. In recommendation with spatial information, we proposed a cross-region collaborative filtering method to deal with the POI (Point of Interest) recommendation when the user travels to a new place; in this model, the long-term and short-term preferences are considered respectively. All these models are evaluated in real life data sets with the state-of-the-art methods.

Keywords: recommender systems, side information, latent factor model

Acknowledgements

I would like to thank my senior supervisor Dr. Ke Wang for his invaluable guidance and support. I also want to thank my supervisor Dr. Jianling Sun for his insightful commentary and valuable input.

I would also like to express my gratefulness to all other students in our group for their kind help and encouragement. They are Weipeng Lin, Hongwei Liang, Chao Han, Aungon Nag Radon, Zhilin Zhang, Yue Wang, Ryan McBride, Yongmin Yan and Peng Wang.

Finally, I want to thank my family for their continuous support and forever love.

Table of Contents

Approval ii

Abstract iii

Acknowledgements iv

Table of Contents v

List of Tables ix

List of Figures x

1 Introduction 1

1.1 Background 2

1.1.1 Taxonomy 2

1.1.2 Latent Factor Model 3

1.1.3 Topic Model 5

1.2 Recommendation with Side Information 5

2 Related Work: Beyond the Rating Matrix 9

2.1 Recommendation with Content Information 9

2.1.1 Similarity-based Approaches 10

2.1.2 Latent Factor Models meet Topic Models 11

2.1.3 Latent Factor Models with Regularization 12

2.1.4 Factorization Machines 13

2.2 Recommendation with Temporal Information 14

2.2.1 Time Partition or Decay Model 15

2.2.2 Temporal CF with Adaptive Neighbourhoods 15

2.2.3 Latent Factor Models with Temporal Bias 16

2.2.4 Kalman Filtering . . . 17
2.2.5 Tensor Factorization . . . 17
2.3 Recommendation with Spatial Information . . . 18
2.3.1 Location as Items . . . 18
2.3.2 Generative Models with Location . . . 19
2.3.3 Latent Factor Models with Location . . . 20

3 Feature-centric Recommendation 23
3.1 Motivation . . . 23
3.1.1 Our Approach . . . 24
3.1.2 Comparison with Related Work . . . 25
3.2 Feature-Centric Recommendation . . . 26
3.2.1 Extracting User-Feature Rating Matrix . . . 27
3.2.2 Predicting Item Ratings by Heuristic . . . 28
3.2.3 Predicting Item Ratings through Regression . . . 29
3.3 Experimental Evaluation . . . 30
3.3.1 Data Sets . . . 31
3.3.2 Evaluated Methods . . . 31
3.3.3 Evaluation Metrics . . . 32
3.3.4 Experimental Results . . . 33
3.4 Summary . . . 36

4 Recommendation by Blending Content and Attributes 38
4.1 Motivation . . . 38
4.1.1 Discussions . . . 40
4.1.2 Comparison with Related Work . . . 41
4.2 Model . . . 41
4.2.1 Item Information Processing . . . 43
4.2.2 First Cut Solutions . . . 43
4.2.3 Content + Attributes Model . . . 44
4.3 Experimental Evaluation . . . 46
4.3.1 Data Sets . . . 46
4.3.2 Evaluated Methods . . . 46
4.3.3 Evaluation Metrics . . . 47

4.3.4 Experimental Results . . . 48
4.4 Summary . . . 49

5 Temporal Matrix Factorization **51**
5.1 Motivation . . . 51
5.1.1 Contributions . . . 52
5.1.2 Comparison with Related Work . . . 53
5.2 Temporal Probabilistic Matrix Factorization . . . 54
5.2.1 Introducing Transition Matrix . . . 54
5.2.2 Modeling . . . 55
5.2.3 Inference . . . 57
5.3 The Fully Bayesian Model (BTMF) . . . 57
5.3.1 Modeling . . . 58
5.3.2 Inference . . . 59
5.4 Experimental Evaluation . . . 62
5.4.1 Evaluated Methods . . . 62
5.4.2 Data Sets . . . 62
5.4.3 Evaluation Metrics . . . 63
5.4.4 Experimental Results . . . 63
5.5 Summary . . . 65

6 Cross-region Collaborative Filtering **70**
6.1 Motivation . . . 71
6.1.1 Contributions . . . 73
6.1.2 Comparison with Related Work . . . 75
6.2 Methodology . . . 76
6.2.1 Preliminaries . . . 76
6.2.2 Predicting Content Preferences . . . 77
6.2.3 Predicting Location Preferences . . . 78
6.2.4 Recommending POIs . . . 80
6.2.5 Discussion . . . 80
6.3 Experimental Evaluation . . . 82
6.3.1 Data Sets . . . 82
6.3.2 Evaluated Methods . . . 82

6.3.3 Evaluation Metrics 84
6.3.4 Experimental Results 85
6.3.5 "New city" Testing 90
6.4 Summary 91

7 Conclusion **93**

Bibliography **96**

List of Tables

Table 3.1 Example for feature-level preferences 24
Table 3.2 Statistics of data sets 30
Table 3.3 $RMSE$ and MAE of four data sets 33
Table 3.4 Paired t-Test(2-tail) of FCR-a and FCR-r 35

Table 4.1 RMSE and coverage results of different methods. Lower values on RMSE and higher values on coverage are better. 48

Table 5.1 Statistics of data sets. 62
Table 5.2 RMSE (mean±standard error) of different models. $D = 20$. The best performer is in **boldface** and the second best performer is in *italic*. . 65

Table 6.1 MRR (mean ± standard error). $K = 10$ 85
Table 6.2 Performances of CRCF for different seed sizes m 88
Table 6.3 Runtime of learning the location recommender of CRCF (min) vs K . 89

List of Figures

Figure 2.1 Content information in recommender systems 10

Figure 2.2 Example from Rendle's work [52] for representing a recommendation problem. Every row represents a feature vector $\mathbf{x}_i$ with its corresponding target y_i 14

Figure 3.1 The feature-centric recommendation approach 26

Figure 3.2 Construct the user-feature rating matrix (right) from the original ratings (left) 27

Figure 3.3 Recall@k of four data sets. Vary k (x-axis) 35

Figure 3.4 Feature selection for FCR-r on Lastfm (left) and MovieLens (right). y-axis represents RMSE and x-axis represents the percentage of the features with lower correlations removed 36

Figure 3.5 Representative features on Lastfm 37

Figure 4.1 Heterogeneous recommendation scenario: each item has content and attribute information. Each user marks the likes($\surd$) and dislikes($\times$) for some items and the rest are unknown(?) in the rating matrix. $\surd$ in the information table means the item has this attribute 39

Figure 4.2 Rating distributions of the data set. The left subfigure shows the number of ratings by each user and the right subfigures shows the number of ratings on each item (logistic scaled), all descent sorted. 46

Figure 4.3 Recall performance of different methods. The left top subfigure shows overall performance of different methods while varying k. The rest subfigures exhibit the individual level of performance each method while fixing $k = 200$. 50

Figure 5.1 Evolution of user i's preferences 52

Figure 5.2 Graphical representations of TMF (left) and BTMF (right), with parameters and hyperparameters of time window t and $t+1$ shown only 56
Figure 5.3 Recall of different models. Every two rows refer to a data set and each column refers to a testing window. For each data set, the first row observes the effect of varying k with fixed $D = 20$ while the second row observes the effect of varying D with fixed $k = 300$. Higher values are better. 67
Figure 5.4 Recall of different models (cont.). Every two rows refer to a data set and each column refers to a testing window. For each data set, the first row observes the effect of varying k with fixed $D = 20$ while the second row observes the effect of varying D with fixed $k = 300$. Higher values are better. 68
Figure 5.5 Recall of different models (cont.). Every two rows refer to a data set and each column refers to a testing window. For each data set, the first row observes the effect of varying k with fixed $D = 20$ while the second row observes the effect of varying D with fixed $k = 300$. Higher values are better. 69

Figure 6.1 Locations of the Yelp data partitioned into regions 72
Figure 6.2 Different matrices for 3 users (A,B,C), 3 POIs (I, II, III), 4 features (f1,f2,f3,f4), and 2 regions (r1,r2). POI I has features f1, f2 and is located in region r1, POI II has features f1, f3 and is located in region r2, POI III has features f2, f4 and is located in region r1. Users may select a subset of the features when rating a POI. 74
Figure 6.3 Combining the predicted content rating and location rating to compute the top-n POIs in CRCF 81
Figure 6.4 Recall@n vs n (the x-axis). $K = 10$ 87
Figure 6.5 Performance vs D (the x-axis). $K = 10$ 89
Figure 6.6 Performances for different numbers of regions K 90
Figure 6.7 Recall@n vs n (the x-axis) in "new city" testing. $K = 10$ 91

Chapter 1

Introduction

The recommender systems have attracted a lot of attentions during the past few years, especially as the social networking services emerge and large amount of data become available. The goal of recommender systems is to predict the rating or preference that the user will give to an item. Recommender systems are quite popular recently because they help users to filter useful information from large amount of complex or redundant data efficiently. For example, Amazon[1] recommends interesting products to users when users purchase related products, and Youtube[2] recommends videos the users may be interested in. Other items such as movies, musics, books and travel packages are commonly recommended due to different applications, and users also could be recommended to other users in social networking services like Twitter[3] and Facebook[4].

Recommender systems are distinguished from search engines in that the user can get personalized feedback without building a query and requesting the results. This is often realized by estimating user's preferences on items and recommending those items featuring the maximal predicted preference. The prerequisite for determining such recommendations is the original rating matrix, i.e., the historic activities that the user rate items, as well as some rich side information. The side information refers to the additional information beyond the rating matrix, e.g., the time when the user performs the rating, or the description of the items. For example, there are thousands of movies in Youtube and it is impossible for a user to watch all the movies or reviews to conclude the best choice. At this time, a proper recommendation based on the user's rating behaviors (like/dislike) and the reviews

[1] http://www.amazon.com/
[2] http://www.youtube.com/
[3] http://www.twitter.com/
[4] http://www.facebook.com/

(side information) reduces the time on searching and increases the user satisfaction. In this thesis, the main research focus is on how to exploit the side information to improve the recommendation performance.

One of the key events that energized research in recommender systems was the Netflix[5] prize. From 2006 to 2009, Netflix sponsored a competition, offering a grand prize of 1 million dollars to the team that could take an offered data set of over 100 million movie ratings and return recommendations that were 10% more accurate than those offered by the company's existing recommender system. This competition energized the search for new and more accurate algorithms. Gradually, recommender system became an independent research area from informational retrieval and data mining, with the milestone that the ACM Conference on Recommender Systems (RecSys) was founded in 2007.

Next, we introduce the background knowledge of recommender systems and rich side information in the following sections.

1.1 Background

The recommender systems in the simplest form have two entities: *user* and *item*, where the item could be movie, product or music etc. Suppose there are I users and J items, and each user i could give a *rating* r_{ij} to an item j, which indicates the user's personalized preference to the item. This forms an $I \times J$ rating matrix R and such ratings could be explicitly presented, e.g., the rating ranges from 1 to 5, or implicitly presented, e.g., the user purchases the item.

1.1.1 Taxonomy

Three basic approaches in recommender systems are content-based filtering, collaborative filtering and hybrid approaches:

Content-based filtering: In this approach, keywords or features are used to describe the items and a user profile is built using the past item ratings to summarize the types of items this user likes. This approach follows the design principle that **liking a feature in the past leads to liking the feature in future**. A drawback is that if a user A has not liked any feature of an item x in the past, x will not be recommended to the user, even though many users with the same profile as A like x.

[5] `https://www.netflix.com/`

Some recommender systems leverage content-based filtering to provide movie recommendations, a few such examples include Rotten Tomatoes[6] and Internet Movie Database (IMDb)[7].

Collaborative filtering: This approach addresses the above problem by collaborative learning: if a user A liked some items that were also liked by user B, A is likely to share the same preference with B on another item. The design principle of this approach is that **liking same items (as other users) leads to liking more same items**.

There are also a large number of recommender systems leveraging collaborative filtering, e.g., Last.fm[8] recommends music based on a comparison of the listening habits of similar users. Facebook, Twitter, LinkedIn[9] and other social networks use collaborative filtering to recommend new friends, groups, and other social connections [53].

Hybrid approaches combine content-based filtering and collaborative filtering. For example, the collaboration via content approach in Pazzani et al.'s work [50] is based on collaborative techniques but also maintains the content-based profile for each user. These content-based profiles, and not the commonly rated items, are then used to calculate the similarity between two users.

Another taxonomy divides recommendation methods into memory-based and model-based method categories:

Memory-based methods usually adopt similarity metrics to obtain the distance between pairwise users or pairwise items according to the historic rating data, and then make rating predictions based on such similarity metrics. The memory-based methods are typically easy to implement but inefficient since they need to explore the entire rating data when predicting each new rating.

Model-based methods create a model to generate the recommendations, where the model parameters are usually learnt from the rating data and the rating prediction is made afterwards with the inferred parameters. The advantage of model-based methods is that predicting a new rating is efficient since only the parameters of the model are needed.

1.1.2 Latent Factor Model

The latent factor model [47, 56] is one of the most successful model-based collaborative filtering methods widely applied in recommender systems, which lays the foundation of our

[6] `http://www.rottentomatoes.com/`
[7] `http://www.imdb.com/`
[8] `http://www.last.fm/`
[9] `https://www.linkedin.com/`

proposed models in this thesis. In the general framework of latent factor model, the key idea is to factorize the rating matrix into the latent user vectors and latent item vectors in the low dimensional latent space. Latent user vector u_i represents user i's personal interests while latent item vector v_j expresses item j's features. By inferring the latent vectors, the predicted rating r^*_{ij} is then computed by the inner product $(u_i^T v_j)$.

The following is the common objective function for latent factor model, that is, observed ratings r_{ij} in the rating matrix are involved in a supervised approach to minimize the regularized squared error loss respect to $U = (u_i)_{i=1}^I$ and $V = (v_j)_{j=1}^J$:

$$E = min_{U,V} \sum_{i,j} \frac{\varepsilon_{ij}}{2}(r_{ij} - u_i^T v_j)^2 + \frac{\lambda_u}{2} \sum_i \|u_i\|^2 + \frac{\lambda_v}{2} \sum_j \|v_j\|^2 \tag{1.1}$$

where ε_{ij} is a binary indicator that is equal to 1 if user i rated item j and equal to 0 otherwise; λ controls the extent of regularization and is determined by cross validation.

Two main learning algorithms to minimize the Eq. (1.1) are (stochastic) gradient descent (GD) and alternating least squares (ALS). GD has the advantages on easy implementation and fast running time while ALS is good at parallelization for massive data sets.

Let us take GD as an example for model learning. A local minimum can be achieved by iteratively applying gradient descent method. In each iteration, first take the gradient of E respect to variable u_i, v_j:

$$\frac{\partial E}{\partial u_i} = \lambda_u u_i - \sum_j (r_{ij} - u_i^T v_j) v_j \tag{1.2}$$

$$\frac{\partial E}{\partial v_j} = \lambda_v v_j - \sum_i (r_{ij} - u_i^T v_j) u_i \tag{1.3}$$

Then update each variable by taking steps proportional to the negative of the gradient based on recent values:

$$u_i^{t+1} = u_i^t - \eta \frac{\partial E}{\partial u_i^t} \quad v_j^{t+1} = v_j^t - \eta \frac{\partial E}{\partial v_j^t} \tag{1.4}$$

where η is a parameter called learning rate and u_i^t, v_j^t stand for the value of u_i, v_j at iteration t. The predicted rating r^*_{ij} is computed by $r^*_{ij} = u_i^T v_j$ after enough iterations until the above equations (1.4) reach the convergence.

1.1.3 Topic Model

Topic models are a suite of algorithms that uncover the hidden thematic structure in document collections. We introduce the simplest topic model, latent Dirichlet allocation (LDA) [7] here, since it is frequently mentioned in this thesis as a tool for content analysis in recommender systems. LDA is a generative model that allows sets of observations to be explained by unobserved variables that explain why some parts of the data are similar. In particular, observations are words of document collections and unobserved variables are the per-document topic distribution and the per-topic word distribution. Each document is a mixture of a small number of latent topics and each word's creation is attributable to one of the document's topics. Let W be the size of the dictionary (i.e., vocabulary size) for document collections. For a given number of latent topics K, θ_d denotes the K-dimensional topic distribution for a document d and φ_j denotes the W-dimensional word distribution for a topic j. The generative process of LDA is as follows:

1. choose $\varphi_j \sim Dir(\beta)$ where $j \in [1, ..., K]$
2. choose $\theta_d \sim Dir(\alpha)$ for each document d
3. for each word position that belongs to document d
   (a) choose a topic $z \sim Mul(\theta_d)$
   (b) choose a word $w \sim Mul(\varphi_z)$

where β is the parameter of the Dirichlet prior on the per-topic word distribution, α is the parameter of the Dirichlet prior on the per-document topic distributions, $Dir(\cdot)$ stands for the Dirichlet distribution and $Mul(\cdot)$ stands for the Multinomial distribution.

Varitional Bayesian [7] and Gibbs sampling [22] are commonly used methods to infer the latent variables φ and θ.

1.2 Recommendation with Side Information

Although the rating matrix provides the basis for recommender systems, we observe that new recommendation scenarios are emerging that offer promising new information that goes beyond the rating matrix. This rich side information includes the user-generated text, the check-in location where the user rates an item, etc. These side information complement the original rating matrix for personalized recommendation.

For example, the content information can assist to address the cold start issue in recommendation, which is impossible for the collaborative filtering on the rating matrix because there is no rating for the cold user/item. In Gantner et al.'s work [18], the authors consider cold-start recommendation by learning a mapping function from the features of users and items to the latent vectors of users and items, and using this function to predict the latent vectors of new users and new items. Thus, the rating can be predicted through these latent vectors for new users and new items.

In this thesis, we focus on making use of the side information to improve real life recommender systems, where the side information is available and provides effective indicators to better recommendations. However, such kinds of side information also lead to some research issues:

- **Compatibility**. We believe that the side information and rating data are compatible with each other in recommender systems. However, the current methods prefer to incorporate the side information into the traditional models (e.g, as regularization terms), in which the rating matrix still has a dominant contribution to the final rating prediction. A sophistic model should balance the side information and the rating matrix in recommendation.

- **Complexity**. The recommender system may involve different kinds of side information, especially in heterogeneous networks. Each kind of side information is unique and the interplay between them is complex. It is an open question to blend different kinds of side information in heterogeneous networks.

- **Dynamics**. The real life recommender system is dynamic where the user preference may change over time. Accordingly, temporal approaches should be proposed beyond the previous static settings. However, there are few works considering the user interest shift and it still remains difficult to find out the time when the user begins to change her interests.

- **Reachability**. The recommendation is failed if the user is unavailable to adopt the recommended items. Particularly, the recommended POIs should be reachable in location-based recommender systems, which is often neglected by the state-of-the-art. It is still a challenge how to recommend when the user goes to a new place.

In this thesis, we try to address these issues in terms of different types of side information: content, temporal and spatial. We seek for novel approaches to incorporate these

side information with some domain knowledge to improve the recommender systems. The general data model in this thesis is that, given I users and J items with the $I \times J$ rating matrix R, and side information as input, the output is the predictions for unknown ratings in the matrix for recommendation.

The rating matrix remains the same but the side information may have a different format in each chapter. The rest of this thesis is organized as follows. First, we surveyed some representative methods related to these side information (Chapter 2). We categorized them and discussed their strengths and shortcomings. Then we presented our original works.

In recommendation with content information, we observed that a user likes an item because of some specific features of the item (Chapter 3). The side information here refers to the features for the item. Although most recommendation approaches are item-centric, we proposed a feature-centric model to analyze the feature-level preferences instead of the item-level preferences, thus, make prediction according to feature-level preferences. We introduced several strategies to generate the item ratings through feature-level preferences, and evaluated them on four data sets.

We further proposed a recommendation by blending content and attributes in academic networks (Chapter 4). The side information here refers to the text and attributes for the item. Academic networks are heterogeneous with the information of authorship, citation, published venue and year. The proposed model combines the unstructured text data with structured attributes data using topic modeling and latent factor model. Again, the experimental evaluation demonstrated the improvement of recommendation quality of the proposed model.

In recommendation with temporal information (Chapter 5), we proposed temporal matrix factorization to model the user's interest shift over time; such changes are essential for developing accurate recommender systems. The side information here refers to the time when rating the item. We proposed the transition matrix to model the evolution of latent user vector in the low-dimensional latent space. Both temporal matrix factorization and its fully Bayesian version are evaluated on six data sets.

In recommendation with spatial information (Chapter 6), we proposed a cross-region collaborative filtering method to deal with the POI recommendation when the user travels to a new place. The side information here refers to POI location and user's reviews. In this model, the long-term and short-term preferences are learnt from content information and location information respectively. The final recommendation considered both the long-term

and short-term preferences. This method was evaluated on two data sets with substantial improvement.

Finally, Chapter 7 concluded this thesis with some directions for future work.

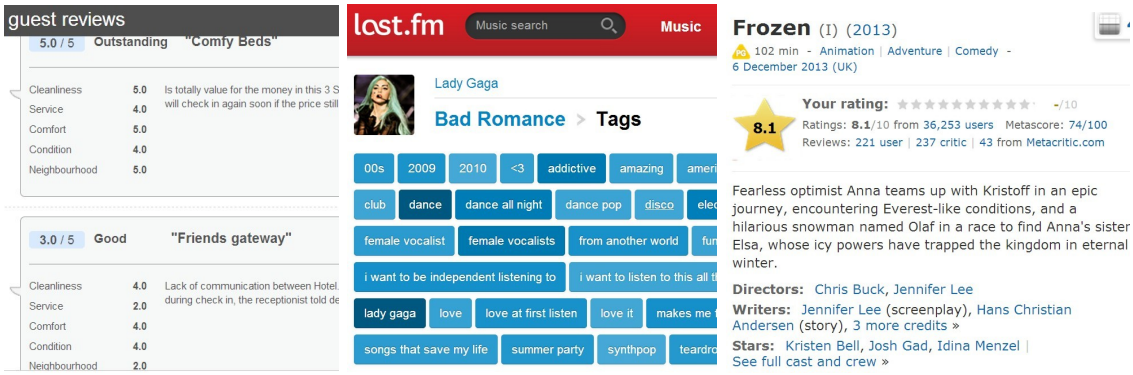
Chapter 2

Related Work: Beyond the Rating Matrix

In this chapter, we surveyed some representative models dealing with side information in recommender systems. Three typical side information, i.e., content information, temporal information and spatial information, are the main focus. These side information complement the original rating matrix for personalized recommendation. In general, content information helps to recognize and explain the user interests in an explicit way; temporal information helps to model the evolution of user interests over time; spatial information helps to indicate the user mobility pattern and activity areas for feasible recommendation. By appropriately involving the side information, the recommender systems achieve better performances beyond the rating matrix.

2.1 Recommendation with Content Information

This section surveyed the recent development of recommender systems with descriptive information. The content information could be unstructured texts or reviews, and features or tags that describes the attributes of the item. Such content information can be found in many rating systems on a daily basis. Figure 2.1 shows the screen shots for three types of content information beyond the ratings in recommender systems. Part (a) shows an interface for hotel reviews where a user can rate specific aspects of a hotel. Part (b) shows a way of expressing preferences attaching personal tags to an item. Part (c) shows a rating for a movie; although the user may rate only the movie, the preference on movie features can



(a) Reviews (b) Tags (c) Features

Figure 2.1: Content information in recommender systems

be derived from the set of features of each movie rated. The content information considered in this thesis is typically the user generated tags or reviews for items and item's attributes.

The content information usually manifests the user's reasons for rating the item and is an important source to be considered when recommending items in the future. Below, we summarize multiple recommendation techniques utilizing content information beyond the rating matrix.

2.1.1 Similarity-based Approaches

Recall that content-based filtering usually generates recommendations based on item profiles and item ratings. It treats recommendation as a user-specific classification problem and learns a classifier for the user's likes and dislikes. Some collaborative filtering techniques such as k nearest neighbors (kNN) also need to establish a set of similar users (or items). In both approaches, the content information can be used to measure similarity of items or users. We call them similarity-based approaches.

The kNN family of algorithms [1] is one of the most widely used memory-based collaborative filtering methods in recommender systems. The key idea behind it is to establish a set of similar users (or items), called the nearest neighbors, whose ratings over the target item (user) are then extrapolated in order to compute a rating prediction. Take the user-based kNN as example, a set of k neighbors of user i is determined according to a certain similarity measure. For example, the Pearson Correlation similarity is computed as follows:

$$sim(i,i') = \frac{\sum_{j \in S_{ii'}} (r_{ij} - \bar{r_i})(r_{i'j} - \bar{r_{i'}})}{\sqrt{\sum_{j \in S_{ii'}} (r_{ij} - \bar{r_i})^2} \sqrt{\sum_{j \in S_{ii'}} (r_{i'j} - \bar{r_{i'}})^2}} \tag{2.1}$$

where $r_{ij} = \emptyset$ means no rating is observed by user i to item j, $S_{ii'} = \{j \in J | r_{ij} \neq \emptyset \wedge r_{i'j} \neq \emptyset\}$ represents the set of items co-rated by both user i and i', and $\bar{r_i}$ represents the average rating of user i.

For each user i, the neighbors can be identified as k users with the largest similarity values. Then the predicted rating is given by:

$$r^*_{ij} = c + \sum_{i' \in N(i)} sim(i, i') \times r_{i'j} \tag{2.2}$$

where $N(i)$ indicates the neighborhood of user i and c is the normalization parameter.

For example, Sen et al. [58] predicted users' ratings for items based on inferred preferences for tags, where tags are used to measure the similarity between items. Several similarity metric such as Cosine or Pearson correlation are adopted. The model in Gedikli et al.'s works [19, 20] improves upon this by predicting tag preferences in the context of an item. The drawback is that these methods infer the preferences for the user's own tags; if an unrated item is attached with tags that the user has never used before, no prediction can be made for the item.

2.1.2 Latent Factor Models meet Topic Models

By taking advantages of the latent factor model for predicting the rating of items [55, 56], and several recent works enjoyed the ability to analyze text information with topic models and combined two models together for recommendation.

The collaborative topic regression (CTR) [65] studies the recommendation for scientific articles with each article being modeled by topic modeling on the text content of the article. In particular, recall that in the latent factor model, there is a vector v_j representing the item j in the latent space. With content information, topic modeling assigns a topic distribution θ_j to each item j with a Gaussian noise ϵ_j, such that:

$$v_j = \theta_j + \epsilon_j \tag{2.3}$$

and the objective function turns to:

$$E = min_{U,V} \sum_{i,j} \frac{\varepsilon_{ij}}{2} (r_{ij} - u_i^T v_j)^2 + \frac{\lambda_u}{2} \sum_i \|u_i\|^2 + \frac{\lambda_v}{2} \sum_j \|v_j - \theta_j\|^2 - \sum_j f(\theta_j^T \beta_w) \tag{2.4}$$

where f is a function to penalize the topic distribution θ_j with the word distribution β_w. An EM-style algorithm is proposed to learn the maximum a posteriori (MAP) estimates, where all the parameters (latent vectors u, v, topic distribution θ and word distribution β) are learnt and updated step by step.

Agarwal et al. [4] proposed fLDA, where the rating is modeled by the regression predictor of u_i and the regression predictor of v_j with the latent topic distribution of the item:

$$r_{ij} = a^T u_i + b^T v_j + s_i^T \bar{z}_j \tag{2.5}$$

where a, b and s_i are regression weights and $\bar{z}_j$ is the average topic distribution for item j. s_i is a user-dependent regression weights and Eq. (2.5) enforces all latent factors (including content) to contribute to the rating, not only item and user factors.

The advantage of these models is that two classes of algorithms successfully benefit with each other after the merge. But the topic model is based on discovering the co-occurrence relationship between words and works well for the textual data such as reviews. It could be inappropriate to apply it to other kinds of content information such as features or tags.

2.1.3 Latent Factor Models with Regularization

The content information also can be adopted to regularize the latent factor models. The followings are the related work in this direction. Given the content information represented by features, the regression-based latent factor model [3] incorporated features and past interactions to regress the latent vectors. For example, if x_i and x_j represent the user feature vector and the item feature vector respectively in the latent space, the latent user vector u_i and the latent item vector v_j turn to:

$$u_i = ax_i + \epsilon_i,\ v_j = bx_j + \epsilon_j \tag{2.6}$$

where a, b are regression weights and ϵ is the noise introduced. After these latent vectors are initialized by Eq. (2.6), the rest is similar in the basic latent factor model. In this way, users or items with similar features tend to have similar latent vectors, so features have indirect impact on the final ratings. Agarwal et al. [5] further extended their previous work [3] by modeling user-generated opinionated texts.

In Zhen et al.'s work [75], the tag information acts as a new regularization term in matrix factorization to constrain the latent vectors between users who frequently used similar tags.

