

**INTERACTIVE MUSIC RECOMMENDATION: CONTEXT,
CONTENT AND COLLABORATIVE FILTERING**

XINXI WANG
(B.E. Harbin Institute of Technology University)

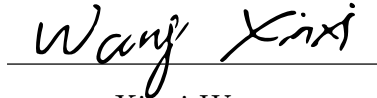
A THESIS SUBMITTED
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE

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.

A handwritten signature in black ink, reading "Wang Xinxi", is written over a horizontal line.

Xinxi Wang

26 Nov 2014

ACKNOWLEDGEMENTS

Many people have given their time and talent in helping me to complete this dissertation.

First and foremost, I would like to express my special appreciation and thanks to my advisor, professor Wang Ye, director of the NUS Sound and Music Computing Lab. I wish to thank you for supporting my research and for allowing me to grow freely as a research scientist.

I would especially like to thank professor David Rosenblum for your valuable feedback and advice when I had little research experience. I would particularly acknowledge professor David Hsu for your advice and encouragement during the most difficult time when writing this thesis.

I would thank Haotian “Sam” Fang for the meticulous editing work that he has done.

I am deeply thankful to my parents and brother for their love, support, and sacrifices. Without them, this thesis would never have been written. This last word of acknowledgement I have saved for my dear wife Xianglian Wang, who has been with me all these years and has made them the best years of my life.

Contents

List of Tables	III
List of Tables	III
List of Figures	IV
List of Figures	IV
1 Introduction	1
1.1 Contributions	3
1.2 Chapter Plan	4
2 Related Work	6
2.1 Recommendation	6
2.2 Music Recommendation	8
2.2.1 Collaborative Filtering	8
2.2.2 Content-Based Music Recommendation	10
2.2.3 Context-Aware Music Recommendation	11
2.2.4 Hybrid Music Recommendation	17
2.3 Deep Learning in Music Recommendation	19
2.3.1 Deep Learning	19

2.3.2	Deep Learning in Music Recommendation and Related Tasks	20
2.4	Reinforcement learning in Music Recommendation	21
2.4.1	Reinforcement Learning	21
2.4.2	Reinforcement Learning in Recommender Systems	23
3	Context-Aware Music Recommendation for Daily Activities	26
3.1	Introduction	26
3.2	Unified Probabilistic Model	29
3.2.1	Problem Formulation	30
3.2.2	Probability Models	30
3.2.3	Music-Context Model	32
3.2.4	Initialization	35
3.2.5	Sensor-Context Model	36
3.3	System Implementation	38
3.4	Experiments	40
3.4.1	Datasets	40
3.4.2	Sensor Data Collection	43
3.4.3	Music-Context Model Evaluation	43
3.4.4	Sensor-Context Model Evaluation	46
3.4.5	User Study	48
3.5	Conclusion	53
4	Content-Based and Hybrid Music Recommendation Using Deep Learning	55
4.1	Introduction	55
4.2	Recommendation Models	58

4.2.1	Collaborative Filtering via Probabilistic Matrix Factorization	59
4.2.2	Content-Based Music Recommendation	60
4.2.3	Hybrid CF and Content-Based Music Recommendation	65
4.3	Experiments	67
4.3.1	Dataset	67
4.3.2	Implementation and Training of deep belief network	69
4.3.3	Evaluation Metrics	70
4.3.4	Probabilistic Matrix Factorization	71
4.3.5	Content-Based Music Recommendation	72
4.3.6	Hybrid Music Recommendation	75
4.4	Conclusion	78
5	Interactive Music Recommendation	79
5.1	Introduction	79
5.2	A Bandit Approach to Music Recommendation	83
5.2.1	Personalized User Rating Model	83
5.2.2	Interactive Music Recommendation	89
5.3	Bayesian Models and Inference	93
5.3.1	Exact Bayesian model	93
5.3.2	Approximate Bayesian Model	94
5.4	Experiments	99
5.4.1	Experiment setup	99
5.4.2	Simulations	103
5.4.3	User Study	109
5.5	Discussion	115
5.6	Conclusion	116

6 Conclusion and Future Work	118
6.1 Conclusion	118
6.2 Future Work	119
Appendices	121
Appendix A	122
A.1 Conditional distributions for the approximate Bayesian model	122
A.2 Variational inference	124
A.3 Variational lower bound	127
Bibliography	129

Abstract

As the World Wide Web becomes the major source of digital music, music recommendation systems have become prevalent. By analyzing related information, e.g., user listening history, music audio content, music recommenders make accurate predictions and thus greatly ease the process of music selection for users and also boost the revenue of online music merchants. However, results produced by existing music recommenders are still not satisfactory because of their ignorance of important relevant information or the drawbacks of the underlying modeling techniques. To better satisfy users' music needs, this thesis strives to improve recommendation performance from three aspects.

First, traditional music recommendation systems rely on collaborative filtering or content-based technologies to satisfy users' long-term music playing needs. To satisfy users' short-term music information needs better, we developed the first *context-aware* music recommendation system that recommends songs to match the target user's daily activities including *sleeping, running, studying, working, walking* and *shopping*.

Second, existing *content-based* music recommendation systems typically employ a *two-stage* approach. They first extract traditional audio content features such as Mel-frequency cepstral coefficients and then predict user preferences. However, these traditional features, originally not created for music recommendation, cannot capture all relevant information in the audio and thus put a cap on recommendation performance. By using a novel deep-learning based model, we unify the two stages into an automated process that simultaneously learns features from audio content and makes personalized recommendations. The features are then incorporated into collaborative filtering to

form an effective *hybrid* recommendation method.

Third, current music recommender systems typically act in a greedy manner by recommending songs with the highest user ratings. Greedy recommendation, however, is suboptimal over the long term: it does not actively gather information on user preferences and fails to recommend *novel* songs that are potentially interesting. A successful recommender system must balance the needs to *explore* user preferences and to *exploit* this information for recommendation. We then present a new approach to music recommendation by formulating this exploration-exploitation trade-off as an *interactive* reinforcement learning task. Moreover, our approach is a single unified model for both music recommendation and playlist generation, which are usually separated by traditional systems.

