School of Computing

ENHANCING COLLABORATIVE FILTERING MUSIC RECOMMENDATION BY BALANCING EXPLORATION AND EXPLOITATION

XING ZHE
(B.Eng., Renmin University of China)
2012

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
SCHOOL OF COMPUTING
NATIONAL UNIVERSITY OF SINGAPORE

2014

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

---

19 August 2014

Abstract

Collaborative filtering (CF) techniques have shown great success in music recommendation applications. However, traditional collaborative-filtering music recommendation algorithms work in a *greedy* way, invariably recommending songs with the highest predicted user ratings. Such a purely *exploitative* strategy may result in suboptimal performance over the long term. Using a reinforcement learning approach, we introduce *exploration* into CF and try to strike a balance between exploration and exploitation. In order to learn users' musical tastes, we use a Bayesian graphical model that takes account of both CF latent factors and recommendation novelty. Moreover, we designed a Bayesian inference algorithm to efficiently estimate the posterior rating distributions. To the best of our knowledge, this is the first attempt to remedy the greedy nature of CF approaches in music recommendation. Results from both simulation experiments and user study show that our proposed approach significantly improves music recommendation performance.

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Wang Ye, who has encouraged and supported me with great patience, to senior PhD student Wang Xinxi, who has provided valuable ideas and suggestions throughout this project, and to Haotian "Sam" Fang for proofreading my thesis. I am also grateful to the subjects in our user study for their time and effort. Lastly, I would like to thank everyone who has generously helped me throughout my studies at School of Computing, National University of Singapore.

This study is funded by the National Research Foundation (NRF) and managed through the multi-agency Interactive & Digital Media Programme Office (IDMPO) hosted by the Media Development Authority of Singapore (MDA) under Centre of Social Media Innovations for Communities (COSMIC).

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Motivation

Internet has dramatically changed the way people consume music. Nowadays, we can easily access a large amount of music collections via the Internet. However, given such huge music data, how to quickly find musical pieces that we like becomes a critical problem. Using music retrieval systems, we have to think about appropriate queries and execute queries repeatedly by ourselves [49]. These poor user experiences have created needs for music recommendation services. In order to save user's time and effort in music discovery and to satisfy user's different musical preferences, numerous music recommender systems (*e.g.*, Pandora[1], Last.fm[2], Allmusic[3] and Songza[4]) have emerged and shown increasing importance in our daily lives. These music recommender systems are trying their best to automatically identify user's musical preferences and accordingly recommend

[1] http://www.pandora.com/
[2] http://www.last.fm/
[3] http://www.allmusic.com/
[4] http://daily.songza.com/

probably-preferred songs from large scale music databases.

Various music recommendation algorithms can be classified into five categories: metadata-based [32, 40], content-based [9, 28, 29], collaborative filtering (CF) [21, 26], context-based [25, 34, 45] and hybrid methods [41, 43, 48, 49]. Among all these categories, content-based approaches and collaborative filtering (CF) approaches have been the most traditional and prevailing recommendation strategies.

Content-based music recommendation algorithms analyze acoustic features of the songs that target user has rated highly in the past. They then recommend only the songs that have a high degree of acoustic similarity to the user's favorites. On the other hand, collaborative filtering (CF) music recommendation algorithms assume that people tend to get good recommendations from someone with similar preferences. People who share similar preferences are called "near neighbors". The target user's ratings are predicted according to his neighbors' ratings, and then songs rated highly by the neighbors but not yet considered by the target user will be recommended to him.

These two traditional music recommendation approaches, however, share a common weakness. They always generate "safe" recommendations by selecting songs with the highest predicted user ratings, and such a purely *exploitative* strategy may result in suboptimal performance over the long term due to the lack of *exploration*. Selecting a song with the highest predicted user rating is called a *greedy* recommendation, and the recommender system is *exploiting* its current knowledge about the target user's preference. If instead the recommender system selects one of the non-greedy recommendations, we say that it is *exploring* because this can enable the

recommender system to improve its prediction about the target user's true preference for the recommended non-greedy song.

To understand why greedy recommendation strategy is not good enough and may result in suboptimal performance over the long term, we will first offer an intuitive explanation here then give more details in Chapter 2.2.

In a music recommendation algorithm, the user preference is only estimated based on the current rating information available in the recommender system. As the predicted user ratings are estimators of the true user ratings, they are intrinsically inaccurate. As a result, *uncertainty* always exists in the predicted user ratings and may give rise to a situation where some of the non-greedy recommendations deemed almost as good as the greedy ones are actually better than them. Without exploration, however, we will never know which ones are better. With the appropriate amount of exploration, the recommender system could gather more rating data and gain more knowledge about the user's true preferences before using them for recommendation. Therefore, rather than merely exploiting the rating data available, a smarter recommender system prefers to explore user preferences actively. At the same time, the key to achieving better recommendation performance is to balance exploration and exploitation.

Currently, the literature of music recommendation research has rarely addressed the weakness of purely exploitative strategies. Wang *et al.* [46], only recently tried to mitigate the greedy problem in content-based music recommendation algorithms. However, no work has tackled this problem in the collaborative filtering (CF) context.

We are thus motivated to remedy the greedy nature of collaborative filtering (CF) approaches in the music recommendation context. We aim

to develop a CF-based music recommendation algorithm that can strike a balance between exploration and exploitation in order to enhance long-term recommendation performance.

To do so, we introduce exploration into collaborative filtering by formulating the music recommendation problem as a reinforcement learning task called *n-armed bandit* problem [39]. A Bayesian graphical model taking account of both collaborative filtering latent factors and recommendation novelty is proposed to learn the user preferences. The lack of efficiency becomes a major challenge, however, when we adopt an off-the-shelf Markov Chain Monte Carlo (MCMC) sampling algorithm[5] for the Bayesian posterior estimation. We are thus prompted to design a much faster sampling algorithm for Bayesian inference. We carried out both simulation experiments and a user study to show the efficiency and effectiveness of our proposed approach.

1.2 Contributions

The main contributions of this thesis are summarized as follows[6]:

- To the best of our knowledge, this is the first work in music recommendation to temper CF's greedy nature by investigating the exploration-exploitation trade-off using a reinforcement learning approach.

- Compared to an off-the-shelf MCMC algorithm, a much more efficient sampling algorithm is proposed to speed up Bayesian posterior estimation.

[5] http://mcmc-jags.sourceforge.net/

[6] Preliminary results of our work have been published in Proceedings of ISMIR 2014 [47].

- Experimental results from both simulation experiments and user study show that our proposed approach enhances the performance of CF-based music recommendation significantly.

1.3 Organization

The rest of the thesis is organized as follows. Chapter 2 reviews related work and introduces necessary background knowledge. Chapter 3 describes our proposed algorithm in detail. Chapter 4 presents evaluation results. We summarize this work and discuss some of the limitations in Chapter 5. Potential future research directions are suggested in Chapter 6.

Chapter 2

Related Work

In this chapter, we will give a literature survey on existing work that is relevant to our proposed approach. Necessary background knowledge will also be introduced.

2.1 Music Recommendation

In the past decade, online music recommendation services have been gaining popularity and significance. Music recommender systems try to identify a user's musical taste and automatically recommend songs from a huge database in order to satisfy the user's preference. The key to user satisfaction and loyalty is matching users with their most preferred songs.

Problem Formulation: Most commonly, a music recommendation problem can be formulated as follows. In a music recommender system, there are m users and n songs. Let $\mathbf{R} = \{r_{ij}\}_{m \times n}$ denote the user-song interaction matrix. There are two types of interaction data. One type is high-quality *explicit feedback* data, which directly indicates user's interest in songs, including ratings of songs given by users, or like/dislike opinions

	Angel	Believe	Cherish	Friday	My Love
Amy	5			2	4
Sam	4	2	5	3	3
Helen	3	3	2		1
Tom	3		4		

Table 2.1: A fragment of the user-song rating matrix for a music recommender system.

about songs given by users. The other type is *implicit feedback* data, which indirectly reflects user preferences for songs, including listening history or search patterns. Apart from interaction data, we may also have additional song metadata (*e.g.*, song title, artist name and genre tag) or user demographic data (*e.g.*, user's age, gender and occupation). Table 2.1 shows an example of a user-song rating matrix (explicit feedback data), where each rating is on a scale of 1 (weakest preference) to 5 (strongest preference). The empty cells in the table mean that the users have not rated the corresponding songs.

The major task of a music recommender system is to predict the ratings of the non-rated user/song pairs based on all the information available in the system and then generate appropriate recommendations according to the predicted ratings. Therefore, the most important two components of a recommender system are the *prediction* component and the *recommendation* component. Different algorithms and strategies used in these two components will make a huge difference in the overall recommendation quality of the system.

In the following sections, we will summarize some state-of-the-art approaches used in music recommender systems and discuss their strengths and weaknesses.

2.1.1 Metadata-based Approaches

In different music data collections [5, 14], various types of metadata information are associated with the music audio files, including title of the song, album name, band or artist's name, music genre, lyrics, year of release, and much more. They are described using textual information and are supplied by experts or the creators [13]. The main idea of metadata-based music recommendation approaches [32, 40] is very intuitive: analyze the metadata of the songs that have been given high ratings by the target user, and then apply fundamental information retrieval techniques to search for musical pieces that belong to similar albums, artists or genres.

Advantages: Metadata-based approaches are based on text processing [35] and information retrieval [3]. These two research directions have been extensively studied so that many existing techniques can be easily implemented and applied to the recommender system. In addition, a genre-based music recommendation approach alone can achieve decent recommendation accuracy because most users often like to listen to a limited number of music genres.

Limitations: Creating and collecting metadata information is time-consuming and requires expertise knowledge, therefore, metadata is not always available in the recommender system. With the emergence and development of Web 2.0, social media websites (*e.g.*, Last.fm[1]) allow users to create tags for albums, songs and artists, which has significantly enriched the metadata information. However, at the same time, user-generated tags have also introduced a lot of noise into the metadata and brought difficulties into text analysis. Another limitation of metadata-based approaches is

[1] http://www.last.fm/

that they may easily lead to predictable recommendations. For example, recommending songs by artists that the target user already knows well does not show the power of recommendation because it fails to give any interesting surprise to the user.

2.1.2 Content-based Approaches

Content-based music recommendation algorithms [9,28,29] analyze acoustic features of music that the target user has rated highly in the past. Then, only the music that has a high degree of acoustic similarity to the user's favorites would be recommended. Commonly used audio features include Mel Frequency Cepstral Coefficient (MFCC), Zero Crossing Rate, Chroma, Spectral Centroid, Spectral Flux, and so on.

Advantages: Since music audio files already exist in the music recommender system, no additional data or information sources are required in the content-based recommendation approaches. When there is no metadata or user-song interaction data available in the recommender system, a content-based approach becomes an optimal choice.

Limitations: Content-based techniques are limited by the audio features selected. It is difficult to determine which underlying acoustic features are suitable and effective in music recommendation scenarios, because these features were not originally designed for music recommendation. With the development of deep learning techniques, this problem will hopefully be solved in the near future [44]. Another shortcoming is that the music recommended by content-based methods often lack variety, because they are all supposed to be acoustically similar to each other. Ideally, the user should be provided with a range of music from different genres rather than a homo-

geneous set [1]. In addition, purely content-based music recommendation algorithms are typically far from satisfactory due to the serious semantic gap between low-level audio features and high-level user preferences.