In particular, the objective function turns to:

$$E = min_{U,V} \sum_{i,j} \frac{\varepsilon_{ij}}{2}(r_{ij} - u_i^T v_j)^2 + \frac{\lambda_u}{2} \sum_i \|u_i\|^2 + \frac{\lambda_v}{2} \sum_j \|v_j\|^2 + \frac{\lambda_t}{2} \sum_i \sum_{i'} sim(i, i')\|u_i - u_{i'}\|^2 \tag{2.7}$$

where the last term is the additional regularization term, $sim(i, i')$ is the similarity measure using tag information. Intuitively, this term forces the latent vectors of similar users to be close and this idea is also adopted with social network information if i and i' are friends [45].

One drawback of these models is that the content information as regularization only has an indirect impact on the rating prediction, which may not be suitable since some features or tags could be a direct indicator to user's preferences. For example, if a user is the superfan of Lady Gaga and is interested in any music or activities related to Lady Gaga, at this time, the tag "Lady Gaga" is a significant indicator of the user's preferences. However, this is impeded by modeling the tags as regularization terms.

2.1.4 Factorization Machines

The factorization machine (FM) [52] models multidimensional variable interactions (user, item, feature, etc.) through latent vectors. In detail, FM is a class of models that combines the advantages of support vector machines (SVM) and factorization models. FM has a good performance to incorporate various content information for sparse interactions typical in recommender systems at a low computational cost. FM allows stochastic gradient descent (SGD) and alternating least squares (ALS) optimization as well as Bayesian inference using Markov Chain Monte Carlo (MCMC). The advantages of FM lie on both efficiency and the ability of generalization.

In FM, the original rating matrix is transformed to a representation of regression as shown in Figure 2.2. The ith row $\mathbf{x}_i \in \mathbb{R}^p$ describes one case with p real-valued variables and y_i is the rating (target) in this case. Alternatively, one can describe this setting as a set of tuples $(\mathbf{x}, y)$, where (again) $\mathbf{x} \in \mathbb{R}^p$ is a feature vector and y is its corresponding target. Each column k in the feature vector is associated with a latent vector $v_k, k \in [1...p]$. FM models all nested interactions up to order d between the p input variables in $\mathbf{x}$. When $d = 2$, the regression is defined as:

$$\hat{y}(\mathbf{x}) := w_0 + \sum_{k=1}^{p} w_k x_k + \sum_{k=1}^{p} \sum_{k'=k+1}^{p} x_k x_{k'} (v_k^T v_{k'}) \tag{2.8}$$

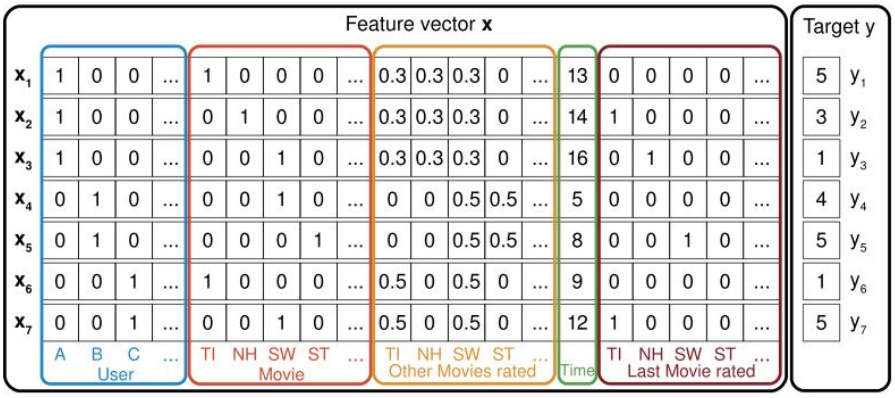


Figure 2.2: Example from Rendle's work [52] for representing a recommendation problem. Every row represents a feature vector $\mathbf{x}_i$ with its corresponding target y_i

where w_0 and w_k are weighting parameters. Note that the user-item interactions can be regarded as the inner product $u_i^T v_j$ in the latent factor model, but other nested interactions have no clear explanations in rating predictions. Besides, the machine is too "general" that does not incorporate any domain knowledge. In many cases, a domain-specific modeling dose better jobs than a general method.

FM allows the modeling of higher-order interactions in a way different from tensor factorization models. Besides, as discussed in Rendle's work [52], many previous models such as regression-based latent factor model [3] can be equivalently seen as the special case of FM. Several works extended FM afterwards, for example, the co-factorization machine [24] couples the learning of two FMs to study two aspects of Twitter data.

2.2 Recommendation with Temporal Information

This section surveyed the recent development of recommender systems with temporal information. Typically, static methods only learn a global model from a series of unordered ratings and are challenged when user behaviors change over time. By incorporating the temporal dynamics into the static models, additional improvements are observed with the advantage of the potential of temporal information. We summarize multiple temporal recommendation techniques, that is, utilizing temporal information beyond the rating matrix.

2.2.1 Time Partition or Decay Model

The basic assumption of these approaches is that users change their preferences as time goes by, so recent ratings better reflect their present tastes. For time partition model, it simply partitions the rating data by time, learns a model using the data in each partition and assigns different weights to each model. This approach may not work well because it not only misses the dependence of preferences for some users across time windows, but also accelerates the well known data sparsity problem.

Ding et al. [15] uses a time weighting scheme for a similarity based collaborative filtering approach, which decays the similarities to previously rated items as time difference increases at the prediction time. Such time weighting scheme $w(t)$ is usually a time decay function $w(t) = \exp(-\delta t)$, where δ is the decay rate. Then Ding et al. [15] modified the rating prediction by kNN in Eq. (2.2):

$$r^*_{ij} = c + \sum_{i' \in N(i)} sim(i, i') \times w(elapse(r_{i'j})) \times r_{i'j} \tag{2.9}$$

where $N(i)$ indicates the neighbourhood of user i the same as in Eq. (2.2), c is the normalization parameter, and $elapse(r)$ returns the elapsed time since rating r was done until the prediction time. In this way, older ratings have less weight in rating predication.

Later on, Ding et al. [16] and Ma et al. [46] extended this idea of time weighting scheme for better predication. Lee et al. [37] proposed a CF approach based on implicit feedback with the launch time of the item and the purchase time. Lee et al. [38] further conducted comprehensive analysis with different weighting schemes according to the temporal information. All these models showed some promising results.

As discussed above, the time decay scheme may miss a long-term effect for some users, since it seems to "truncate" the ratings in the very old time, thus, leading to the information loss. Besides, the weighting scheme is also hard to be estimated.

2.2.2 Temporal CF with Adaptive Neighbourhoods

Lathia et al. [35] proposed an algorithm for dynamically updating the neighbourhood size for each user and concluded that the neighbourhood size formed by kNN is changing over time. Their model adopted a metric called *time averaged* RMSE (TA_RMSE) [34], and outperformed the global model with static k. The adaptive neighbourhood size k is

computed by [35, 36]:

$$\forall i \in I : k_{i,t+1} = \max_{k \in P}(e_{i,t} - TA\_RMSE_{i,t,k}) \tag{2.10}$$

where $k_{i,t+1}$ is the k value for predicted ratings of user i in the time window $[t, t+1]$, P is a set of potential candidate values for k ($P = \{0, 20, 35, 50\}$ used in Lathia et al.'s work [35]), $e_{i,t}$ indicates the TA_RMSE achieved until time t for user i and the last term $TA\_RMSE_{i,t,k}$ is the other possible values for this user with a different k. Clearly, Eq. (2.10) returns the parameter k that maximized the improvement on the current error for each user.

One drawback of this approach is not to fully consider the user's changing preference over time, since the computation of the adaptive neighbourhood size k is based on all the past ratings of the user, which does not emphasize the user's current preference and not decline the importance of ratings in the past.

2.2.3 Latent Factor Models with Temporal Bias

Koren et al. [29] introduced the bias terms to the latent factor model. The intuition of introducing the linear biases is that the average rating for each user is essentially different. For example, a critical user may rate a good movie 3-star since her average rating is only 2-star, while for a common user, the rating of this movie is 5-star. If the former rating value is explained by an *interaction* of the form $(u_i^T v_j)$, the *bias* part to a rating value is only related to either user or item and independent of any interactions.

Let us denote 1-dimensional parameters α_i and β_j be the linear bias for user i and item j respectively, which control the user-specific and item-specific biases. Then Eq. (1.1) turns to the following objective function with such linear biases:

$$\min_{U,V} \sum_{i,j} \varepsilon_{ij}(r_{ij} - \alpha_i - \beta_j - u_i^T v_j)^2 + \lambda(\sum_i \|u_i\|^2 + \sum_j \|v_j\|^2 + \sum_i \alpha_i^2 + \sum_j \beta_j^2) \tag{2.11}$$

Koren [30] discussed the differences between temporal recommendation and concept drift [64] and presented the first temporal model for the matrix factorization approach by introducing time-variant biases for each user and each item. For example, the bias terms for user i and item j is:

$$b_{ij}(t) = \alpha_i(t) + \beta_j(t) \tag{2.12}$$

where $\alpha_i(t)$ and $\beta_j(t)$ are real valued functions changing over time.

In particular, Koren's work [30] associated a day t with an integer $Bin(t)$ and modeled the item bias into a stationary part and a time-variant part:

$$\beta_j(t) = \beta_j + \beta_{j,Bin(t)} \tag{2.13}$$

For the user bias, Koren's work [30] considered both gradual concept drifts and sudden drifts.

$$\alpha_i(t) = \alpha_i + \lambda \cdot dev_i(t) + \alpha_{i,t} \tag{2.14}$$

where $dev_i(t)$ captures the gradual drift and $\alpha_{i,t}$ denotes the day-specific variability.

The time-variant biases are the differences at each time window, so the model complexity becomes higher to infer these parameters. This approach works for prediction only if two windows share similar biases.

2.2.4 Kalman Filtering

Lu et al. [43] proposed a spatio-temporal model for collaborative filtering, in which a linear Kalman filtering [27] is adopted to estimate the temporal structure. Sun et al. [61] modeled temporal effects using Kalman filtering with a transition process parameter for each user. These parameters are time dependent. For example, the evolution of the latent user vector u_i is linear according to the transition process $A_{i,t}$:

$$u_{i,t+1} = A_{i,t}u_{i,t} + \epsilon_{i,t} \tag{2.15}$$

where $\epsilon_{i,t}$ is the noise parameter. This forms a dynamic system with rest part remaining similar to the traditional latent factor model. The Rauch-Tung-Striebel (RTS) smoother [51] is adopted to infer the parameters in this system.

Chua et al. [13] further extended Sun et al.'s work [61] in the scenario of temporal adoption. One limitation of Kalman filtering is that the transition process parameter is user-supplied. Clearly, specifying such parameters for all users at all time points is impractical.

2.2.5 Tensor Factorization

Another approach to deal with temporal information is tensor factorization. Xiong et al. [68] proposed the user-item-time tensor factorization to model temporal effects and

provided a fully Bayesian treatment to automatically tune parameters and control the model complexity.

Extending the idea of the latent factor model, the three-dimensional tensor r_{ij}^t is the user i rating on item j at time window t, and can be expressed as the inner product of three D-dimensional latent vectors.

$$r_{ij}^t \approx < u_i, v_j, s_t > \equiv \sum_{d=1}^{D} u_i(d) v_j(d) s_t(d) \tag{2.16}$$

where s_t is the additional latent feature vector for time window t.

The tensor factorization treats time as a universal dimension shared by all users. This is argued by Xiang et al. [67] that time dimension is a local effect. The local effect of this work [67] argues for that each user has its own time change patterns, which is against "shared by all users". Xiang et al. [67] used a graph connecting users, items, and sessions to model users' long-term preferences and short-term preferences. The links between user and item nodes represent that items are viewed by a user at any time (long-term preferences) and the links between user-session and item nodes indicate items are viewed by a user during a specific time session (short-term preferences).

2.3 Recommendation with Spatial Information

This section surveyed the recent development of recommender systems with spatial information. We summarize multiple location-based recommendation techniques, that is, utilizing location information beyond the rating matrix.

2.3.1 Location as Items

Early works in location-based recommender systems simply treated location as spatial items, and traditional approaches for items can also be adopted without much modification. For example, the works [76, 78] studied recommender systems for locations/activities using trajectory data such as GPS data. In Zheng et al.'s work [78], the interest of location and user experience in a location are considered mutually reinforcing each other and a HITS-like algorithm is proposed to learn their scores. Like hub/authority scores, the interest of location and user experiences are global, i.e., not specific to a pair of location and user. In Zheng et al.'s work [76], a location-activity matrix is extracted from GPS data with

the help of activity correlation mining and location feature extraction, and a location is recommended for a given activity, or an activity is recommended for a given location.

In Bao et al.'s work [6], the rating data is partitioned by regions and categories of locations, and for each region and each category, "local experts" are extracted based on the historical data in that region and category. As pointed out in Section 2.2.1 dealing with temporal information, partitioning the rating data by regions will accelerate data sparsity and reduces the level of collaborative filtering.

Later on, the distance is measured as additional weight. In Ye et al.'s work [71], the geographical influence of location is modeled by a naive Bayesian method and power-law distribution of distance, under the assumption that the distances of POI pairs visited by a user are independent. Given a collection l_i of POIs visited by a user i, the likelihood of visiting a POI j is modeled by the conditional probability $P(j|l_i)$ computed by $\prod_{j' \in l_i} P[d(j, j')]$, where the distance probability $P[d(j, j')]$ is assumed to follow a power-law distribution.

2.3.2 Generative Models with Location

Rather than the simple treatment to location in the above mentioned methods, generative models assign a latent variable to location, which will have a direct impact on the probability of recommending spatial items. For example, Kurashima et al. [33] extended topic model with geographical influence [71] in location recommendation. Geographical influence suggests that locations that are closer to user's visited locations tend to have a high probability to be recommended. Based on this idea, the proposed Geo topic model still has topic distributions θ_i for each user i and location distributions ϕ_z for each topic z, the probability that user i with check-in history l_i (l_i represents the set of all check-ins by user i) visits location j is calculated by the following equation:

$$P(j|i, l_i, \theta, \phi) = \sum_z P(z|i, \theta) P(j|z, l_i, \phi) \tag{2.17}$$

$P(j|z, l_i, \phi)$ is the probability that location j is chosen form topic z, considering the geographical influence:

$$P(j|z, l_i, \phi) = \frac{1}{c} \exp(\phi_z(j)) \sum_{k \in l_i} \exp(-\frac{\beta}{2} d(j, k)) \tag{2.18}$$

where $d(j,k)$ represents the distance between location j and k, β is the weighting parameter and c is the normalization parameter. The final recommendation are those locations j the with high probability in Eq. (2.17).

Hu et al. [25] proposed a spatial topic model by incorporating user posts as well as user movements into a geographical topic model where regions and topics are random latent variables associated with each post. Their model predicts the location for a given document based on a collection of geo-tagged posts, which tends to give high probabilities to locations close to user's frequently visited locations before. More specifically, the location j is generated according to not only the geographical region r, but also the topic z learnt from the content (posts):

$$j \sim P(j|r, z, \phi, \mu, \Sigma) \tag{2.19}$$

where ϕ is the word distribution in topic model, and μ, Σ are the priors in Gaussian distribution.

The above mentioned generative models are all based on modeling the probability distribution of the POIs for a user based on the observed data, therefore, favor the POIs in the user's frequently visited region (e.g., the home city). Such methods are not suitable if the user travels to a *new* region where she has not generated any post or check-in activity. For example, if a user checked-in POIs in Beijing in the past and is currently visiting New York City where she has not checked-in any POI, those generative models tend to recommend the POIs in Beijing because they have the highest estimated probability.

Yin et al. [72] presents a topic model called LCA-LDA to incorporate item content and user's visit history. However, their focus is to make appropriate recommendations for a user visiting a new region. By treating each POI as word, their model learns the topic distributions for each user and each region (e.g., city), and for a user coming to a region, the recommendation of venues is derived from the distributions learnt for that region. The drawback of this model is not to consider the distance or location information for POIs across regions, which could be an important factor in real life recommendation.

2.3.3 Latent Factor Models with Location

One drawback of the generative models is that the generated probability distributions cannot be interpreted as ratings directly. This probability is usually used to rank the items for top-k recommendation but inapplicable to the task of rating prediction. Some works based on the latent factor model address this issue by incorporating the spatial information.

Cheng et al. [9] leveraged the user check-in information to build a user-location matrix which contains each user's check-in frequency on locations, and adopted a multi-center Gaussian model to model the probability of a user's check-in behavior over spatial items and to predict those unknown frequency to the locations that the user has never been to. In this work, Gaussian models are adopted for location distributions.

Similar to Cheng et al.'s work [9], Liu et al. [42] proposed a topic and location-aware matrix factorization model for location recommendation. First, the interest topics of users are learnt by mining the textual information associated with POIs. Then, the topic and the location influence are added into the matrix factorization framework and both contribute to the final rating prediction. More specifically, the distribution over the observed ratings and the textual information is as follows:

$$P(r|U,V,TL,\sigma) = \prod_{i=1}^{I}\prod_{j=1}^{J}[\mathcal{N}(r_{ij}|f(u_i^T v_j, TL_{ij}),\sigma^2)]^{\varepsilon_{ij}} \tag{2.20}$$

where TL is the topic and location influence predefined, $\mathcal{N}(\cdot|\mu,\sigma^2)$ is a Gaussian distribution with mean μ and variance σ^2, and $f(u_i^T v_j, TL_{ij}) = u_i^T v_j \times TL_{ij}$ is a function to approximate the rating of user i to POI j.

Liu et al. [41] proposed a geographical probabilistic factor model taking latent preferences, user mobility and item popularity into consideration. Their model assigns a user to a latent region learnt from user's previously visited POIs and sets the region center as the user's location. Each POI location l_j is drawn from a region-dependent multivariate normal distribution:

$$l_i \sim \mathcal{N}(\mu_k, \sigma_k) \tag{2.21}$$

where k is a region with center μ_k and each user is associated with a certain region as the common activity area. Then the user-POI distance is modeled as:

$$d(i,j) = \sqrt{\|\mu_k - l_j\|_2} \tag{2.22}$$

If $u_i^T v_j$ reflects the user i's intrinsic preference to POI j, the final rating is monotonically decreases with the distance between them:

$$r_{ij}^* = u_i^T v_j (d_0 + d(i,j))^{-\gamma} \tag{2.23}$$

where $(d_0 + d(i,j))^{-\gamma}$ is a power-law like parametric term to model the distance factor in the decision making process.

The location-based recommendation also benefits from the research in social science. For example, some interesting findings in Cho et al.'s work [12] motivates the research in modeling human mobility or location-based recommendation. One is that the short-ranged travel is periodic and not affected by the social network, while the long-distance travel is more influenced by social ties. Another is that users tend to move within a small number of regions, e.g., around the home area or work places.

Chapter 3

Feature-centric Recommendation

We proposed our first model dealing with content information in this chapter. Typically a user prefers an item (e.g., a movie) because she likes certain features of the item (e.g., director, genre, producer). This observation motivates us to consider a *feature-centric recommendation* approach to item recommendation: instead of directly predicting the rating on items, we predict the rating on the features of items, and use such ratings to derive the rating on an item. This approach offers several advantages over the traditional item-centric approach: it incorporates more information about why a user chooses an item, it generalizes better due to the denser feature rating data, it explains the prediction of item ratings through the predicted feature ratings. Another contribution is turning a principled item-centric solution into a feature-centric solution, instead of inventing a new algorithm that is feature-centric. We demonstrate this approach by turning the traditional item-centric latent factor model into a feature-centric solution and demonstrate its superiority over item-centric approaches.

3.1 Motivation

A user likes an item because of some *specific features* of the item. When a user likes an item, she may like some features of the item but is not impressed with other features; consequently, two users may like the same item for different reasons. For example, a user may like the rate of a hotel but not its service, while another user may like the cleanliness of the hotel but nothing else. If the above observation holds, the design principle "liking same items (as other users) leads to liking more same items" practised by the standard collaborative filtering may not work.

Table 3.1: Example for feature-level preferences

	W={anti-allergy, rose}	X={anti-allergy, rose}	Y={anti-allergy, orange}	Z={sun proof, rose}
A	like (due to anti-allergy)	like (due to anti-allergy)	? (like)	? (unknown)
B	like (due to rose)	like (due to rose)		like (due to rose)

To explain our point, let us consider the toy example in Table 3.1 with four items W={anti-allergy, rose}, X={anti-allergy, rose}, Y={anti-allergy, orange}, and Z={sun proof, rose} where {} contains the features for the item. In the history, suppose that user A loves W and X due to "anti-allergy" and user B loves W, X, and Z due to "rose". In traditional collaborative filtering, it will recommend Z to user A since A and B both like W and X. On the other hand, for content-based filtering, Y has the feature "anti-allergy" and Z has the feature "rose" as in W and X. Therefore, the score is the same, but it is obvious that A will love Y more than Z from the view of feature level.

User's preferences on features are available in many real life rating systems. For example, in the hotel rating systems a user can rate specific features (cleanliness, service, etc.) of a hotel. Other systems [58] allow users to explicitly express preferences on features by attaching personal tags to an item. The ratings of features (e.g., director, actor) can also be implicitly inherited from the features of items (e.g., movie). Such information manifests the user's reasons for rating the item, led by different preferences on features.

3.1.1 Our Approach

Our approach is motivated by the above discussion, therefore, if we can predict user's preference on features, we are able to predict user's preference on an item; if we can express observed preference on features in the same format as observed preference on items, any principled collaborative filtering algorithm for items can be applied to predict user's preference on features. This thinking leads to a *feature-centric recommendation* approach in which features of items are the "King": given the usual user-item rating matrix as well as the feature information, we first convert item ratings into feature ratings and obtain a user-feature rating matrix. We directly perform collaborative filtering on the user-feature rating matrix, which practices our design principle "liking same features (as other users) leads to liking more same features". The output is a model for predicting a user's rating on a feature.

To predict a user's rating on an unrated item, we need to integrate the predicted feature ratings to derive the rating for the item. While sum and average are obvious choices, features are not equally representative, e.g., the feature "anti-allergy" clearly has a more significance than the other features for user A. So we present two novel integration approaches, one is heuristic based and one is regression based, to give different significance for feature preferences.

The key innovation is that our approach transforms the item ratings into feature ratings and later the modeling is fully at the feature level. To this end, our approach takes the advantages of collaborative filtering and content-based filtering, e.g., even if items are not shared among users, features of items may still be shared, which helps address the cold start problem of a new item. As the final prediction on the item rating is an integration of a set of feature ratings, our modeling actually involves no "items", which is fundamentally different to the modeling of the existing feature based recommendations [8, 23]. Their modelings all have "items" (e.g., latent item vectors) which directly affect the final prediction while features are used as a finer description to regularize the item ratings. However, we fully rely on the feature ratings and may avoid some side effects introduced by items.

3.1.2 Comparison with Related Work

The regression-based latent factor model [3] incorporated features and past interactions to regress the latent vectors. Items with similar features tend to have similar latent vectors, so features have indirect impact on the final ratings. Agarwal et al. [5] further extended their previous work [3] by modeling user-generated opinionated texts. In Gantner et al.'s work [18], the features of users and items are used to predict the latent factors of new users and new items; existing users and items do not benefit from the available feature information. The collaborative topic regression (CTR) [65] studies the recommendation for scientific articles with each article being modeled by topic modeling on the text content of the article. The factorization machine (FM) [52] models multidimensional variable interactions (user, item, feature, etc.) through latent vectors. The tensor factorization [28] generalizes the rating matrix with additional context information. The co-factorization machine [24] couples the learning of two FMs to study two aspects of tweeter data. All these methods treat the features of an item equally as side information and no preference is captured on features.

Sen et al. [58] predicted users' ratings for items based on inferred preferences for tags, but the preferences are global for all items, that is, a tag is either liked or disliked for all

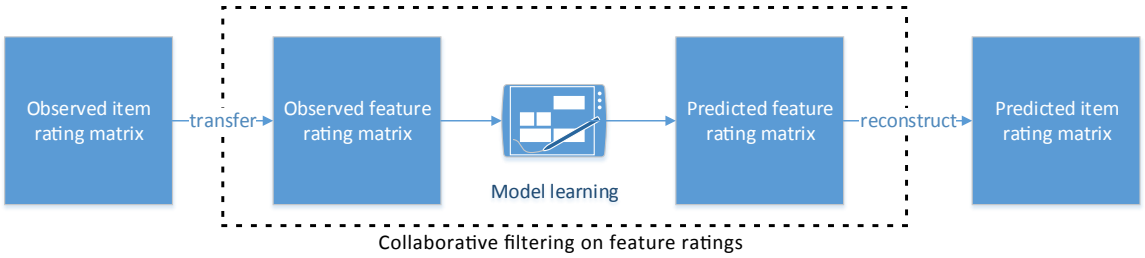


Figure 3.1: The feature-centric recommendation approach

items. The works [19, 20] improved upon this by predicting tag preferences in the context of an item. All these methods infer the preferences for the user's own tags; if an unrated item is attached with tags that the user has never used before, no prediction can be made for the item. Our method does not have this problem because it employs collaborative filtering on feature ratings, which can predict a rating for any pair of user and feature. The tag-aware recommendation in Tso et al.'s work [63] models a 3-way relation $< user, item, tag >$ by 2-way relations $< user, tag >$, $< item, tag >$, and $< user, item >$, which cannot express the 3-way information that a user i tags an item j using a tag t. There is a similar problem with Zhou et al.'s model [79]. In Zhen et al.'s work [75], the tag information acts as a new regularization term in matrix factorization to constrain the latent vectors between users who used similar tags. Features play a more central role in our model in that we learn latent vectors for features and predict the rating of features. If a user prefers an item because of specific features of the item, our feature-centric approach is more sensitive to user preferences.

3.2 Feature-Centric Recommendation

We present a feature-centric recommendation approach to utilize user's feature preferences to improve recommendation of items. This approach is shown in Figure 3.1 where the dashed box encompasses a standard collaborative filtering method. We consider the latent factor model for this box, but it could be replaced with any other collaborative filtering methods. We discuss the key steps of extracting an observed feature rating matrix and predicting item ratings from feature ratings in the following sections.

	W	X	Y	Z
A	4	4		
B	5	4		4

(a) User-item rating matrix R

	anti-allergy	rose	sun proof
A	{4,4}		
B		{5,4,4}	

(b) User-feature rating matrix R'

Figure 3.2: Construct the user-feature rating matrix (right) from the original ratings (left)

3.2.1 Extracting User-Feature Rating Matrix

We assume there are I users, J items, T features. Each item j is associated with a bag of features, denoted by B_j. For a feature t, S_t denotes the set of items j such that $t \in B_j$. The original rating data can be represented by an $I \times J$ user-item rating matrix R in which each element r_{ij} indicates user i's rating value to item j.

In addition, when the user i rates the item j, the user may optionally rate or select (such as tagging) some features t of item j. If the user rates the feature t for item j, $h_{it}(j)$ denotes this rating. If the user i selects the feature t but does not rate t, $h_{it}(j) = r_{ij}$, which is user i's rating on item j. If the user i does not select any feature at all when rating the item j, $h_{it}(j) = r_{ij}$ for all features t of item j as we believe that the user implicitly selects all features. In all other cases, $h_{it}(j)$ is undefined.

User-feature rating matrix. For user i and feature t, $\{h_{it}(j)\}$ denotes the bag of defined ratings $h_{it}(j)$ for all items $j \in S_t$. We extract an $I \times T$ *user-feature rating matrix* denoted by R', there is one row for each user i, one column for each feature t, and the entry for (i, t) is equal to $\{h_{it}(j)\}$. As the toy example in Section 3.1, if user A rates the item W with the rating 4 and the item X with the rating 4, and selects the feature "anti-allergy" for both, so $h_{A,anti-allergy}(W) = 4$ and $h_{A,anti-allergy}(X) = 4$, and the entry for ($A$,"anti-allergy") is $\{4, 4\}$, as shown in Figure 3.2.

We adopt latent factor model [56] on R' to produce the latent user vector u_i for each user i and the latent feature vector f_t for each feature t, and the objective is to minimize $\sum_{j \in S_t} \varepsilon_{ijt}(h_{it}(j) - u_i^T f_t)^2$ where ε_{ijt} is equal to 1 if $h_{it}(j)$ is defined, and is equal to 0 otherwise. User i's predicated rating on feature t is given by $u_i^T f_t$. This part is the standard method for the latent factor model except that items are substituted by features. The interested reader please refer to Salakhutdinov et al.'s work [56] for the detailed inference.

3.2.2 Predicting Item Ratings by Heuristic

We present two heuristic strategies for integrating the predicted feature ratings to derive the predicted item rating for a user in this section. One strategy is using the average of feature ratings to predict the rating r^*_{ij} for item j, that is,

$$r^*_{ij} = \frac{1}{|B_j|} \sum_{t \in B_j} u_i^T f_t \tag{3.1}$$

This prediction treats all features in B_j equally because each $u_i^T f_t$ has the weight $1/|B_j|$. It does not take into account, for example, whether the user occasionally selects the feature t by chance or consistently selects the feature. Below, we present another strategy that consider such differences.