Extensive evaluation results have demonstrated the effectiveness of the developed methods, and future directions are then discussed.

List of Tables

3.1	Comparison of Classifiers	36
3.2	Summary of the Grooveshark Dataset. <i>Distinct Songs</i> indicates the number of distinct songs from the playlists for the specified context, while <i>Observations</i> indicates the total number of songs including duplicates.	41
3.3	Inter-Subject Agreement on Music Preferences for Different Activities	44
3.4	Activity Classification Accuracy	48
3.5	Questionnaire 1	49
3.6	Context Inference and Recommendation Accuracy Before and After Adaptation	52
3.7	Questionnaire 2	53
4.1	Frequently used symbols	58
4.2	Dataset statistics	69
4.3	Predictive performance of CF and content-based methods using DBN (Root Mean Squared Error). WS and CS stand for warm-start and cold-start, respectively.	74
4.4	Predictive performance of our hybrid method with the features learnt by our HLDBN model and the baseline CB2 model (Root Mean Squared Error)	74
4.5	Comparison between hybrid methods using features learnt from HLDBN and traditional features (mean Average Precision)	76
4.6	Traditional audio features used	77
5.1	Table of symbols	84
5.2	Music Content Features	102

List of Figures

2.1	Three paradigms for incorporating context information into traditional recommender systems [Adomavicius and Tuzhilin, 2008]	11
2.2	Probabilistic model for combining CF and music audio content [Yoshii <i>et al.</i> , 2006]	18
3.1	Graphical representation of the ACACF Model. Shaded and unshaded nodes represent observed and unobserved variables, respectively.	32
3.2	Context-aware mobile music recommender.	39
3.3	Retrieval performance of the music-context model.	46
3.4	Average recommendation ratings. The error bars show the standard deviation of the ratings.	51
4.1	Probabilistic matrix factorization	59
4.2	Hierarchical linear model with deep belief network	61
4.3	Hybrid recommendation	66
5.1	Uncertainty in recommendation	80
5.2	Proportion of repetitions in users' listen history. This boxplot shows a five-number summary: minimum, first quartile, median, third quartile, and maximum.	86
5.3	Zipf's law of song repetition frequency	86
5.4	Examples of $U_n = 1 - e^{-t/s}$. The line marked with circles is a 4-segment piecewise linear approximation.	87
5.5	Relationship between the multi-armed bandit problem and music recommendation	90
5.6	Graphical representation of the Bayesian models. Shaded nodes represent observable random variables, while white nodes represent hidden ones. The rectangle (plate) indicates that the nodes and arcs inside are replicated for N times.	94
5.7	Regret comparison in simulation	106

5.8	Time efficiency comparison	106
5.9	Accuracy versus training time	107
5.10	Sample efficiency comparison	109
5.11	User evaluation interface	110
5.12	Performance comparison in user study	111
5.13	The uncertainty of Bayes-UCB decreases faster than that of Greedy	112
5.14	Distributions of song repetition frequency	113
5.15	Four users' diversity factors learnt from the approximate Bayesian model	115

Chapter 1

Introduction

Music recommendation systems help users find music from large music databases, and an effective system is one that consistently recommends songs that match a user's preference. By analyzing information about users (e.g. music listening history, demographic, contextual information) or songs (e.g. metadata, audio content), music recommenders can predict users' favorite songs very accurately, and thus, they greatly ease the music selection process for the users and boost the revenue of online music merchants. Currently, music recommender systems can be classified according to their methodologies into four categories: collaborative filtering (CF), content-based methods, context-based methods, and hybrid methods [Song *et al.*, 2012].

Collaborative filtering [Resnick *et al.*, 1994] considers every user's listening history; it recommends songs by considering those preferred by other like-minded users. For instance, if user A and user B have similar music preferences, then songs liked by A but not yet considered by B will be recommended to B . The state-of-the-art method for performing CF is non-negative matrix factorization (MF). Although CF is the most widely used and accurate

method, it suffers from the notorious *cold-start problem* [Schein *et al.*, 2002].

The cold-start problem is the issue that the system cannot accurately recommend songs to new users whose preference are unknown (the *new-user problem*) or recommend new songs to users (the *new-song problem*). It could cause a new user to immediately leave a recommender because of a few inaccurate recommendations. It could even become a vicious circle for new recommenders that have little user data: scarce data results in poor recommendation quality, which further limit the chance of attracting users and gathering more data. Solving the cold-start problem is thus crucial for recommenders.

Content-based methods [Casey *et al.*, 2008] recommend songs that have similar audio content to the user’s preferred songs. For instance, if user A likes song S , then songs having content (i.e., musical features e.g. genre, mood, and instrument) similar to S will be recommended to A . Content-based systems mitigate the new-song problem, but they still suffer from the new-user problem, and their recommendation quality is largely determined by audio content features.

Context-based (a.k.a *context-aware*) recommendation systems [Wang *et al.*, 2012a] recommend songs to match various aspects of the user’s context, e.g., activities, environment, or physiological states. They have become increasingly popular in recent years with the advent of sensor-rich and computationally powerful smartphones. Real time user contextual information helps better satisfy users’ short-term music needs and also mitigates the new-user problem. However, they are limited by the richness of the available sensor information.

Hybrid methods [Yoshii *et al.*, 2006] combine two or more of the above methods. By taking advantages of content information, hybrid CF and content-based methods improve the recommendation performance and mitigates the

new-song problem. However, they still suffer from the new-user problem, and similar to the content-based method, their performance depends on the audio content features.

Theoretically, the cold-start problem is caused by the lack of information about the users or songs that is required for making good predictions, i.e. the *uncertainty* about the users or songs. Content-based method and the hybrid method mitigate the cold-start problem because the additional audio content or user context information decreases the uncertainty of the songs or users.