2.1.3 Collaborative Filtering (CF) Algorithms

Collaborative filtering methods automatically make predictions about the preferences of the target user by collecting preference information from many other like-minded users. They are based on the assumption that if user A has the same interests as user B in an item, then the items liked by user B are very likely to satisfy user A's preferences. Actually, this strategy is commonly used by people in daily life because we usually ask opinions and advice from others who have similar preferences. To some extent, collaborative filtering is a method that simulates and automates the word-of-mouth recommendation process in real life. Various collaborative filtering (CF) algorithms are usually classified into two general classes, namely memory-based (also called neighborhood-based) CF and model-based CF [7].

Memory-based CF algorithms [16,17,22,36] compute recommendations directly based on the entire raw rating data in the recommender system. They rely on some heuristic similarity measures between users or items. According to the similarity measure used, memory-based CF can be further divided into two categories: user-oriented and item-oriented. User-oriented CF methods [16,17,22] rely on the similarity measure between users. They first search for neighbors who have similar rating histories to the target user. Then the target user's ratings can be estimated as weighted average of his neighbors' ratings. Finally, songs with the highest predicted ratings will

be recommended. In contrast, item-oriented CF methods [36] rely on the similarity measure between items. They recommend songs that are rated similarly to the ones for which the target user has shown strong preference. The item-oriented CF algorithm has been used in the world's largest online retailer, Amazon[2].

In contrast to memory-based CF, model-based CF algorithms [21, 23, 31, 52] work in a different fashion as recommendations are not directly computed based on the collection of raw rating data. Using various machine learning and data mining techniques, a model is first learned in order to discover latent factors that account for the observed ratings, which is then used to predict unknown ratings. Model-based CF algorithms have shown prominent prediction power in some well-known competitions of recommendation tasks (*e.g.*, the Netflix Prize Challenge [4], the Yahoo! Music KDD-Cup [14] and the Million Song Dataset Challenge [30]).

Advantages: Collaborative filtering has gained great success in online recommender systems. It is acknowledged that collaborative filtering approaches are the most prevailing and popular algorithms being used in existing recommendation services. Compared to other algorithms, collaborative filtering usually achieves better recommendation accuracy.

Limitations: Even though collaborative filtering tends to achieve higher recommendation accuracy, it suffers from three notorious drawbacks: *cold-start*, *data sparsity* and *scalability* problem. The first two problems are related to each other. In the prediction phase, a sufficient amount of rating data is required to search for near neighbors or learn a decent model. When a new user or a new item is first introduced into the recommender

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

system, there is no interaction data for it at all and thus results in the cold-start problem. Even for existing users or items, without enough rating data available, recommendation quality of the CF algorithm will degrade substantially. Additionally, the computational bottleneck in conventional memory-based CF is the search for neighbors among a large user population of potential neighbors. Thus, improving the efficiency of the recommendation algorithm and solving the scalability problem is also challenging.

2.1.4 Context-aware Approaches

Traditional music recommender systems focus on satisfying long-term user preferences, but context-aware approaches put more emphasis on user's current context (*e.g.*, user's mood [25], activity [45], location [37] and Web documents the user is reading [8]). Context-based recommendation algorithms detect or infer the user's current context and then recommend songs that match the user's current context.

Advantages: User musical preferences are complicated, and they are a combined result of many external and internal factors. Therefore, different environments will lead to different user preferences. Context-aware recommendation approaches are getting increasingly popular because they aim to satisfy short-term user preferences. In addition, the dramatic expansion of mobile internet and mobile devices creates new needs and opportunities for context-based recommendation algorithms.

Limitations: Contextual data is not always available in the recommender system, and sometimes people are reluctant to provide their environmental information (*e.g.,* geospatial data). Currently, automatically detecting and inferring a user's context is inaccurate. More effort is needed to

improve the relevant techniques. Another limitation is that context-based recommender systems require additional devices to finish the recommendation task (*e.g.,* sensor and smart phone).

2.1.5 Hybrid Methods

Hybrid recommendation is a method that combines two or more different recommendation approaches together. Hybrid methods [41, 43, 49] highlight the necessity of following multimodal approaches so as to alleviate limitations of methods that solely depend on audio content or user rating data. Yoshii *et al.* [49] use a probabilistic graphical model to combine content-based and collaborative filtering music recommendation algorithms. Tiemann *et al.* [43] combine a content-based and a social recommendation algorithm using ensemble learning methods. A recent work by Tan *et al.* [41] creatively uses a hypergraph model to combine rich social media information including six different types of objects and nine different types of relations for music recommendation.

Advantages: Since hybrid recommendation methods combine multiple techniques, they can overcome the shortcomings of solely using one class of recommendation approach. Thus, hybrid recommendation approaches often achieve better recommendation performance.

Limitations: Hybrid methods require different data sources, which increases the difficulty in collecting data. In addition, combining multiple approaches often results in a very complicated model, thus efficiency issues become a critical problem.

2.1.6 Summary

In summary, according to the approaches used in the *prediction* phase, various music recommendation algorithms can be classified into the five categories introduced in the previous sections. Table 2.2 presents a summary of these algorithms.

<table>
<tr><th colspan="5">Music Recommendation Algorithms</th></tr>
<tr><th colspan="3">Category</th><th>Data</th><th>Advantages</th><th>Limitations</th></tr>
<tr><td colspan="3">Metadata-based</td><td>song title, album name, artist name, genre, …</td><td>easy to implement, high efficiency</td><td>difficult data collection, require expertise knowledge, noise in the free text, difficult to verify information correctness, predictable recommendations</td></tr>
<tr><td colspan="3">Content-based</td><td>music audio files</td><td>no additional data is required</td><td>difficult to select effective features, huge semantic gap, lack variety</td></tr>
<tr><td rowspan="3">Collaborative Filtering</td><td rowspan="2">Memory-based</td><td>User-oriented</td><td rowspan="3">user-song interaction data (explicit feedback or implicit feedback)</td><td rowspan="3">high recommendation accuracy and quality</td><td rowspan="3">cold-start, data sparsity, scalability problem</td></tr>
<tr><td>Item-oriented</td></tr>
<tr><td colspan="2">Model-based</td></tr>
<tr><td colspan="3">Context-aware</td><td>geospatial data, environmental sound, weather, surrounding text, …</td><td>satisfy short-term user preferences</td><td>require specific devices, difficult data collection, inaccurate context detecting and inferring</td></tr>
<tr><td colspan="3">Hybrid Methods</td><td>all types of data listed above, social data (friendship relations, affinity group membership relations, …)</td><td>high recommendation accuracy and quality</td><td>difficult data collection, complex model, efficiency issues</td></tr>
</table>

Table 2.2: A summary of various music recommendation algorithms.

2.2 Greedy Recommendation Strategy

Chapter 2.1 reviews five different categories of music recommendation algorithms. The major differences between these recommendation approaches lie in the *prediction* phase of the algorithms. However, no matter what different methods are used in the prediction phase, various recommendation algorithms adopt almost the same strategy in the *recommendation* phase: rank the candidate songs according to their predicted ratings and then recommend the songs with the highest predicted ratings (some recommender systems may also generate a list of top-N recommended songs). We call this strategy a *greedy* recommendation strategy.

It seems reasonable to recommend the songs with the highest predicted ratings because people assume that it can maximize user satisfaction. Now the greedy recommendation strategy is very popular in existing music recommender systems, so much so that many system designers fail to notice the drawbacks of the greedy strategy.

Since the predicted ratings are estimated values based on the data available in the recommender system, they always carry uncertainty. This uncertainty may result in a situation where the target user may probably show stronger preference for a non-greedy song than the greedy song. Therefore, over the long term, the greedy recommendation strategy may lead to suboptimal performance. To better illustrate this point, we will give a simple example in subsequent sections.

2.2.1 A Probabilistic Perspective

Before introducing a concrete example, we first need to reconsider the music recommendation problem from a probabilistic perspective due to the ever-existing uncertainty.

In the music recommender system, a user can listen to a song multiple times. Affected by a broad range of external and internal factors (*e.g.*, mood, location and activity), different ratings may be given by the target user each time he listens to the same song. Therefore, we can treat the user rating as a random variable with an underlying probability distribution which is unknown to the recommender system. Commonly, we can assume that the underlying probability distribution is a normal distribution.

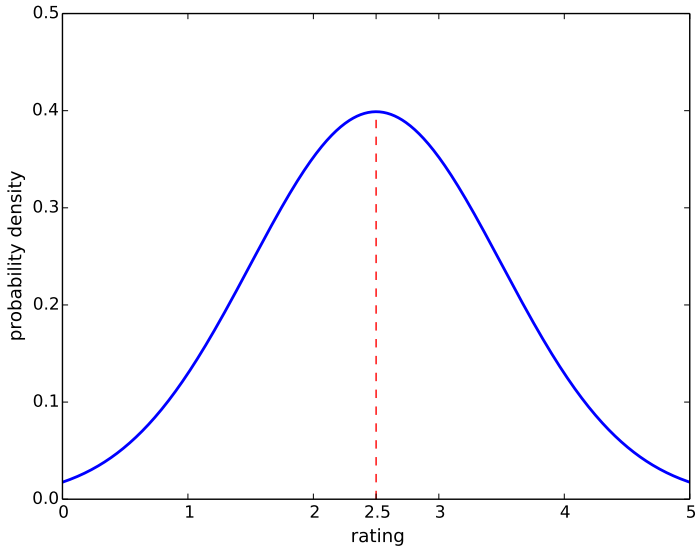


Figure 2.1: An example of the underlying probability distribution of the user rating. This is a normal distribution with mean $\mu = 2.5$ and variance $\sigma^2 = 1$.

Figure 2.1 shows a normal distribution, where the mean μ is 2.5 and the variance σ^2 is 1. It conveys a piece of information that no matter how

the user will be affected by all the complicated factors, over the long term, it can be expected that on average he will give this song a rating of 2.5. The mean μ is a very important unknown parameter that the recommender system cares about because the mean is the expected rating that the user is likely to give to the song. Since the mean is unknown, the major task of the music recommender system is thus to *estimate* the mean of the user rating for each candidate song j. These predicted mean ratings then become the important knowledge the recommender system relies on so as to make appropriate recommendation.

Following a greedy strategy, the system merely *exploits* its current knowledge and recommends the song with the highest predicted mean rating (*i.e.* recommend song j^* that has maximum estimated mean rating $\hat{\mu}_{j^*}$).

2.2.2 Bayesian Estimation

In the prediction (or estimation) process, a Bayesian method is usually preferred over a Frequentist method, because the Bayesian method can represent uncertainty about the unknown parameter [6]. Bayesian estimation uses probability to quantify the uncertainty, thus the unknown parameter is treated as a random variable rather than a fixed value. Bayesian method also allows us to inject our *priori* knowledge of the estimated parameter, and then use *evidence* (*i.e.* the observed data) to update and refine our estimation of the parameter.

Figure 2.2 shows an example of Bayesian estimation process. Suppose we want to estimate the mean of a Gaussian distribution (the correct mean is 0.8). At the beginning, our initial prior distribution (a Gaussian dis-

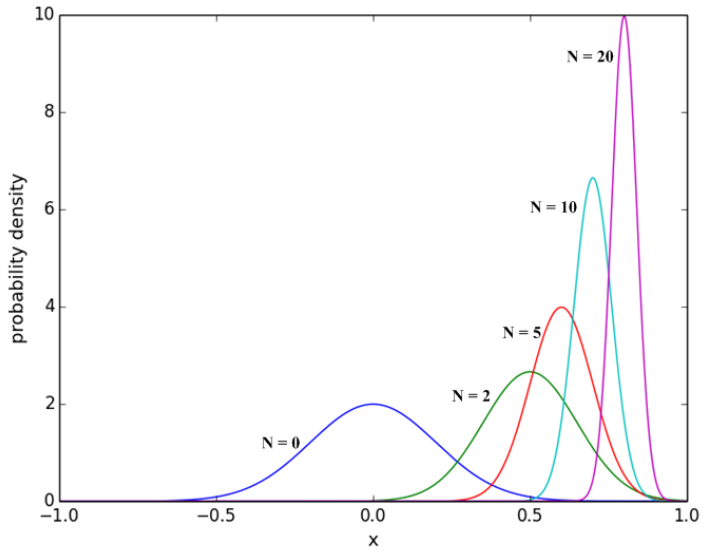


Figure 2.2: An example of Bayesian estimation. N is the number of observed data samples. As we gradually get more observed data (*i.e.* N becomes larger), the estimated mean gets closer to the correct value 0.8, the posterior distribution becomes sharper, and the variance gets smaller.

tribution with mean = 0) may be a very flat and broad (*i.e* with big variance) distribution. As we gradually collect more observed data to perform Bayesian update, the estimated mean shifts toward the true value, the posterior distribution (*i.e.* our estimation of the parameter given the data) is sharpened, and the variance becomes smaller, which means that we are getting more confident about our estimation.