The second strategy is to introduce a weighting scheme specific to individual users. If a user i selects a feature t frequently, the user is more interested in t. Besides relative frequency of selection, the absolute number of selection also matters. For example, selecting a feature twice out of 3 ratings has the same frequency as selecting a feature 20 times out of 30 ratings, but the latter has more statistical significance. This strategy suggests that features are not equally representative.

We propose a single weighting scheme to account for both relative frequency and statistical significance. Suppose that a user i has rated N items, among them, the feature t was selected s times. We can regard this as one sample where the event t was observed s times in N trials, where $\hat{p} = \frac{s}{N}$ is the *observed proportion*. We want to bound the *true proportion* that the user selects the feature t. Let $CI(s, N)$ denote the confidence interval for user i selecting feature t. We adopt the following Wilson score [66] interval since it is an improvement over the usual normal approximation.

$$CI(s, N) = [c - \sigma, c + \sigma] \tag{3.2}$$

where

$$c = \frac{1}{1 + \frac{1}{N} z^2} (\hat{p} + \frac{1}{2N} z^2)$$

$$\sigma = \frac{1}{1 + \frac{1}{N} z^2} z \sqrt{\frac{1}{N} \hat{p}\,(1 - \hat{p}) + \frac{1}{4N^2} z^2}$$

z is the $1 - \frac{1}{2}\alpha$ percentile of a standard normal distribution and α is the error percentile. For a 95% confidence level the error α is 5%, so $1 - \frac{1}{2}\alpha = 0.975$ and $z = 1.96$.

A larger c and a smaller interval size σ represent a more significant selection of t. Therefore, we use the mid-point of the lower bound $c - \sigma$ and the interval center c to measure the weight of selecting t by user i: $l_{it} = \frac{1}{2}(c - \sigma + c) = c - \frac{1}{2}\sigma$. The predicted rating r^*_{ij} of item j by user i is then defined as

$$r^*_{ij} = \frac{1}{L_j} \sum_{t \in B_j} l_{it} u_i^T f_t \tag{3.3}$$

where $L_j = \sum_{t \in B_j} l_{it}$. Note that this weighting scheme is unique to features, not items, because only a feature can be selected by a user multiple times.

3.2.3 Predicting Item Ratings through Regression

In the previous section, the weighting schemes are computed by heuristic, which may not capture the essential by model fitting. In this section, we propose a regression model to automatically learn the global weighting for features. We assume that there exists a weighting vector $\mathbf{w}$, with $w_t \in \mathbf{w}$ representing the importance for each feature t. The training set contains historic ratings, where $\mathbf{x}_k$ is the kth input data and y_k is the kth output data, in particular, if this rating is made by user i, $y_k = r_{ij}$, $\mathbf{x}_k(t) = u_i f_t$ if $t \in B_j$ and $\mathbf{x}_k(t) = 0$ otherwise. The regression model is trained by the relationship between input and output data: $y_k = \mathbf{w}^T \mathbf{x}_k + b_i$. Following support vector regression (SVR) [60], our goal is to find optimal weighting $\mathbf{w}$ and user specific bias b_i that fit the model best. The optimization problem becomes,

$$\min_{\mathbf{w}, \xi, \xi^*} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2} \sum_k (\xi_k^2 + \hat{\xi}_k^2) \tag{3.4}$$

$$s.t. \quad |\mathbf{w}^T \mathbf{x}_k + b_i - y_k| \leq \varepsilon + \xi_k; \tag{3.5}$$

$$\xi_k, \hat{\xi}_k \geq 0; \forall k. \tag{3.6}$$

where $\xi_k, \hat{\xi}_k$ are slack variables and C is a constant.

The primal Lagrangian is,

$$\begin{aligned} \mathcal{L}_P = \frac{1}{2}\|\mathbf{w}\|^2 &+ \sum_k \alpha_k (\mathbf{w}^T \mathbf{x}_k + b_i - y_k - \varepsilon - \xi_k) \\ &+ \sum_k \hat{\alpha}_k (y_k - \mathbf{w}^T \mathbf{x}_k - b_i - \varepsilon - \hat{\xi}_k) + \frac{C}{2} \sum_k (\xi_k^2 + \hat{\xi}_k^2) \end{aligned} \tag{3.7}$$

Table 3.2: Statistics of data sets

	Delicious	**Lastfm**	**DBLP**	**Movielens**
User	1867	2100	6815	1857
Item	69223	18744	78745	4721
Feature	40897	12647	81901	8288
Item ratings	104799	71064	436704	20607
Feature ratings	437593	186479	2554597	36885
Density (item)	8.1×10^{-4}	1.8×10^{-3}	8.1×10^{-4}	2.3×10^{-3}

where $\alpha_k, \hat{\alpha}_k$ are Lagrange multipliers. We take the derivatives with respect to $\mathbf{w}, b_i, \xi_k, \hat{\xi}_k$, leading to the Karush-Kuhn-Tucker (KKT) conditions [32] as follows:

$$\mathbf{w} = \sum_i (\alpha_k - \hat{\alpha}_k) x_k, \quad \xi_k = \alpha_k / C, \quad \hat{\xi}_k = \hat{\alpha}_k / C \tag{3.8}$$

We skip the comprehensive inference as SVR is a principled method. More details refer to Smola et al.'s work [60]. Once the optimal weighting vector $\mathbf{w}$ and the bias term b_i are found, the predicted rating for specific item j is given by:

$$r_{ij}^* = \sum_{t \in B_j} w_t u_i^T f_t + b_i \tag{3.9}$$

Feature selection. We conduct a 2-step feature selection that aims at the representative features. (1) we calculate the cosine similarity to quantify correlations between various predictors (feature ratings) and the item ratings to identify the best predictors. According to the result, we remove features with relatively low similarity by a threshold. (2) we fit the model with the remaining features. If the model performance is close to the original one, we believe that the removed features are less representative. This procedure is optional but helps the prediction accuracy.

3.3 Experimental Evaluation

We report our findings on the evaluation of the proposed feature-centric recommendation against well known baselines using real life data sets. We first introduce data sets, baseline methods, and evaluation metrics.

3.3.1 Data Sets

We employed four data sets: *Delicious*, *Lastfm*, *DBLP*, and *Movielens*. The first two data sets were recommended as benchmark data sets for studying recommender systems by the 2011 HetRec conference[1]. These data sets contain user's tagging information on bookmarks and music songs, which expresses user's ratings or preferences on items. We treat tags as the features of an item. Delicious contains 1867 users' ratings on 69223 items with 40897 unique features. Lastfm contains 2100 users' ratings on 18744 items with 12647 unique features. The third data set *DBLP* contains authors, papers and citation information from an academic network. We treated authors as users, papers as items, each publishing/citation of a paper as user's rating on the paper, and treated the venues and authors of a paper as the features of the paper. After removing the users with fewer than 10 papers from the original DBLP data set[2], the final data set contains 6815 users' ratings on 78475 items with 81858 unique features. All the above data sets have binary ratings. The fourth data set *Movielens*, also recommended by the 2011 HetRec conference, was collected from a movie review system. This data set has the ratings ranged from 1 to 5. We removed those movies without any ratings. The resulting data set has 1857 users' ratings on 4721 items with 8288 unique features (i.e., tags). The statistics of these data sets are found in Table 3.2. We conducted 10-fold cross validation for all data sets.

3.3.2 Evaluated Methods

The first baseline is the probabilistic matrix factorization that ignores features of items:

Probabilistic matrix factorization (denoted PMF): This method adopts matrix factorization on the user-item rating matrix [56]. Following Salakhutdinov et al.'s work [56], we set the parameters $\lambda_u = \lambda_v = 0.01$.

The next four baselines consider features of an item. All the baselines were previously proposed in the literature.

Collaborative topic regression (denoted CTR): This is matrix factorization with topic modeling applied to features of items [65]. Following Wang et al.'s work [65], we set the parameters $\lambda_u = \lambda_v = 0.01, \alpha = \frac{50}{D}$ and $\beta = 0.01$.

Factorization machine (denoted FM): This is the factorization machine approach in Rendle's work [52]. FM takes selected features into consideration equally while our model

[1]http://www.grouplens.org/data sets/hetrec-2011/
[2]http://arnetminer.org/citation

gives more importance to representative features. We run the code of Rendle's work [52] with the default settings. Note that we need not compare with Chen et al.'s work [8] since it can be modeled by FM as indicated in Rendle's work [52].

Regression latent factor model (denoted RLFM): This is the regression based latent factor model [3]. RLFM incorporated features as side information to regress the latent vectors so as to improve the performance. We run the code of Agarwal et al.'s work [3] with the default settings.

Similarity based method (denoted SIM): This is a content-based filtering approach that uses the SVM regression [20] to predict the user's ratings on items according to inferred feature preferences. Note that the computation of SIM is not in the latent space.

The next method is the feature-centric approach proposed in this chapter's work:

Feature-centric recommendation (denoted FCR): This is the feature-centric solution proposed in Section 3.2. We denote FCR-a, FCR-u and FCR-r for different integration strategies, i.e., averaged heuristic, user-specific heuristic and regression model. FCR-a is computed by Eq. (3.1), FCR-u is computed by Eq. (3.3), and FCR-r is computed by Eq. (3.9).

For all methods except for SIM, we adopt the dimensionality of $D = 20$ for latent vectors and the learning rate of $\eta = 0.0001$.

3.3.3 Evaluation Metrics

RMSE (root mean squared error) and **MAE** (mean absolute error) quantify the difference between the rating values predicted by a recommender and the true values in the testing set. These two metric are defined as follows: $RMSE = \sqrt{\frac{1}{n}\sum_{i,j}(r_{ij} - r^*_{ij})^2}$, $MAE = \frac{1}{n}\sum_{i,j}|r_{ij} - r^*_{ij}|$, where r_{ij} is the true rating value, r^*_{ij} is the predicted rating value, and n is the number of ratings in the testing set. The smaller these values are, the better the result is. As pointed out in Koren's work [30], achievable RMSE values lie in a quite compressed range and small improvements in RMSE terms can have a significant impact on the quality of the top few presented recommendations.

Recall@k quantifies the fraction of rated items that are in the top-k of the ranking list sorted by their estimated ratings from among all rated items in the test set. For each user i: $Recall@k = \frac{|N(k;i)|}{|N(i)|}$, where $|\cdot|$ denotes the number of elements in a set, $N(i)$ is the set of items rated by i in the testing set and $N(k;i)$ is the subset of $N(i)$ contained in the top-k list of all items sorted by their estimated ratings.

Table 3.3: $RMSE$ and MAE of four data sets

	Delicious		**Lastfm**	
Methods	**RMSE**	**MAE**	**RMSE**	**MAE**
Baselines				
PMF	0.8907 ± 0.0046	0.8081 ± 0.0048	0.4449 ± 0.0040	0.3179 ± 0.0022
CTR	0.7844 ± 0.0004	0.7431 ± 0.0005	0.5078 ± 0.0025	0.4084 ± 0.0025
FM	0.3551 ± 0.0017	0.2906 ± 0.0020	0.3239 ± 0.0022	0.2534 ± 0.0030
RLFM	0.4182 ± 0.0010	0.3978 ± 0.0010	0.3208 ± 0.0014	0.2235 ± 0.0015
SIM	0.4001 ± 0.0008	0.3872 ± 0.0011	0.3269 ± 0.0013	0.2941 ± 0.0013
Proposed Methods				
FCR-a	0.3169 ± 0.0020	0.2396 ± 0.0016	0.3790 ± 0.0023	0.3062 ± 0.0022
FCR-u	0.2572 ± 0.0023	0.1645 ± 0.0014	0.2455 ± 0.0032	0.1468 ± 0.0020
FCR-r	$\mathbf{0.2176} \pm 0.0011$	$\mathbf{0.1513} \pm 0.0009$	$\mathbf{0.1868} \pm 0.0020$	$\mathbf{0.1066} \pm 0.0016$

	DBLP		**Movielens**	
Methods	**RMSE**	**MAE**	**RMSE**	**MAE**
Baselines				
PMF	0.5060 ± 0.0021	0.3800 ± 0.0020	1.1271 ± 0.0247	0.8622 ± 0.0197
CTR	0.4943 ± 0.0021	0.3653 ± 0.0018	1.0880 ± 0.0191	0.8315 ± 0.0156
FM	0.1821 ± 0.0023	0.1167 ± 0.0020	1.2049 ± 0.0229	0.9467 ± 0.0194
RLFM	0.2297 ± 0.0007	0.1930 ± 0.0007	1.0662 ± 0.0215	0.8056 ± 0.0145
SIM	0.3064 ± 0.0003	0.3032 ± 0.0003	1.0137 ± 0.0190	0.7616 ± 0.0123
Proposed Methods				
FCR-a	0.2043 ± 0.0016	0.1393 ± 0.0017	0.9966 ± 0.0148	0.7770 ± 0.0105
FCR-u	0.1204 ± 0.0010	$\mathbf{0.0739} \pm 0.0006$	$\mathbf{0.9515} \pm 0.0150$	0.7306 ± 0.0092
FCR-r	$\mathbf{0.1064} \pm 0.0004$	0.0841 ± 0.0003	0.9724 ± 0.0201	$\mathbf{0.7208} \pm 0.0125$

3.3.4 Experimental Results

RMSE and MAE. Table 3.3 shows RMSE and MAE with standard errors for different methods. Among them, FM and RLFM are recent developed models which can incorporate features for better predications. The best performers for each data set are highlighted in bold face. All reported RMSE and MAE are the average of the 10 runs in the 10-fold cross-validation.

First, PMF performs poorly on all data sets since it only considers the item ratings and ignores the feature information. When this matrix is sparse, the ratings as only the similarity information among users are hardly enough to make accurate recommendation. CTR slightly improve the performance over PMF by using features as the side information to regularize the original matrix. The improvement is not significant since the role of features is limited to regularization; there is no direct participation in rating prediction. On Lastfm, their performances are even worse than PMF. The next three baselines, FM, RLFM, SIM, further improves the performance with similar improvement.

FCR-a is our proposed feature-centric method with averaged heuristic and performs close to FM because these methods involve features in matrix factorization and infer latent

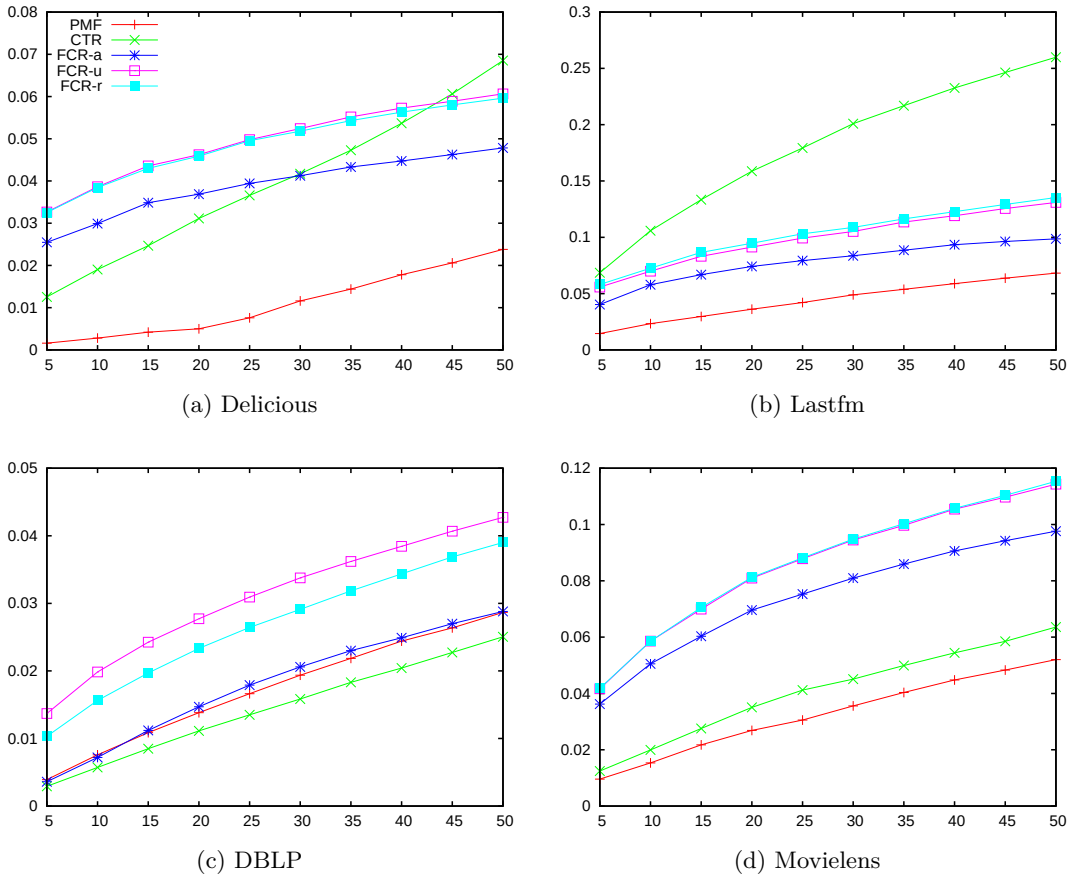
feature vectors for prediction. However, FM involves the pairwise interactions between latent item vectors and latent feature vectors, which may be complex and improper in real life applications; on the contrary, our model provides a simple and clean solution.

By incorporating the user-specific heuristic, FCR-u achieves significant improvements over FCR-a. This weighting scheme gives more trust to features that are more frequently selected by users. Through a regression model, FCR-r finds proper weighting for features and achieves better results compared to FCR-u.

Overall, FCR is the best performing method for all data sets, which verifies the effectiveness of the proposed feature-centric approach. Among the three integration strategies, FCR-r performs the best on Delicious and Lastfm, which suggests that the regression model yields better weighting for features. For example, the rating density is very sparse for DBLP and Delicious and matrix factorization on traditional user-item rating matrix works poorly. In the feature-centric FCR, this problem does not occur because the features of items act as a media for collaborative learning between users. For the DBLP data set, the users working in AI areas may focus on AAAI, IJCAI, ICML conferences (which are features of papers), so even the citation/publishing data is sparse, the interests of users are closely related through similar publishing venues. For the Movielens data set, the improvement is least mainly because user's interests are more diverse on movies, leading to less collaborative learning effect through features.

Recall. Figure 3.3 shows the recall@k performance of different methods. Since the codes of FM, RLFM and SIM output only RMSE/MAE, the required data are inaccessible to us for calculating recall, so we have to skip these methods in this study. PMF performs worst on all data sets. CTR learns the topic distribution from item's features, which indirectly groups the items with similar features. The improvement is obvious compared to PMF, except on DBLP. For DBLP, the topics learnt from features may be less meaningful since an author may have more diverse interests. For example, a music of Lady Gaga is always tagged with "Pop" or "Dance", but a researcher could publish papers in different conferences like SIGKDD, SIGMOD and ICML.

Except for Lastfm, FCR always performs best. A cross-examination with Table 3.3 suggests that the small RMSE/MAE on Lastfm does not yield a high recall. This is not surprising because RMSE considers only rated items in the testing set whereas recall ranks all items including unrated items. If many unrated items receive a high rating, which does not affect RMSE, rated items in the testing set will be pushed down in the ranking list, resulting in a low recall. This problem with recall was also noted in Cremonesi et



(a) Delicious (b) Lastfm

(c) DBLP (d) Movielens

Figure 3.3: Recall@k of four data sets. Vary k (x-axis)

al.'s work [14]. Among the three variations of FCR, FCR-u and FCR-r achieve the better performances than FCR-a. This suggests that assigning different weights to features boosts the items with user preferred features in the ranking list.

t-Test. To further verify the statistical significance of the improvement introduced by the regression model, we conducted the paired t-Test (2-tail) on FCR-a and FCR-r over 10 folds. As shown in Table 3.4, the t-Test results (p-values) are less than 0.01, which suggests that the improvement of FCR-r over FCR-a is statistically significant.

Table 3.4: Paired t-Test(2-tail) of FCR-a and FCR-r

t-Test	Delicious	Lastfm	DBLP	Movielens
RMSE	2.4×10^{-15}	1.5×10^{-16}	5.1×10^{-16}	1.4×10^{-4}
MAE	2.2×10^{-15}	1.6×10^{-17}	8.1×10^{-15}	1.3×10^{-7}

Representative features. We perform the 2-step feature selection for FCR-r and study the representative features. Figure 3.4 presents the RMSE results on Lastfm and

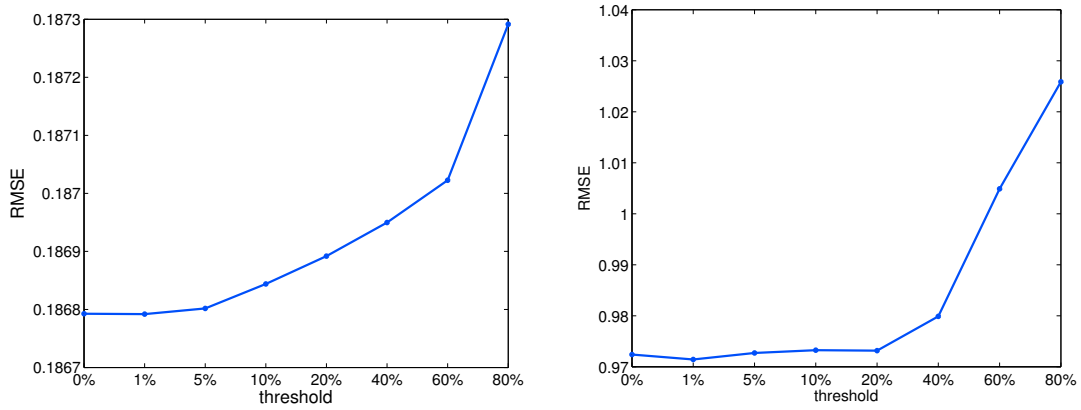


Figure 3.4: Feature selection for FCR-r on Lastfm (left) and MovieLens (right). y-axis represents RMSE and x-axis represents the percentage of the features with lower correlations removed

MovieLens when removing those features with lower correlations. We observe that the lowest 1% features have no contributions to the model since the RMSE is slightly improved without them. With more features removed, the RMSE is steady at first, indicating those features are less representative, i.e., lowest 5% on Lastfm and lowest 20% on Movielens. Then the RMSE goes up steeply when removing those features with higher correlations. Note that the results are still better than most baselines even if the lowest 80% of the features are removed. This also coincides with our thinking that the features with higher correlations are more representative. Figure 3.5 demonstrates the representative features on Lastfm with visualization tools.

3.4 Summary

In summary, the proposed feature-centric approach demonstrates superiority over item-centric approaches. This superiority is especially obvious for a sparse rating matrix in which case collaborative filtering on features is a much better option than collaborative filtering on items because feature ratings are denser than item ratings. Content-based filtering, i.e., SIM, extends items with content/features, and more recently, several works extend collaborative filtering (i.e., the latent factor model) to items with content and features, i.e., CTR and RLFM. The improvement is limited because features are used as auxiliary information such as a new regularization term in matrix factorization. Our feature-centric approach acknowledges the upmost importance of features in item preferences by allowing the features to play a central role from rating capturing to model building to rating pre-

Figure 3.5: Representative features on Lastfm

diction, which yields significant improvements. The studies also suggested that a model based regression is a viable weighting strategy for integrating feature ratings and finding representative features. Those representative features should be highly considered in real life recommendations.

Chapter 4

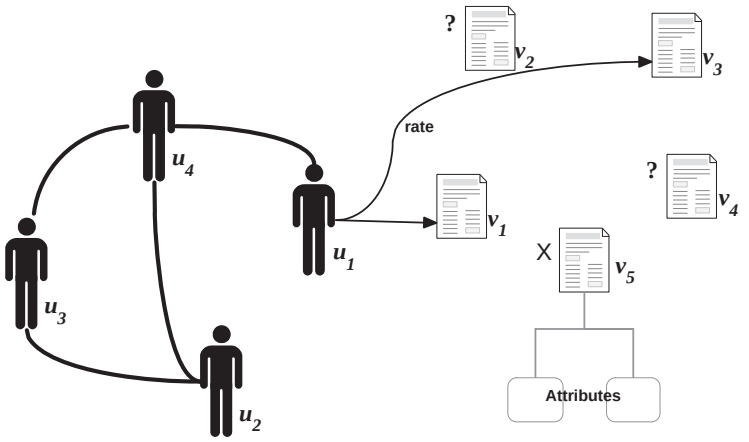
Recommendation by Blending Content and Attributes

This chapter focuses on the recommendation with heterogeneous information, in particular, the precise recommendation of scientific papers in academic networks where both items' content and attributes exist and have to be profoundly exploited. Different from conventional collaborative filtering cases with only a user-item rating matrix, we study the standard latent factor model and extend it to a heterogeneous one, which models the interaction of different kinds of information. This latent model is called "Content + Attributes", which incorporates latent topics and descriptive attributes using probabilistic matrix factorization and topic modeling to figure out the final recommendation results in heterogeneous scenarios. We conduct extensive experiments on the DBLP data set and the experimental results show that our proposed model outperforms the baseline methods.

4.1 Motivation

The content information added to the recommender systems would help improve the performance, which is demonstrated in the previous chapter with homogeneous information. In this chapter, we start to consider the case in the heterogeneous scenarios where usually more than one single type of information exists.

We mainly focuses on the recommendation in academic networks, which is far more complex than the conventional cases such as recommending movies or books. It is a heterogeneous scenario to recommend scientific articles in academic networks: each author has several co-authors and may write and cite some papers; this co-authorship forms social



User-item Utility Matrix

	v_1	v_2	v_3	v_4	v_5
u_1	√	?	√	?	×
u_2	×	×	√	?	√
u_3	√	×	√	?	?
u_4	√	√	?	?	?

Content and Attributes Information of Items

	CONTENT / WORDS					ATTRIBUTES						
	W_1	W_2	W_3	...	W_n	A_1	A_2	A_3	C_1	C_2 ...	C_k	*Year*
v_1	6	5	0	…	0	√			√			2008
v_2	0	0	8	…	6	√				√		2008
v_3	3	6	0	…	1		√				√	2010
v_4	5	7	1	…	0	√			√			2012
v_5	5	9	2	…	1			√			√	2010

Figure 4.1: Heterogeneous recommendation scenario: each item has content and attribute information. Each user marks the likes(√) and dislikes(×) for some items and the rest are unknown(?) in the rating matrix. √ in the information table means the item has this attribute

networks and such publishing or citation reflects implicit ratings for items; each paper has *content* (e.g., title and abstract in plain text) and *attributes* (e.g., author, venue, publish year), as shown in Fig. 4.1. However, conventional collaborative filtering methods cannot deal with the paper recommendation well if they still follow the pattern of friend's friend or item similarity. We have to recognize the diversity in the academic networks and admit that users prefer not all the papers of their co-authors. Moreover, without considering attribute information, the similarity-based methods may return less qualified papers. This is because, the attributes remain a good indicator of user's interests as the users may follow the big guy or prefer top conferences in research areas. This motivates us to develop a novel model to recommend scientific articles by taking advantage of all the heterogeneous information. Next, we present some discussions on how these heterogeneous information matter.

4.1.1 Discussions

According to the toy example shown in Fig. 4.1, there are 5 items binding with content and attributes (u and v represent the user and item in this section temporarily). Assume that there are two topics "database" and "data mining" and three authors. According to the content, suppose that v_1, v_3, v_4 and v_5 are in one topic (e.g., data mining) and v_2 is in the other (e.g., database). v_1, v_2 and v_4 are written by the same author A_1, v_3 and v_5 are written by A_2 and A_3 respectively.

Item v_1 and v_2 have the same attributes but different topics in content; Item v_1 and v_3 have similar content but different attributes. The underlying pattern which involves both content and attributes should be that u_1 prefers the "data mining" topic except that of author A_3, u_2 prefers the "data mining" topic except that of author A_1, u_4 prefers both topics from author A_1. If we consider content alone, it is difficult to explain why u_2 likes v_3 but dislikes v_1 although they share the same topic and why u_4 likes both v_1 and v_2 although they have different topics. If we consider attributes alone, it is difficult to explain why u_1 likes v_1 but dislikes v_2 although they are written by the same author. In fact, both content and attribute information should be collaborated to determine a user's interest.

One possible solution is to assign certain weights to content and attributes (i.e., w_1 and w_2), build one recommender using content and another recommender using attributes, and then integrate them according to different weights. However, this approach does not consider correlation of content and attributes into a unified model, and it is difficult to find right weights from separate recommenders. For example, the recommender using attributes would highly recommend v_2 to u_3, which lead to lower the value of w_2 to balance the final result. In that case, the recommender using content becomes dominant and would recommend v_5 to u_1 and v_1 to u_2 by mistake. The problem of contradicted results by two recommenders is hard to resolve.

Another solution is simply to combine content and attribute matrix together and treat each attribute value as a word. At this time, the effect of attributes may be overwhelmed by content because the number of attributes is much less than that of words. For instance, v_3 and v_5 should have been very similar in this case but u_1 likes v_3 and dislikes v_5.

Moreover, for out of matrix prediction, e.g., item v_4 has never been rated or a new user joins the network, traditional collaborative filtering methods cannot deal with this cold start problem, but we can address it through item's information and user's social network.