Utilizing additional information is one approach for decreasing the uncertainty; another one is to optimize the interactive music recommendation process in a holistic way. Indeed, music recommendation is an *interactive process* between the target user and the recommendation system: the system recommends a song to the target user, and then the user either explicitly informs the system that he likes/dislikes the song, or gives implicit feedback such as skipping the song or listening repeatedly. As the number of interactions increases, the system knows more about the target user and thus the uncertainty is reduced. The key to optimizing this interactive process holistically is to first recommend informative songs to quickly reduce the uncertainty about the target user's preference - this is termed as "*exploration*". Then the system gradually switches to songs that match the user's preference, which is termed as "*exploitation*". Either too much exploration or too much exploitation results in suboptimal performance, and thus balancing the two is important.

1.1 Contributions

Motivated by the above observations, this thesis contributes from the following aspects:

- We present the first context-aware recommender system we are aware of that recommends songs explicitly for everyday user activities including *sleeping, running, studying, working, walking* and *shopping* [Wang *et al.*, 2012a; Wang *et al.*, 2012b]. It not only satisfies users' short-term needs better but also mitigates the new-user problem.
- We develop a novel content-based recommendation model based on probabilistic graphical model and the deep belief network [Wang and Wang, 2014]. It significantly improves the accuracy of content-based music recommendation. To mitigate the new-song problem of collaborative filtering and improve its accuracy, the learnt features are then incorporated into collaborative filtering to form an accurate hybrid method.
- We present the first approach to balance exploration and exploitation based on *reinforcement learning* and particularly multi-armed bandit in order to improve music recommendation performance over the long term and mitigate the new-user problem [Wang *et al.*, 2014; Xing *et al.*, 2014]. Moreover, while most existing systems separate music recommendation and playlist generation, this work provides a more principled approach by jointly optimizing them in a unified model.

1.2 Chapter Plan

The chapter structure of the rest of this thesis is organized as follows:

- Chapter 2 surveys various other works in the field of music recommendation.
- Chapter 3 presents our context-aware mobile music recommender system.

Chapter 1. Introduction

- Chapter 4 shows our content-based and hybrid recommendation models based on deep learning.
- Chapter 5 describes our multi-armed bandit approach to interactive music recommendation.
- Chapter 6 concludes this thesis and shows a few future study directions.

Chapter 2

Related Work

2.1 Recommendation

Recommendation systems have been developed for a variety of online applications. The most popular ones are for movies [Bell *et al.*, 2009], music [Song *et al.*, 2012], news [Das *et al.*, 2007], books [Herlocker *et al.*, 1999], and products in general. Most of these systems use a common approach i.e. collaborative filtering (CF), which is by far the most accurate compared with other approaches such as content-based ones. The main idea behind collaborative filtering is that if user A and B have similar interest, items liked by A yet not considered by B will be recommended to B . Collaborative filtering can be classified into two categories: memory-based and model-based. Currently the most CF method is matrix factorization (Sec 2.2.1), a model-based approach.

One notorious drawback of CF is the cold-start problem i.e. it cannot handle new users/items. The reason is that CF recommends based on history data about the user/item, which new users/items lack. To mitigate this problem, many methods were proposed, e.g., content-based methods [Mooney and

[Roy, 2000](#)], which try to extract useful information from items' content such as news text, music audio content.

Music recommendation, one of the most popular recommendation applications, shares some major characteristics with other recommendation systems. For example, it has to predict user preference based on history data such as ratings or listening history. However, it also has its own specialties.

First, people in different context prefer different music, so music recommendation needs to be contextualized. For example, many users prefer soothing music when they are going to sleep but energetic music when running. This requirement provides a very unique chance for integrating the prevailing context-aware and mobile computing technologies into music recommendation. However, for book/movie/news recommendation, contextual information either has little impact on user preference or is too difficult to be used.

Second, music recommendation should repeat songs. People do not listen to completely new songs all the time; instead, they usually repeat songs they already know. For book recommendation in Amazon, however, it makes little sense to recommend the same book to the same user twice.

Third, people usually listen to a list of songs one after another in a short while, which makes the sequential and interactive properties of music recommendation very prominent compared with book/movie recommendations. How to effectively and efficiently take advantage of user feedbacks immediately after he/she listens to a song and how to plan in real time a sequence of songs to achieve the maximum effectiveness are all interesting and important research problems that are unique to music recommendation.

Forth, compared with other recommendation systems, content-based music recommendation provides the unique chance for testing feature-learning techniques. Feature learning for textual media like books or news is relatively easy

while it may be overwhelming for movies. Learning features from music audio data is challenging but possible.

2.2 Music Recommendation

Currently music recommender systems can be classified into four categories: collaborative filtering (CF), content-based methods, context-based methods and hybrid methods.

2.2.1 Collaborative Filtering

The main idea behind collaborative filtering is that if user A and user B have similar music preferences, then songs liked by A but not yet considered by B will be recommended to B . This approach has been proved to be fairly effective and widely adopted in many practical web-shopping services such as iTunes music store¹ and Amazon² [Adomavicius and Tuzhilin, 2005]. The state-of-the-art method for performing CF is matrix factorization, which is well summarized in [Koren *et al.*, 2009].

In matrix factorization models, every user i and song j are represented as two vectors \mathbf{p}_i and \mathbf{q}_j respectively in a joint latent factor space, and the rating that user i gives to item j can then be approximated by the dot product of \mathbf{p}_i and \mathbf{q}_j :

$$r_{ij} \approx \mathbf{p}_i' \mathbf{q}_j \quad (2.1)$$

This can also be written as the following matrix format:

$$\mathbf{R} \approx \mathbf{P}' \mathbf{Q} \quad (2.2)$$

¹<http://www.apple.com/itunes/>

²<http://www.amazon.com/>

where \mathbf{R} , usually called the user-item ratings matrix, is the collected ratings from all users for all songs. Matrices \mathbf{P} and \mathbf{Q} are the latent factors for all users and all songs, respectively. This method essentially factorizes the user-item matrix into two matrices with lower ranks. One possible simple approach for the factorization is the singular value decomposition (SVD). However, directly applying the SVD algorithm to \mathbf{R} can cause a severe overfitting problem because \mathbf{R} is usually very sparse for real-world recommendation systems. To address it, regularization can be applied:

$$\min_{\mathbf{P}, \mathbf{Q}} \sum_{i,j} \|\mathbf{R} - \mathbf{P}'\mathbf{Q}\|^2 + \lambda (\|\mathbf{P}\|^2 + \|\mathbf{Q}\|^2) \quad (2.3)$$

where λ , balancing bias and variance, needs to be tuned. The optimization procedure is usually implemented as a stochastic gradient decent algorithm or an alternative least square algorithm. To make the latent factors more interpretable, Zhang *et al.* further restrict the elements in \mathbf{P} and \mathbf{Q} to be non-negative [Zhang *et al.*, 2006].