Due to the advantages of Bayesian estimation over Frequentist estimation, we will adopt a Bayesian method to estimate the expected ratings of songs in all subsequent examples.

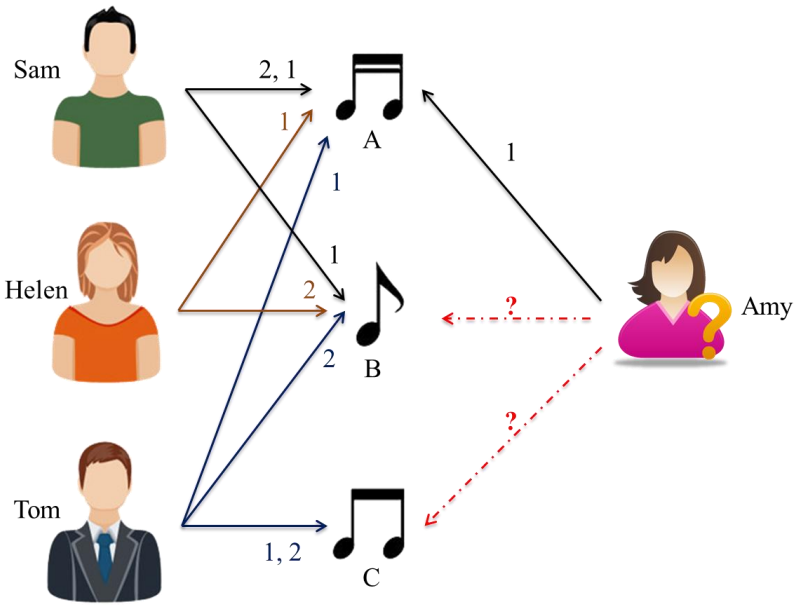
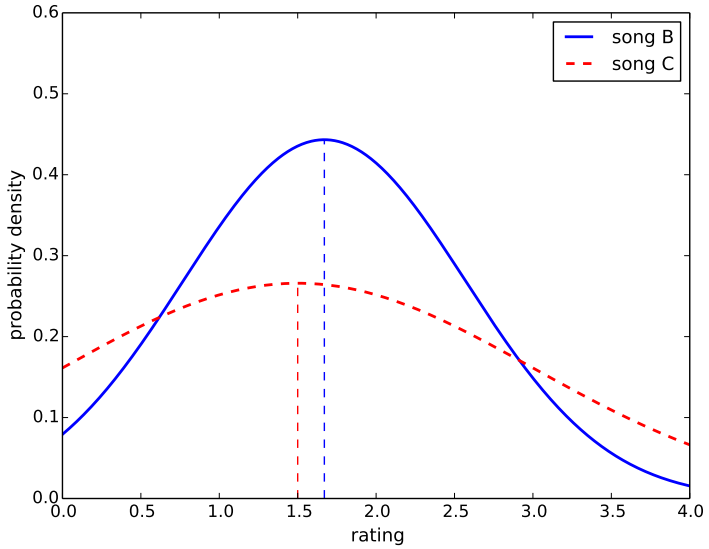


Figure 2.3: A simple example of the music recommender system.

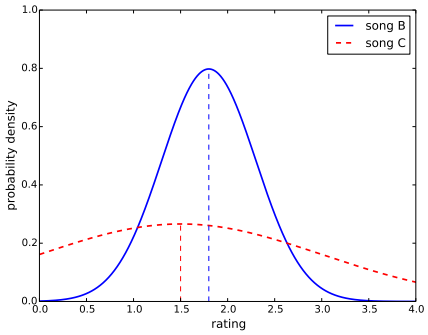
2.2.3 Limitations of The Greedy Strategy

As shown in Figure 2.3, there are four users {Sam, Helen, Tom, Amy} and three songs {A, B, C} in the music recommender system. Suppose Amy is the target user, and the recommender system is going to recommend a song from two candidate songs {B, C} to Amy. Sam has listened to song A twice, and the two ratings he has given to song A are 2 and 1. Similarly, Tom has listened to song C twice, and ratings are 1 and 2.

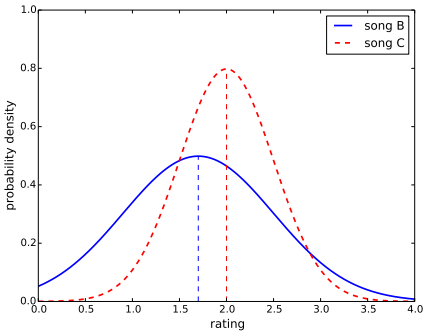
Since no interaction data between Amy and the candidate songs is available in the system, based on the idea of collaborative filtering, the recommender system collects preference information from other users to make rating predictions about the candidate songs {B, C}. Thus the predicted mean ratings for song B and song C are 1.667 and 1.5, respectively. Figure 2.4a shows the estimated posterior distribution of the mean rating. Suppose the true expected ratings for song B and C are 1.8 and 2, respectively. A



(a) The initial estimation of the mean rating.



(b) Our estimation of the mean rating after several runs of update under greedy strategy.



(c) Our estimation of the mean rating after several runs of update under non-greedy strategy.

Figure 2.4: Our estimation of the mean rating under different recommendation strategies.

greedy strategy will recommend song B to Amy. After collecting Amy's rating feedback for song B, the predicted rating will approach the correct value 1.8 (see Figure 2.4b). Then song B always has a higher predicted rating than song C, therefore, the greedy strategy keeps recommending song B and never has a chance to recommend song C so as to find out its true

expected rating. Since song C actually has a higher expected rating than song B, over the long term, the greedy strategy can only achieve suboptimal performance.

At the beginning, variance in the predicted rating of song C is larger than song B, it is thus worthwhile to recommend song C and explore Amy's true preference for it, so as to decrease the variance of our estimation of song C's mean rating . After recommending song C, Amy will give a rating feedback which has the mean of 2, therefore, predicted mean rating for song C will gradually shift toward the correct value 2, and the variance will become smaller. After several runs of non-greedy recommendation, the system is able to find out that Amy likes song C better than song B (Figure 2.4c), and then keeps recommending song C to Amy. This strategy can thus achieve better recommendation performance in the long run.

2.2.4 Solving The Greedy Problem

In the music recommendation research domain, we know only one piece of relevant work on addressing the greedy problem: Wang *et al.* [46] proposed a reinforcement learning approach to balance exploration and exploitation in music recommendation. However, this work is based on a content-based recommendation method. One major drawback of their personalized user rating model is that low-level audio features are used to represent the content of songs. This purely content-based approach is not satisfactory due to the semantic gap between low-level audio features and high-level user preferences. Moreover, songs recommended by content-based methods often lack variety because they are all acoustically similar to each other. Another limitation is that, they use a piecewise linear ap-

proximation of the model to speed up Bayesian inference, which leads to inconvenient parameters tunning process.

While no work has attempted to address the greedy problem of collaborative filtering approaches in the music recommendation context, Karimi *et al.* [18, 19] have investigated this problem in other recommendation applications (*e.g.*, movie recommendation). However, their active learning approach [18] merely explores items to optimize the prediction accuracy on a pre-determined test set. No attention is paid to the exploration-exploitation trade-off problem. In their other work [19], the recommendation process is split into two steps. In the exploration step, they select an item that brings maximum change to the user parameters, and then in the exploitation step, they pick the item based on the current parameters. This work takes balancing exploration and exploitation into consideration, but only in an ad hoc way. In addition, their approach is evaluated using only an offline and pre-determined dataset. In the end, their algorithm is not practical for deployment in online recommender systems due to its low efficiency.

Similar to our work, Li *et al.* [27] also formulate their news article recommendation problem as an n-armed Bandit problem. They treat user-click feedback as reward, and their reward function is a linear function of the news articles' feature vectors. A LinUCB approach is then proposed to learn the weights of the linear reward function. The differences between our work and their work lie in the following three aspects. First, compared to other recommendation problems, music recommendation has its specific nature: in the music recommender system, a user can listen to a song multiple times, however, recommending an already-consumed news article,

book or movie doesn't make much sense. This special repeatability makes music recommendation a unique problem because temporal factors need to be considered in the rating model. The reward function in our approach is nonlinear as a result of the additional novelty score, therefore, we resort to a more sophisticated Bayesian-UCB approach. Second, Li *et al.* use offline methods to evaluate their algorithm while we carry out online evaluation due to the interactiveness and dynamic property of our proposed algorithm. Third, our approach is based on collaborative filtering while their approach is based on contextual information. The focus of our study is on balancing between exploration and exploitation as as to remedy the greedy nature of the CF-based recommendation techniques.

2.3 Reinforcement Learning

In this paper, in order to temper the greedy nature of collaborative filtering music recommendation, we use a reinforcement learning approach to investigate the exploration-exploitation trade-off. We introduce necessary background knowledge in this section.

Different from supervised learning that learns from a ground truth dataset containing correct input/output examples, reinforcement learning needs to learn from its interactions with an unknown environment. Reinforcement learning is a category of machine learning techniques that investigates the problem of how to take actions in an environment so as to maximize a cumulated reward [39]. No external expertise knowledge will tell the reinforcement learning algorithm which actions to take, and the algorithm's suboptimal actions will not be explicitly corrected. The learn-

ing algorithm has to discover the optimal actions by trying them. In other words, the reinforcement learning algorithm must be able to learn from its own experience.

In reinforcement learning domain, online performance is a focus of study, which involves a key problem of finding a balance between exploration of the unknown environment and exploitation of the current knowledge. The exploration-exploitation trade-off has been thoroughly studied in the *n-armed Bandit* problem [39].

2.3.1 n-armed Bandit Problem

The n-armed bandit problem assumes a slot machine with n levers. Pulling a lever generates a random payoff (also called reward) chosen from an unknown and lever-specific probability distribution. The objective is to maximize the expected total payoff over a given number of action selections, say, over 1000 plays.

More formally, the n-armed bandit problem can be formulated as follows: Let $\mathcal{L} = \{1, 2, ..., n\}$ be the set of all levers of the slot machine. The reward r_i of pulling each lever $i \in \mathcal{L}$ follows an underlying probability distribution p_i which is unknown to us. We have totally N rounds to play the slot machine. At the k^{th} round, we can choose to pull an lever $I_k \in \mathcal{L}$ and receive a random reward r_{I_k} sampled from the probability distribution p_{I_k}. Our objective is to carefully choose the lever to pull at each round $((I_1, I_2, ..., I_N) \in \mathcal{L}^N)$ so as to maximize the expected cumulated reward $E[\sum_{k=1}^{N} r_{I_k}]$.

In the n-armed bandit problem, *exploration* is to randomly pull levers to gain knowledge of their distribution p_i, and *exploitation* is to pull the lever

that yields maximum expected reward based on the current estimation.