From above discussions, in order to make precise recommendation in academic networks, we propose a latent factor model especially incorporating the content and attribute information. We take both information into consideration as they are good indicators of user's preferences, and try to avoid the partial and biased recommendation. Our proposed model can automatically optimize the contribution of content and attributes to the final recommendation. The generalized recommendation model aims to overcome the drawbacks of former homogeneous recommendation methodologies.

4.1.2 Comparison with Related Work

Some works adopted side information such as social networks or item content to improve the recommendation. The works [44, 45, 69] incorporate social networks for social recommendation, based on the assumption that users should have close interests with their friends in the social network. Ma et al. [45] introduced the social regularization to conform user i to the friend f, through an individual based regularization term $\|u_i - u_f\|^2$. Yang et al. [69] proposed the friendship-interest propagation (FIP) mode that utilizes the inner product ($u_i^T u_f$) between two users i and f. As indicated in Shen et al.'s work [59], existing approaches have largely ignored the heterogeneity and diversity of the networks. It is not equivalent to say that preferences between friends should be similar even though they share certain interests.

Agarwal et al. [4] proposed fLDA to combine matrix factorization with topic modeling, which lets latent item vector and topic assignments, as well as latent user factor, contribute the rating prediction. Wang et al. [65] introduced collaborative topic regression (CTR) which captures item's content in latent space and derives the latent item vector through the topic proportion vector with Gaussian noise. Both models provide a good solution incorporating item's content in latent space, but cannot satisfy the requirements in dealing with the complex situations discussed in Section 4.1.1. The major improvement in this chapter's work is to further leverage the descriptive attributes into the latent factor model and automatically tune their contributions to the latent item vectors, not with fixed settings in CTR.

4.2 Model

The formulation of recommendation task in academic networks. The recommender system has several inputs: (1) I users and J items with the user-item utility matrix R, in

which each element $r_{ij} \in \{0, 1\}$ indicates user i's preference to item j. $r_{ij} = 1$ means the user rates the item (publish or cite it) and $r_{ij} = 0$ means the user does not rate the item (dislikes or unknown); (2) items' content and attributes, as shown in Figure 4.1. For each user, the task is to recommend scientific papers that are not rated by this user before. We assume that the latent topics and latent aspects for each item are obtained by topic modeling and we can represent users and items in the latent low-dimensional space of dimension D, with latent user vector $u_i \in \mathbb{R}^D$ and latent item vector $v_j \in \mathbb{R}^D$ through matrix factorization. The prediction r^*_{ij} represents that user i likes item j or not with inner product in their latent space $r^*_{ij} = u_i^T v_j$.

Observed ratings in the rating matrix are involved in a supervised approach to minimize the regularized squared error loss respect to $U = (u_i)_{i=1}^I$ and $V = (v_j)_{j=1}^J$:

$$min_{U,V} \sum_{i,j} \frac{\varepsilon_{ij}}{2}(r_{ij} - u_i^T v_j)^2 + \frac{\lambda_u}{2} \sum_i \|u_i\|^2 + \frac{\lambda_v}{2} \sum_j \|v_j\|^2 \tag{4.1}$$

where ε_{ij} is a binary indicator that is equal to 1 if user i rated item j and equal to 0 otherwise.

In PMF [56], the matrix factorization is generalized as a probabilistic model, where latent user vector $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_D)$, latent item vector $v_j \sim \mathcal{N}(0, \lambda_v^{-1} I_D)$ and user-item rating $r_{ij} \sim \mathcal{N}(u_i^T v_j, \varepsilon_{ij}^{-1})$. A local minimum of Eq. (4.1) can be achieved by applying gradient descent algorithm in U and V. The final results can be used to predict user i's preference on item j by r^*_{ij}. The disadvantage of PMF is that it cannot deal with the cold start problem.

In CTR [65], the latent topic vector θ_j is incorporated into PMF framework and the latent item vector is further confined by setting $v_j \sim \mathcal{N}(\theta_j, \lambda_v^{-1} I_D)$. CTR can address the cold start problem of new items, but is not fit for the feature information on items.

In this chapter's work, we focus on the heterogeneity of content and attribute information of items and build a recommendation system to combine these two sides into a unified model (Section 4.2.3). We call it "unified" because both content and attributes are integrated into a single model. We first propose a probabilistic topic model to process the item information, then introduce two naive solutions to import attribute information and later propose our unified model to achieve better performances.

4.2.1 Item Information Processing

Each item has its unstructured information (i.e., content) and structured information (i.e., attributes). Both of them are useful in recommending items to users as discussed in the introduction. Usually, every item may contain hundreds of words in content and several attributes, and the item set may overall contain tens of thousands of words in vocabulary and attribute values. So we have to seek for an unsupervised method to reduce them to a low dimension.

For content information, the topic modeling methods such as LDA [7] can be used to achieve the goal of dimension reduction. The intuition behind LDA is that documents exhibit multiple topics which are represented by distributions over words. For paper data set, the content of each item consists of title and abstract, which is a probabilistic mixture of latent topics. If we have K topics, we can adopt LDA to learn a latent topic vector in a K-dimensional space for each item.

An attribute can be numerical (i.e., year) or categorical (i.e., venue and author). The values of categorical attributes can be treated as words in content, while the values of numerical attributes need to be processed in order to avoid the sparse problem. For example, the attribute "year" in a paper data set can be ranged from late 20th century to now and each paper only has one value for this attribute. If we take a separated view on it, the connection of papers sharing the same topic but publishing in adjacent years would be concealed. It is also probable that the number of published papers is skewed on "year", much more in some years while much less in other years. So we need to partition the numerical attributes to ensure the effectiveness. The values in the attribute "year" are divided into four classes: (2005-now), (2000-2004), (1995-1999) and (before 1995) from newly published to long before. After such transforming, the attribute information has the same format as content information.

4.2.2 First Cut Solutions

This section introduces two first cut solutions to deal with the additional attribute information in the recommendation systems. Both methods leverage the attributes of items to complement the only content based method.

The first method is **weighted attributes method**, which builds two different recommenders on content and attributes separately. Based on the content vector and attribute vector learned by LDA, weighted attributes method applies CTR to obtain two predicted

ratings from content part and attribute part, denoted as $r_{ij}^{*(c)}$ and $r_{ij}^{*(a)}$. A weight $w \in [0, 1]$ is chosen to adjust the contribution of content and attributes, so the final rating of user i's preference on item j is:

$$r_{ij}^{*} = (1 - w)r_{ij}^{*(c)} + wr_{ij}^{*(a)} \tag{4.2}$$

The second method is **overwhelming attributes method**, which treats the attribute values as single word in content. This method merges the attribute values and words, and then adopts LDA to obtain the latent topic vectors mixed with attribute information. CTR is applied based on these latent vectors to figure out the final ratings. Compared to the words in content, the number of attribute values is very little, so the attribute values are largely overwhelmed by words, leading to a limited improvement.

4.2.3 Content + Attributes Model

This section introduces our Content + Attributes model to address the heterogeneity issue in recommendation systems. The model has loose coupling and integrality on content and attributes, as well as self-adaptation to latent item vectors. This model can offer more precious recommendations to users by considering both content and attributes.

After item information processing, we adopt LDA to reduce both content and attributes into a low dimensional space and apply them into the matrix factorization. Note that for item j, we have obtained the K-dimensional latent topic vectors θ_j for content and the L-dimensional latent aspect vectors γ_j for attributes, we should extend these vectors into a combined latent vector in D-dimensional space ($D = K + L$). In our model, the combined latent vector ϑ_j for item j is defined as follows:

$$\vartheta_j =< \theta_j(1), ..., \theta_j(K), \gamma_j(1), ..., \gamma_j(L) > \tag{4.3}$$

where the first K elements keep the topic information for content and the last L elements keep the aspect information for attributes.

After the above transformations, we now describe the generative process for observed ratings of the utility matrix in Content + Attributes model:

1. For each user i, draw latent user vector $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_D)$;

2. For each item j, draw latent item offset $\epsilon_j \sim \mathcal{N}(0, \lambda_v^{-1} I_D)$, and set latent item vector $v_j = \epsilon_j + \tau\vartheta_j$;

3. For each user-item (i, j), draw the rating $r_{ij} \sim \mathcal{N}(u_i^T v_j, 1)$.

Note that the latent item vector $v_j = \epsilon_j + \tau\vartheta_j$ indicates v_j should be close to scaled ϑ_j, and τ is the adjustment factor to adapt the importance of ϑ_j to v_j

So after combining adjusted latent topic vector for content and latent aspect vector for attribute, we modify Eq. (4.1) to follow our generative process and set our goal of object function that is to minimize E given such variables and parameters as follows:

$$E = \sum_{i,j} \frac{\varepsilon_{ij}}{2}(r_{ij} - u_i^T v_j)^2 + \frac{\lambda_u}{2}\sum_i u_i^T u_i + \frac{\lambda_v}{2}\sum_j (v_j - \tau\vartheta_j)^T(v_j - \tau\vartheta_j) \tag{4.4}$$

A local minimum can be achieved by iteratively applying gradient descent method. In each iteration, first take the gradient of E respect to variable u_i, v_j and τ:

$$\frac{\partial E}{\partial u_i} = \lambda_u u_i - \sum_j (r_{ij} - u_i^T v_j) v_j \tag{4.5}$$

$$\frac{\partial E}{\partial v_j} = \lambda_v v_j - \lambda_v \tau\vartheta_j - \sum_i (r_{ij} - u_i^T v_j) u_i \tag{4.6}$$

$$\frac{\partial E}{\partial \tau} = -\lambda_v \sum_j \vartheta_j^T (v_j - \tau\vartheta_j) \tag{4.7}$$

Then update each variable by taking steps proportional to the negative of the gradient based on recent values:

$$u_i^{t+1} = u_i^t - \eta\frac{\partial E}{\partial u_i^t} \quad v_j^{t+1} = v_j^t - \eta\frac{\partial E}{\partial v_j^t} \quad \tau^{t+1} = \tau^t - \eta\frac{\partial E}{\partial \tau^t} \tag{4.8}$$

where η is a parameter called learning rate and u_i^t, v_j^t, τ^t stand for the value of u_i, v_j and τ at iteration t.

We can obtain the final results after enough iterations until the above equations (4.8) reach the convergence. Similar to the previous methods, we can use the latent user vector u_i and latent item vector v_j to predict user i's rating value on item j: $r_{ij}^* = u_i^T v_j$.

The Content + Attributes model introduces adjustment factor τ to automatically optimize the contribution of content and attributes to latent item vectors, because the closeness of ϑ_j to u_j is unknown. Unlike CTR that fixes $u_j \approx \theta_j$, we use τ to control the scale of vectors and achieve better predictions.

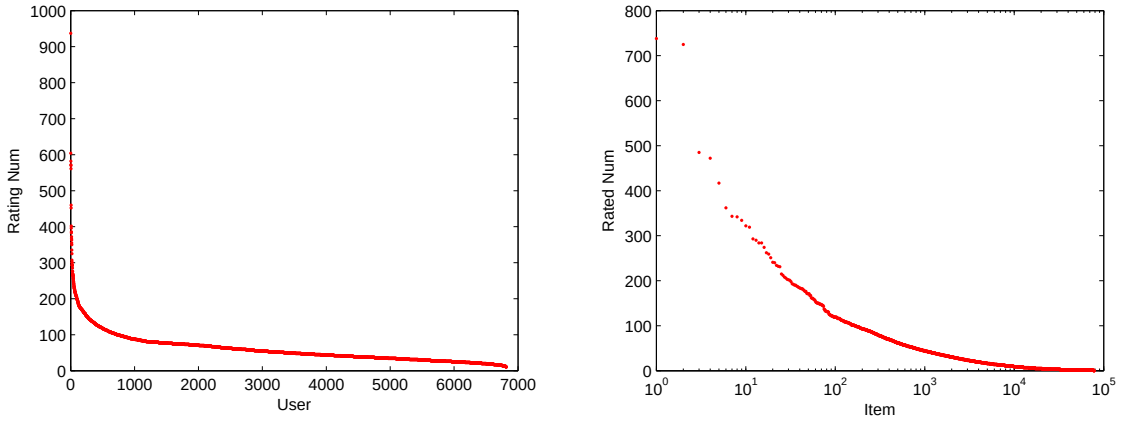


Figure 4.2: Rating distributions of the data set. The left subfigure shows the number of ratings by each user and the right subfigures shows the number of ratings on each item (logistic scaled), all descent sorted.

4.3 Experimental Evaluation

4.3.1 Data Sets

The original data set[1] is from DBLP, containing over 1.5 million item (papers) and around 700 thousand users (authors). We preprocess this data set to select those items with complete content (i.e., title and abstract) and attributes (i.e., author, venue and year), to remove the users with fewer than 10 papers. The final data set consists of 6815 users and 78475 items with 436704 user-item ratings. The rating distribution of the processed data set is exhibited in Fig. 4.2, which is following the power law distribution. We randomly select 10% percent of ratings from 1600 users who have more than 80 ratings, i.e., 26328 user-item ratings (approximately 6% of total ratings) as testing set withheld in model learning and use others as training set for learning the latent vectors. We conduct validations on the testing set and report the averaged results of 5 repeated experiments to measure the performance of different methods.

4.3.2 Evaluated Methods

Followings are five methods compared in the experiments all with the same learning rate $\eta = 0.001$:

[1] http://arnetminer.org/citation

Probabilistic matrix factorization (denoted PMF): This is the first baseline method only adopting matrix factorization on utility matrix. PMF is widely used in the collaborative filtering community. Following Salakhutdinov et al.'s work [56], we set the parameters $\lambda_u = \lambda_v = 0.01$.

Collaborative topic regression (denoted CTR): This is the second baseline method incorporating topic model into matrix factorization. Following Wang et al.'s work [65], we set the parameters $\lambda_u = \lambda_v = 0.01, \alpha = \frac{50}{K}$ and $\beta = 0.01$.

Weighted attributes method (denoted WAM): This is the third baseline algorithm mentioned in section 4.2.2 which builds two recommenders on content and attributes respectively, and take different weight on attributes. The parameter settings are the same as CTR, and weight w varies from 0.1 to 0.5. Note that WAM is identical to CTR when $w = 0$.

Overwhelming attributes method (denoted OAM): This is the fourth baseline algorithm mentioned in section 4.2.2 combining words and attribute values, which neglects the heterogeneity of content and attributes. The parameter settings are the same as CTR.

Content + Attributes model (denoted CAT): This is our model proposed in section 4.2.3 with adapted adjustment factor τ. In CAT, τ is initialized as 1. Other parameter settings are the same as CTR.

4.3.3 Evaluation Metrics

We adopt the following metrics to evaluate the performance of different methods.

RMSE (root mean squared error) quantifies the difference between rating values implied by a recommender and the true values in the testing set. This metric is defined as follows: $RMSE = \sqrt{\frac{1}{N}\sum_{i,j}(r_{ij} - r^*_{ij})^2}$ where r_{ij} is the true rating value, r^*_{ij} is the predicted rating value and N is the number of ratings in the testing set.

Coverage indicates the retrieval ratio from total ratings in the testing set using a transforming function $\sigma(x)$, which classifies the estimated rating values into "likes" and "dislikes" by a threshold: $Coverage = 1 - \frac{1}{N}\sum_{i,j}(r_{ij} - \sigma(r^*_{ij}))$ where $\sigma(x) = \begin{cases} 1 & \text{if } x \geq 0.5 \\ 0 & \text{if } x < 0.5 \end{cases}$.

Recall@k quantifies the fraction of rated items that are in the top-k of the ranking list sorted by their estimated ratings from among all rated items in the test set. For each user i: $Recall@k = \frac{|N(k;i)|}{|N(i)|}$, where $|\cdot|$ denotes the number of elements in a set, $N(i)$ is the set of items rated by i in the testing set and $N(k;i)$ is the subset of $N(i)$ contained in the top-k

list of all items sorted by their estimated ratings. The metric is designed in the condition that a high coverage may be biased due to excessive estimated rating values.

We report average RMSE, coverage and recall over the whole testing set.

4.3.4 Experimental Results

We vary the number of dimension $D \in \{10, 20, 40, 80\}$, when the dimension is low or high, the matrix factorization cannot converge well on RMSE for training set due to less features or overfitting problem. So we report the results of the best performance at $D = 20$. The mean rating values of the predicted utility matrix is 0.2-0.3, which would not lead to a biased coverage.

Table 4.1: RMSE and coverage results of different methods. Lower values on RMSE and higher values on coverage are better.

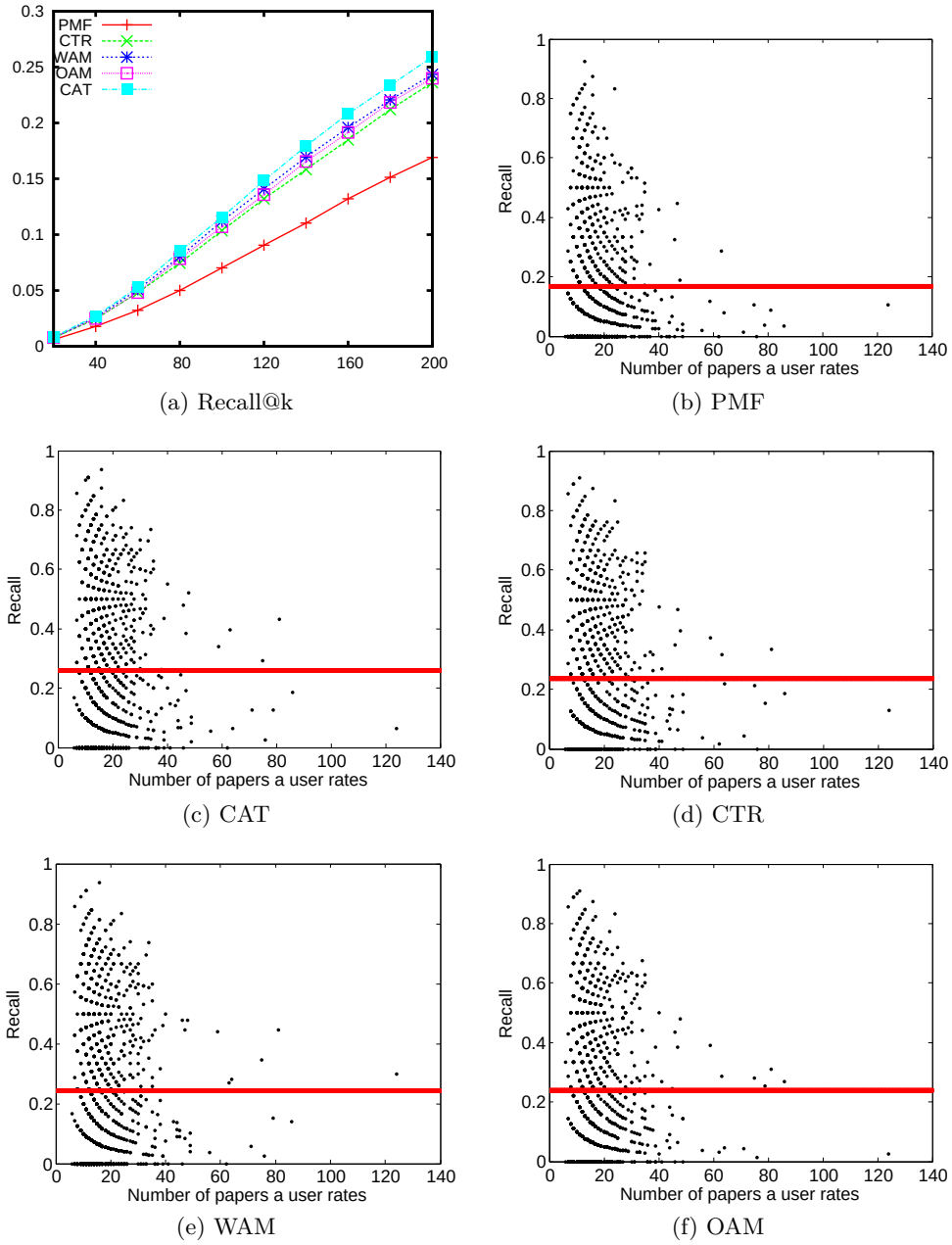
Models	RMSE	Coverage
PMF	0.3077 ± 0.0019	0.8810 ± 0.0034
CTR	0.2775 ± 0.0020	0.9090 ± 0.0042
WAM	0.2691 ± 0.0014	0.9158 ± 0.0033
OAM	0.2700 ± 0.0022	0.9157 ± 0.0031
CAT	$\mathbf{0.2580} \pm 0.0017$	$\mathbf{0.9237} \pm 0.0038$

The experiment results for RMSE and coverage are shown in Table 4.1, with the best performers for each metric highlighted in bold face. Obviously, the conventional PMF without considering any auxiliary information performs worst. This is because, when this matrix is sparse, the ratings as only the similarity information among users are hardly enough to make accurate recommendation. Two naive solutions combining attribute information work slightly better than CTR, which only considers the content information. All these methods takes additional information beyond the rating matrix to achieve better performances. However, as discussed in Section 4.1.1, these solutions lost specific patterns represented by attributes. In contrast, CAT incorporates both content and attributes into a unified model which automatically adapts the contribution to the final latent item vectors. As pointed in Koren's work [29], achievable RMSE values lie in a quite compressed range and small improvements in RMSE terms can have a significant impact on the quality of the top few presented recommendations. Our CAT model achieves a 16% and 7% improvement over PMF and CTR on RMSE, which verifies the effectiveness of further incorporating the attribute information.

The experiment results for recall are shown in Fig. 4.3, with an overall subfigure and five individual level subfigures (each dot represents a user and the red line reports the average) for each method. We observe that CTR, WAM and OAM perform very close, due to that both WAM and OAM are minor variations of CTR. With no surprise, PMF without any side information still performs worst on recall. CAT model performs best on recall, with a 53% and 10% improvement over PMF and CTR, as well as more dense points above the average line. The advantage over CTR is from that CAT further considers the attribute information. In other words, more side information achieves better recommendation performance.

4.4 Summary

This chapter mainly discussed a heterogeneous recommendation scenario in academic networks where abundant information exist. We proposed a unified model to incorporate item's content and attributes for better prediction. The experiments on real life data set demonstrated its effectiveness compared with baseline methods. The studies suggested that the traditional approach (PMF) on the rating matrix only is not a good predictor for scientific paper recommendation. Incorporating more side information such as content and attributes improve the recommendation performance.



(a) Recall@k (b) PMF

(c) CAT (d) CTR

(e) WAM (f) OAM

Figure 4.3: Recall performance of different methods. The left top subfigure shows overall performance of different methods while varying k. The rest subfigures exhibit the individual level of performance each method while fixing $k = 200$.

Chapter 5

Temporal Matrix Factorization

User preferences change over time and capturing such changes are essential for developing accurate recommender systems. Despite its importance, only a few works in collaborative filtering have addressed this issue. In this chapter, we consider evolving preferences and we model user dynamics by introducing and learning a transition matrix for each user's latent vectors between consecutive time windows. Intuitively, the transition matrix for a user summarizes the time-invariant pattern of the evolution for the user. We first extend the conventional probabilistic matrix factorization and then improve upon this solution through its fully Bayesian model. These solutions take advantage of the model complexity and scalability of conventional Bayesian matrix factorization, yet adapt dynamically to user's evolving preferences. We evaluate the effectiveness of these solutions through empirical studies on six large-scale real life data sets.

5.1 Motivation

A founding principle of collaborative filtering is that if two users share similar interests on some items, they also likely share similar interests on other items. This simple preference propagation model from one item to another is challenged when user behaviors change over time. For example, a user rated cartoon movies at earlier years, then action movies some years later, and romantic movies more recently, and for another user, such a path of changes may be different. Another changing scenario voiced in Mcauley et al.'s work [48] is that user's expertise level may upgrade from amateur to connoisseur over time. In both cases, the exact change in preferences depends on the user because such changes are a reflection of user's life experiences. As a result, even though two users rated cartoon movies

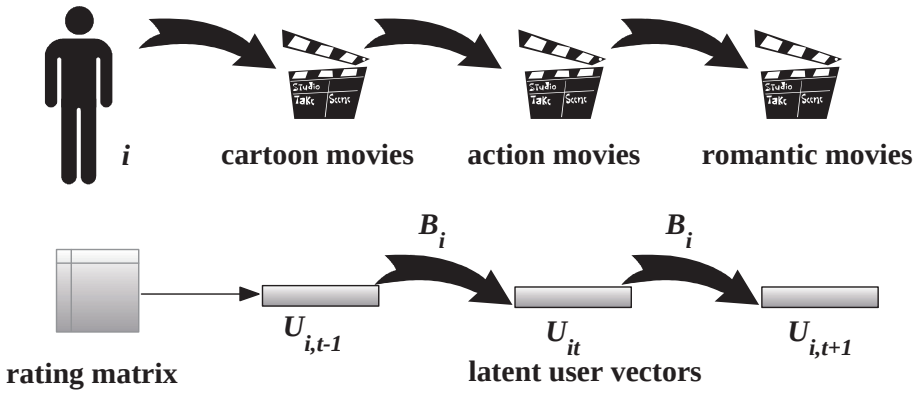


Figure 5.1: Evolution of user i's preferences

similarly some years ago, they may rate the same movie differently or like very different kinds of movies at a later time. In this situation, it is largely irrelevant to predict current preferences based on the similarity of preferences a few years ago.

On the other hand, it is possible that, for other users, the similarity of past preferences remains a good indicator of the similarity of current preferences. Therefore, it does not work to simply partition the rating data by time and learn a model using the data in each partition. This method not only misses the dependence of preferences for some users across time windows, but also accelerates the well known data sparsity problem. In addition, as pointed out in Koren's work [30], temporal dynamics in recommendation are different from concept drift studied in machine learning [64]. In the case of recommendation, the evolution of each user's behaviors is determined by a potentially different set of factors, and there is no single global concept to be learnt.

5.1.1 Contributions

We assume that user preferences evolve gradually: the preference of user i at time t depends on the preference of the user at time $t-1$. Such temporal dependence is the basic idea of many statistical dynamic approaches such as hidden Markov model [21] and Kalman filter [27]. We model the temporal dependence for each user i through a $D \times D$ *transition matrix* B_i, where D is the dimensionality in the latent space: the latent vector u_{it} of user i at time t is a linear combination, specified by the rows of B_i, of the user's latent vector $u_{i,t-1}$ at time $t-1$, that is, u_{it} has the mean $B_i u_{i,t-1}$. This relationship is illustrated in Fig. 5.1. Intuitively, B_i captures the time-invariant pattern of the evolution for user i. For example,

if user i increasingly prefers the movies directed by James Cameron over time, the entry (j, j) in B_i will have a value larger than 1, assuming that the jth latent factor corresponds to James Cameron. The conventional static model can be treated as the special case of having the identity transition matrix B_i. Learning the transition matrices that help predict unknown ratings in the next time point is the main task in this chapter.

The contributions of this chapter are as follows: (1) We propose temporal probabilistic matrix factorization (TMF) and its fully Bayesian treatment model (BTMF), by incorporating a transition matrix into the conventional matrix factorization methods. This approach provides a clean solution by capturing temporal dynamics through the transition matrix and leveraging the principled Bayesian matrix factorization methodology. (2) We present gradient descent and MCMC to infer the parameters and transition matrices of these two models. (3) We conduct extensive experiments on six large-scale data sets. The empirical results demonstrate appealing improvements over the conventional matrix factorization and the state-of-the-art time-aware methods.

Although predicting the future rating is the focus in this chapter, the learnt transition matrices have other applications. For example, since the transition matrix for a user captures the time-invariant aspect of user's evolution patterns, we can group users using learnt transition matrices as the features and develop a customized recommendation strategy for each group. A further investigation of this topic is beyond the scope of this chapter.

5.1.2 Comparison with Related Work

Ding et al. [15] uses a time weighting scheme for a similarity based collaborative filtering approach, which decays the similarities to previously rated items as time difference increases at the prediction time. As discussed above, the time decay scheme may miss a long-term effect for some users. Our method learns the temporal dependence from the whole set of ratings without limiting any part of the data. Xiong et al. [68] proposed the user-item-time tensor factorization to model temporal effects. In a recommender system, the time dimension is a local effect and should not be compared across all (user,item) pairs [67]. Xiang et al. [67] used a graph connecting users, items, and sessions to model users' long-term preferences and short-term preferences. Sun et al. [61] modeled temporal effects using Kalman filtering with a transition process parameter for each user similar to our transition matrix, but their transition parameters are time-dependent and user-supplied, and their model was evaluated only on generated data. Clearly, specifying such parameters for all

users at all time points is impractical. In contrast, our transition matrices are time-invariant and are learnt automatically by the model from observed data. Chua et al. [13] further extended Sun et al.'s work [61] in the scenario of temporal adoption.

Our work is most related to Koren's work [30], which presented the first temporal model for the matrix factorization approach by introducing time-variant biases for each user and each item. The time-variant biases are the differences at each time window, but do not summarize the underlying pattern for such differences. This approach works for prediction only if two windows share similar biases. In contrast, our time-invariant transition matrices capture properties that are independent of time, and thus, can be used for prediction in a new time window. Beyond prediction, time-invariant properties also helps understand the mechanism that underpins the temporal dynamics.