Although CF is one of the most widely adopted methods, it suffers from two problems. The first problem is the *cold-start* problem. When a new user joins the system, no data of this user can be used to predict his/her preferences, i.e. the *new user* problem; similarly, the system cannot recommend a newly added song to users accurately, i.e. the *new song* problem [Adomavicius and Tuzhilin, 2005]. The second problem is that the recommended songs tend to come from a small number of artists, who are often very familiar to the user. This indicates that much room is still remaining for better utilizing the Long-Tail effect [Anderson, 2006].

2.2.2 Content-Based Music Recommendation

Content-based methods recommend a user songs whose audio content are similar to that of the user’s favorites [Adomavicius and Tuzhilin, 2005]. Usually, the audio content similarity of two songs is calculated based on their audio feature vectors, the most effective ones of which are timbre and rhythm features [Song *et al.*, 2012]. Content-based methods solve the the new song problem to some extent and also result in better artists diversity than CF. However, the accuracy of content-based methods is usually limited for the following reasons.

First, traditional audio content features were not created for music recommendation or music related tasks. For example, MFCC was originally used for speech recognition [Mermelstein, 1976]. They only became attached to music recommendation after the discovery that they can describe high-level music concepts like genre, timbre, and melody. However, these features are by no means optimal for music recommendation. The gap between these features and high level meaning is usually called the *semantic gap*. To address the gap, feature learning methods (e.g. [Lee *et al.*, 2009]) could developed to learn a better representation of songs, but little work has tried so in music recommendation.

Second, the distance function between two audio feature vectors is usually designed in an ad hoc way and not optimized with respect to the recommendation objective. They are usually chosen from a very restrictive set of distance functions such as Euclidean distance [Chen and Chen, 2001; Zhang *et al.*, 2009], Earth Mover’s distance [Logan and Salomon, 2001], or Pearson correlation distance [Bogdanov *et al.*, 2010; Bogdanov *et al.*, 2013]. While two recent works tried to employ machine learning techniques to automat-

ically learn a similarity metric [McFee *et al.*, 2012a; Liu, 2013], they still relied on traditional features. Attempts have been made to perform feature selection or transformation on traditional features [Zhang *et al.*, 2009; Bogdanov *et al.*, 2010], but they remain suboptimal as the traditional features may fail to take into account essential information.

2.2.3 Context-Aware Music Recommendation

2.2.3.1 Context-Aware Recommendation Systems

Context information, which could improve recommendation accuracy significantly, has been utilized in many recommendation systems. As shown in Figure 2.1, context-aware recommender systems can be classified into three categories according to the paradigms that context information is incorporated in: contextual pre-filtering, contextual post-filtering, and contextual modelling [Adomavicius and Tuzhilin, 2008].

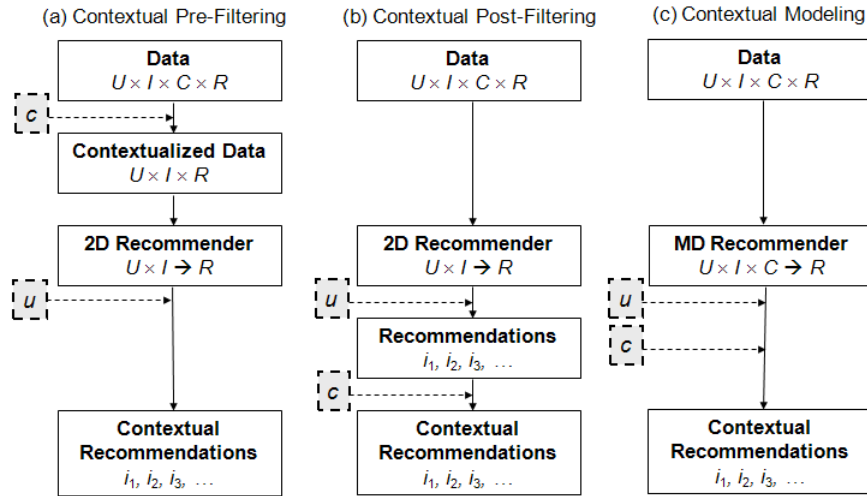


Figure 2.1: Three paradigms for incorporating context information into traditional recommender systems [Adomavicius and Tuzhilin, 2008]

As shown in Figure 2.1, contextual pre-filtering uses the target user’s context to pre-filter the user-item rating matrix so that the remaining users have similar context to the target user’s. Then it recommends using collaborative filtering based on the pre-filtered rating matrix. The advantage of this paradigm is that most traditional recommendation methods can be immediately applied after the pre-filtering phase. In context post-filtering (Figure 2.1), the context information is used to filter the recommendations generated by CF so that the remaining items suit the target user’s context. Post-filtering and pre-filtering share the same advantage, i.e. simplicity. In practice, which of them should be used depends on the application. Contextual modelling tries to integrate context information directly into CF. For example, the context information can be integrated as an additional dimension into the user-item rating matrix, and matrix factorization for three dimensional matrices can then be used for recommendation. Contextual modelling ((Figure 2.1) can better capture the correlation between context, users and items, but new models need to be developed.

This classification provides some insight on designing new approaches to incorporate context information in CF-based recommendation systems, but it is not exhaustive. For example, context-aware recommendation systems which do not rely on collaborative filtering belong to none of the three classes.