Researchers have come up with various algorithms that try to provide principled ways to solve the n-armed bandit problem, including ϵ-greedy, Boltzmann exploration, pursuit algorithms [42], upper confidence bounds (UCB) [2], Bayes-UCB [20] and so on. For more details on these algorithms, please refer to [24, 39].

In this paper, we formulate the music recommendation as an n-armed bandit problem (see Chapter 3.2.1) and adopt one of state-of-the-art algorithms called Bayes-UCB [20] to strike a balance between exploration and exploitation. In the Bayes-UCB algorithm, the expected reward U_i of lever i is predicted using Bayesian estimation. Thus U_i is treated as a random variable instead of a fixed value, and the posterior distribution of U_i given the observed reward history $\mathcal{D}$, denoted as $p(U_i|\mathcal{D})$, will be updated and refined when a new reward data is received. At each round of play, the algorithm will select the lever that has the maximum fixed-level quantile of the posterior distribution $p(U_i|\mathcal{D})$.

Chapter 3

Proposed Approach

We first present one of the most powerful techniques for collaborative filtering (CF) music recommendation, namely a low-rank matrix factorization model. Then, we point out major limitations of this traditional and popular CF algorithm. Finally, our improved approach will be described in detail.

3.1 Matrix Factorization for Collaborative Filtering

Suppose we have m users and n songs in the music recommender system. Let $\mathbf{R} = \{r_{ij}\}_{m \times n}$ denote the user-song rating matrix, where each element r_{ij} represents the rating of song j given by user i.

Matrix factorization models assume that characteristics of songs and user preferences can be explained by a number of latent factors, therefore these methods map users and songs to a joint latent factor space of dimensionality f. In this low-dimensional latent factor space, every user is

associated with a user feature vector $\mathbf{u}_i \in \mathbb{R}^f, i = 1, 2, ..., m$, and every song is associated with a song feature vector $\mathbf{v}_j \in \mathbb{R}^f, j = 1, 2, ..., n$.

For a given song j, elements of $\mathbf{v}_j$ measure the extent to which the song contains the latent factors. For a given user i, elements of $\mathbf{u}_i$ measure the extent to which he likes these latent factors. The user rating can thus be approximated by the inner product of the corresponding user feature vector and song feature vector:

$$\hat{r}_{ij} = \mathbf{u}_i^T \mathbf{v}_j \tag{3.1}$$

Let $\mathbf{U} = [\mathbf{u}_i]$ denote the user feature matrix, where $\mathbf{u}_i \in \mathbb{R}^f$ $(i = 1, 2, ..., m)$ represents the i^{th} column of $\mathbf{U}$, and let $\mathbf{V} = [\mathbf{v}_j]$ denote the song feature matrix, where $\mathbf{v}_j \in \mathbb{R}^f$ $(j = 1, 2, ..., n)$ represents the j^{th} column of $\mathbf{V}$. The algorithm learns feature matrix $\mathbf{U}$ and $\mathbf{V}$ by minimizing the following objective function that is also used in [52]:

$$\sum_{(i,j)\in I} (r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2 + \lambda (\sum_{i=1}^{m} n_{u_i} \|\mathbf{u}_i\|^2 + \sum_{j=1}^{n} n_{v_j} \|\mathbf{v}_j\|^2) \tag{3.2}$$

where I is the index set of all the known ratings, λ is a regularization parameter, n_{u_i} is the number of ratings given by user i, and n_{v_j} is the number of ratings for song j. This objective function consists of two parts: the first part $\sum_{(i,j)\in I}(r_{ij} - \mathbf{u}_i^T \mathbf{v}_j)^2$ is the squared error function and the second part $\lambda(\sum_{i=1}^{m} n_{u_i} \|\mathbf{u}_i\|^2 + \sum_{j=1}^{n} n_{v_j} \|\mathbf{v}_j\|^2)$ is a regularization term to avoid overfitting.

We adopt the *alternating least squares* (ALS) technique [52] to minimize Equation (3.2). The process is as follows: First, we fix matrix $\mathbf{V}$, take the partial derivative of Equation (3.2) with respect to $\mathbf{u}_i$, set it to zero and

solve it, thus we can derive the updating rule for each $\mathbf{u}_i$:

$$\forall i, \mathbf{u}_i = (\mathbf{V}_{I_{u_i}} \mathbf{V}_{I_{u_i}}^T + \lambda n_{u_i} \mathbf{E})^{-1} (\mathbf{V}_{I_{u_i}} \mathbf{R}_{i,I_{u_i}}^T) \tag{3.3}$$

where I_{u_i} denotes the set of songs that user i has rated, $\mathbf{V}_{I_{u_i}}$ is a matrix containing all columns $j \in I_{u_i}$ of $\mathbf{V}$, $\mathbf{E}$ is a $f \times f$ identity matrix, and $\mathbf{R}_{i,I_{u_i}}$ is a row vector where columns $j \in I_{u_i}$ of the i^{th} row of $\mathbf{R}$ are selected.

Similarly, we then fix matrix $\mathbf{U}$, take the partial derivative of Equation (3.2) with respect to $\mathbf{v}_j$, set it to zero and solve it, thus we obtain the following updating rule for each $\mathbf{v}_j$:

$$\forall j, \mathbf{v}_j = (\mathbf{U}_{I_{v_j}} \mathbf{U}_{I_{v_j}}^T + \lambda n_{v_j} \mathbf{E})^{-1} (\mathbf{U}_{I_{v_j}} \mathbf{R}_{I_{v_j},j}) \tag{3.4}$$

where I_{v_j} denotes the set of users who have rated song j, $\mathbf{U}_{I_{v_j}}$ is a matrix containing all columns $i \in I_{v_j}$ of $\mathbf{U}$, $\mathbf{E}$ is a $f \times f$ identity matrix, and $\mathbf{R}_{I_{v_j},j}$ is a column vector where rows $i \in I_{v_j}$ of the j^{th} column of $\mathbf{R}$ are selected.

An advantage of ALS is that the algorithm computes each $\mathbf{u}_i$ independently of the other user feature vectors and computes each $\mathbf{v}_j$ independently of the other song feature vectors [23]. This advantage allows us to implement a multi-threaded parallel ALS algorithm so as to make the collaborative filtering process much more efficient.

In our parallel ALS algorithm, each thread is responsible for updating an independent subset of the column vectors of matrix $\mathbf{U}$ or $\mathbf{V}$. The stopping criterion of the ALS algorithm will be achieved when the change in root mean square error (RMSE) on the validation set is less than 10^{-4}. The detailed steps are presented in Algorithm 1.

Algorithm 1 Multi-threaded Parallel ALS for Collaborative Filtering

```
Initialize matrix V with small random numbers;
while stopping criterion is not satisfied do
  Fix matrix V, create k threads to update U;
  for all thread t = 0 → k − 1 do
    Update U's columns u_i, i = t × (m/k), ..., (t + 1) × (m/k) − 1, using
    Equation (3.3);
  end for
  Fix matrix U, create k threads to update V;
  for all thread t = 0 → k − 1 do
    Update V's columns v_j, j = t × (n/k), ..., (t + 1) × (n/k) − 1, using
    Equation (3.4);
  end for
end while
```

Even though matrix factorization model is a powerful tool for collaborative filtering (CF) [23], this traditional CF technique has two major drawbacks:

1. It fails to take recommendation *novelty* into consideration. A user can listen to the same song multiple times, but each time he listens to it, the novelty of this song may be different to him. For example, if the system keeps recommending the same song to the target user just because he has given this song a very high rating before, he will quickly get bored with this song, and the novelty of this song will degrade dramatically.

2. It works *greedily*, always recommending songs with the highest predicted mean ratings, while a better approach may be to actively explore a user's preferences rather than to merely exploit available rating information. Chapter 2.2 has illustrated the limitations of the greedy recommendation strategy in detail.

To address these two drawbacks, we propose to use a reinforcement

learning approach to modify the original matrix factorization model for music recommendation. Technical details about this improved approach will be described in subsequent sections.

3.2 A Reinforcement Learning Approach

Music recommendation is an interactive process. The system repeatedly choose among n different songs to recommend. After each recommendation, it receives a rating feedback (or *reward*) chosen from an unknown probability distribution. The goal of the recommender system is to maximize user satisfaction, *i.e.* the expected total reward, in the long run. Similarly, reinforcement learning explores an environment and takes actions to maximize the cumulated reward. It is thus fitting to treat music recommendation as a well-studied reinforcement learning task called *n-armed bandit* (introduced in Chapter 2.3.1).

3.2.1 Problem Formulation

To formulate music recommendation problem as an n-armed bandit problem, we can treat the recommender system as the player, treat the target user as the slot machine, treat songs in the recommender system as levers of the slot machine, and treat rating for a song as the reward of pulling the corresponding lever.

More formally, the interactive music recommendation problem can be formulated as follows: Let $\mathcal{S} = \{1, 2, ..., n\}$ be the set of all songs in the recommender system. The rating feedback R_i given by the target user for each song $i \in \mathcal{S}$ follows an underlying probability distribution p_i which

is unknown to the recommender system. Suppose the system has totally N chances to recommend a song to the target user. At the k^{th} iteration, the system selects a song $I_k \in \mathcal{S}$ to recommend and will receive a rating feedback R_{I_k} sampled from the probability distribution p_{I_k}. The objective of the music recommender system is to wisely select a recommended song at each recommendation iteration($(I_1, I_2, ..., I_N) \in \mathcal{S}^N$) so as to maximize the user satisfaction over the long term (*i.e.* maximize the expected cumulated rating $E[\sum_{k=1}^{N} R_{I_k}]$).

3.2.2 Modeling User Rating

To address drawback (1) pointed out in Chapter 3.1, we assume that a song's rating is mainly affected by two factors: the extent to which the user likes the song in terms of each CF latent factor, and the dynamically changing novelty of the song. The former is quantified as the *CF score*, and the latter is quantified as the *novelty score*.

From Equation (3.1), we define the CF score as:

$$U_{CF} = \boldsymbol{\theta}^T \mathbf{v} \tag{3.5}$$

where vector $\boldsymbol{\theta}$ is a parameter indicating the user's preferences for different CF latent factors and $\mathbf{v}$ is the song feature vector learned from the parallel ALS CF algorithm (Algorithm 1).

For the novelty score, we adopt the formula used in [46]:

$$U_N = 1 - e^{-t/s} \tag{3.6}$$

where t is the time elapsed since when the song was last heard, s is a param-

eter indicating the relative strength of the user's memory, and $e^{-t/s}$ is the well-known forgetting curve proposed by Ebbinghaus *et al.* [15]. Clearly, the larger the elapsed time t is, the more novel the song is to the target user. On the other hand, the larger the memory strength s is, the less novel the song is to the user. Equation (3.6) assumes that the novelty of a song decreases immediately when the user listens to it and then gradually recovers as time goes by. The novel score is on a per-song basis, and it seems to be sparse because there are relatively very few songs that are heard by the target user in the whole dataset. As a result, someone may suggest that we should define our novelty score based on a larger group of songs such as the musical genres. However, we must argue that the purpose of introducing a novelty score into the rating model is not to distinguish every song in the dataset, but to degrade the priority of recommending those already-heard and high-scored songs. Only the songs that are heard by the target user can have different novelty scores, while those non-heard songs all have the same novelty score (*i.e.* $U_N = 1$). Therefore, the sparseness issue is not our concern.