5.2 Temporal Probabilistic Matrix Factorization

We present the temporal probabilistic matrix factorization (TMF) to model user temporal dynamics. We assume that time is represented by a series of consecutive time windows. Matrix factorization models map both users and items to a joint latent factor space of a low dimensionality D, such that ratings are modeled as inner products in that space. Suppose we have M items, N users, S time windows, and rating values from 1 to K. Let r_{ijt} denote the rating of user i for item j at time window t, $U \in \mathbb{R}^{D \times N \times S}$ and $V \in \mathbb{R}^{D \times M \times S}$ denote latent user and latent item (factor) hypermatrices, with column vectors u_{it} and v_{jt} representing user-specific and item-specific latent (factor) vectors, which describe user's interests and item's features, at time window t, respectively.

5.2.1 Introducing Transition Matrix

We assume that there is a temporal dependence between the latent user vectors u_{it} and $u_{i,t-1}$ and we model this dependency by a transition hypermatrix $B \in \mathbb{R}^{N \times D \times D}$ for all users. In particular, B_i, the $D \times D$ transition matrix for user i, models the transition of user i's preferences in the *latent* space from the previous time window to the next; so we expect u_{it} to have the mean $B_i u_{i,t-1}$. In plain English, this says that for each user i, the jth latent factor in the next time window is a linear combination, as specified by the jth row of B_i, of the latent factors in the previous time window. As each latent factor captures some intrinsic feature of items (e.g., movie's genre, movie's director, etc.), B_i captures the

time-invariant aspect of the user i's evolution in this intrinsic feature space. It makes sense not to model this dependency for items since item's features are stable and less correlated.

For example, consider the $D = 2$ latent space. The identity transition matrix $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ represents a stable latent user vector that does not change much over time; the transition matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ represents the alternating changing pattern between two latent user vectors (a, b) and (b, a); the transition matrix $\begin{bmatrix} 1.1 & 0 \\ 0 & 1 \end{bmatrix}$ represents a gradual shift pattern toward the first factor. In a higher dimensional latent space, a different subspace could undergo a different pattern, therefore, several patterns could occur at the same time.

At the current time t, we learn the latent user vector u_{it} and latent item vector v_{jt} for each user i and each item j, as well as the transition matrix B_i from *all* the ratings collected up to time t. With the learnt parameters, we can predict the rating of the user i on the item j at a future time window by the rule $r^*_{ij} = (B_i u_{it})^T v_{jt}$. The model performance is measured by the *root mean squared error (RMSE)* on the testing set $\{r_{ij}\}$:

$$RMSE = \sqrt{\frac{\sum_{i,j}(r_{ij} - r^*_{ij})^2}{n}} \tag{5.1}$$

where n is the number of ratings in the testing set. Using all the ratings collected up to the current time t helps capture the long-term effect discussed in Section 1 while modeling temporal changes through transition matrices. As the current time advances to the time $t + 1$, the above learning process is repeated on all the ratings collected up to time $t + 1$. The choice of the granularity of time dictates the trade-off between the freshness of updates and the efficiency of learning.

5.2.2 Modeling

We develop a temporal probabilistic model with Gaussian observation noises to learn the parameters U, V, and B. This is done by extending the conventional probabilistic matrix factorization (PMF) model [56] with the transition hypermatrix B. The conditional distribution over the observed ratings R is

$$p(R|U, V, \sigma) = \prod_{t=1}^{S} \prod_{i=1}^{N} \prod_{j=1}^{M} [\mathcal{N}(r_{ijt}|u_{it}^T v_{jt}, \sigma^2)]^{\varepsilon_{ijt}} \tag{5.2}$$

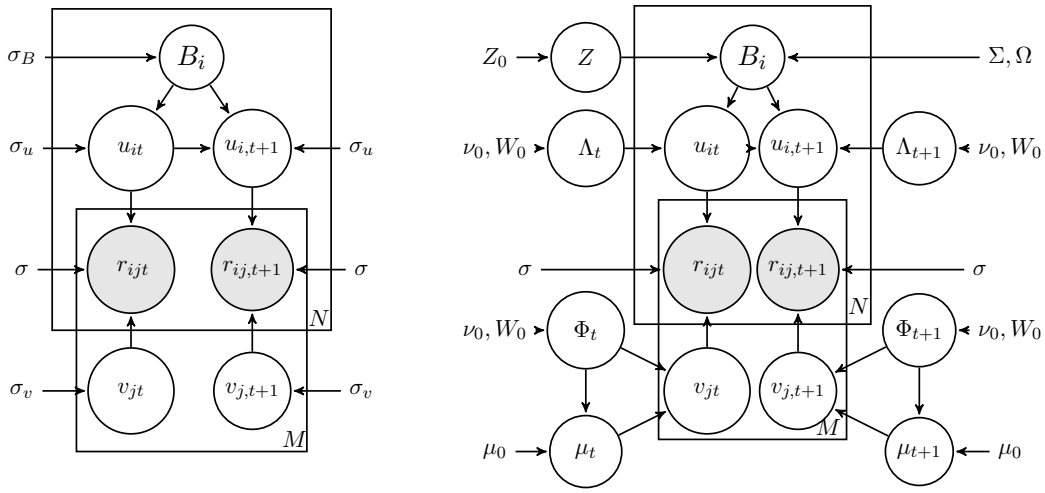


Figure 5.2: Graphical representations of TMF (left) and BTMF (right), with parameters and hyperparameters of time window t and $t+1$ shown only

where $\mathcal{N}(x|\mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and ε_{ijt} is the indicator variable that is equal to 1 if user i rated item j at time window t and equal to 0 otherwise.

As in Salakhutdinov et al.'s work [56], we place zero-mean spherical Gaussian priors on latent item vectors in V:

$$p(V|\sigma_v) = \prod_{t=1}^{S}\prod_{j=1}^{M} p(v_{jt}|\sigma_v) = \prod_{t=1}^{S}\prod_{j=1}^{M} \mathcal{N}(v_{jt}|\mathbf{0}, \sigma_v^2\mathbf{I}) \tag{5.3}$$

To model latent user vectors in U, we place Gaussian priors with mean $B_i u_{i,t-1}$ on latent user vectors u_{it} to model the temporal dependence of time window t on time window $t-1$:

$$p(u_{it}|B_i, u_{i,t-1}, \sigma_u) = \mathcal{N}(u_{it}|B_i u_{i,t-1}, \sigma_u^2\mathbf{I}) \tag{5.4}$$

where $p(u_{i1}|B_i, u_{i0}, \sigma_u) = \mathcal{N}(u_{i1}|\mathbf{0}, \sigma_u^2\mathbf{I})$ by defining $u_{i0} = \mathbf{0}$. Integrating out variables i and t gives:

$$\begin{aligned} p(U|B, \sigma_u) &= \prod_{t=1}^{S}\prod_{i=1}^{N} p(u_{it}|B_i, u_{i,t-1}, \sigma_u) \\ &= \prod_{t=1}^{S}\prod_{i=1}^{N} \mathcal{N}(u_{it}|B_i u_{i,t-1}, \sigma_U^2\mathbf{I}) \end{aligned} \tag{5.5}$$

The latent transition matrix B_i is placed with matrix-variate normal distribution $\mathcal{MN}(Z, \Sigma, \Omega)$ where Z is a matrix containing the expectation of each element of B_i, and Σ, Ω are two co-

variance matrices. In this case, we set $Z = \mathbf{I}$, i.e., a $D \times D$ identity matrix, and $\Sigma = \Omega = \sigma_B \mathbf{I}$:

$$p(B|\sigma_B) = \prod_{i=1}^{N} \mathcal{MN}(B_i|\mathbf{I}, \sigma_B \mathbf{I}, \sigma_B \mathbf{I}) \tag{5.6}$$

5.2.3 Inference

Following Eq. (5.2-5.6) and the graphical representation of TMF shown in Fig. 5.2 , the posterior distribution over the user and item vectors is given by

$$\begin{aligned} &p(U, V, B|R, \sigma, \sigma_v, \sigma_u, \sigma_B) \\ \propto &p(U|B, \sigma_u) p(V|\sigma_v) p(B|\sigma_B) p(R|U, V, \sigma) \end{aligned} \tag{5.7}$$

Our goal is to find the values of u_{it}, v_{jt}, and B_i that maximize the log-posterior of Eq. (5.7), which is equivalent to minimizing the sum-of-squared-errors objective function with quadratic regularization terms:

$$\begin{aligned} &\frac{1}{2} \sum_{t=1}^{S} \sum_{i=1}^{N} \sum_{j=1}^{M} \varepsilon_{ijt} (r_{ijt} - u_{it}^T v_{jt})^2 + \frac{\lambda_v}{2} \sum_{t=1}^{S} \sum_{j=1}^{M} \|v_{jt}\|_{Fro}^2 \\ &+ \frac{\lambda_u}{2} \sum_{t=1}^{S} \sum_{i=1}^{N} \|u_{it} - B_i u_{i,t-1}\|_{Fro}^2 + \frac{\lambda_B}{2} \sum_{i=1}^{N} \|B_i - \mathbf{I}\|_{Fro}^2 \end{aligned} \tag{5.8}$$

where $\lambda_u = \sigma^2/\sigma_u^2$, $\lambda_v = \sigma^2/\sigma_v^2$, $\lambda_B = \sigma^2/\sigma_B^2$ and $\|\cdot\|_{Fro}^2$ denotes the Frobenius norm. We adopt gradient descent with learning rate η in U, V and B to find the local minimum of the objective function in Eq. (5.8).

5.3 The Fully Bayesian Model (BTMF)

One drawback of TMF is that it is hard to search appropriate values of the hyperparameters $\sigma, \sigma_u, \sigma_v, \sigma_B$ to control the model complexity. A possible solution is to integrates out all model parameters U, V, B and hyperparameters $\sigma, \sigma_u, \sigma_v, \sigma_B$ to achieve the *predictive distribution* given observed data. In this section, we extend TMF to a fully Bayesian treatment called BTMF, in which both parameters and hyperparameters are sampled from the predictive distribution through the MCMC method. Though the mathematical development is a bit involved, the spirit of the extension is essentially the same as extending the probabilistic matrix factorization to its fully Bayesian treatment [55].

5.3.1 Modeling

BTMF introduces priors for the hyperparameters to control the model complexity, as shown in Fig. 5.2. Instead of Eq. (5.3) with fixed settings, BTMF models two hyperparameters, the mean vector μ_t and the precision matrix Φ_t, for each latent item vector v_{jt}, as in Salakhutdinov et al.'s work [55]; the prior distribution is assumed to be Gaussian:

$$p(v_{jt}|\mu_t, \Phi_t) = \mathcal{N}(v_{jt}|\mu_t, \Phi_t^{-1}) \tag{5.9}$$

And we place Gaussian-Wishart priors on μ_t, Φ_t:

$$\begin{aligned} &p(\mu_t, \Phi_t|\mu_0, \beta_0, W_0, \nu_0) \\ &= p(\mu_t|\Phi_t, \mu_0, \beta_0)p(\Phi_t|W_0, \nu_0) \\ &= \mathcal{N}(\mu_t|\mu_0, (\beta_0\Phi_t)^{-1})\mathcal{W}(\Phi_t|W_0, \nu_0) \end{aligned} \tag{5.10}$$

Here $\mathcal{W}$ is the Wishart distribution with ν_0 degrees of freedom and a $D \times D$ scale matrix W_0:

$$\mathcal{W}(\Lambda|W_0, \nu_0) = \frac{1}{C}|\Lambda|^{(\nu_0-D-1)/2}\exp(-\frac{1}{2}\mathrm{Tr}(W_0^{-1}\Lambda))$$

For each latent user vector u_{it}, the mean vector is given by $B_i u_{i,t-1}$ and we place Wishart priors on the user hyperparameter Λ_t:

$$p(u_{it}|B_i, u_{i,t-1}, \Lambda_t) = \mathcal{N}(u_{it}|B_i u_{i,t-1}, \Lambda_t^{-1}) \tag{5.11}$$

$$p(\Lambda_t|W_0, \nu_0) = p(\Lambda_t|W_0, \nu_0) = \mathcal{W}(\Lambda_t|W_0, \nu_0) \tag{5.12}$$

For each transition matrix B_i, there are three hyperparameters, i.e., the mean matrix Z and two covariance matrices Σ, Ω; contrary to Eq. (5.6), the prior distribution is assumed to be matrix normal:

$$p(B_i|Z, \Sigma, \Omega) = \mathcal{MN}(B_i|Z, \Sigma, \Omega) \tag{5.13}$$

To be simplified, we place no priors on Σ and Ω (Indeed, the priors for variance matrices of matrix normal distribution are hyper inverse Wishart distributions but they have little effects). We place the prior Z_0 to control the expectation matrix Z and set $\Sigma = \Omega = \mathbf{I}$.

For the sake of convenience, we define the hyperparameters $\Theta_U = \{\Lambda_{t=1...S}\}$, $\Theta_V = \{\mu_{t=1...S}, \Phi_{t=1...S}\}$ and $\Theta_B = \{Z\}$ controlled by priors $\Theta_0 = \{\mu_0, \nu_0, \beta_0, W_0, Z_0\}$. The predictive distribution of the rating value r_{ij}^* for user i and item j at future time window can

be obtained by marginalization:

$$\begin{aligned} & p(r_{ij}^*|R,\Theta_0) \\ = & \iint p(r_{ij}^*|u_{it},v_{jt},B_i)p(U,V,B|R,\Theta_U,\Theta_V,\Theta_B) \\ & p(\Theta_U,\Theta_V,\Theta_B|\Theta_0)\mathrm{d}\{U,V,B\}\mathrm{d}\{\Theta_U,\Theta_V,\Theta_B\} \end{aligned} \tag{5.14}$$

The exact evaluation of this predictive distribution is analytically intractable due to the complexity of the posterior. MCMC-based methods [49] use the Monte Carlo approximation to the predictive distribution given by

$$p(r_{ij}^*|R,\Theta_0) \approx \frac{1}{\vartheta}\sum_{\kappa=1}^{\vartheta} p(r_{ij}^* \mid u_{it}^\kappa, v_{jt}^\kappa, B_i^\kappa) \tag{5.15}$$

where ϑ is the given maximal iteration number and $u_{it}^\kappa, v_{jt}^\kappa, B_i^\kappa$ are samples at κth iteration.

5.3.2 Inference

To compute r_{ij}^* using Eq. (5.15), we need to sample the variables U, V, B and Θ_U, Θ_V, Θ_B in turn from its distribution conditional on the current values of all other variables, according to Gibbs sampling. Below, we describe these conditional distributions.

Sampling u_{it} and hyperparameter Λ_t: Due to the use of conjugate priors for the parameters and hyperparameters in our model, the conditional distribution over the latent user vector u_{it}, conditioned on other variables $(V, R, B...)$ and the hyperparameters (Θ_U, σ), is Gaussian:

$$\begin{aligned} p(u_{it}|V,R,B,u_{i,t-1},\Theta_U,\sigma) &= \mathcal{N}(u_{it}|\mu_u^*,[\Lambda_u^*]^{-1}) \\ &\propto \prod_{j=1}^{M}[\mathcal{N}(r_{ijt}|u_{it}^T v_{jt},\sigma^2)]^{\varepsilon_{ijt}} p(u_{it}|B_i,u_{i,t-1},\Lambda_t) \end{aligned} \tag{5.16}$$

where

$$\Lambda_u^* = \Lambda_t + \frac{1}{\sigma^2}\sum_{j=1}^{M}\varepsilon_{ijt}[v_{jt}v_{jt}^T],$$

$$\mu_u^* = [\Lambda_u^*]^{-1}(\frac{1}{\sigma^2}\sum_{j=1}^{M}\varepsilon_{ijt}[v_{jt}r_{ijt}] + \Lambda_t B_i u_{i,t-1}).$$

The conditional distribution over the user hyperparameters Λ_t conditioned on the latent user feature hypermatrix U and transition hypermatrix B is given by the Wishart

distribution:

$$p(\Lambda_t|U,B,\Theta_0) = \mathcal{W}(\Lambda_t|W_0^*,\nu_0^*) \tag{5.17}$$

where

$$[W_0^*]^{-1} = W_0^{-1} + \sum_{i=1}^{N}(u_{it} - B_i u_{i,t-1})(u_{it} - B_i u_{i,t-1})^T,$$

$$\nu_0^* = \nu_0 + N.$$

Sampling v_{jt} and hyperparameters μ_t, Φ_t: This part is the same as the conventional fully Bayesian case [55]. The conditional distribution over the latent item vector v_{jt}, conditioned on other variables (U, R) and the hyperparameters (Θ_V, σ), is Gaussian:

$$\begin{aligned} p(v_{jt}|U,R,\Theta_V,\sigma) &= \mathcal{N}(v_{jt}|\mu_v^*,[\Phi_v^*]^{-1}) \\ &\propto \prod_{i=1}^{N}[\mathcal{N}(r_{ijt}|u_{it}^T v_{jt},\sigma^2)]^{\varepsilon_{ijt}} p(v_{jt}|\mu_t,\Phi_t) \end{aligned} \tag{5.18}$$

where

$$\Phi_v^* = \Phi_t + \frac{1}{\sigma^2}\sum_{i=1}^{N}\varepsilon_{ijt}[u_{it}u_{it}^T],$$

$$\mu_v^* = [\Phi_v^*]^{-1}(\frac{1}{\sigma^2}\sum_{i=1}^{N}\varepsilon_{ijt}[u_{it}r_{ijt}] + \Phi_t\mu_t).$$

The conditional distribution over the item hyperparameters μ_t, Φ_t conditioned on the latent item vector hypermatrix V is given by the Gaussian-Wishart distribution:

$$p(\mu_t,\Phi_t|V,\Theta_0) = \mathcal{N}(\mu_t|\mu_0^*,(\beta_0^*\Phi_t)^{-1})\mathcal{W}(\Phi_t|W_0^*,\nu_0^*) \tag{5.19}$$

where

$$\mu_0^* = \frac{\beta_0\mu_0 + M\bar{V}}{\beta_0 + M}, \quad \beta_0^* = \beta_0 + M, \quad \nu_0^* = \nu_0 + M,$$

$$[W_0^*]^{-1} = W_0^{-1} + \bar{C} + \frac{\beta_0 M}{\beta_0 + M}(\mu_0 - \bar{v})(\mu_0 - \bar{v})^T,$$

$$\bar{v} = \frac{1}{M}\sum_{j=1}^{M}v_{jt}, \quad \bar{C} = \sum_{j=1}^{M}(v_{jt} - \bar{v})(v_{jt} - \bar{v})^T.$$

Sampling B_i and hyperparameter Z: Like Eq. (5.16) and (5.18), we assume that the conditional distribution over each transition matrix B_i is matrix normal, and the conditional distribution is determined by a series of multivariate normal distribution and a prior distribution according to $p(B_i|U,\Theta_B) \propto p(U|B_i)p(B_i|\Theta_B)$. The approximation to this

conditional distribution is as follows:

$$\begin{aligned} p(B_i|U,\Theta_B) &= \mathcal{MN}(B_i|Z^*,\Sigma,\Omega) \\ &\propto \prod_{t=1}^{S}\mathcal{N}(u_{it}|B_i u_{i,t-1},\Lambda_u^{-1})p(B_i|Z,\Sigma,\Omega) \end{aligned} \tag{5.20}$$

where

$$Z^* = (\sum_{t=1}^{S}[u_{it}u_{i,t-1}^T] + Z)(\sum_{t=1}^{S}[u_{i,t-1}u_{i,t-1}^T] + \mathbf{I})^{-1},$$

$$\Sigma = \Omega = \mathbf{I}.$$

Analogously, the conditional distribution over Z conditioned on the transition hypermatrix B is given by the matrix normal distribution:

$$p(Z|B,\Theta_0) = \mathcal{MN}(Z|Z_0^*,\Sigma,\Omega) \tag{5.21}$$

where

$$Z_0^* = \frac{Z_0 + N\bar{B}}{1+N}, \quad \bar{B} = \frac{1}{N}\sum_{i=1}^{N}B_i, \quad \Sigma = \Omega = \mathbf{I}.$$

Algorithm 1 shows the process of the Gibbs sampling for BTMF. $x \sim y$ means sampling the random variable x following the distribution y.

Algorithm 1 Gibbs sampling for BTMF

Initialize the model parameters $\{ U^1, V^1, B^1 \}$
for $\kappa = 1 \to \vartheta$ **do**
  Sample the hyperparameters by Eq. (5.17) (5.19) (5.21) for all time windows:
    $\Theta_U^\kappa \sim p(\Theta_U|U^\kappa, B^\kappa, \Theta_0)$,
    $\Theta_V^\kappa \sim p(\Theta_V|V^\kappa, \Theta_0)$,
    $\Theta_B^\kappa \sim p(\Theta_B|B^\kappa, \Theta_0)$.
  For each time window $t = 1, ..., S$, sample each latent user vector by Eq. (5.16), sample each latent item vector by Eq. (5.18):
    $u_{it}^{\kappa+1} \sim p(u_{it}|V^\kappa, R, B^\kappa, u_{i,t-1}^\kappa, \Theta_U^\kappa, \sigma)$,
    $v_{jt}^{\kappa+1} \sim p(v_{jt}|U^\kappa, R, \Theta_V^\kappa, \sigma)$.
  For each user $i = 1, ..., N$, sample the transition matrix in parallel by Eq. (5.20):
    $B_i^{\kappa+1} \sim p(B_i|U^\kappa, \Theta_B^\kappa)$.
end for

5.4 Experimental Evaluation

We conducted extensive experiments to evaluate the prediction accuracy of the proposed methods. We first introduce the experimental settings and then present the main findings.

5.4.1 Evaluated Methods

We evaluate **TMF** proposed in Section 5.2 and **BTMF** proposed in Section 5.3 against the following methods: **PMF** refers to the probabilistic matrix factorization model widely used in collaborative filtering [56]. **BPMF** refers to the fully Bayesian treatment to PMF achieving better results [55]. **timeSVD** refers to the time sensitive algorithm applied successfully in Netflix data set [30]. **Ensemble** refers to an ensemble method by adopting PMF in separate windows and tuning the relevant weights.

We set $\eta = 0.001, \lambda_u = \lambda_v = 0.01$ in PMF, timeSVD, Ensemble and TMF, with other internal parameters with default settings in timeSVD and $\lambda_B = 0.01$ in TMF, and set $\nu_0 = D, \beta_0 = 2, W_0 = \mathbf{I}, \mu_0 = \mathbf{0}$ in BPMF and BTMF, with additional $Z_0 = \mathbf{I}$ in BTMF.

5.4.2 Data Sets

We conduct experiments on six data sets: MovieLens[1] (movie ratings), BeerAdvocate[2] (beer ratings), FineFoods[2] (food ratings), Epinions [26] (product ratings), EachMovie[1] (movie ratings), and Flixster [26] (movie ratings). We keep the first two data sets unchanged because they have a balanced scale, and preprocess the other four data sets by removing the users with less than 20 ratings as in Marlin et al.'s work [47]. All data sets contain ratings ranging from 1 to 5, except for the Eachmovie data set which ranges from 1 to 6. The statistics of the processed data sets are shown in Table 5.1.

Table 5.1: Statistics of data sets.

Data set	#User	#Item	#Rating	Timespan
MovieLens	2113	9801	824600	1997.11-2008.12
BeerAdvocate	33387	66051	1586259	1998.1-2012.10
FineFoods	7590	27385	141294	1999.10-2012.10
Epinions	14077	96291	470557	1999.3-2000.12
EachMovie	36658	1623	2580267	1996.2-1997.8
Flixster	36492	48277	7729741	2005.12-2009.11

[1] http://www.grouplens.org/node/12
[2] http://snap.stanford.edu/data/#reviews [48]

The first three data sets span a longer period of time than the last three. For each data set, we created $S = 10$ time windows as follows. We partition each of the first three data sets yearly and merge several oldest windows (which have less data) into one window to create a total of 10 time windows. For the last three data sets, we partition them bimonthly or half-yearly to get 10 time windows. The last three windows are used for testing. For each testing window, we use all the time windows prior to it, except for Ensemble, as the training set, and run the experiment five times and report the average results for reliability. For Ensemble, we use each of the q most recent windows prior to the testing window to learn a model and average the scores of these models. We report the best result for all possible q.

5.4.3 Evaluation Metrics

We evaluate the accuracy of estimated ratings based on RMSE and Recall@k for the testing data. *RMSE*, defined in Eq. (5.1), is the root mean squared error measuring the difference between the estimated rating values and the true rating values, thus, the prediction accuracy on the individual rating level. *Recall@k* for a user i is defined as the ratio $\frac{|N(k;i)|}{|N(i)|}$, where $|\cdot|$ denotes the number of elements in a set, $N(i)$ is the set of items rated by i in the testing set and $N(k;i)$ is the subset of $N(i)$ contained in the top-k list of all items sorted by their estimated ratings. We report the average of the recall over all users in the testing set. In practice, recall is more useful in recommender systems since it takes a global view on all items and a high recall indeed reflects the user's adoption.

5.4.4 Experimental Results

The followings are the findings on Recall@k and RMSE.

Recall@k. Fig. 5.3-5.5 presents the recall performance of different models at three testing windows of six data sets. The left three columns show Recall@k at the latent dimensionality $D = 20$ while varying k from 50 to 500. The right three columns show Recall@300 while varying D from 10 to 50. We observed that our proposed model TMF always performs better than its conventional counterpart PMF that has no temporal consideration. Ensemble, whose data partitioning misses global patterns and suffers from the data sparsity issue, works no better than PMF. These evidences suggest that considering the effect of temporal dependence and dynamics through the transition matrix help improve the accuracy of recommendation.

The fully Bayesian treatment BTMF beats PMF, BPMF, Ensemble, and TMF in all six data sets. The performance of BTMF is steady with respect to the dimensionality D. The relative ranks of items are better predicted by our temporal models due to the reality that user's subsequent ratings depend more on recent ratings, which is modeled by $r^*_{ij} = (B_i u_{it})^T v_{jt}$, where t is the current time. However, simply focusing on recent ratings, like Ensemble, will miss patterns represented by anterior ratings. Our models address this problem by capturing the temporal dependence and the users' interest shift through the transition matrix B_i learnt from the full set of rating data.

The results of BTMF are competitive even compared with the time-aware timeSVD: BTMF beats timeSVD on MovieLens, FineFood and EachMovie data sets and performs similarly on other data sets. timeSVD aims to capture temporal effects by incorporating a time-variant bias for each user and item at every individual time window, whereas BTMF captures temporal effects through incorporating a single time-invariant transition matrix for each user. Unlike time-variant biases, the time-invariant transition matrix captures the temporal pattern that holds all times, and thus, is suitable for prediction in the next time window. In contrast, time-variant biases are differences at each individual time window and do not summarize the underlying pattern for such differences. This approach works for prediction only if two windows share similar biases. Indeed, for the faster changing FineFoods and MovieLens, we observed a more significant improvement of BTMF over timeSVD.

Different with timeSVD regressing the ratings dynamically, our BTMF further models the preference shift associated with latent user vectors. This relaxes the restriction that prediction should be within the same time period of training in timeSVD. Usually, timeSVD adopts the latest components learnt from the training data to predict future ratings, which works well only if the rating patterns in the future time window is close to recent patterns. Our models, however, try to catch the pattern transition across time windows and better estimate the future ratings with new patterns.

RMSE. Table 5.2 shows the RMSE of different models. As pointed out in Koren's work [30], achievable RMSE values lie in a quite compressed range and small improvements in RMSE terms can have a significant impact on the quality of the top few presented recommendations. First of all, the two fully Bayesian matrix factorization models, BPMF and BTMF, achieve better performance than their non-Bayesian counterparts, PMF and TMF. The boldface and italic highlight the best and second best performers, respectively. Since timeSVD and BTMF perform best on recall, we focus on these two methods here.

Table 5.2: RMSE (mean±standard error) of different models. $D = 20$. The best performer is in **boldface** and the second best performer is in *italic*.