2.2.3.2 Context-Aware Music Recommendation Systems

In music recommendation, increasingly many context-aware systems are proposed in order to utilize user context information to better satisfy their short-term needs.

XPod is a mobile music player that selects songs matching a users’ emotional and activity states [Dornbush *et al.*, 2007]. The player uses an external

physiological data collection device called BodyMedia SensorWear. Compared to the activities we consider, the activities considered in XPod are very coarse-grained, namely *resting*, *passive* and *active*. User ratings and metadata are used to associate songs with these activities, but recommendation quality is not evaluated.

In addition to XPod, many other context-aware music recommenders (CAMRs) exploit user emotional states as a context factor. Park *et al.* were probably the first to propose the concept of context-aware music recommendation [Park *et al.*, 2006]. They used a fuzzy Bayesian network to infer a user’s mood (*depressed*, *content*, *exuberant*, or *anxious/frantic*) from context information including weather, noise, time, gender and age. Music is then recommended to match the inferred mood using mood labels manually annotated on each available song. In the work of Cunningham *et al.*, user emotional states are deduced from user movements, temperature, weather and lighting of the surroundings based on reasoning with a manually built knowledge base [Cunningham *et al.*, 2008]. Rho *et al.* built an ontology to infer a user’s mood from context information including time, location, event, and demographic information, and then the inferred mood is matched with the mood of songs predicted from music content [Rho *et al.*, 2009; Han *et al.*, 2010]. However, we believe that with current mobile phones, it is too difficult to infer a user’s mood automatically.

Some work tries to incorporate context information into CF. Lee *et al.* proposed a context-aware music recommendation system based on case-based reasoning [Lee and Lee, 2008]. In this work, in order to recommend music to a target user, other users who have similar context as the target user are selected, and then CF is performed among the selected users and the target user. This work considers time, region and weather as the context information.

Similarly, Su *et al.* use context information that includes physiological signal, weather, noise, light condition, motion, time and location to pre-filter users and items [Su *et al.*, 2010]. Both context and content information are used to calculate similarities and are incorporated into CF.

SuperMusic [Lehtiniemi, 2008], developed at Nokia, is a streaming context-aware mobile service. Location (GPS and cell ID) and time are considered as the context information. Since that application cannot show a user’s current context categories explicitly, many users get confused about the application’s “situation” concept: “If I’m being honest, I didn’t understand the concept of situation. . . . I don’t know if there is music for my situation at all?” [Lehtiniemi, 2008]. This inspired us to design our system recommending music explicitly for understandable context categories such as *working*, *sleeping*, etc.

Resa *et al.* studied the relationship between temporal information (e.g., day of week, hour of day) and music listening preferences such as genre and artist [Resa, 2010]. Baltrunas *et al.* described similar work that aims to predict a user’s music preference based on the current time [Baltrunas and Amatriain, 2009]. Several other works exploit physiological signals to generate music playlists automatically for exercising [Wijnalda *et al.*, 2005; Elliott and Tomlinson, 2006; Oliveira and Oliver, 2008]. Only the tempo attribute of music was considered in those works, whereas in our work, several music audio features including timbre, pitch and tempo are considered. Kaminskas *et al.* recommend music for places of interest by matching tags of places with tags on songs [Kaminskas and Ricci, 2011]. In other work by Baltrunas *et al.*, a song’s most suitable context is predicted from user ratings [Baltrunas *et al.*, 2010], and a CAMRS was built specifically for the context of riding in a car [Baltrunas *et al.*, 2011].

Reddy *et al.* proposed a context-aware mobile music player but did not de-

scribe how they infer context categories from sensor data or how they combine context information with music recommendation [Reddy and Mascia, 2006]. In addition, they provide no evaluation of their system. Seppänen *et al.* argue that mobile music experiences in the future should be both personalized and situationalized (i.e., context-aware) [Seppänen and Huopaniemi, 2008]. Bostrom *et al.* also tried to build a context-aware mobile music recommender system, but the recommendation part was not implemented, and no evaluation is presented [Boström, 2008].

While we use context to refer to mood, activities or physiology states of the user or the physical/social environment around him/her, some other work use context to refer to web context like tags or text surrounding the song on the web [Turnbull *et al.*, 2009], which is beyond the scope of our discussion.

2.2.3.3 Music Content Analysis in CAMRSs

Only a relatively small number of CAMRSs described in the literature use music content information. To associate songs with context categories such as emotional states, most of them use manually supplied metadata or annotation labels or ratings [Park *et al.*, 2006; Dornbush *et al.*, 2007; Oliveira and Oliver, 2008; Lee and Lee, 2008; Cunningham *et al.*, 2008; Lehtiniemi, 2008; Kaminskis and Ricci, 2011; Baltrunas *et al.*, 2010], or implicit feedback [Dornbush *et al.*, 2007]. In other cases, content is not directly associated with context, but is used instead to measure the similarity between two songs in order to support content-based recommendation [Lehtiniemi, 2008; Su *et al.*, 2010].

There are mainly two types of methods to automatically associate music audio content with high level categories: multi-class classification and tagging. The multi-class classification method is used by Rho, Han *et al.*: Emotion classifiers are first trained, and then every song is classified into a single emotional

state [Rho *et al.*, 2009; Han *et al.*, 2010]. The method that we use to associate music content with daily activities is based on a tagging method called Autotagger [Bertin-Mahieux *et al.*, 2008; Zhao *et al.*, 2010b]. Similar methods have been proposed by others [Turnbull *et al.*, 2007], but Autotagger is the only one evaluated on a large dataset. All these methods were used originally to annotate songs with multiple semantic tags, including genre, mood and usage. Although their tags include some of our context categories such as *sleeping*, the training dataset used in these studies (the CAL500 dataset discussed in Section 3.4.1.1) is too small (500 songs with around 3 annotations per song), and evaluations were done together with other tags. From their reported results, it is difficult to know whether or not the trained models capture the relationship between daily activities and music content.