We model the final user rating by combining these two scores:

$$U = U_{CF}U_N = (\boldsymbol{\theta}^T\mathbf{v})(1 - e^{-t/s}) \tag{3.7}$$

It is worth noting that different users may have different musical tastes and memory strengths, therefore, each user is associated with a pair of parameters $\boldsymbol{\Omega} = (\boldsymbol{\theta}, s)$ that are unknown to the recommender system and need to be learned from the user's rating history. More technical details about learning these parameters will be described in Chapter 3.2.3.

Let R_j denote the rating of song j given by the target user. From a

probabilistic perspective, R_j is a random variable that follows an unknown probability distribution p_j. We assume that the expectation of R_j is the U_j defined in Equation (3.7):

$$\mathbb{E}[R_j] = U_j = (\boldsymbol{\theta}^T \mathbf{v}_j)(1 - e^{-t_j/s}) \tag{3.8}$$

Given this assumption, the major task of the music recommendation algorithm is thus to predict or estimate the expected rating U_j for each candidate song j in the system.

A traditional recommendation strategy will first obtain the song feature vector $\mathbf{v}_j$ and the elapsed time t_j of each song j to compute the mean rating U_j using Equation (3.7) and then recommend the song with the highest predicted mean rating. We call this a *greedy* recommendation as the system is merely *exploiting* its current knowledge of the user preferences. By selecting one of the non-greedy recommendations and gathering more user feedback, the system *explores* further and gains more knowledge about the user preferences. If we knew the user's true preferences (*i.e.* the true value of the user parameter $\boldsymbol{\Omega} = (\boldsymbol{\theta}, s)$), then it would be trivial to solve the recommendation problem by just recommending the greedy song because the predicted mean rating is exactly the true mean rating. However, the value of user parameter $\boldsymbol{\Omega} = (\boldsymbol{\theta}, s)$ is learned based on currently observed data (*i.e.* the target user's rating history). Therefore, the predicted mean rating we compute using Equation (3.7) is just an estimator of the true mean rating, and it may contain inaccuracy. A greedy recommendation would result in suboptimal performance over the long term. This is because several non-greedy recommendations may be deemed nearly as good but come with substantial *variance* (or uncertainty), and it is thus possible

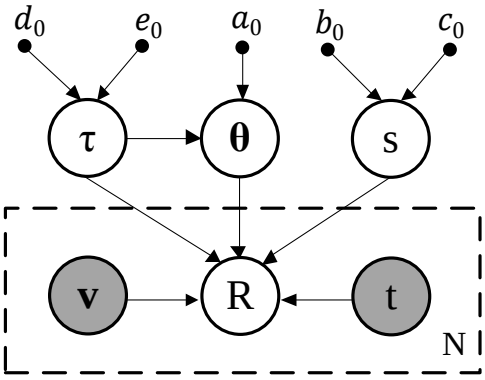


Figure 3.1: Bayesian Graphical Model.

that some of them are actually better than the greedy recommendation. Without exploration, however, we will never know which ones they are.

In order to counter the greedy nature of collaborative filtering, *i.e.* the drawback (2) pointed out in Chapter 3.1, we introduce exploration into music recommendation to balance exploitation. To do so, we adopt one of the state-of-the-art algorithms developed in the n-armed bandit problem, namely the Bayesian Upper Confidence Bounds (Bayes-UCB) [20]. In the Bayes-UCB, the expected rating U_j is a random variable rather than a fixed number. Given the target user's rating history $\mathcal{D}$, the posterior distribution of U_j, denoted as $p(U_j|\mathcal{D})$, needs to be estimated. At each recommendation iteration, the song with the highest fixed-level quantile value of $p(U_j|\mathcal{D})$ will be recommended to the target user.

3.2.3 Bayesian Graphical Model

To estimate the posterior distribution $p(U_j|\mathcal{D})$, we adopt the Bayesian graphical model shown in Figure 3.1. The corresponding probability de-

pendency is defined as follows:

$$R|\mathbf{v}, t, \boldsymbol{\theta}, s, \sigma^2 \sim \mathcal{N}(\boldsymbol{\theta}^T \mathbf{v}(1 - e^{-t/s}), \sigma^2) \tag{3.9}$$

$$\boldsymbol{\theta}|\sigma^2 \sim \mathcal{N}(\mathbf{0}, a_0 \sigma^2 \mathbf{I}) \tag{3.10}$$

$$s \sim Gamma(b_0, c_0) \tag{3.11}$$

$$\tau = 1/\sigma^2 \sim Gamma(d_0, e_0) \tag{3.12}$$

$\mathbf{I}$ is the $f \times f$ identity matrix. $\mathcal{N}$ represents Gaussian distribution with parameters mean and variance. *Gamma* represents Gamma distribution with parameters shape and rate. $\boldsymbol{\theta}$, s, and τ are parameters. a_0, b_0, c_0, d_0, and e_0 are hyperparameters of the priors.

Suppose at current iteration $h+1$, we have gathered h observed recommendation history $\mathcal{D}_h = \{(\mathbf{v}_i, t_i, r_i)\}_{i=1}^h$. Recall that, in our rating model, each user is describe as a pair of parameters $\boldsymbol{\Omega} = (\boldsymbol{\theta}, s)$. According to the Bayes theorem, the posterior distribution of these parameters given the history data is:

$$p(\boldsymbol{\Omega} \mid \mathcal{D}_h) \propto p(\boldsymbol{\Omega}) p(\mathcal{D}_h \mid \boldsymbol{\Omega}) \tag{3.13}$$

Then the posterior probability density function (PDF) of the expected rating U_j of song j can be estimated as:

$$p(U_j|\mathcal{D}_h) = \int p(U_j|\boldsymbol{\Omega}) p(\boldsymbol{\Omega}|\mathcal{D}_h) \boldsymbol{d\Omega} \tag{3.14}$$

Since Equation (3.13) has no closed form solution, we are unable to directly estimate the posterior PDF in Equation (3.14). To solve this problem, we thus turn to a Markov Chain Monte Carlo (MCMC) algorithm to draw an adequate amount of samples of parameters $\boldsymbol{\Omega} = (\boldsymbol{\theta}, s)$. We then substitute

every parameter sample into Equation (3.7) to obtain a sample of U_j. Finally, the posterior PDF in Equation (3.14) can be approximated by the histogram of the samples of U_j.

After estimating the posterior PDF of each song's expected rating, we follow the Bayes-UCB approach [20] to achieve a balance between exploration and exploitation, *i.e.* recommend song j^* that maximizes the following quantile function:

$$j^* = \arg \max_{j=1,\ldots,|\mathcal{S}|} Q\left(\alpha, p(U_j|\mathcal{D}_h)\right) \tag{3.15}$$

where $\alpha = 1 - \frac{1}{h+1}$, $|\mathcal{S}|$ is the total number of songs in the recommender system, and the quantile function Q returns the value x such that $\mathbf{Pr}(U_j \leq x|\mathcal{D}_h) = \alpha$. The pseudo code of our exploration-exploitation balanced music recommendation algorithm is presented in Algorithm 2.

It is worth mentioning that, different from some typical recommendation problems which may recommend a list of top-N items (*e.g.*, images and query suggestions), we recommend only the top song at each iteration. The reason is that a user only has one pair of ears, and only one song can be heard by the user at a time unlike other types of visual information. What's more, our interactive music recommender system is just like an online radio station application. Recommending one song per iteration is enough for our application scenario, so there is no need to recommend a list of songs.

Algorithm 2 Exploration-Exploitation Balanced Music Recommendation

for $h = 1 \rightarrow N$ **do**
  if $h == 1$ **then**
    Recommend a song randomly;
  else
    Draw samples of $\boldsymbol{\theta}$ and s based on $p(\boldsymbol{\Omega} \mid \mathcal{D}_{h-1})$;
    for song $j = 1 \rightarrow |\mathcal{S}|$ **do**
      Obtain $\mathbf{v}_j$ and t_j of song j and compute samples of U_j using Equation (3.7);
      Estimate $p(U_j|\mathcal{D}_{h-1})$ using histogram of the samples of U_j;
      Compute quantile value $q_j^h = Q\left(1 - \frac{1}{h}, p(U_j|\mathcal{D}_{h-1})\right)$;
    end for
    Recommend song $j^* = \arg\max_{j=1,\ldots,|\mathcal{S}|} q_j^h$;
  end if
  Collect user rating feedback r_h and update $p(\boldsymbol{\Omega} \mid \mathcal{D}_h)$;
end for

3.3 Efficient Sampling Algorithm

When we use an off-the-shelf MCMC sampling algorithm[1], Bayesian inference becomes very slow because it takes a long time for the Markov chain to converge. In response, Wang *et al.* [46] proposed an approximate Bayesian model using piecewise linear approximation. However, not only is the original Bayesian model altered, tuning the numerous (hyper)parameters is also tedious.

The efficiency of the MCMC sampling is highly related to the proposal distribution selected. A better proposal distribution will lead to faster convergence of the Markov chain and hence reduce the time of Bayesian inference. In contrast, with a bad proposal distribution, it takes a long time for the Markov chain to converge. In this paper, we present a better way to improve efficiency. Given that it is simple to sample from a conditional distribution, we develop a specific Gibbs sampling algorithm to hasten

[1] http://mcmc-jags.sourceforge.net/

convergence of the Markov chain.

Given N observed recommendation history $\mathcal{D} = \{\mathbf{v}_i, t_i, r_i\}_{i=1}^N$, the conditional distribution $p(\boldsymbol{\theta}|\mathcal{D}, \tau, s)$ is still a Gaussian distribution and can be obtained as follows:

$$
\begin{aligned}
p(\boldsymbol{\theta}|\mathcal{D}, \tau, s) &\propto p(\tau)p(\boldsymbol{\theta}|\tau)p(s)\prod_{i=1}^{N} p(r_i|\mathbf{v}_i, t_i, \boldsymbol{\theta}, s, \tau) \\
&\propto p(\boldsymbol{\theta}|\tau)\prod_{i=1}^{N} p(r_i|\mathbf{v}_i, t_i, \boldsymbol{\theta}, s, \tau) \\
&\propto \exp\left(-\frac{1}{2}\boldsymbol{\theta}^T(a_0\sigma^2\mathbf{I})^{-1}\boldsymbol{\theta}\right) \times \exp\left(\sum_{i=1}^{N} -\frac{1}{2\sigma^2}\left(r_i - \boldsymbol{\theta}^T\mathbf{v}_i(1-e^{-t_i/s})\right)^2\right) \\
&\propto \exp\left[-\frac{1}{2}\boldsymbol{\theta}^T\left(\frac{\tau}{a_0}\mathbf{I} + \tau\sum_{i=1}^{N}(1-e^{-t_i/s})^2\mathbf{v}_i\mathbf{v}_i^T\right)\boldsymbol{\theta} + \left(\tau\sum_{i=1}^{N} r_i(1-e^{-t_i/s})\mathbf{v}_i^T\right)\boldsymbol{\theta}\right] \\
&\propto \exp\left(-\frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\Lambda}\boldsymbol{\theta} + \boldsymbol{\eta}^T\boldsymbol{\theta}\right) \\
&\propto \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \qquad (3.16)
\end{aligned}
$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, respectively the mean and covariance of the multivariate Gaussian distribution, satisfy:

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Lambda} = \tau\left(\frac{1}{a_0}\mathbf{I} + \sum_{i=1}^{N}(1-e^{-t_i/s})^2\mathbf{v}_i\mathbf{v}_i^T\right) \qquad (3.17)$$

$$\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1} = \boldsymbol{\eta}^T = \tau\left(\sum_{i=1}^{N} r_i(1-e^{-t_i/s})\mathbf{v}_i^T\right) \qquad (3.18)$$