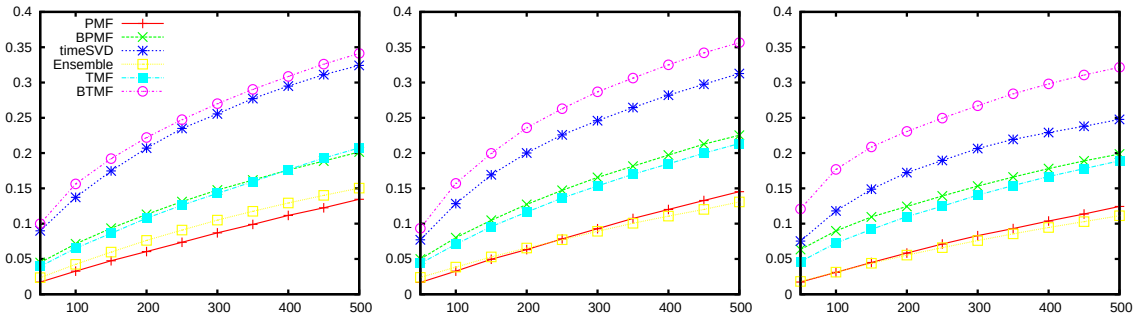
	MovieLens			BeerAdvocate		
Algorithm	**Window 1**	**Window 2**	**Window 3**	**Window 1**	**Window 2**	**Window 3**
PMF[56]	0.9661 ± 0.0018	0.9478 ± 0.0026	0.9036 ± 0.0016	0.7006 ± 0.0015	0.6247 ± 0.0014	0.6368 ± 0.0010
BPMF[55]	0.9351 ± 0.0029	0.9350 ± 0.0057	0.8843 ± 0.0015	0.6917 ± 0.0005	0.6226 ± 0.0005	0.6325 ± 0.0010
timeSVD[30]	0.9170 ± 0.0021	0.9049 ± 0.0010	0.8683 ± 0.0013	**0.6424** ± 0.0013	**0.5634** ± 0.0017	**0.5648** ± 0.0009
Ensemble	0.9531 ± 0.0007	0.9642 ± 0.0020	0.9105 ± 0.0027	0.6984 ± 0.0007	0.6269 ± 0.0008	0.5941 ± 0.0032
TMF	*0.8909* ± 0.0020	*0.8801* ± 0.0020	*0.8557* ± 0.0027	0.6660 ± 0.0007	0.5800 ± 0.0012	0.5882 ± 0.0022
BTMF	**0.8712** ± 0.0006	**0.8454** ± 0.0006	**0.8310** ± 0.0004	*0.6524* ± 0.0003	*0.5708* ± 0.0003	*0.5779* ± 0.0002
	FineFoods			**Epinions**		
PMF[56]	1.2671 ± 0.0032	1.2078 ± 0.0073	1.2429 ± 0.0097	1.2419 ± 0.0087	1.2268 ± 0.0037	1.1924 ± 0.0048
BPMF[55]	1.2537 ± 0.0003	1.1792 ± 0.0002	1.1940 ± 0.0006	1.1892 ± 0.0004	1.1818 ± 0.0005	1.1542 ± 0.0003
timeSVD[30]	*1.2484* ± 0.0008	**1.1684** ± 0.0009	*1.1893* ± 0.0007	1.1707 ± 0.0013	1.1655 ± 0.0017	1.1357 ± 0.0009
Ensemble	1.2528 ± 0.0004	1.1779 ± 0.0018	1.1956 ± 0.0020	1.1929 ± 0.0017	1.1847 ± 0.0008	1.1582 ± 0.0002
TMF	1.2506 ± 0.0047	1.1820 ± 0.0058	1.2089 ± 0.0041	*1.1125* ± 0.0016	*1.1117* ± 0.0012	*1.0978* ± 0.0015
BTMF	**1.2476** ± 0.0004	*1.1744* ± 0.0002	**1.1891** ± 0.0001	**1.1024** ± 0.0002	**1.1067** ± 0.0002	**1.0937** ± 0.0001
	EachMovie			**Flixster**		
PMF[56]	1.3163 ± 0.0015	1.3099 ± 0.0044	1.3608 ± 0.0025	1.1299 ± 0.0082	1.0511 ± 0.0056	1.0899 ± 0.0067
BPMF[55]	*1.2894* ± 0.0016	**1.2855** ± 0.0017	**1.3134** ± 0.0014	1.1100 ± 0.0008	*1.0337* ± 0.0015	*1.0489* ± 0.0024
timeSVD[30]	1.3696 ± 0.0010	1.3736 ± 0.0026	1.4233 ± 0.0019	*1.0865* ± 0.0007	1.0390 ± 0.0003	1.0539 ± 0.0006
Ensemble	1.4659 ± 0.0053	1.4778 ± 0.0064	1.5371 ± 0.0052	1.1466 ± 0.0024	1.0572 ± 0.0022	1.0713 ± 0.0006
TMF	1.3477 ± 0.0024	1.3493 ± 0.0019	1.3909 ± 0.0030	1.1555 ± 0.0038	1.0723 ± 0.0011	1.1290 ± 0.0004
BTMF	**1.2867** ± 0.0080	*1.2892* ± 0.0098	*1.3414* ± 0.0143	**1.0802** ± 0.0038	**1.0329** ± 0.0022	**1.0436** ± 0.0018

With 6 data sets and 3 testing windows for each, there are 18 test cases. Among these 18 test cases, BTMF performs best in 12 cases and second best in 6 cases, whereas timeSVD performs best in 4 cases and second best in 3 cases. For the four cases where timeSVD performs best (i.e., three cases for BeerAdvocate and one case for FineFoods), timeSVD is only slightly better than BTMF.

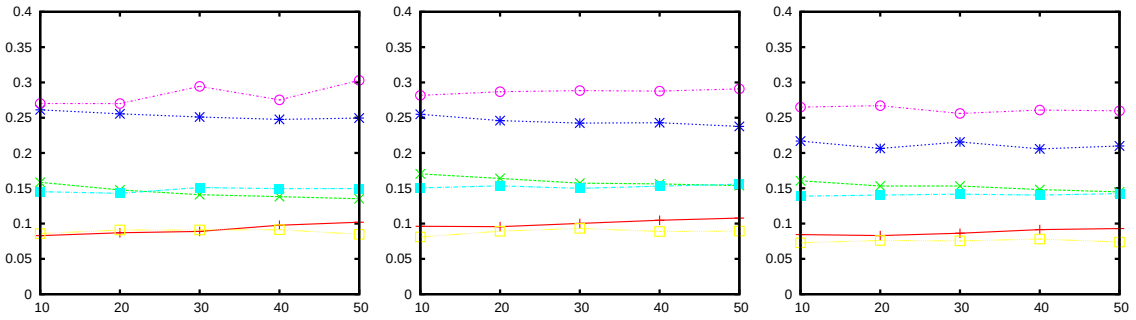
5.5 Summary

This chapter proposed two temporal matrix factorization methods, TMF and BTMF, to predict user preferences that evolve over time. The key idea is to model the evolution by a latent transition matrix that captures the time-invariant property of user's temporal dynamics, thus, the "pattern of evolution" for a user. The experimental results demonstrated improved prediction over previous methods. Two elements contribute to this improvement. One is considering temporal effects in the full time range without discarding any part of the data. Ensemble, which uses several recent time windows, performs poorly. The second element is the time-invariant transition matrix that captures a property essential for prediction. The time-variant biases used by the timeSVD model are fit for prediction only if such biases are similar for adjacent time windows. The transition matrix also provides a way to

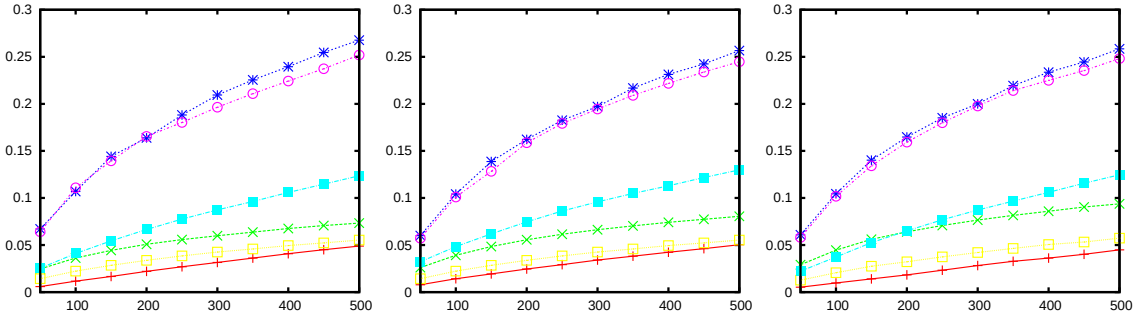
understand the pattern of evolution for a user. We observed two types of transition matrices learnt. The first type has non-zero entries on the main diagonal, which captures those inactive users who have very few ratings observed. The second type has non-zero entries outside the main diagonal and such entries represent certain preference shifts among latent factors, and a different distribution of non-zero entries captures a different shift pattern.



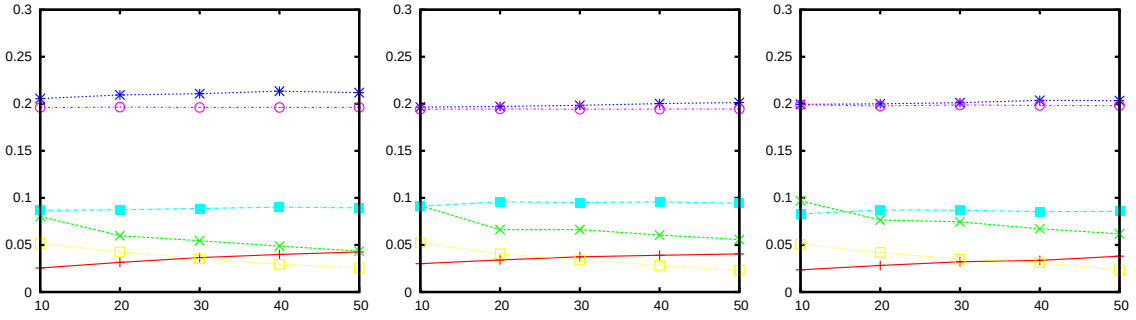
(a) Recall@k at MovieLens, $D = 20$. Vary k



(b) Recall@300 at MovieLens. Vary D

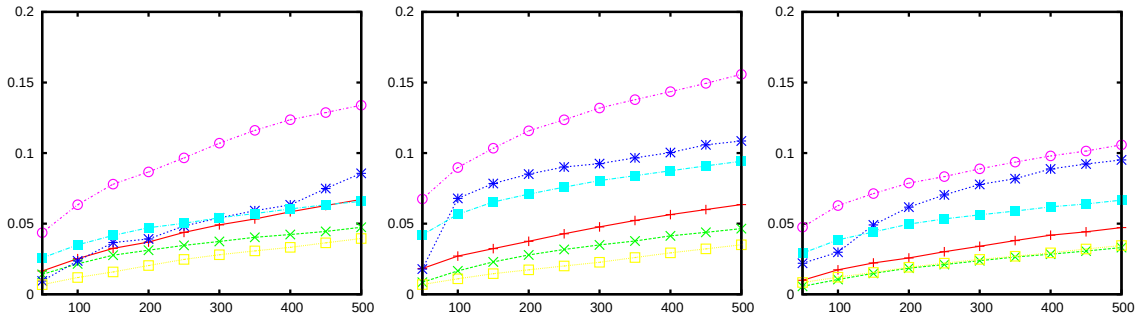


(c) Recall@k at BeerAdvocate, $D = 20$. Vary k

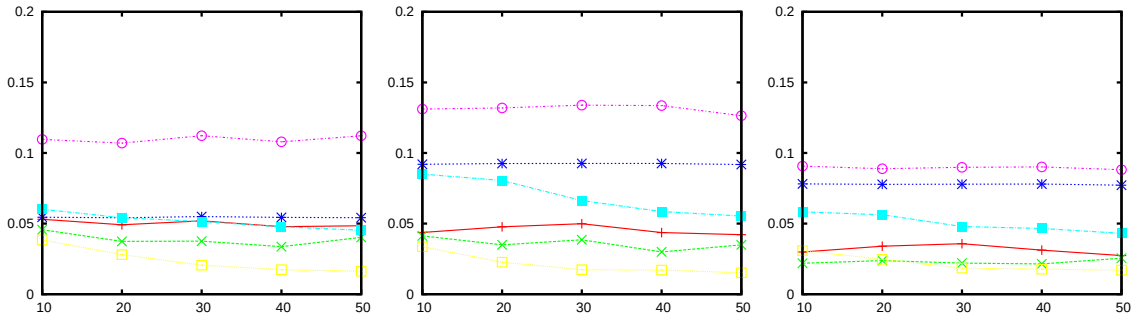


(d) Recall@300 at BeerAdvocate. Vary D

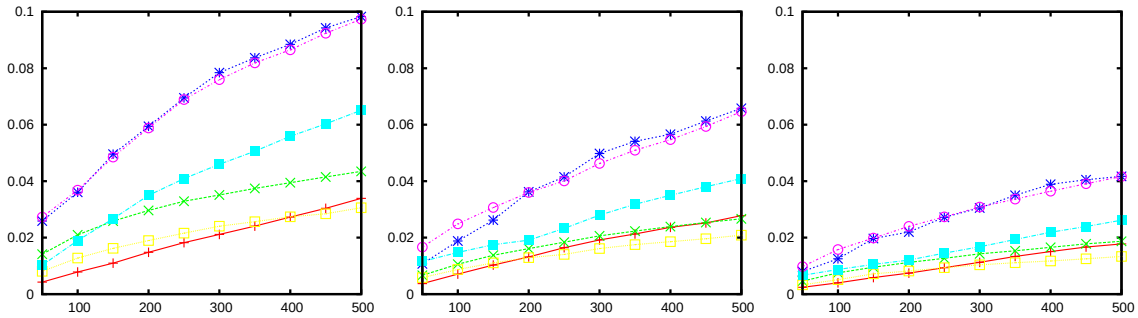
Figure 5.3: Recall of different models. Every two rows refer to a data set and each column refers to a testing window. For each data set, the first row observes the effect of varying k with fixed $D = 20$ while the second row observes the effect of varying D with fixed $k = 300$. Higher values are better.



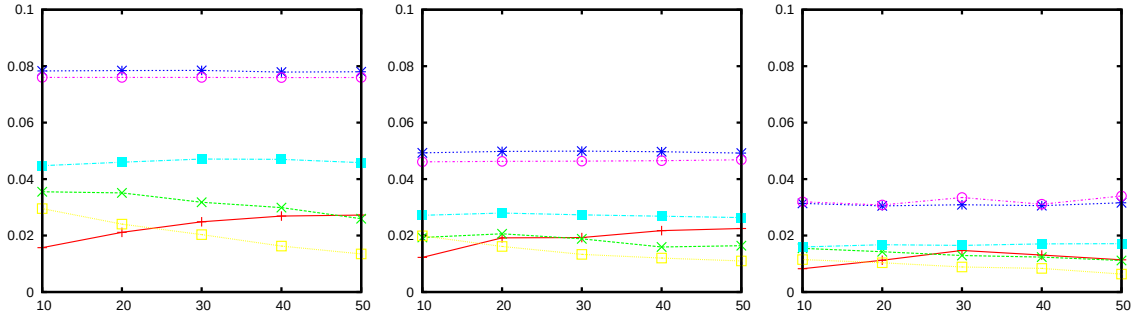
(a) Recall@k at FineFoods, $D = 20$. Vary k



(b) Recall@300 at FineFoods. Vary D

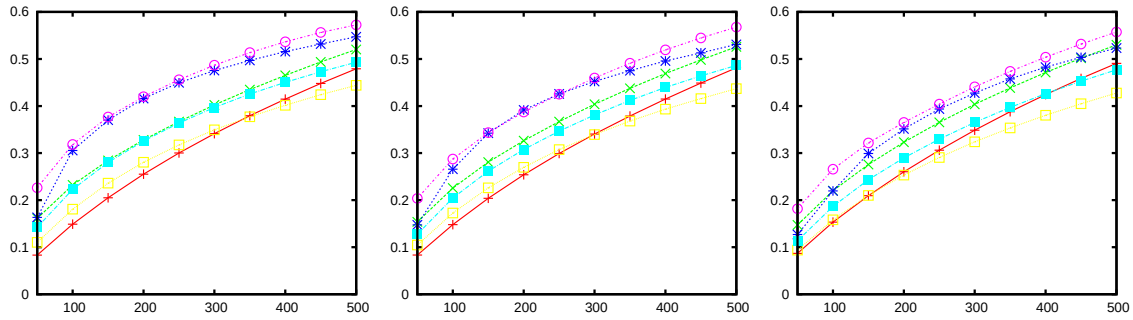


(c) Recall@k at Epinions, $D = 20$. Vary k

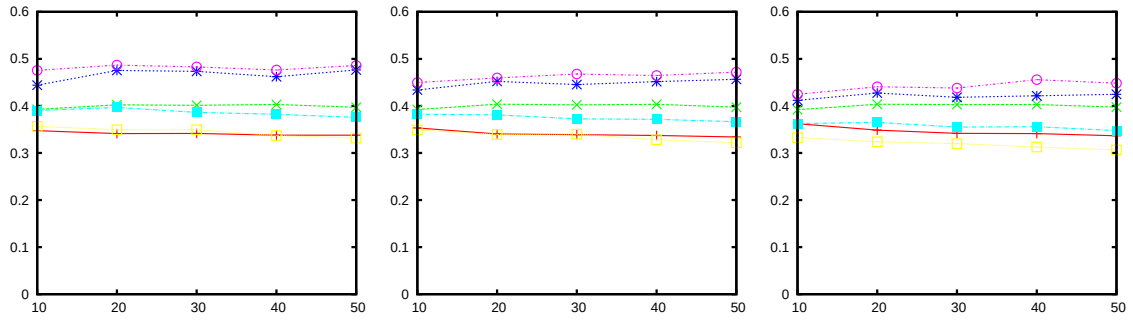


(d) Recall@300 at Epinions. Vary D

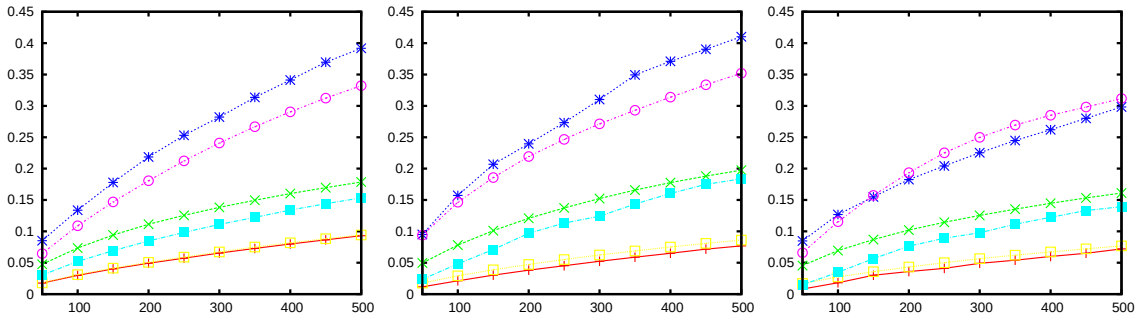
Figure 5.4: Recall of different models (cont.). Every two rows refer to a data set and each column refers to a testing window. For each data set, the first row observes the effect of varying k with fixed $D = 20$ while the second row observes the effect of varying D with fixed $k = 300$. Higher values are better.



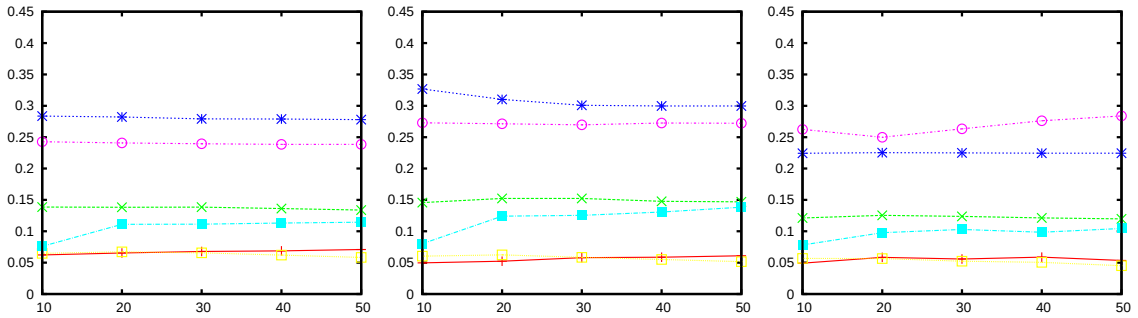
(a) Recall@k at Eachmovie, $D = 20$. Vary k



(b) Recall@300 at Eachmovie. Vary D



(c) Recall@k at Flixster, $D = 20$. Vary k



(d) Recall@300 at Flixster. Vary D

Figure 5.5: Recall of different models (cont.). Every two rows refer to a data set and each column refers to a testing window. For each data set, the first row observes the effect of varying k with fixed $D = 20$ while the second row observes the effect of varying D with fixed $k = 300$. Higher values are better.

Chapter 6

Cross-region Collaborative Filtering

Recommending points of interest (POIs) to a user according to the user's current location and past check-in activities is the focus in this chapter. Previously proposed probabilistic and topic model based methods predict the POIs based on the distribution of the POIs visited in the past, assuming that the next POI for the user follows the same distribution. Such methods tend to recommend the POIs in the cities or regions that the user has visited before because only such cities or regions have observed ratings for the user. Thus, these works are not suitable for a user who travels to a new city or region where she has not checked in any POI previously. To address this issue, we distinguish the user preferences on the content of POIs from the user preferences on the POIs themselves. The former is long-term and is independent of where POIs are located, and the latter is short-term and is constrained by the proximity of the location of the POI and the user's current location. This distinction motivates a location-independent modeling of user's content preferences of POIs, and a location-aware modeling of user's location preferences of POIs. The final recommendation of POIs is derived by combining the predicted rating on content and the predicted rating on location of POI. We evaluate this method using the Yelp and Foursquare data sets. This approach has superiority over the state-of-the-art and works well in the "new city" situation in which the user has not rated any of the POIs in the current region.

6.1 Motivation

Location based social network (LBSN) services such as Yelp and Foursquare enable users to "check in" at a certain point of interest (POI), such as hotel/restaurant/museum, via their mobile devices. A user may rate and comment about the POI after visiting it, and subsequently other users may consider those ratings and comments before selecting the POIs for their visits. The availability of rating data and LBSN services open up an array of new research topics and problems in both academia and industry [57, 70], such as user behavior analysis, movement pattern study [10, 39], and various real-world applications [11, 73, 77]. Among them, an emerging application is to recommend suitable POIs as users travel to a region or location [25, 41, 72]. A key difference from traditional recommender systems is that the distance between the location of the user and the location of a POI will influence the user's adoption of the recommended POI. For example, recommending a Chinese restaurant in Beijing to a user who is currently visiting New York City will fail, even if the user loves Chinese food very much.

We assume that each POI has two essential descriptions, *location* and *content*. Location refers to the geographical position such as longitude and latitude of the POI. Content refers to any other descriptive information about a POI and its function, such as features (e.g., categories of art works for a museum), tags attached by users, and comments created by users. After a user visits a POI, the user may rate the POI and select some features or create tags or comments for it. The content of a POI is defined by the collection of terms created or features selected by all users for the POI. Given some observed rating data and content of POIs, the POI recommendation problem is to recommend suitable POIs to a user who is currently visiting a location, by factoring in the match between the user's interest and the content of the POI, and the geographical distance between the POI and the current location of the user. Figure 6.1 is a POI map in Phoenix from the Yelp data set partitioned into 10 regions. The POIs in the orange and skyblue regions are less attractive to a user visiting the brown region, and more so for the POIs in the yellow and dark blue regions.

One immediate solution is to partition the observed ratings by geographical regions of location (e.g., cities) and build one regional recommender for each region using the rating data in that region. As a user moves to a region, the recommender for that region is used for POI recommendation. Though very simple, this solution has a number of drawbacks. (1) The restriction to the rating data in a single region accelerates "data sparsity", which leads to poor rating prediction. (2) There is a lack of knowledge transfer due to the following

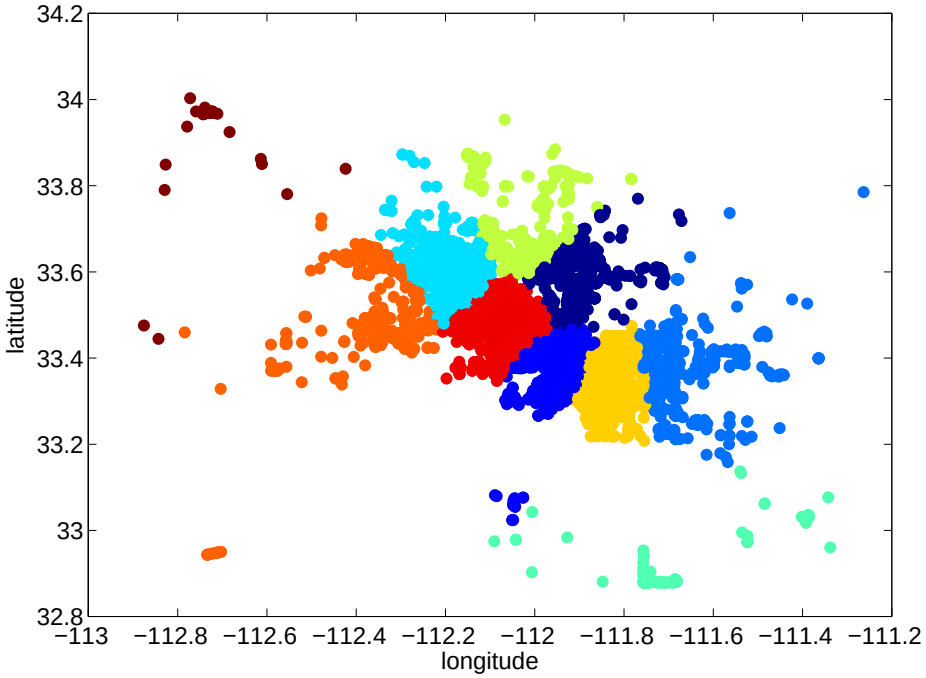


Figure 6.1: Locations of the Yelp data partitioned into regions

"new city" problem: suppose that two users x and y have visited many common POIs in their home city A and suppose that user x has also visited the city B but y has not. When y visits the city B for the first time, thus a "new city" to y, the recommender for B will mistakenly consider x and y as sharing little common interests due to the lack of observed data for y in the city B. Note that the "new city" problem is different from the "cold start" problem since the POIs of the "new city" all exist in the system, but they are "new" to a specific user who has not rated any POI in that city. (3) If the user's current region has no suitable POI, this method tends to recommend irrelevant POIs because collaborative filtering in one region does not benefit from those in other regions.

Recently, some topic modeling based methods [25][72][33] have incorporated content and location information of POIs. The main idea is to infer the probability of visiting a POI/location based on the observed check-in data and user generated data such as tweets [25], and recommend the POIs/locations to a user that have the highest estimated probability, assuming that the check-in pattern of the user does not change. These approaches may not work if the user is currently visiting a region where she does not have much check-in data previously, as discussed before. In this case, these methods still recommend the POIs from the "home cities" that the user used to visit in the past because such POIs have the

highest estimated probability, but such POIs are not interesting to the user who is currently visiting a different region.

Our insight is to distinguish user's preferences on *the content of POIs* from user's preferences on *the POIs themselves*: The former reflects the user's long-term interests developed through life time and is independent of the user's current location, whereas the latter is short-term and is constrained by where the POI and the user are located. Importantly, the long-term preferences on content are *transferrable via location*. For example, a user who loves arts, e.g., visited arts museums frequently in the past, tends to look for arts museums when visiting a new city. This observation motivates a cross-region collaborative filtering approach to POI recommendation: we infer the user's content preferences from the observed rating data across all regions *independent of where the user is*, and infer user's location preferences based on *where the user is currently located*. As a user travels to a new region, content-suitable POIs in the new region are identified through the location independent content preferences. The details and contributions are summarized as follows.

6.1.1 Contributions

We present a new solution, *cross-region collaborative filtering (CRCF)*, to address both long-term content preferences and short-term location preferences of users. We model the long-term content preferences by a location independent *content recommender* learnt from a user-feature rating matrix derived from all rating data, and use this recommender to predict the user's rating on the content of a POI. We model the short-term location preferences by a *location recommender*, and use this recommender to predict the user's rating on the location of a POI. In particular, we partition the entire geographical region into a number of local regions (e.g., cities), and for every user i and every region k, we create a "virtual user" (i, k) to model the user i visiting the region k, and learn the location recommender from a "virtual-user"-POI rating matrix whose entry $((i, k), j)$ is the "perceived rating" on the location of POI j by the virtual user (i, k), that is, assuming that the user i is visiting the region k. Both recommenders are learnt by collaborative filtering on their matrices. The final predicted rating of a POI is derived from the predicted rating on content and the predicted rating on location.

An example of these matrices is shown in Figure 6.2. For example, (A,II)=4 in the original rating matrix (a) indicates the observed rating 4 on POI II by the user A when the user *actually visited* the POI II in the region r2. In the "virtual-user"-POI rating matrix

	I	II	III		bag of features	region
user A	4	4		I	f1, f2	r1
user B	5	2	5	II	f1, f3	r2
user C	5		4	III	f2, f4	r1

(a) The left indicates the traditional rating matrix and the right indicates the content and region of each POI

	f1	f2	f3	f4
A	{4,4}		4	
B	2	{5,5}		5
C	5	5		4

(b) User-feature matrix for learning the content recommender

	I	II	III
(A,r1)	4	2	
(A,r2)	2	4	
(B,r1)	5	1	5
...		...	

(c) "Virtual-user"-POI matrix for learning the location recommender

Figure 6.2: Different matrices for 3 users (A,B,C), 3 POIs (I, II, III), 4 features (f1,f2,f3,f4), and 2 regions (r1,r2). POI I has features f1, f2 and is located in region r1, POI II has features f1, f3 and is located in region r2, POI III has features f2, f4 and is located in region r1. Users may select a subset of the features when rating a POI.