2.2.3.4 Context Inference in CAMRSs

None of the existing CAMRSs tries to infer user activities using a mobile phone. XPod uses an external device for classification—the classified activities are very low-level, and classification is performed on a laptop [Dornbush *et al.*, 2007]. While activity recognition using mobile phones is itself not a new idea, none of the systems that have been studied can be updated incrementally to adapt to a particular user [Saponas *et al.*, ; Brezmes *et al.*, 2009; Berchtold *et al.*, 2010; Khan *et al.*, 2011; Kwapisz *et al.*, 2011; Lee and Cho, 2011]. In one remarkable work, Berchtold *et al.* proposed an activity recognition service supporting online personalized optimization [Berchtold *et al.*, 2010]. However, their model needs to search in a large space using a genetic algorithm, which requires significant computation. And to update the model to adapt to a particular user, all of that user’s sensor data history is needed, thus requiring significant storage.

2.2.4 Hybrid Music Recommendation

Hybrid methods combine two or more of the above methods. In subsequent part of this thesis, we will use “hybrid method” to refer to “hybrid collaborative filtering and content-based method” as it is the most popular hybrid form and also the focus of this thesis. Hybrid CF and content-based methods have been explored extensively in recommenders for other products such as movies [Porteous *et al.*, ; de Campos *et al.*, 2010; Shan and Banerjee, 2010; Park *et al.*, 2013]. Although such approaches can potentially generalize to music recommendation, they have efficiency issues: (1) they use full Bayesian inference [Porteous *et al.*, ; Shan and Banerjee, 2010; Park *et al.*, 2013] or Monte Carlo simulation [Agarwal and Chen, 2009] and are thus slow; (2) they have been applied to a dataset with only thousands of users and items and about 1 million ratings.

To our knowledge, Yoshii *et al.* [Yoshii *et al.*, 2006] are the first to combine CF and content-based methods in music recommendation. In this work, MFCC features were quantized into codewords and used together with rating data in the three-way aspect model, a probabilistic model, originally proposed in [Popescul and Ungar, 2001] for bibliographic recommendation. The model is shown in Figure 2.2, where users, songs, and audio content are assumed to be independent given the latent variable Z . Every time the song with the highest probability $p(m|u)$ is recommended. Parameters of the model are learnt by the Expectation-Maximization (EM) algorithm. Almost concurrently, Li *et al.* [Li *et al.*, 2007] built a probabilistic hybrid approach to unify CF and traditional features.

While Yoshii and Li’s works were promising starting points for *model-based* hybrid methods, subsequent studies all focused on content similarity

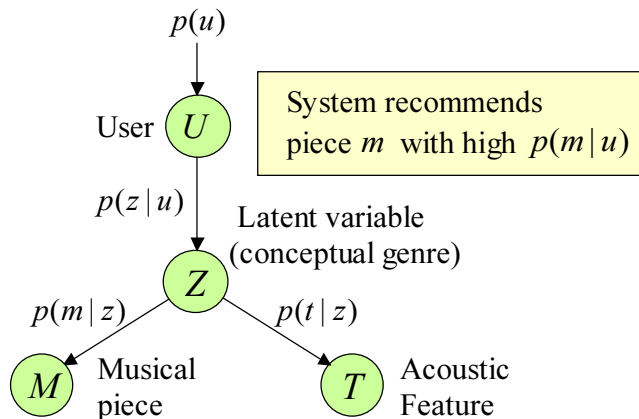


Figure 2.2: Probabilistic model for combining CF and music audio content [Yoshii *et al.*, 2006]

based methods. Castillo [Del Castillo, 2007] proposed a hybrid recommender by linearly combining the results of a content similarity based recommender and a collaborative filtering based one. Tiemann *et al.* [Tiemann and Pauws, 2007] and Shruthi *et al.* [Shruthi *et al.*,] developed approaches that successfully fused CF and content similarities but revealed little information about the fusion process. Bu *et al.* [Bu *et al.*, 2010] and Shao *et al.* [Shao *et al.*, 2009] used hyper-graphs to combine usage data and content similarity information. Domingues *et al.* [Domingues *et al.*, 2012] first obtained song similarities based on CF and content features separately before integrating the two kinds of similarities into a hybrid similarity metric. Similarly, Bogdanov *et al.* [Bogdanov *et al.*, 2013] also combined content similarity and Last.fm’s similarity, which is likely based on CF. Combining the similarities of different modalities is relatively easy and may work to some extent in practice, but the similarity metrics are usually selected in an ad hoc way, which results in suboptimal recommendation performance.

Hybrid methods integrate both user rating data and the music content data

and thus mitigate the new song problem and lead to better recommendation accuracy while maintaining good recommendation diversity. However, they still suffer from the new user problem similar to content-based methods.

Collaborative filtering, content-based methods and hybrid methods satisfy users long-term preferences well. However, since they do not take into account the short-term variations of users' preferences, which are usually influenced by users' context such as activities, they cannot satisfy users' short-term needs well.

2.3 Deep Learning in Music Recommendation

2.3.1 Deep Learning

Deep learning methods mimic the architecture of mammalian brains. They can automatically learn features at multiple levels directly from low-level data without resorting to manually crafted features. We give a very brief introduction to *deep belief networks* (DBN), which will be used in this thesis, and refer the readers to Bengio *et al.* [Bengio, 2009] for a more comprehensive review of deep learning techniques.

A deep belief network is a generative probabilistic graphical model with many layers of hidden nodes at the top and one layer of observations at the bottom. Connections are allowed between two adjacent layers but not between the same layer. Connections of the top two layers are undirected while the rest are directed. Jointly training all layers is computationally intractable, so Hinton *et al.* [Hinton *et al.*, 2006] developed an efficient algorithm to train the model layer by layer from bottom to top in a greedy manner. This unsupervised training process is usually called *pre-training*. Afterward, the DBN

can be converted to a *multi-layer perceptron* (MLP) for supervised learning. This stage is called *finetune* and is usually implemented as *back-propagation*. It is also possible to directly train a MLP using back-propagation without the pre-training step, but this is prone to overfitting, especially when the MLP is deep (has more than two hidden layers). Pre-training may help by implicitly effecting a form of regularization [Erhan *et al.*, 2010].