Similarly, the conditional distribution $p(\tau|\mathcal{D}, \boldsymbol{\theta}, s)$ remains a Gamma distribution and can be derived as:

$$
\begin{aligned}
p(\tau|\mathcal{D},\boldsymbol{\theta},s) &\propto p(\tau)p(\boldsymbol{\theta}|\tau)p(s)\prod_{i=1}^{N}p(r_i|\mathbf{v}_i,t_i,\boldsymbol{\theta},s,\tau) \\
&\propto p(\tau)p(\boldsymbol{\theta}|\tau)\prod_{i=1}^{N}p(r_i|\mathbf{v}_i,t_i,\boldsymbol{\theta},s,\tau) \\
&\propto \tau^{d_0-1}\exp(-e_0\tau)\times\exp\left(-\frac{1}{2}\boldsymbol{\theta}^T(a_0\sigma^2\mathbf{I})^{-1}\boldsymbol{\theta}\right)\times \\
&\left(\sigma\sqrt{2\pi}\right)^{-N}\exp\left(\sum_{i=1}^{N}-\frac{1}{2\sigma^2}\left(r_i-\boldsymbol{\theta}^T\mathbf{v}_i(1-e^{-t_i/s})\right)^2\right) \\
&\propto \tau^{\alpha-1}\exp(-\beta\tau) \\
&\propto Gamma\,(\alpha,\beta) \qquad (3.19)
\end{aligned}
$$

where α and β are respectively the shape and rate of the Gamma distribution and satisfy:

$$\alpha = d_0 + \frac{f+N}{2} \qquad (3.20)$$

$$\beta = e_0 + \frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{2a_0} + \frac{1}{2}\sum_{i=1}^{N}\left(r_i-\boldsymbol{\theta}^T\mathbf{v}_i(1-e^{-t_i/s})\right)^2 \qquad (3.21)$$

The conditional distribution $p(s|\mathcal{D},\boldsymbol{\theta},\tau)$ has no closed form expression. We thus adopt the Metropolis-Hastings (MH) algorithm [11] with a proposal distribution $q(s_{t+1}|s_t)=\mathcal{N}(s_t,1)$ to draw samples of s. Our efficient Gibbs sampling algorithm is presented in Algorithm 3.

Algorithm 3 Gibbs Sampling for Bayesian Inference

Initialize $\boldsymbol{\theta}$, s, τ;
for $t = 1 \rightarrow BURN\_IN + SAMPLE\_SIZE$ **do**
  Sample $\boldsymbol{\theta}^{(t+1)} \sim p(\boldsymbol{\theta}|\mathcal{D}, \tau^{(t)}, s^{(t)})$;
  Sample $\tau^{(t+1)} \sim p(\tau|\mathcal{D}, \boldsymbol{\theta}^{(t+1)}, s^{(t)})$;
  $s_{tmp} = s^{(t)}$;
  for $i = 1 \rightarrow K$ **do** # MH Step
    Draw $y \sim \mathcal{N}(s_{tmp}, 1)$;
    $\alpha = min\left(\frac{p(y|\mathcal{D}, \boldsymbol{\theta}^{(t+1)}, \tau^{(t+1)})}{p(s_{tmp}|\mathcal{D}, \boldsymbol{\theta}^{(t+1)}, \tau^{(t+1)})}, 1\right)$;
    Draw $u \sim Uniform(0, 1)$;
    if $u < \alpha$ **then**
      $s_{tmp} = y$;
    end if
  end for
  $s^{(t+1)} = s_{tmp}$;
end for
return Last $SAMPLE\_SIZE$ sets of samples ($\boldsymbol{\theta}$, s, τ);

Chapter 4

Experiments

We conduct experiments to:

- determine the optimal parameter setting for our matrix factorization model, Gibbs sampling algorithm and Bayesian graphical model,
- learn the collaborative filtering latent factors for each song,
- show the efficiency of our proposed Gibbs sampling algorithm for Bayesian inference,
- show the effectiveness of our exploration-exploitation balanced music recommendation algorithm in terms of recommendation performance.

4.1 Dataset

The Taste Profile Subset[1] used in the Million Song Dataset Challenge [30] provides over 48 million triplets (user, song, play count) describing the listening history of over 1 million users and 380,000 songs. In the music recommendation domain, this is one of the largest publicly available

[1] http://labrosa.ee.columbia.edu/millionsong/tasteprofile

# Users	# Songs	# Observations	% Density
100,000	20,000	20,699,820	1.035%

Training	Validation	Test
16,619,732	1,969,562	2,110,526

Table 4.1: Dataset size statistics. *Density* is the percentage of entries in the user-song interaction matrix that have observations.

collaborative filtering datasets. Since the raw audio data is absent in the dataset, we obtain 30-second audio clips for songs in the dataset from 7digital.com[2]. According to the computational resource we have, performing collaborative filtering on the entire dataset is impractical due to the huge amount of data. Therefore, we select 20,000 songs with top listening counts and 100,000 users who have listened to the most songs. Since listening history data is a form of implicit feedback data and only contains positive examples, we need to perform preprocessing on the dataset using the approach proposed in [33]: First, all the non-zero play counts are mapped to value 1 in the user-song interaction matrix. Then, adopt user-oriented negative sampling method to randomly draw the same amount of negative examples as the positive examples on a per-user basis. Finally, the negative examples are mapped to value 0 in the user-song interaction matrix. Thus, we get our collaborative filtering dataset ready for matrix factorization. We randomly split the dataset into three disjoint parts: training set (80%), validation set (10%), and test set (10%). The detailed statistics of the final dataset we used are shown in Table 4.1.

[2]http://www.7digital.com/

4.2 Learning CF Latent Factors

First, we need to determine the optimal values for the two parameters in the matrix factorization model, *i.e.* λ, the regularization parameter, and f, the dimensionality of the latent feature vectors.

The training set is used to learn the CF latent factors, and the convergence criterion of the ALS algorithm (Algorithm 1) is achieved when the change in root mean square error (RMSE) on the validation set is less than 10^{-4}. Then we use the learned latent factors to predict the ratings on the test set[3].

We first fix $f = 55$ and vary λ from 0.005 to 0.1; minimal RMSE is achieved at $\lambda = 0.025$ (experimental results are shown in Figure 4.1).

We then fix $\lambda = 0.025$ and vary f from 10 to 80, and $f = 75$ yields minimal RMSE (shown in Figure 4.2).

Finally, we adopt the optimal value $\lambda = 0.025$ and $f = 75$ to perform the ALS CF algorithm and obtain the learned latent feature vector of each song in our dataset. These latent feature vectors will later be used in the proposed music recommendation algorithm.

4.3 Efficiency Study

To show that our Gibbs sampling algorithm makes Bayesian inference significantly more efficient, we conduct simulation experiments to compare it with an off-the-shelf MCMC algorithm developed in JAGS[4]. We

[3]We ran our parallel ALS algorithm on a 64-processor Linux server. All processors are AMD Opteron 6376 @ 2.3GHz. It takes about 6.3 minutes to finish one ALS iteration, and the converged solution (with 15 ALS iterations on average) can be computed within 1.6 hours.

[4]http://mcmc-jags.sourceforge.net/

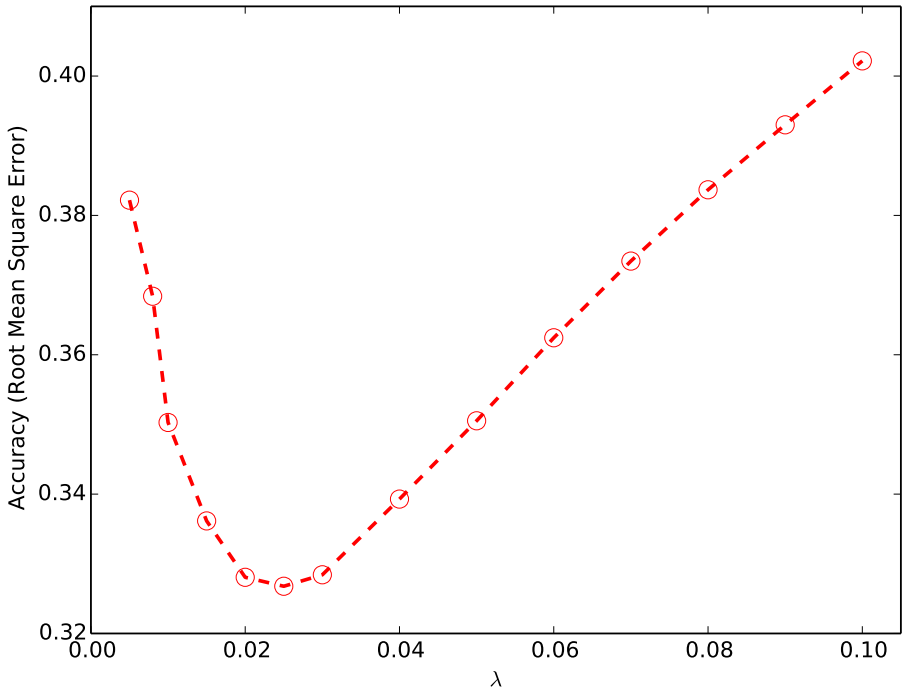


Figure 4.1: Fix $f = 55$, RMSE results of CF with different λ values.

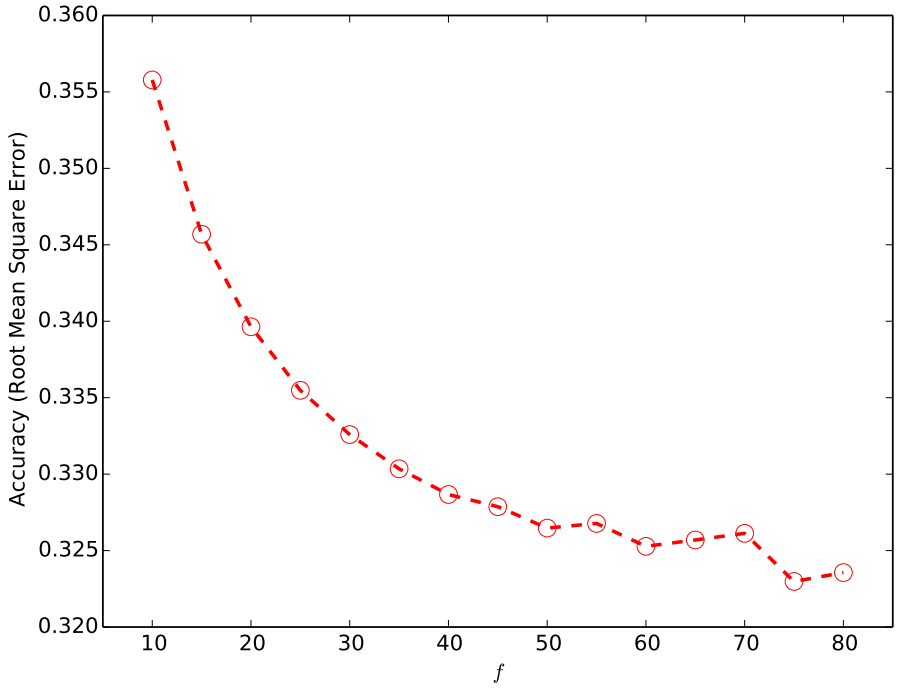


Figure 4.2: Fix $\lambda = 0.025$, RMSE results of CF with different f values.

implemented the Gibbs algorithm in C++, which JAGS uses, for a fair comparison.

For each data point $d_i \in \{(\mathbf{v}_i, t_i, r_i)\}_{i=1}^n$ in the simulation experiments, $\mathbf{v}_i$ is randomly chosen from the latent feature vectors learned in Chapter4.2. t_i is randomly sampled from $uniform(50, 2592000)$, *i.e.* between a time gap of 50 seconds and one month. r_i is calculated using Equation (3.7) where elements of $\boldsymbol{\theta}$ are sampled from $\mathcal{N}(0, 1)$ and s from $uniform(100, 1000)$.