(c), ((A,r1),II)=2 indicates the perceived rating on II by A assuming that the user is in the region r1. The perceived rating is lower than the observed rating because the user is in a different region from that of the POI. The generation of the perceived rating ((A,r1),II) will take into account the observed rating (A,II)=4 as well as the distance between the POI II and the user's current region r1. Thus, ((A,r1),II) is a result of transferring the observed knowledge in other regions, i.e., II's region r2, to the current region of the user, i.e., r1, while factoring the impact of distance.

Our method is fundamentally different from the distance adjustment to the ratings of any collaborative filtering, e.g., filter the POIs by a circle or some distance penalty functions [40], because the distance adjustment still prefers the near POIs and further needs user efforts such as setting the radius of the circle. Consider the example if you are vegetarian and the only vegetarian restaurant in this city is at some distance from your current location, our method likely recommends the vegetarian restaurant because the content recommender ranks the restaurant highly and the location recommender does not use a hard distance threshold, while the distance adjustment approach either cannot find the vegetarian restaurant or recommends a non-vegetarian restaurant.

This approach has several novel features. (1) Both location and content recommenders are based on collaborative filtering on the rating data across *all* regions, an important factor to address the data sparsity issue discussed earlier. (2) Our method addresses the "new city" problem by explicitly modeling a user i visiting any region k through the virtual user (i, k) in the location-aware matrix used by the location recommender. (3) Our method could recommend POIs from a neighboring region if the current region of the user does not have suitable POIs. In contrast, the regional recommender tends to recommend irrelevant POIs in this case. (4) This method is easy to implement; all it requires is the principled matrix factorization method and the preprocessing/postprocessing steps that wraps up the factorization method. We have evaluated our method on two LBSN service data sets, Yelp and Foursquare. The study shows that our method achieves a considerable improvement over the state-of-the-art methods in terms of mean reciprocal rank and recall.

6.1.2 Comparison with Related Work

Hu et al. [25] proposed a spatial topic model by incorporating user posts as well as user movements into a geographical topic model where regions and topics are random latent variables associated with each post. Their model predicts the location for a given document based on a collection of geo-tagged posts, which tends to give high probabilities to locations close to user's frequently visited locations before. Liu et al. [41] proposed a geographical probabilistic factor model taking latent preferences, user mobility and item popularity into consideration. Their model assigns a user to a latent region learnt from user's previously visited POIs and sets the region center as the user's location.

All of the above mentioned probabilistic and topic model methods are based on modeling the probability distribution of the POIs for a user based on the observed data, therefore, favor the POIs in the user's frequently visited region (e.g., the home city). Such methods are not suitable if the user travels to a *new* region where she has not generated any post or check-in activity. By modeling a virtual user (i, k), our method will recommend POIs specifically for a user i traveling to a region k. For example, if a user checked-in POIs in Beijing in the past and is currently visiting New York City where she has not checked-in any POI, our method will recommend matching POIs in New York City, whereas the probability and topic model methods tend to recommend the POIs in Beijing because they have the highest estimated probability.

Yin et al. [72] presented a topic model called LCA-LDA to incorporate item content and user's visit history. Similar to ours, their focus is to make appropriate recommendations for a user visiting a certain region. By treating each POI as word, their model learns the topic distributions for each user and each region (e.g., city), and for a user coming to a region, the recommendation of venues is derived from the distributions learnt for that region. Their model does not consider the distance or location information for POIs across regions, which is an important factor in our modeling. In Bao et al.'s work [6], the rating data is partitioned by regions and categories of locations, and for each region and each category, "local experts" are extracted based on the historical data in that region and category. As pointed out before, partitioning the rating data by regions will accelerate data sparsity and reduces the level of collaborative filtering. Our method takes the geographical space as partitioned regions (for modeling geographical influence), but collaborative filtering is always performed on the rating data for all regions.

6.2 Methodology

This section presents our model. We start with the description of data representations and tasks. Our model basically consists of two components, i.e., the content recommender and the location recommender. Figure 6.2 gives a quick overview of them in terms of data presentation.

6.2.1 Preliminaries

We have I users, J POIs, and F features (or terms) for POIs. Each user i may give some numeric ratings (say 1 to 5) to several POIs j that she has visited. The observed rating data is represented in an $I \times J$ user-POI matrix M (see Figure 6.2(a)), in which an entry r_{ij} represents the observed rating on POI j given by user i. If user i did not rate POI j, r_{ij} is undefined. When rating a POI j, a user i may optionally select some features of j, T_{ij}, to indicate the features preferred. If a user i does not rate a POI j, we assume $T_{ij} = \emptyset$. The content of POI j is defined as $T_j = \bigcup_i T_{ij}$. Each POI also has a geographical location, i.e., latitude and longitude. We assume that a function will return the distance between any two geographical locations.

For example, if POIs are hotels, the ratings could be 1-5 stars and the features of a hotel can be cleanliness, service, rate, comfort, etc. If a user i gives the 5-star rating to a hotel j and explicitly selects cleanliness and comfort, $r_{ij} = 5$ and $T_{ij} = \{cleanliness, comfort\}$,

which means that the user particularly likes cleanliness and comfort of the hotel. If a user i rates a hotel j but selects no feature, we assume that the user implicitly selects all the features, i.e., T_{ij} contains all features of j. Here T_{ij} contains the features both explicitly and implicitly selected. Alternatively, features can be terms in a review or tags created for a POI, and T_{ij} is the collection of terms or tags for POI j created by user i.

The task of POI recommendation is to recommend top-n POIs to a user that suit the user's interests and are close to the user's current location. The key is to predict a user's rating on unrated POIs based on the observed rating data, the content and location of POIs, and the user's current location. Our solution first predicts the rating on the content of POIs and the rating on the locations of POIs independently, and then combines these ratings to derive an overall ranked list of POIs. Below, we describe these steps.

6.2.2 Predicting Content Preferences

First, we construct the content recommender to predict the user's rating on a feature of a POI, e.g., cleanliness of a hotel. The user's preference on the content of a POI is estimated by aggregating the predicted ratings on all the features of the POI. To predict the rating on the features of a POI, we adopt the feature-centric recommendation from Zhang et al.'s work [74], which applies collaborative filtering on the features of items instead of items themselves. First, we transform the original user-POI matrix M into a $I \times F$ user-feature matrix M_1 (see Figure 6.2(b)), where each row represents a user and each column represents a feature f [74]. An entry (i, f) in M_1 is the aggregated rating on the feature f over the POIs j that contain f and are rated by user i:

$$g_{if} = agg(\{r_{ij} | f \in T_{ij} \text{ and } r_{ij} \text{ is defined}\}) \tag{6.1}$$

In this chapter, $agg(X)$ returns the average of the values in X. Therefore, g_{if} is the averaged observed rating on f by user i. $agg(X)$ is undefined if X is empty. Then we extract the latent user vectors u_i for users i and the latent feature vector v_f for features f. This can be done by performing the standard matrix factorization [31] on M_1. The goal is to minimize the regularized squared error loss between the rating g_{if} and the predicted rating $u_i^T v_f$, as in Salakhutdinov et al.'s work [56]. For completeness, we include the algorithm of matrix factorization as follows.

$$E = \frac{1}{2}\sum_{i,f} \varepsilon_{if}(g_{if} - u_i^T v_f)^2 + \frac{\lambda_u}{2}\sum_i \|u_i\|^2 + \frac{\lambda_v}{2}\sum_f \|v_f\|^2 \tag{6.2}$$

where λ_u and λ_v are regularization parameters; ε_{if} is a binary indicator that is equal to 1 if user i has selected feature f (g_{if} is observed) and equal to 0 otherwise. Taking the gradient of E respect to the variables u_i and v_f, we get

$$\frac{\partial E}{\partial u_i} = \lambda_u u_i - \sum_f (g_{if} - u_i^T v_f) v_f \tag{6.3}$$

$$\frac{\partial E}{\partial v_f} = \lambda_v v_f - \sum_i (g_{if} - u_i^T v_f) u_i \tag{6.4}$$

A local minimum of E is found by iteratively updating each variable with a step proportional to the negative of the gradient based on the recent values with the learning rate η:

$$u_i^{\kappa+1} = u_i^{\kappa} - \eta \frac{\partial E}{\partial u_i^{\kappa}}, \;\; v_f^{\kappa+1} = v_f^{\kappa} - \eta \frac{\partial E}{\partial v_f^{\kappa}} \tag{6.5}$$

This iterative process stops until convergence. The outcome is the latent user vectors u_i for all users i and the latent feature vectors v_f for all features f.

The predicted rating of user i on the content of a POI j is computed by aggregating the predicted ratings $u_i^T v_f$ over all the features f in T_j:

$$r_{ij}^* = agg(\{u_i^T v_f | f \in T_j\}) \tag{6.6}$$

The reason for choosing $agg(X)$ as the average of the values in X is that it is simple and works well in our experiments. Other choices of the aggregation function are possible, such as the average of the differences from the averaged rating of all the items rated by the concerned user, or a regression based weighting inspired by multi-criteria ratings [17].

6.2.3 Predicting Location Preferences

The second important factor in recommending POIs is the user's location preference on a POI. We construct a location recommender to predict this user location preference, which depends on the location of the POI and the location of the user. To model this dependency, we partition the geographical space into K regions for some specified K, where a region

covers the locations within a close proximity and each region has a mass center. Since we approximate the distance between any two locations from two regions by the distance between the centers of the two regions, a region should be "small enough". For example, in New York City, regions could be areas that corresponded to community districts, such as Manhattan, Brooklyn, and Queens, etc. The choice of the definition of regions often depends on the application. Our method assumes that the partition of regions is given.

Assuming that user i is in some region k, we want to estimate the user's rating on the *location* of POI j, where j is not necessarily in the region k. Clearly, this rating may not be the same as the observed rating r_{ij} generated when user i actually visited POI j where both the user and the POI are in the same region. When the user is in the region k, her rating on the location of j should be reduced according to the distance from the region k to the location of j. We estimate this distance-adjusted rating by $l_{(i,k),j} = \frac{r_{ij}}{dis(k,j)}$, where $dis(k,j)$ denotes the normalized distance between the center of the region k and the center of the region for POI j:

$$dis(k,j) = \begin{cases} 1 & \text{if } j \text{ is in region } k \\ 1 + \dfrac{\|c(k) - c(\sigma(j))\|_2}{MIN} & \text{otherwise} \end{cases} \tag{6.7}$$

where $\sigma(j)$ is the region index of the POI j, $c(x)$ is the center of a region x, and MIN is the minimum pairwise distance of region centers. Intuitively, $l_{(i,k),j}$ is the observed rating r_{ij} adjusted by the normalized distance between the regions of the POI j and the region of user i. This adjustment is consistent with Tobler's first law of geography [62], which implies that the propensity of a user for a POI is inversely proportional to geographic distance between the user and the POI. On the other hand, if the user is obsessed with the content of the POI, she will be less deterred by geographical distance. This is the reason for incorporating the rating r_{ij} in the distance-adjusted rating. If r_{ij} is not observed, $l_{(i,k),j}$ is undefined. Note that if the POI j is in the same region k as the user i, the distance influence disappears and $l_{(i,k),j} = r_{ij}$.

We can define an $(I \times K) \times J$ matrix M_2 (see Figure 6.2(c)) with $l_{(i,k),j}$ as follows. For each user i and each region k, we create a row denoted (i,k), and for each POI j, we create a column j. The entry for the row (i,k) and the column j, denoted by $((i,k),j)$, contains the distance-adjusted rating $l_{(i,k),j}$ if it is defined. Intuitively, each row (i,k) simulates a virtual user that is the instance of user i visiting the region k and the entry $((i,k),j)$ simulates the perceived rating on the location of the POI j of this virtual user.

Once having constructed M_2, we can infer unfilled ratings in M_2 by performing matrix factorization on M_2 to learn the latent user vectors u_{ik} for virtual users (i,k), and the latent location vectors v_j for the POIs j. The objective function to be minimized is

$$E = \frac{1}{2}\sum_{i,j,k}\varepsilon_{ij}(l_{(i,k),j} - u_{ik}^T v_j)^2 + \frac{\lambda_u}{2}\sum_{i,k}\|u_{ik}\|^2 + \frac{\lambda_v}{2}\sum_{f}\|v_j\|^2 \tag{6.8}$$

The computation of u_{ik} and v_j is similar to the previous subsection. The predicted location rating by a user i at the region k for the POI j is estimated by the inner product of u_{ik} and v_j:

$$l^*_{(i,k),j} = u_{ik}^T v_j \tag{6.9}$$

6.2.4 Recommending POIs

We believe that the content recommender captures the long-term preferences while the location recommender captures the short-term preferences for users. So we first generate a set of POIs that match user's interests and later filter them by location. For a given positive integer n, we would like to recommend top-n interesting POIs to a user i who is currently visiting a region k. Note that these POIs are not necessarily in the region k. This is done in four steps as shown in Figure 6.3. (i) For each POI j, predict the content rating of user i on j using Eq. (6.6) and sort all POIs by their predicted content rating. (ii) Select the top-m POIs from the ranked list as the seed set, where $m > n$. The POIs in the seed set are considered matching user interests in content, and the POIs not in the seed set are never recommended to the user. (iii) Reorder the POIs in the seed set according to their predicted location rating computed by Eq. (6.9). (iv) Recommend the top-n POIs in the reordered list. These POIs have the most preferred locations among those that match user interests. We call this method *cross-region collaborative filtering (CRCF)*.

Of course, there are other ways to generate the final recommendation except the re-ranking mentioned above, for example, pick the nearby POIs first according to location recommender and then choose those matching user's interests through content recommender. In this chapter, however, we do not consider these in the experiments.

6.2.5 Discussion

The CRCF method has a number of properties that are worth noting. First, the recommended POIs are not necessarily from the current region of the user. If the current region

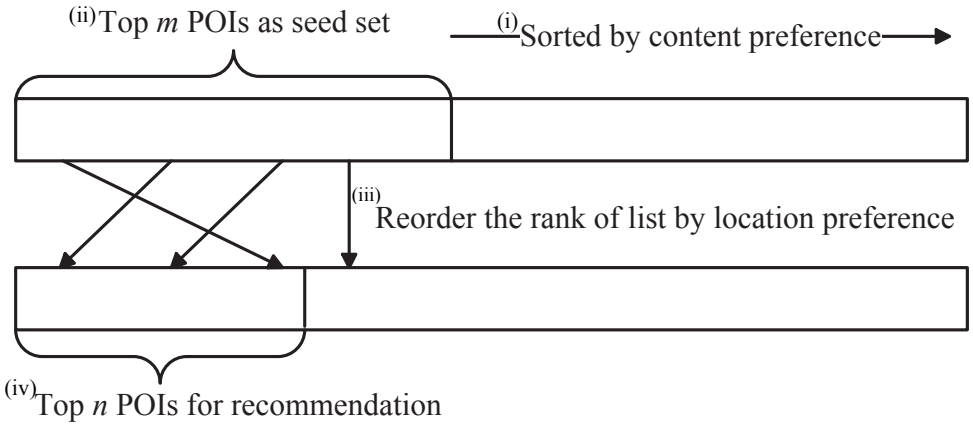


Figure 6.3: Combining the predicted content rating and location rating to compute the top-n POIs in CRCF

of the user has POIs with a high predicted content rating, such POIs tend to be included in the seed set and selected for recommendation. If the current region does not have POIs with a high predicted content rating, POIs from other regions with a high predicted content rating will be included in the seed set. The choice of the size m of the seed set dictates the relative weight of content preference and location preference. A larger m leads to more flexible content preferences to favor local POIs in the current region of the user, and a smaller m will favor POIs with strong content preferences, so local POIs with weak content preferences may not get into the seed set. Thus, m can be used to control the trade-off between content preference and location preference.

Our method assumes that the geographical space is partitioned into K regions so that the locations of the POIs within a region can be approximated by the region center. The granularity of regions has an implication on the computation and the accuracy of location modeling. A finer granularity yields a more accurate modeling of the location of POIs, but leads to a larger number K of regions, thus, a larger matrix M_2. For a small-to-medium geographical space, such as a city or state, a K in hundreds usually suffices. For a large geographical space, such as a country or the world, K is likely larger in order to keep each region small. This could impose a high cost on space and computation. One solution is to partition a large geographical space, e.g., a country, into small-to-medium subspaces, e.g., states, and apply our method to each subspace independently. This approach makes sense because a user visiting a state usually considers the POIs from the same state and the problem of data sparsity is less severe for a state (than a small district).

Our method can handle the "new city" problem without special arrangements. Suppose that a user i visits a city k for the first time, therefore, has not rated any POI in the city. From the user's point of view, the city k is a "new city". Our method models this scenario by the virtual user (i, k) in M_2 with the perceived rating $l_{(i,k),j}$ for POI j. As long as user i has rated some POIs in some cities, not necessarily in the current city k, the recommendation for user i visiting the region k does not require special handling.

6.3 Experimental Evaluation

This section evaluates the effectiveness of the CRCF method proposed in Section 6.2. First, we describe our data sets, the methods for comparison, and our evaluation metrics.

6.3.1 Data Sets

We adopt the Yelp[1] and Foursquare[2] data sets in our experiments. Both data sets were previously used for recommendation evaluation in Hu et al.'s work[25]. The Yelp data set contains 45,981 users, 229,906 ratings, 11,537 POIs, plus user reviews on POIs. We preprocessed the reviews by removing stop words and infrequent words occurring in < 100 reviews, and using the remaining 8,519 keywords as features. The feature set or content of a POI consists of all keywords contained in the reviews about the POI. The Foursquare data set contains 20,784 users, 153,477 ratings, 7,711 POIs, and user published tweets when checking-in at a POI. Like Yelp, we obtained 1,377 features after preprocessing the tweets. To partition the space into regions, we apply the K-means clustering algorithm to the locations of all POIs due to its simplicity. Other sophisticate clustering or tessellation methods [2, 54] are also possible.

6.3.2 Evaluated Methods

We compare the following methods, which we either implemented or got the codes from their authors.

Probabilistic matrix factorization (denoted PMF): This method adopts matrix factorization on the user-item rating utility matrix [56] where POIs are treated as items, which ignores the content and location information of POIs. In PMF, matrix factorization is generalized as a probabilistic model where a latent user vector $u_i \sim \mathcal{N}(0, \lambda_u^{-1} I_D)$, a latent

[1] `http://www.yelp.com/dataset_challenge/`
[2] `https://foursquare.com/`

item vector $v_j \sim \mathcal{N}(0, \lambda_v^{-1} I_D)$, and user-item rating $r_{ij} \sim \mathcal{N}(u_i^T v_j, \varepsilon_{ij}^{-1})$. The predicted user i's rating on item j is given by $u_i^T v_j$.

Partitioned PMF (denoted PAR): This method simulates a location aware version of PMF by building a PMF model for each region using the user-item matrix M containing only the rating data in the region. Note that M still has a column for every POI, but only those in the current region have rating data. As discussed in Section 6.2.5, this approach does not leverage collaborative learning across regions in that only local rating data are used for each region. For a user visiting a region, the recommender built for that region is used to recommend POIs to the user.

Collaborative topic regression (denoted CTR): This is the matrix factorization with topic modeling applied to the features of items [65]. For our data sets, items are POIs. LDA is employed on POI j's features to learn the latent topic vector θ_j, which is incorporated into the PMF framework to confine the search of latent item vectors by setting $v_j \sim \mathcal{N}(\theta_j, \lambda_v^{-1} I_D)$. This method does not consider the location of POIs.

Spatial topic model (denoted STM): This is the generative topic model for POI recommendation incorporating the spatial and textual aspects of POIs [25]. STM gives a global recommendation without considering the user's current location.

Location content aware model (denoted LCA): This is another generative topic model for POI recommendation incorporating content and location information [72]. LCA makes personalized recommendation according to user's current location but does not consider distance while modeling locations.

Location recommender (denoted LOC): This is the location recommender in Section 6.2.3 that performs cross-region collaborative filtering on location ratings while ignoring the content information of POIs. Given a user i visiting a region k, this recommender recommends the top-n POIs ranked by the predicted rating in Eq. (6.9).

Content recommender (denoted CON): This is the content recommender in Section 6.2.2 that performs cross-region collaborative filtering on feature ratings while ignoring the location information of POIs. Given a user i, this recommender recommends the top-n POIs ranked by the predicted rating in Eq. (6.6).

Cross region POI recommender (denoted CRCF): This is the POI recommender in Section 6.2.4 that integrates the predicted ratings by the location recommender and the content recommender. By default, the seed size m used by this recommender is set to 200.

PMF, PAR, and CTR serve as the baselines that treat POIs as regular items. STM and LCA are recent state-of-the-art recommendation methods for POIs. LOC, CON, CRCF are

derived from our methods in Section 6.2 through considering location only, content only, and both location and content of POIs. All latent factor models adopt the dimensionality of $D = 10$ by default for latent vectors and we also vary D from 10 to 50 to study the effectiveness. We use grid search to find the optimal value of the regularization parameters λ_u and λ_v which are searched from [0.001,0.01,0.1,1]. The parameters for topic modeling are $\alpha = \frac{50}{D}, \beta = 0.01$ and the learning rate is $\eta = 0.0001$. Since we could not get the code of the geographical probabilistic factor model proposed in Liu et el,'s work [41], it is not included here.

6.3.3 Evaluation Metrics

For both data sets, we conduct 10-fold cross-validation, i.e., in each fold, 90% of the observed ratings are picked as the training set M and the test set T contains the positive ratings in the remaining 10% data. For Yelp with 1-5 ratings, a positive rating is a rating ≥ 4 (since the average ratings in this data set is around 3.6). For Foursquare with 0/1 ratings, a positive rating is the rating of 1. Since each user actually checked-in the POI of a certain region in the test set, we adopt the region center as the user's current location when the recommendation is performed.

In general, recommendation methods that predict the user's rating on a POI can be evaluated by error based metrics (such as RMSE/MAE) and rank based metrics. However, the topic model based methods STM and LCA predict the probability of visiting a POI, for which the error specific metrics such as RMSE/MAE are not suitable because probabilities are not comparable with ratings. For this reason, we use rank based metrics and adopt a similar procedure as in Cremonesi et al.'s work [14]: For each method described in Section 6.3.2, we first train the recommender using the observed ratings in M. Then, for each test case (i, j) in T where user i rates POI j positively: (i) We predict the ratings for the POI j and for all the other POIs unrated by user i using the recommender. (ii) We form a ranked list by ordering all these POIs according to their predicted ratings. Let $rank(i, j)$ denote the rank of the POI j within this list. The best result corresponds to the case where the POI j precedes all other POIs, i.e., $rank(i, j) = 1$. (iii) We pick the top n ranked POIs from the list, where n is the number of POIs to be recommended. If $rank(i, j) \leq n$, we have a *hit*. Otherwise we have a *miss*. We consider the following two rank based evaluation metrics based on the notion of hits.

Table 6.1: MRR (mean ± standard error). $K = 10$

Methods	Yelp	Foursquare
Traditional recommendation algorithms		
PMF	0.0078 ± 0.0003	0.0092 ± 0.0004
CTR	0.0107 ± 0.0004	0.00112 ± 0.0006
Spatial recommendation algorithms		
STM	0.0102 ± 0.0003	0.0139 ± 0.0006
LCA	0.0159 ± 0.0007	0.0272 ± 0.0011
PAR	0.0254 ± 0.0010	0.0282 ± 0.0012
Our proposed algorithms		
LOC	0.0090 ± 0.0004	0.0119 ± 0.0004
CON	0.0155 ± 0.0005	0.0178 ± 0.0008
CRCF	$\mathbf{0.0690} \pm 0.0014$	$\mathbf{0.0521} \pm 0.0016$

Mean reciprocal rank (MRR) is the average of the reciprocal ranks of the test cases (i, j) in T, defined as

$$MRR = \frac{1}{|T|} \sum_{(i,j) \in T} \frac{1}{rank(i, j)} \tag{6.10}$$

A large value of MRR is desirable since a small value of $rank(i, j)$ gives a higher rank to a positive rating.

Recall@n is designed for top-n recommendation. As explained above, for each test case (i, j) in T, if $rank(i, j) \leq n$, we have one hit, meaning that the positively rated POI j is recommended to the user i. The portion of the test cases in T that that have a hit is defined by recall:

$$Recall@n = \frac{\#hit}{|T|} \tag{6.11}$$

where $\#hit$ is the number of hits for the test cases in T. A higher recall means that more positively rated POIs in T are recommended to a user.

Another metric used in Cremonesi et al.'s work [14] is Precision@n, which is equal to $\frac{Recall@n}{n}$. Since all methods are compared for the same n, it suffices to consider Recall@n.

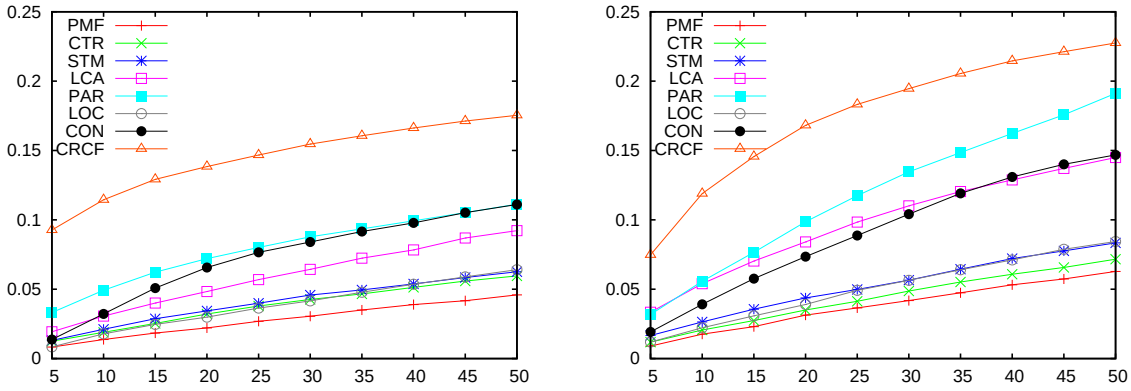
6.3.4 Experimental Results

MRR. Table 6.1 shows MRR of all methods for $K = 10$ regions. PMF performs worst since it uses only the rating data but ignores the feature and location information. The content-aware CTR and STM improve the performance over PMF. STM also considers the

location of POIs, but the improvement is limited on the Foursquare data set because the global recommendation of STM does not consider the user's current location. LCA achieves better results by modeling topic distribution for different regions and making personalized recommendation to suit the user's current region. To our surprise, PAR performs reasonably well, but it fails when the regional recommender has not enough ratings in the "new city" testing (Section 6.3.5). The significantly higher MRR of CRCF, compared to that of PAR, LOC and CON, suggests that considering both content and location information, as well as collaborative filtering across all regions, does improve the performance of POI recommendation.

We further examine the performance of the two best performers, CRCF and PAR. For all test cases, 97% of the top-10 POIs recommended by PAR are from the user's current region, and this percentage is around 30% for CRCF. The rest POIs recommended are from nearby regions. For example, for a test case (i, j) with positive rating on a Chinese restaurant j, CRCF recommends 4 restaurants in user i's current region and other 6 from nearby regions, and all these restaurants are associated with the Chinese food theme. However, PAR recommends all 10 restaurants from the user i's current region, and only 1 of them is related to the Chinese food theme. The higher MRR in the case of CRCF is because more POI j has a higher $rank(i, j)$ than in the case of PAR. In particular, the PAR recommender relies solely on the rating data in the region for user i, which is vulnerable to data sparsity, thus, tends to rank POIs incorrectly. We should mention that our evaluation is in favor of PAR that tends to recommend POIs from the current region because for each test case (i, j) in T, user i has actually visited POI j, so the hit j is always in the current region of user i. If the preferred POIs are not in the current region of the user, we expect that PAR will perform worse. CRCF captures the user's interest and location constraint by considering nearby regions if necessary.

Recall@n. Figure 6.4 shows recall@n. CRCF is the clear overall winner. PAR has the second best performance, followed by LCA. LOC, CTR and STM achieve quite similar results and PMF has the worst performance. The reason for the higher recall of CRCF is the same as for the higher MRR discussed above, i.e., there is a higher portion of hits for the test cases (i, j) in T. From a view of each component of CRCF, we find that CON already beats other algorithms considering content information, which shows our modeling for content preference is feasible. The other component LOC also has a comparable result to other algorithms, but generates a different ranking of POIs with CON (if not, the POIs keep almost no change after re-ranking), which leads to the higher recall of CRCF by the two



(a) Recall@n at Yelp

(b) Recall@n at Foursquare

Figure 6.4: Recall@n vs n (the x-axis). $K = 10$

stage ranking mentioned in Section 6.2.4. In other words, the superiority of CRCF comes from combining the location preference and content preference of users, and collaborative filtering over the rating data of all regions without discarding any data.