2.3.2 Deep Learning in Music Recommendation and Related Tasks

The field of music information retrieval (MIR) has only recently begun to embrace the power of deep learning. Lee *et al.* [Lee *et al.*, 2009] used a convolutional deep belief network to extract features in an unsupervised fashion for tasks such as music genre classification. Results show that the automatically learnt features significantly outperforms MFCC. In Hamel *et al.* [Hamel and Eck, 2010], a deep belief network was used for music genre classification and autotagging, with performance surpassing that based on MFCC and MIM feature sets. In [Humphrey *et al.*, 2012; Humphrey *et al.*, 2013], Humphrey *et al.* proposed that the traditional two-stage machine learning process — feature extraction and classification/regression — should be conducted simultaneously. To classify the rhythm style of a piece of music, Pikrakis applied DBN to engineered features representing rhythmic signatures [Pikrakis, 2013]. Schmidt *et al.* [Schmidt and Kim, 2013] found that DBN easily outperforms traditional features in understanding rhythm and melody based on music audio content.

To the best of our knowledge, the first deep learning based approach for music recommendation was almost concurrently proposed by Oord *et al.* [van den Oord *et al.*, 2013] recently. They first conducted matrix factorization to obtain

latent features for all songs, and then used deep learning to map audio content to those latent features.

2.4 Reinforcement learning in Music Recommendation

The music recommendation process is inherently interactive, in which we need to *explore* users' preference and at the same time *exploit* the learnt knowledge to give good recommendations. Balancing the amount of exploration against exploitation is important for achieving optimal recommendation performance. Reinforcement learning techniques provide a principled solution to this problem. In this section, we survey some of these techniques and their applications in recommender systems.

2.4.1 Reinforcement Learning

Unlike supervised learning (e.g., classification, regression), which considers only prescribed training data, a reinforcement learning (RL) algorithm *actively* explores its environment to gather information and exploits the acquired knowledge to make decisions or predictions.

The multi-armed bandit is a thoroughly studied reinforcement learning problem. For a *bandit (slot machine)* with M arms, pulling arm i will result in a random payoff r , sampled from an unknown and arm-specific distribution p_i . The objective is to maximize the *total payoff* given a number of trials. The set of arms is $\mathcal{A} = \{1 \dots M\}$, known to the player; each arm $i \in \mathcal{A}$ has a probability distribution p_i , unknown to the player. The player also knows he has n rounds of pulls. At the l -th round, he can pull an arm $I_l \in \mathcal{A}$, and

receive a random payoff r_{I_t} , sampled from the distribution p_{I_t} . The objective is to wisely choose the n pulls $((I_1, I_2, \dots, I_n) \in \mathcal{A}^n)$ in order to maximize *total payoff* $= \sum_{l=1}^n r_{I_l}$.

A naive solution to the problem would be to first randomly pull arms to gather information to learn p_i (exploration) and then always pull the arm that yields the maximum predicted payoff (exploitation). However, either too much exploration (the learnt information is not used much) or too much exploitation (the player lacks information to make accurate predictions) results in a suboptimal total payoff. Thus, balancing exploration and exploitation is the key issue.

The multi-armed bandit approach provides a principled solution to this problem. The simplest multi-armed bandit approach, namely ϵ -greedy, chooses the arm with the highest predicted payoff with probability $1 - \epsilon$ or chooses arms uniformly at random with probability ϵ . An approach better than ϵ -greedy is based on a simple and elegant idea called upper confidence bound (UCB) [Auer and Long, 2002]. Let U_i be the true expected payoff for arm i , i.e., the expectation of p_i ; UCB-based algorithms estimate both its expected payoff \hat{U}_i and a confidence bound c_i from past payoffs, so that U_i lies in $(\hat{U}_i - c_i, \hat{U}_i + c_i)$ with high probability. Intuitively, selecting an arm with large \hat{U}_i corresponds to exploitation, while selecting one with large c_i corresponds to exploration. To balance exploration and exploitation, UCB-based algorithms follow the principle of “optimism in the face of uncertainty” and always select the arm that maximizes $\hat{U}_i + c_i$.

Bayes-UCB [Kaufmann *et al.*, 2012] is a state-of-the-art Bayesian counterpart of the UCB approach. In Bayes-UCB, the expected payoff U_i is regarded as a random variable, and the posterior distribution of U_i given the history payoffs \mathcal{D} , denoted as $p(U_i|\mathcal{D})$, is maintained, and the fixed-level *quantile* of

$p(U_i|\mathcal{D})$ is used to mimic the upper confidence bound. Similar to UCB, every time Bayes-UCB selects the arm with the maximum quantile. UCB-based algorithms require an explicit form of the confidence bound, which is difficult to derive in our case, but in Bayes-UCB, the quantiles of the posterior distributions of U_i can be easily obtained using *Bayesian inference*. We therefore use Bayes-UCB in our work.

There are more sophisticated RL methods such as Markov Decision Process (MDP) [Szepesvári, 2010], which generalizes the bandit problem by assuming that the states of the system can change following a Markov process. Although MDP can model a broader range of problems than the multi-armed bandit, it requires much more data to train and is often more expensive computationally.

2.4.2 Reinforcement Learning in Recommender Systems

Previous work has used reinforcement learning to recommend web pages, travel information, books, news, etc. For example, [Joachims *et al.*, 1997] use Q-learning to guide users through web pages. In [Golovin and Rahm, 2004], a general framework is proposed for web recommendation, where user implicit feedback is used to update the system. [Zhang and Seo, 2001] propose a personalized web-document recommender where each user profile is represented as vector of terms whose weights of the terms are updated based on the temporal difference method using both implicit and explicit feedback. In [Srivihok and Sukonmanee, 2005], a Q-learning-based travel recommender is proposed, where trips are ranked using a linear function of several attributes including trip duration, price and country, and the weights are updated according to user feedback. [Shani *et al.*, 2005] use a MDP to model the dynamics of user preference in book recommendation, where purchase history is used as the states and

the generated profit the payoffs. Similarly, in a web recommender [Taghipour and Kardan, 2008], browsing history are used as the states, and web content similarity and user behavior are combined as the payoffs. [Chen *et al.*, 2013] consider the exploration/exploitation tradeoff in the rank aggregation problem — aggregating partial rankings given by many users into a global ranking list. This global ranking list can be used for unpersonalized recommenders but is of very limited use for personalized ones.