To determine the two parameters (*i.e. burn-in* and *sample size*) of the two sampling algorithms and to ensure they draw samples equally effectively, we first check to see if they converge to a similar level.

We generate a test set of 300 data points and vary the size of the training set to gauge the prediction accuracy. The value of K in the Metropolis-Hastings (MH) step of our Gibbs algorithm (Algorithm 3) is set to 5.

While our Gibbs algorithm achieves reasonable accuracy with burn-in = 20 and sample size = 100, the MCMC algorithm gives comparable results only when both parameters are 10000. Figure 4.3 shows their prediction accuracies averaged over 10 trials.

With the parameters *burn-in* and *sample size* determined, we can ensure that the two sampling algorithms draw samples equally effectively, based on which, we can then conduct an efficiency study of the two algorithms.

We vary the training set size from 1 to 1000 and record the time they take to finish the sampling process[5]. The efficiency comparison result is shown in Figure 4.4. (For more details on the numerical results, please refer to Table 4.2). We can see that computation time of both two sampling algorithms grows linearly with the training set size. However, our proposed

[5]We use a computer with Intel Core i7-2600 CPU @ 3.40Ghz and 8GB RAM.

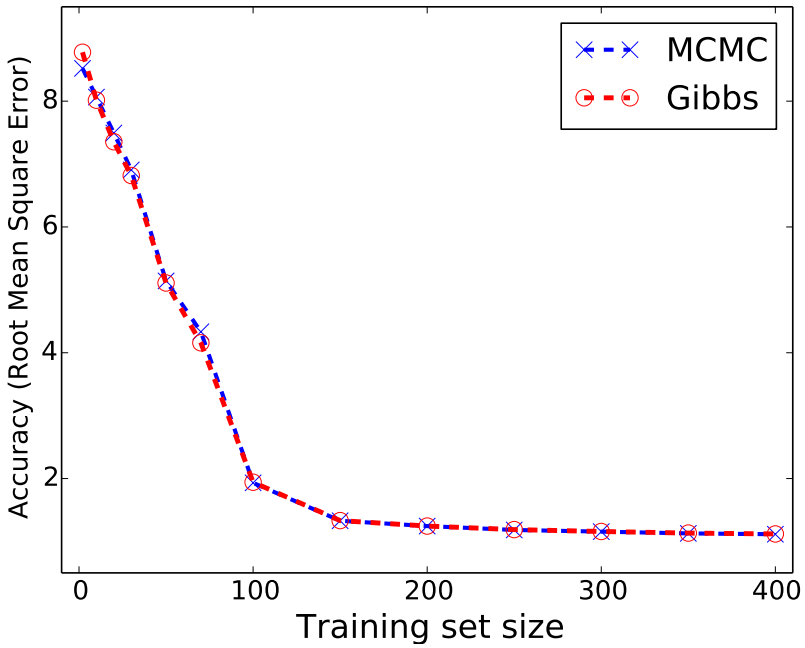


Figure 4.3: Prediction accuracy of the two sampling algorithms. In Gibbs, burn-in=20, sample size=100. The prediction accuracy of MCMC achieves a comparable level when burn-in=10000 and sample size=10000.

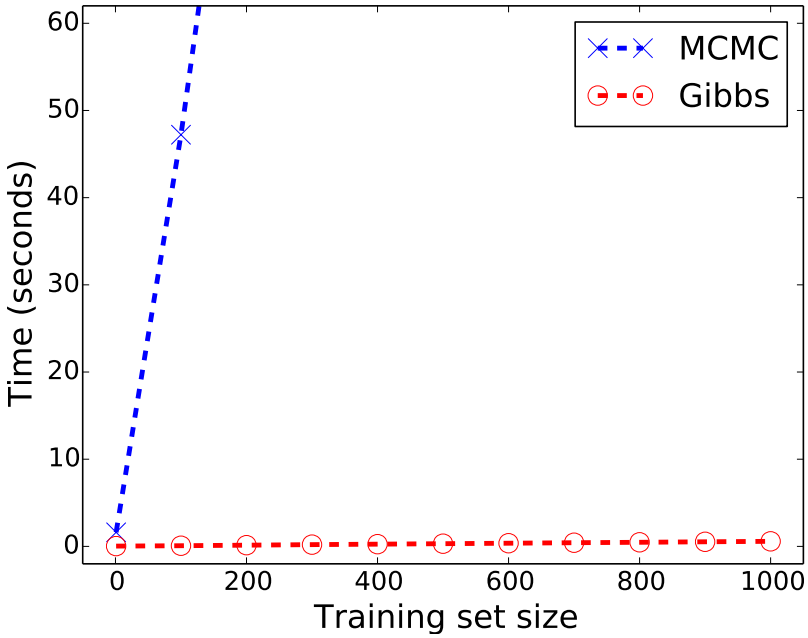


Figure 4.4: Efficiency comparison of the two sampling algorithms. $Time_{MCMC} = 538.762s$ and $Time_{Gibbs} = 0.579s$ when $TrainingSetSize = 1000$.

Training Set Size	$Time_{MCMC}$ (second)	$Time_{Gibbs}$ (second)
1	1.613	0.034
100	47.219	0.075
200	100.780	0.140
300	151.650	0.202
400	203.508	0.255
500	266.518	0.310
600	319.629	0.363
700	375.958	0.421
800	408.789	0.469
900	456.071	0.527
1000	538.762	0.579

Table 4.2: Efficiency comparison of the two sampling algorithms (with detailed numerical results).

Gibbs sampling algorithm is hundreds of times faster than MCMC, suggesting that our proposed approach is practical for deployment in online recommender systems[6].

4.4 Effectiveness Study

We denote our proposed recommendation algorithm as *Bayes-UCB-CF* because it adopts Bayes-UCB approach to temper the greedy nature of CF-based music recommendation. We compare it with two baseline algorithms:

1. the *Greedy* algorithm, representing the traditional recommendation strategy withou exploration-exploitation trade-off. This is to check if balancing exploration-exploitation can improve the performance of music recommendation.

[6]In the online music recommender system prototype we developed, the entire process of generating next recommendation can finish in 2 seconds, which meets the efficiency requirement well.

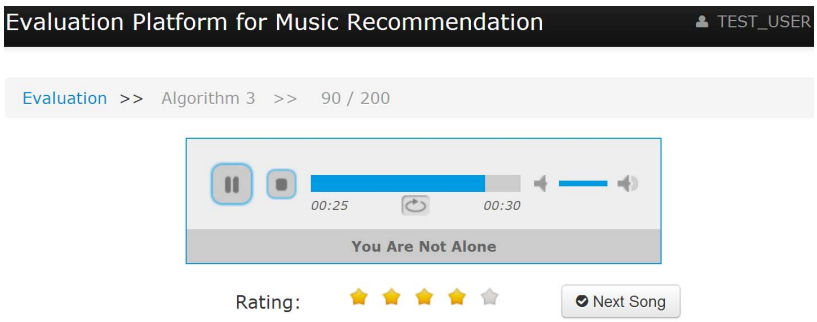


Figure 4.5: Online evaluation platform.

2. the *Bayes-UCB-Content* algorithm [46], which also adopts the Bayes-UCB technique but is content-based instead of CF-based. This is to check if our proposed algorithm can outperform existing work that also attempts to address the greedy problem of traditional music recommendation approaches.

To evaluate the effectiveness of these three algorithms, we conducted an online user study. We perform online evaluation instead of offline evaluation because the latter cannot capture the effect of the elapsed time t in our rating model and the interactiveness of our recommendation approach.

Eighteen undergraduate and graduate students (9 females and 9 males, age 19 to 29) are invited to participate in the user study. The subject pool covers a variety of majors of study and nationalities, including American, Chinese, Korean, Malaysian, Singaporean and Iranian. Subjects receive a small payment for their participation. The user study takes place over the course of three weeks in April 2014 on an online evaluation platform[7] we constructed (Figure 4.5).

The three algorithms evaluated are randomly assigned to numbers 1-3 to avoid bias. For each of these three algorithms, every subject is asked

[7] http://evaluation.smcnus.org/

to evaluate 200 recommendations using a rating scale from 1 to 5. To mimic regular recommendation sessions, subjects are reminded to take a 5-minute break after evaluating 20 songs, which also ensures the quality of the ratings. To minimize the carryover effect, subjects cannot evaluate two different algorithms in one day[8].

At the first recommendation iteration, the recommender system knows nothing about the user, therefore, every song in the dataset has equal probability to be recommended. The user listens to the first song and gives a rating feedback based on his own musical preferences. Then the system learns from the user's feedback, refines its knowledge about the user's preferences and tries to improve its next recommendation. This process is repeated until the 200^{th} recommendation iteration. Intuitively, if we sum up the user's 200 ratings, then the higher the total rating is, the better the recommendation algorithm is.

Therefore, the evaluation metric we used to compare the performance of the three algorithm is the *cumulated average rating*, denoted as $\bar{R}$. Suppose currently we are at the n^{th} recommendation iteration, the cumulated average rating $\bar{R}$ can be calculated as follows:

$$\bar{R} = \frac{1}{m \times n} \sum_{u=1}^{m} \sum_{i=1}^{n} R_{ui} \tag{4.1}$$

where m is the number of subjects in the user study (*i.e.* 18 in our experiment), n is the current recommendation iteration, and R_{ui} is the rating given by the subject u at the i^{th} iteration.

[8]For the user study, hyperparameters of the Bayes-UCB-CF algorithm are set as: $a_0 = 10$, $b_0 = 3$, $c_0 = 0.01$, $d_0 = 0.001$ and $e_0 = 0.001$.

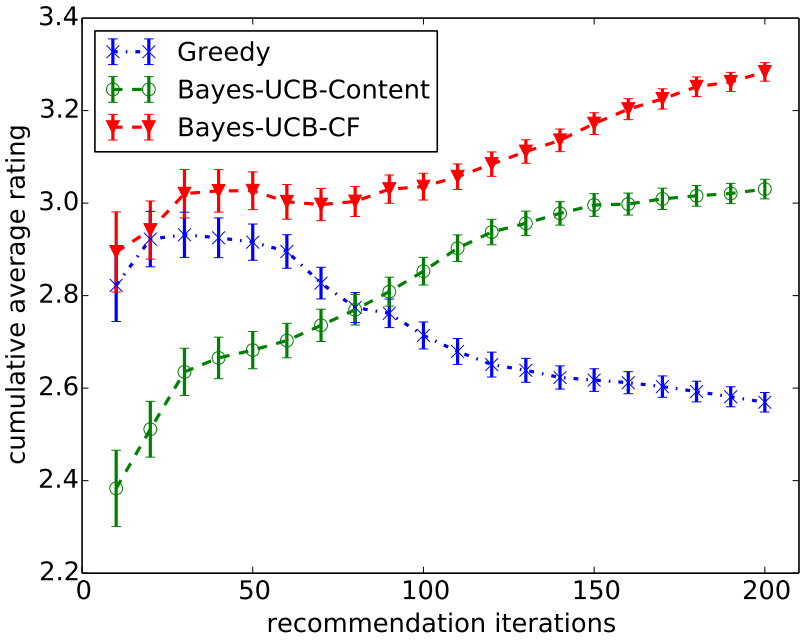


Figure 4.6: Recommendation performance comparison.