The effect of the seed size m. As discussed in Section 6.2.5, CRCF uses the seed size m to determine the top m POIs, ranked by content preferences, that participate in the final ranking by location preferences. A larger m gives flexibility to content preferences and more weight to location preferences. Table 6.2 reports the effect of the seed size m on MRR and recall for CRCF. For a smaller n, a better recall@n is achieved for a smaller m. In fact, a large m and a small n tend to include many less preferred POIs for location preference ranking, which degenerates into the under-performing LOC. For a large data set with over thousands of POIs, we believe that the setting of m at hundreds is a good choice.

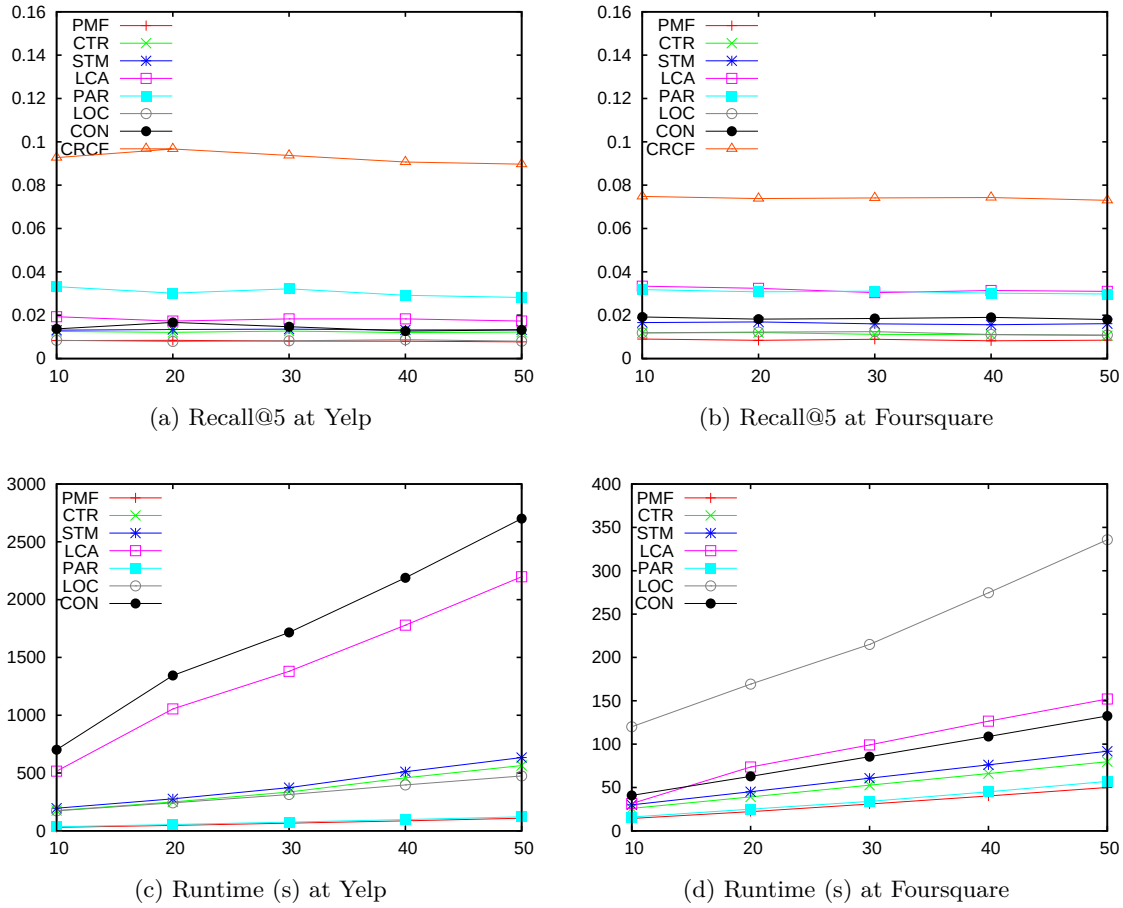
The effect of the dimensionality D. We examine the performance of different methods with respect to the number of dimensionality D. We show the results in Figure 6.5 where D is represented by the x-axis. From $D = 10$ to 50, we observe that all the methods achieve stable performances, with the runtime linearly increased. With D increased, there are more latent factors to infer so that the runtime is accordingly increased. For the Yelp data set which has more content information (each review has much more words than the tweet in the Foursquare data set), the content recommender CON takes the longest time, followed by the algorithms with topic modeling such as LCA and STM. For the Foursquare data set whose content information is less, however, the location recommender LOC takes the longest time since the ratings are replicated for K times.

Table 6.2: Performances of CRCF for different seed sizes m

	Yelp			
Seed size m	MRR	Recall@5	Recall@10	Recall@50
50	0.0585	0.0681	0.0786	0.1118
100	0.0682	0.0826	0.0993	0.1463
200	**0.0690**	0.0927	0.1145	0.1754
300	0.0644	**0.0941**	**0.1195**	0.1913
400	0.0597	0.0898	0.1191	0.1984
500	0.0566	0.0833	0.1183	**0.2045**
	Foursquare			
Seed size m	MRR	Recall@5	Recall@10	Recall@50
50	**0.0603**	**0.0932**	0.1212	0.1627
100	0.0588	0.0893	**0.1284**	0.1972
200	0.0521	0.0752	0.1193	**0.2279**
300	0.0452	0.0636	0.1017	0.2266
400	0.0409	0.0565	0.0897	0.2211
500	0.0373	0.0492	0.0819	0.2113

The effect of the number of regions K. We also examine the performance of CRCF and PAR with respect to the number of regions K. The result is shown in Figure 6.6 where K is represented by the x-axis. For all K tested, CRCF outperforms PAR. The performance of CRCF can be affected by K to a certain extent. The performance of PAR gradually improves as K increases except recall@50. Note that this result is obtained from our test cases (i, j) in T where i has actually visited POI j, therefore, user i and the POI j are always in the same region. Since a larger K produces smaller regions with fewer POIs and PAR tends to recommend local POIs, a larger K increases the chance of hits. However, a larger K also increases the possibility of visiting a "new city" and leads to less POIs in each region. If the hits are not in the region of the user, PAR is expected to perform less desirably.

The number of regions K also impacts the runtime of learning the location recommender of CRCF, shown in Table 6.3, because the matrix M_2 contains K virtual users for each real user. Since the runtime of matrix factorization is proportional to the number of observed ratings in the matrix and since M_2 contains a copy of distance-adjusted ratings for each region, the runtime for factorization based on M_2 is proportional to K. As the performance of CRCF is stable with a different K, we suggest that larger K is unnecessary. As discussed



(a) Recall@5 at Yelp

(b) Recall@5 at Foursquare

(c) Runtime (s) at Yelp

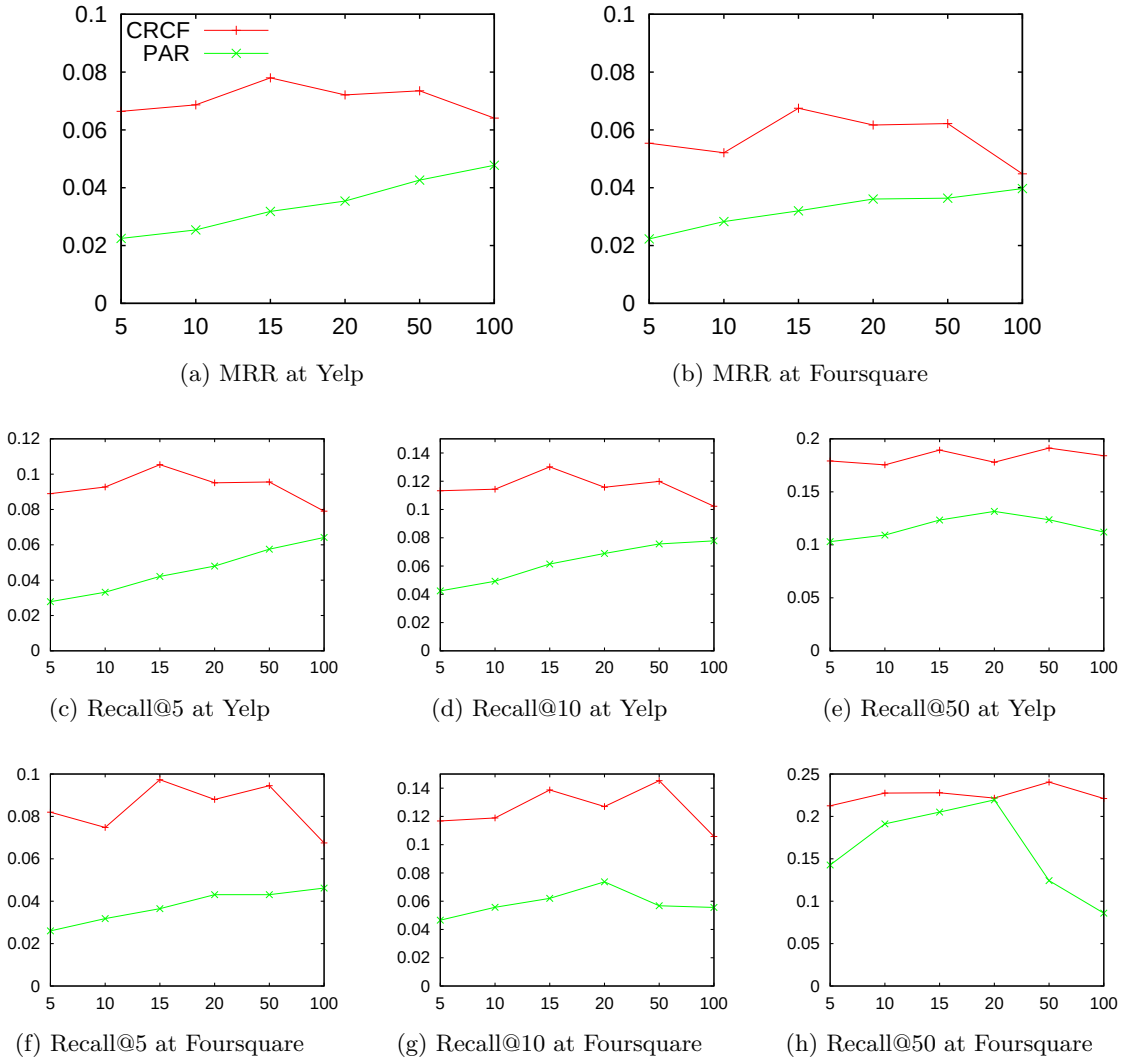
(d) Runtime (s) at Foursquare

Figure 6.5: Performance vs D (the x-axis). $K = 10$

in Section 6.2.5, a large geographical space can be partitioned into multiple small-to-medium geographical spaces and our method can be applied to each small-to-medium geographical space independently. Therefore, K does not have to increase as the size of the geographical space. Note that K does not affect the runtime of learning the content recommender of CRCF.

Table 6.3: Runtime of learning the location recommender of CRCF (min) vs K

K	5	10	15	20	50	100
Yelp	1.46	2.88	4.31	5.97	14.85	29.43
Foursquare	1.04	1.98	2.91	3.85	9.55	18.87



(a) MRR at Yelp (b) MRR at Foursquare

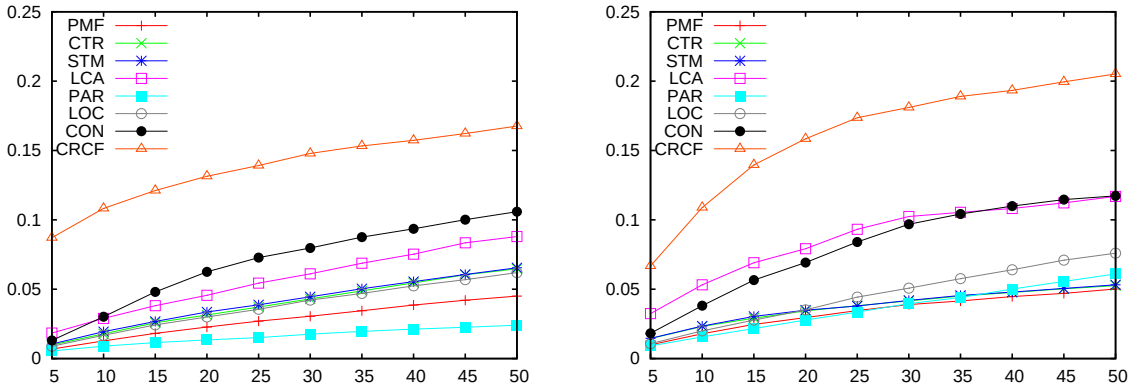
(c) Recall@5 at Yelp (d) Recall@10 at Yelp (e) Recall@50 at Yelp

(f) Recall@5 at Foursquare (g) Recall@10 at Foursquare (h) Recall@50 at Foursquare

Figure 6.6: Performances for different numbers of regions K

6.3.5 "New city" Testing

One claim about our method is the ability to deal with the "new city" problem, i.e., the recommendation to a user who visits a city or region for the first time, therefore, does not have any rating data in the city before. To evaluate this claim, we partition the POIs into $K = 10$ regions by K-means clustering and fix the region index, pick 90% of the observed ratings as the training set and take the rest as the test set such that, for each user, the region in the test data did not appear in the training data. In other words, each user has ratings in 9 regions as "home city" and the other region is treated as "new city". Note that the "home city" is not actually the home of user, but the regions where the user has already



(a) Recall@n at Yelp

(b) Recall@n at Foursquare

Figure 6.7: Recall@n vs n (the x-axis) in "new city" testing. $K = 10$

visited (even a few ratings), and the "new city" refers to the regions where the user has never visited before. In this way, we could conduct 10-fold cross validation to evaluate the performance of each method for a user visiting a "new city".

Figure 6.7 presents the recall@n of all methods in "new city" testing. A cross examination with Figure 6.4 shows that all the methods perform similarly in both general testing and "new city" testing except PAR. A large decrease in recall is observed for PAR, which is not surprising because PAR takes only a portion of training data while others take the whole training data to build the model. In the "new city" setting, the user never rated a POI in the region of testing data, so she is the cold user for PAR in that region. As PAR is the location-aware version of PMF, it definitely performs bad facing the cold users. The other methods, adopting the whole training data, can benefit the collaborative learning across different regions. From the "new city" testing, we suggest that a direct data partition and building regional models is unsatisfactory. Besides, only recommending the POIs in the current region like PAR is not practical.

6.4 Summary

This chapter proposed a cross-region collaborative filtering for location-based recommendation. The empirical studies on real life LBSN service data supported our claim that cross-region collaborative filtering helps address the vulnerability of data sparsity while factoring in the location influence and leads to more correct ranking of POIs. The superior performance of the proposed method also supported our approach of distinguishing user's

long-term content preferences and user's short-term location preferences, and the different strategies for modeling such preferences.

Chapter 7

Conclusion

Data mining in real life recommender systems is a hot topic due to large amount of available data in social media and user's high demands in personalized recommendation. The fundamental of recommender systems is to estimate user's preferences on items accurately, and comprehensive works have been done to predict the user's preferences based on the rating matrix. However, there exist different kinds of side information in real life applications beyond the rating matrix. In this thesis, we focused on recommender systems with rich side information. We believe that content information helps to recognize and explain the user interests in an explicit way; temporal information helps to model the evolution of user interests over time; spatial information helps to indicate the user mobility pattern and activity areas for feasible recommendation. The state-of-the-art treated side information as add-on to the traditional approaches for better performances, as reviewed in Chapter 2.

In this thesis, we proposed several novel methods to handle the side information in recommender systems. In Chapter 3, we proposed a feature-centric recommendation approach to model the feature-level preference because we believe that the user prefers an item due to some specific features on it. Different from previous works that are item-centric, our approach learns the feature-level preferences and the relevant weight for each feature, and then predicts user's preferences on items through the feature-level preference. Evaluation results on four data sets showed that the proposed approach demonstrated its superiority over item-centric approaches in terms of recommendation accuracy.

In Chapter 4, we studied the scientific paper recommendation in heterogeneous academic networks. We pointed out that both content and attributes (e.g., author, venue, etc.) could be a good indicator for paper recommendation and proposed a unified latent factor model to blend both content information and attribute information. Compared to the previous works

with no side information or only content information, our method achieved considerable improvement on DBLP data set.

In Chapter 5, we focused on the temporal effects in recommender systems where user's interests may change over time. We proposed temporal matrix factorization to model such dynamics by introducing and learning a transition matrix for each user's latent vectors. This transition matrix captures the time-invariant pattern of the evolution for user's preferences, and helps to predict user's adoption in future. Both temporal matrix factorization and its fully Bayesian version showed their effectiveness through the empirical studies on six data sets.

In Chapter 6, we studied the problem of recommending POIs to a user according to the user current location and past check-in activities, where the spatial information plays an important role to restrict the feasible areas for recommendation. We proposed a cross-region collaborative filtering to distinguish the user's long-term interests and short-term interests. The long-term interests are learnt through content recommmender which takes the advantage of the feature-centric recommendation, and the short-term interests are learnt through location recommender which considers the proximity of the POI location and the user's current location. The final recommendation is generated by the itemset re-ranking of content recommmender and location recommender. Experiment results shows the effectiveness of our methods on two location-based data sets.

This thesis is ended up with several promising directions for future work. We briefly discuss these directions:

- **Recommendation with constraints**

  Recommendation with constraints is rarely considered in literature, regardless of its practicability in real life applications. For example, there are limited resources in job recommendation and limited budget in travel recommendation. Some users may be unavailable to attend the social event due to the conflict of time so that this constraints should be considered in the event recommendation. Such constraints could be another kind of side information and well-designed recommender systems had better take these constraints into consideration.

- **More complex scenarios**

  This thesis basically dealt with each kind of side information in separate. Although the heterogeneous information is considered in Chapter 4 and both content information and spatial information is considered in Chapter 6, the recommender systems on

a daily basis may meet more complex scenarios. e.g., a recommender system with content, temporal, spatial and budget information. To this end, a comprehensive model taking all information into consideration or ensemble learning of different models in separate is a promising direction.

- **Sequential recommendation**

  Extending the current techniques for recommending a sequence of items in order instead of a single item is another possible future work. We believe that the order is an important factor to influence the user's preferences and the itemset itself should also match the user's interests. Such sequential recommendation can be applied to trip recommendation.

- **Evaluation metrics**

  Currently, the state-of-the-art measures the recommender system in terms of error-specific metrics (RMSE, MAE, etc.) or ranking-specific metrics (recall, MRR, etc.), which only focus on prediction accuracy. It is an interesting study to adopt other evaluation metrics such as diversity, serendipity and novelty for the efficacy of recommender systems.

- **Cold start problem**

  Cold start is a common issue in recommender systems. Although the cold start problem is not the main focus of this thesis, the feature-centric recommendation proposed in Chapter 3 can deal with it if the cold item shares some features already existed in the systems. Further research is still needed to resolve the cold start problem.

- **User feedback**

  User feedback is another kind of side information for improving the recommender systems. Since the user feedback reflects whether the user adopted the recommendation or not, the recommender system should be dynamically updated according to user feedback. Evolving the recommender system with user feedback is currently absent in literature and further research is needed in this direction.

Bibliography

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

[2] Natalia Adrienko and Gennady Adrienko. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):205–219, 2011.

[3] Deepak Agarwal and Bee-Chung Chen. Regression-based latent factor models. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 19–28. ACM, 2009.

[4] Deepak Agarwal and Bee-Chung Chen. flda: matrix factorization through latent dirichlet allocation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 91–100. ACM, 2010.

[5] Deepak Agarwal, Bee-Chung Chen, and Bo Pang. Personalized recommendation of user comments via factor models. In *EMNLP*, pages 571–582. Association for Computational Linguistics, 2011.

[6] Jie Bao, Yu Zheng, and Mohamed F Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 199–208, 2012.

[7] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[8] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *The Journal of Machine Learning Research*, 13(1):3619–3622, 2012.

[9] 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 Conference on Artificial Intelligence*, volume 12, pages 17–23, 2012.

[10] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *International Joint Conferences on Artificial Intelligence*, pages 2605–2611. AAAI Press, 2013.

[11] Zhiyuan Cheng, James Caverlee, Krishna Yeswanth Kamath, and Kyumin Lee. Toward traffic-driven location-based web search. In *ACM International Conference on Information and Knowledge Management*, pages 805–814, 2011.

[12] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090. ACM, 2011.

[13] Freddy Chong Tat Chua, Richard Oentaryo, and Ee-Peng Lim. Modeling temporal adoptions using dynamic matrix factorization. In *IEEE International Conference on Data Mining*, 2013.

[14] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Recsys*, pages 39–46. ACM, 2010.

[15] Yi Ding and Xue Li. Time weight collaborative filtering. In *ACM International Conference on Information and Knowledge Management*, pages 485–492, 2005.

[16] Yi Ding, Xue Li, and Maria E Orlowska. Recency-based collaborative filtering. In *Proceedings of the 17th Australasian Database Conference-Volume 49*, pages 99–107. Australian Computer Society, Inc., 2006.

[17] Matthias Fuchs and Markus Zanker. *Multi-criteria ratings for recommender systems: an empirical analysis in the tourism domain.* Springer, 2012.

[18] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, Steffen Rendle, and Lars Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *IEEE International Conference on Data Mining*, pages 176–185. IEEE, 2010.

[19] Fatih Gedikli and Dietmar Jannach. Rating items by rating tags. In *Proceedings of the 2010 Workshop on Recommender Systems and the Social Web at ACM RecSys*, pages 25–32, 2010.

[20] Fatih Gedikli and Dietmar Jannach. Improving recommendation accuracy based on item-specific tag preferences. *ACM Transactions on Intelligent Systems and Technology (TIST)*, pages 11:1–11:19, 2013.

[21] Zoubin Ghahramani and Michael I. Jordan. Factorial hidden markov models. *Machine learning*, 29(2-3):245–273, 1997.

[22] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[23] Eui-Hong Sam Han and George Karypis. Feature-based recommendation system. In *ACM International Conference on Information and Knowledge Management*, pages 446–452. ACM, 2005.

[24] Liangjie Hong, Aziz S Doumith, and Brian D Davison. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *ACM International Conference on Web Search and Data Mining*, pages 557–566. ACM, 2013.

[25] Bo Hu and Martin Ester. Spatial topic modeling in online social media for location recommendation. In *RecSys*, pages 25–32, 2013.

[26] Mohsen Jamali and Martin Ester. A transitivity aware matrix factorization model for recommendation in social networks. In *International Joint Conferences on Artificial Intelligence*, pages 2644–2649, 2011.

[27] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[28] 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. ACM, 2010.

[29] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[30] Yehuda Koren. Collaborative filtering with temporal dynamics. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 447–456, 2009.

[31] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[32] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492. University of California Press, 1951.

[33] Takeshi Kurashima, Tomoharu Iwata, Takahide Hoshide, Noriko Takaya, and Ko Fujimura. Geo topic model: joint modeling of user's activity area and interests for location recommendation. In *ACM International Conference on Web Search and Data Mining*, pages 375–384, 2013.

[34] Neal Lathia, Stephen Hailes, and Licia Capra. Evaluating collaborative filtering over time. In *Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation*, pages 41–42. Citeseer, 2009.

[35] Neal Lathia, Stephen Hailes, and Licia Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *ACM SIGIR conference on Research and development in information retrieval*, pages 796–797, 2009.

[36] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *ACM SIGIR conference on Research and development in information retrieval*, pages 210–217. ACM, 2010.

[37] Tong Queue Lee, Young Park, and Yong-Tae Park. A time-based approach to effective recommender systems using implicit feedback. *Expert systems with applications*, 34(4):3055–3062, 2008.

[38] Tong Queue Lee, Young Park, and Yong-Tae Park. An empirical study on effectiveness of temporal information as implicit ratings. *Expert Systems with Applications*, 36(2):1315–1321, 2009.

[39] Kenneth Wai-Ting Leung, Dik Lun Lee, and Wang-Chien Lee. Clr: a collaborative location recommendation framework based on co-clustering. In *ACM SIGIR conference on Research and development in information retrieval*, pages 305–314, 2011.

[40] Justin J Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F Mokbel. Lars: A location-aware recommender system. In *ICDE*, pages 450–461. IEEE, 2012.

[41] Bin Liu, Yanjie Fu, Zijun Yao, and Hui Xiong. Learning geographical preferences for point-of-interest recommendation. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1043–1051, 2013.

[42] Bin Liu and Hui Xiong. Point-of-interest recommendation in location based social networks with topic and location awareness. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 396–404, 2013.

[43] Zhengdong Lu, Deepak Agarwal, and Inderjit S Dhillon. A spatio-temporal approach to collaborative filtering. In *Proceedings of the third ACM conference on Recommender systems*, pages 13–20. ACM, 2009.

[44] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In *ACM International Conference on Information and Knowledge Management*, pages 931–940. ACM, 2008.

[45] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *ACM International Conference on Web Search and Data Mining*, pages 287–296. ACM, 2011.

[46] Shanle Ma, Xue Li, Yi Ding, and Maria E Orlowska. A recommender system with interest-drifting. In *International Conference on Web Information Systems Engineering*, pages 633–642. Springer, 2007.

[47] Benjamin Marlin and Richard S. Zemel. The multiple multiplicative factor model for collaborative filtering. In *International Conference on Machine Learning*, pages 73–80, 2004.

[48] Julian McAuley and Jure Leskovec. From amateurs to connoisseurs: Modeling the evolution of user expertise through online reviews. In *International World Wide Web Conference*. ACM, 2013.

[49] Radford M. Neal. Probabilistic inference using markov chain monte carlo methods. *Technical Report CRG-TR-93-1*, 1993.

[50] Michael J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Reivew*, 13:393–408, 1999.

[51] Herbert E Rauch, CT Striebel, and F Tung. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

[52] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57:1–57:22, 2012.

[53] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B Kantor. *Recommender systems handbook*, volume 1. Springer, 2011.

[54] Roberto Rösler and Thomas Liebig. Using data from location based social networks for urban activity clustering. In *Geographic Information Science at the Heart of Europe*, pages 55–72. Springer, 2013.

[55] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *International Conference on Machine Learning*, pages 880–887, 2008.

[56] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1257–1264, 2008.

[57] Jitao Sang, Tao Mei, Jian-Tao Sun, Changsheng Xu, and Shipeng Li. Probabilistic sequential pois recommendation via check-in data. In *ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 402–405, 2012.

[58] Shilad Sen, Jesse Vig, and John Riedl. Tagommenders: connecting users to items through tags. In *International World Wide Web Conference*, pages 671–680. ACM, 2009.

[59] Yelong Shen and Ruoming Jin. Learning personal+ social latent factor model for social recommendation. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1303–1311. ACM, 2012.

[60] Alex J Smola and Bernhard Scholkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[61] John Z Sun, Kush R Varshney, and Karthik Subbian. Dynamic matrix factorization: A state space approach. In *ICASSP*, pages 1897–1900, 2012.

[62] Waldo R Tobler. A computer movie simulating urban growth in the detroit region. *Economic geography*, pages 234–240, 1970.

[63] Karen HL Tso-Sutter, Leandro Balby Marinho, and Lars Schmidt-Thieme. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 1995–1999. ACM, 2008.

[64] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Technical Report TCD-CS-2004-15, Trinity College Dublin*, 2004.

[65] Chong Wang and David M. Blei. Collaborative topic modeling for recommending scientific articles. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 448–456, 2011.

[66] Edwin B Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[67] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal recommendation on graphs via long-and short-term preference fusion. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 723–732, 2010.

[68] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G Schneider, and Jaime G Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SIAM International Conference on Data Mining*, volume 10, pages 211–222, 2010.

[69] Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. Like like alike: joint friendship and interest propagation in social networks. In *International World Wide Web Conference*, pages 537–546. ACM, 2011.

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

[71] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik-Lun Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *ACM SIGIR conference on Research and development in information retrieval*, pages 325–334, 2011.

[72] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. Lcars: a location-content-aware recommender system. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 221–229, 2013.

[73] Jing Yuan, Yu Zheng, and Xing Xie. Discovering regions of different functions in a city using human mobility and pois. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 186–194, 2012.

[74] Chenyi Zhang, Ke Wang, Ee-peng Lim, Qinneng Xu, Jianling Sun, and Hongkun Yu. Are features equally representative? a feature-centric recommendation. In *AAAI Conference on Artificial Intelligence*, pages 389–395, 2015.

[75] Yi Zhen, Wu-Jun Li, and Dit-Yan Yeung. Tagicofi: tag informed collaborative filtering. In *Recsys*, pages 69–76. ACM, 2009.

[76] Vincent W Zheng, Yu Zheng, Xing Xie, and Qiang Yang. Collaborative location and activity recommendations with gps history data. In *International World Wide Web Conference*, pages 1029–1038, 2010.

[77] Yu Zheng, Lizhu Zhang, Zhengxin Ma, Xing Xie, and Wei-Ying Ma. Recommending friends and locations based on individual location history. *ACM Transactions on the Web*, 5(1):1–44, 2011.

[78] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *International World Wide Web Conference*, pages 791–800, 2009.

[79] Tom Chao Zhou, Hao Ma, Irwin King, and Michael R Lyu. Tagrec: Leveraging tagging wisdom for recommendation. In *International Conference on Computational Science and Engineering*, volume 4, pages 194–199. IEEE, 2009.