Figure 4.6 shows the cumulated average ratings (along with their standard errors) of each recommendation algorithm from the beginning till the n^{th} recommendation iteration. The standard error (SE) is computed as:

$$SE = \frac{SD}{\sqrt{m \times n}} = \frac{\sqrt{\frac{1}{m \times n} \sum_{u=1}^{m} \sum_{i=1}^{n} (R_{ui} - \bar{R})^2}}{\sqrt{m \times n}} \tag{4.2}$$

where m is the number of subjects (*i.e.* 18 in our experiment), n is the current recommendation iteration, R_{ui} is the rating given by the subject u at the i^{th} iteration, $\bar{R}$ is the cumulated average rating defined in Equation (4.1), and SD stands for "standard deviation".

From Figure 4.6, we can see that our proposed Bayes-UCB-CF algorithm significantly outperforms Bayes-UCB-Content, suggesting that the latter still fails to bridge the semantic gap between high-level user preferences and low-level audio features. T-tests show that Bayes-UCB-CF

starts to significantly outperform the Greedy baseline after the 46^{th} iteration (p-value < 0.0472). In fact, Greedy's performance decays rapidly after the 60^{th} iteration while others continue to improve. Because Greedy solely exploits, it is quickly trapped at a local optima, repeatedly recommending the few songs with initial good ratings. As a result, the novelty of those songs plummets, and users become bored. Greedy will introduce new songs after collecting many low ratings, only to be soon trapped into a new local optima. By contrast, our Bayes-UCB-CF algorithm balances exploration and exploitation and thus significantly improves the recommendation performance.

Chapter 5

Conclusion

We present a reinforcement learning approach to music recommendation that remedies the greedy nature of the collaborative filtering (CF) approaches by balancing exploitation with exploration. A Bayesian graphical model incorporating both the CF latent factors and recommendation novelty is used to learn user preferences. We also develop an efficient sampling algorithm to speed up Bayesian inference. In CF-based music recommendation, our work is the first attempt to investigate the exploration-exploitation trade-off and to address the greedy recommendation problem. Results from simulation experiments and user study have shown that our proposed approach significantly improves recommendation performance. Limitations and possible improvements are discussed as follows:

- In the initial stage, our interactive music recommender system gives each song in the dataset equal probability, and randomly recommends a song to the target user. This is because we assume that the system knows nothing about the target user. However, it is usually possible that the system has some prior information about the target user's

musical preferences. For example, when a new user signs up in the recommender system, we can ask the new user to provide some information about his musical tastes such as his favorite artists and musical genres. Based on this prior knowledge, we can give higher prior probabilities to a group of songs that belong to the user's favorite artists and genres. In this way, we can probably reduce the time of exploration and improve the overall recommendation performance.

- In our approach, learning the CF latent factors and learning the user's musical preferences are two independent components. The CF latent factors of each song are learned offline and won't be changed anymore. On the other hand, the user's preferences are learned online and will be refined each time we receive a rating feedback from the user. Actually, it would be better to combine these two components together. That is to say, when we collect a rating feedback, we can simultaneously update our estimation about the song's CF latent factors and update our knowledge about the user's preferences.

- In our online experiments, in order to compare the performance of different recommendation algorithms, we ask the subjects to give explicit rating feedback (*i.e.* ratings on a scale of 1-5). However, in real life applications, users are reluctant to give explicit feedback when they listen to music. It is very difficult to gather explicit rating feedback from users, but implicit feedback data (such as "like" or "dislike", listening count, and how quickly the user skips a recommended song) is often easy to collect. Our model can be easily modified and adapted to these recommendation scenarios when only implicit feedback data is available. (*e.g.,* we can treat "like" rate, listening count and lis-

tening time on a recommended song as the reward in our bandit algorithm.)

Chapter 6

Future Work

We suggest potential future research directions in this chapter.

6.1 Increasing Recommendation Diversity

In our proposed algorithm, we only take two factors into consideration, namely the recommendation novelty and the CF latent factors of songs. More factors (*e.g.*, diversity) can be integrated into our user rating model so as to further improve user satisfaction and recommendation performance.

Most of the previous research in recommender systems mainly focuses on designing better algorithms to improve the *accuracy* of recommendation. Recently, there has been a growing interest in investigating the *diversity* of the recommendation results [10, 38, 50, 51].

We have put emphasis on better exploring and modeling user's musical preferences, but less attention has been paid to increasing the diversity of the recommended songs. It is possible that users in the music recommender system would like to listen to more diverse songs, which help to avoid fatigue and increase freshness. Additionally, diversity of the recommended

songs can facilitate a user's music discovery.

One possible way to integrate diversity into our rating model could be described as follows: Different users may have different interests in songs' diversity, which can be denoted as a parameter k and needs to be learned during the interaction between the user and the recommender system. We need to find a method to represent and measure the diversity d, treat it as the third factor, compute a diversity score $U_D = kd$ and then multiply it with the other two scores U_{CF} and U_N to generate the final user rating.

It may also be interesting to try some ad hoc methods which address the diversity problem in post-processing phase. For example, we can keep the approaches used in the prediction phase unchanged, but choose to recommend a non-greedy song from a genre that is different from the previously recommended n songs.

6.2 Hybrid Recommendation Model

In this paper, we mainly focus on enhancing one popular category of music recommendation approaches (*i.e.* the collaborative filtering approach). We can apply our proposed method to other more sophisticated recommendation approaches (*e.g.*, hybrid recommendation approach) in the future.

A year ago, we developed a hybrid social music recommender system. It is a web application embedded into the Facebook platform. Using this recommender system, we are able to collect multiple sources of input data including rating data, friendship data, music sharing data between friends, and so on. Rating predictions are made based on the analysis of data from these multiple sources. However, a greedy recommendation strategy is

used in this hybrid recommender system. Therefore, we plan to deploy the framework proposed in this paper into this hybrid recommender system in order to check if the framework can also improve the performance of other categories of recommendation approaches.

References

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

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[3] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

[4] R. Bell and Y. Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[5] T. Bertin-Mahieux, D. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *ISMIR 2011: Proceedings of the 12th International Society for Music Information Retrieval Conference, October 24-28, 2011, Miami, Florida*, pages 591–596. University of Miami, 2011.

[6] C. Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[7] J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[8] R. Cai, C.g Zhang, C.and Wang, L. Zhang, and W. Ma. Musicsense: contextual music recommendation using emotional allocation modeling. In *Proceedings of the 15th international conference on Multimedia*, pages 553–556. ACM, 2007.

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

[10] P. Castells, S. Vargas, and J. Wang. Novelty and diversity metrics for recommender systems: choice, discovery and relevance. In *International Workshop on Diversity in Document Retrieval (DDR 2011) at the 33rd European Conference on Information Retrieval (ECIR 2011)*, pages 29–36. Citeseer, 2011.

[11] S. Chib and E. Greenberg. Understanding the metropolis-hastings algorithm. *The American Statistician*, 49(4):327–335, 1995.

[12] W. Davis, K. Gfeller, and M. Thaut. *An introduction to music therapy: Theory and practice*. ERIC, 2008.

[13] J Stephen Downie. Music information retrieval. *Annual review of information science and technology*, 37(1):295–340, 2003.

[14] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. The yahoo! music dataset and kdd-cup'11. In *KDD Cup*, pages 8–18, 2012.

[15] H. Ebbinghaus. *Memory: A contribution to experimental psychology.* Number 3. Teachers college, Columbia university, 1913.

[16] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[17] J. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual ACM international conference on SIGIR*, pages 230–237. ACM, 1999.

[18] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme. Active learning for aspect model in recommender systems. In *Symposium on Computational Intelligence and Data Mining*, pages 162–167. IEEE, 2011.

[19] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme. Non-myopic active learning for recommender systems based on matrix factorization. In *International Conference on Information Reuse and Integration*, pages 299–303. IEEE, 2011.

[20] E. Kaufmann, O. Cappé, and A. Garivier. On bayesian upper confidence bounds for bandit problems. In *International Conference on Artificial Intelligence and Statistics*, pages 592–600, 2012.

[21] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item tax-

onomy. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 165–172. ACM, 2011.

[22] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[23] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[24] V. Kuleshov and D. Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

[25] J. Lee and J. Lee. Music for my mood: A music recommendation system based on context reasoning. In *Smart sensing and context*, pages 190–203. Springer, 2006.

[26] S. Lee, Y. Cho, and S. Kim. Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations. *Information Sciences*, 180(11):2142–2155, 2010.

[27] L. Li, W. Chu, J. Langford, and R. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World Wide Web*, pages 661–670. ACM, 2010.

[28] Q. Li, B. Kim, D. Guan, et al. A music recommender based on audio features. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 532–533. ACM, 2004.

[29] B. Logan. Music recommendation from song sets. In *ISMIR*, 2004.

[30] B. McFee, T. Bertin-Mahieux, D. P.W. Ellis, and G. R.G. Lanckriet. The million song dataset challenge. In *Proceedings of international conference companion on World Wide Web*, pages 909–916. ACM, 2012.

[31] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[32] A. Nanopoulos, D. Rafailidis, P. Symeonidis, and Y. Manolopoulos. Musicbox: Personalized music recommendation based on cubic analysis of social tags. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(2):407–412, 2010.

[33] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Eighth IEEE International Conference on Data Mining*, pages 502–511. IEEE, 2008.

[34] H. Park, J. Yoo, and S. Cho. A context-aware music recommendation system using fuzzy bayesian networks with utility theory. In *Fuzzy systems and knowledge discovery*, pages 970–979. Springer, 2006.

[35] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of*. Addison-Wesley, 1989.

[36] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[37] M. Schedl and D. Schnitzer. Hybrid retrieval approaches to geospatial music recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 793–796. ACM, 2013.

[38] M. Slaney and W. White. Measuring playlist diversity for recommendation systems. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, pages 77–82. ACM, 2006.

[39] R. Sutton and A. Barto. *Introduction to reinforcement learning*. MIT Press, 1998.

[40] P. Symeonidis, M. Ruxanda, A. Nanopoulos, and Y. Manolopoulos. Ternary semantic analysis of social tags for personalized music recommendation. In *ISMIR*, volume 8, pages 219–224, 2008.

[41] S. Tan, J. Bu, C. Chen, and X. He. Using rich social media information for music recommendation via hypergraph model. In *Social media modeling and computing*, pages 213–237. Springer, 2011.

[42] MAL Thathachar and PS Sastry. A class of rapidly converging algorithms for learning automata. 1984.

[43] M. Tiemann and S. Pauws. Towards ensemble learning for hybrid music recommendation. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 177–178. ACM, 2007.

[44] A. Van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651, 2013.

[45] X. Wang, D. Rosenblum, and Y. Wang. Context-aware mobile music recommendation for daily activities. In *Proceedings of the 20th ACM international conference on Multimedia*, pages 99–108. ACM, 2012.

[46] X. Wang, Y. Wang, D. Hsu, and Y. Wang. Exploration in interactive personalized music recommendation: A reinforcement learning approach. *arXiv preprint arXiv:1311.6355*, 2013.

[47] Z. Xing, X. Wang, and Y. Wang. Enhancing collaborative filtering music recommendation by balancing exploration and exploitation. In *Proceedings of the 15th Conference of the International Society for Music Information Retrieval (ISMIR 2014), October 27-31, 2014, Taipei, Taiwan*, pages 445–450, 2014.

[48] K. Yoshii and M. Goto. Continuous plsi and smoothing techniques for hybrid music recommendation. In *ISMIR*, pages 339–344, 2009.

[49] K. Yoshii, M. Goto, K. Komatani, T. Ogata, and H. Okuno. Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *ISMIR*, volume 6, page 7th, 2006.

[50] M. Zhang and N. Hurley. Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 123–130. ACM, 2008.

[51] Y. Zhang, D. Séaghdha, D. Quercia, and T. Jambor. Auralist: introducing serendipity into music recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 13–22. ACM, 2012.

[52] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. Springer, 2008.