In the seminal work done by [Li *et al.*, 2010], news articles are represented as feature vectors; the click-through rates of articles are treated as the payoffs and assumed to be a linear function of news feature vectors. A multi-armed bandit model called LinUCB is proposed to learn the weights of the linear function. Our work differs from this work in two aspects. Fundamentally, music recommendation is different from news recommendation due to the sequential relationship between songs. Technically, the additional novelty factor of our rating model makes the reward function nonlinear and the confidence bound difficult to obtain. Therefore we need the Bayes-UCB approach and the more sophisticated Bayesian inference algorithms (Section 5.3). Moreover, we cannot apply the offline evaluation techniques developed in [Li *et al.*, 2011], because we assume that ratings change dynamically over time. As a result, we must conduct online evaluation with real human subjects.

Although we believe reinforcement learning has great potential in improving music recommendation, it has received relatively little attention and found only limited application. [Liu *et al.*, 2009] use MDP to recommend music based on a user’s heart rate to help the user maintain it within the normal range. States are defined as different levels of heart rate, and biofeedback is used as payoffs. However, (1) parameters of the model are not learnt from exploration, and thus exploration/exploitation tradeoff is not needed; (2) the

work does not disclose much information about the evaluation of the approach. [Chi *et al.*, 2010] uses MDP to automatically generate playlist. Both SARSA and Q-learning are used to learn user preference, and, states are defined as mood categories of the recent listening history similar to [Shani *et al.*, 2005]. However, in this work, (1) exploration/exploitation tradeoff is not considered; (2) mood or emotion, while useful, can only contribute so much to effective music recommendation; and (3) the MDP model cannot handle long listening history, as the state space grows exponentially with history length; as a result, too much exploration and computation will be required to learn the model. Independent of and concurrent with our work, [Liebman and Stone, 2014] build a DJ agent to recommend playlists based on reinforcement learning. Their work differs from ours in that: (1) exploration/exploitation tradeoff is not considered; (2) the reward function does not consider the novelty of recommendations; (3) their approach is based on a simple tree-search heuristic, while ours the thoroughly studied muti-armed bandit; (4) not much information about the simulation study is disclosed, and no user study is conducted.

The active learning approach [Huang *et al.*, 2008; Karimi *et al.*, 2011] *only explores* songs in order to optimize the predictive performance on a pre-determined test dataset. Our approach, on the other hand, requires no test dataset and balances *both* exploration and exploitation to optimize the entire interactive recommendation process between the system and users. Since many recommender systems in reality do not have test data or at least have no data for new users, the bandit approach is more realistic compared with the active learning approach.

Chapter 3

Context-Aware Music Recommendation for Daily Activities

3.1 Introduction

Most of the existing music recommendation systems that model users' long-term preferences provide an elegant solution to satisfying long-term music information needs [Adomavicius and Tuzhilin, 2005]. However, according to some studies of the psychology and sociology of music, users' short-term needs are usually influenced by the users' *context*, such as their emotional states, activities, or external environment [North *et al.*, 2004; Levitin and McGill, 2007; Reynolds *et al.*, 2008]. For instance, a user who is running generally will prefer loud, energizing music. Existing commercial music recommendation systems such as Last.fm and Pandora cannot satisfy these short-term needs very well. However, the advent of smart mobile phones with rich sensing

capabilities makes real-time context information collection and exploitation a possibility [Saponas *et al.*, ; Brezmes *et al.*, 2009; Berchtold *et al.*, 2010; Khan *et al.*, 2011; Kwapisz *et al.*, 2011; Lee and Cho, 2011]. Considerable attention has focused recently on *context-aware music recommender systems* (CAMRSs) in order to utilize contextual information and better satisfy users' short-term needs [Camurri *et al.*, 2010; Su *et al.*, 2010; Resa, 2010; Han *et al.*, 2010; Kaminskas and Ricci, 2011].

Existing CAMRSs have explored many kinds of context information, such as location [Kim *et al.*, 2006; Lee and Lee, 2008; Lehtiniemi, 2008; Camurri *et al.*, 2010; Kaminskas and Ricci, 2011], time [Park *et al.*, 2006; Leake *et al.*, 2006; Lehtiniemi, 2008; Baltrunas and Amatriain, 2009; Resa, 2010; Su *et al.*, 2010], emotional state [Park *et al.*, 2006; Dornbush *et al.*, 2007; Reynolds *et al.*, 2008; Cunningham *et al.*, 2008; Rho *et al.*, 2009; Han *et al.*, 2010], physiological state [Kim *et al.*, 2006; Oliveira and Oliver, 2008; Liu *et al.*, 2009; Su *et al.*, 2010; Zhao *et al.*, 2010a], running pace [Wijnalda *et al.*, 2005; Elliott and Tomlinson, 2006; Oliveira and Oliver, 2008], weather [Park *et al.*, 2006; Su *et al.*, 2010], and low-level activities [Dornbush *et al.*, 2007]. To the best of our knowledge, none of the existing systems can recommend suitable music *explicitly* for *daily activities* such as working, sleeping, running, and studying. It is known that people prefer different music for different daily activities [North *et al.*, 2004; Levitin and McGill, 2007]. But with current technology, people must create playlists manually for different activities and then switch to an appropriate playlist upon changing activities, which is time-consuming and inconvenient. A music system that can detect users' daily activities in real-time and play suitable music automatically thus could save time and effort.

Most existing collaborative filtering-based systems, content-based systems

and CAMRSs require explicit user ratings or other manual annotations [Adomavicius and Tuzhilin, 2005]. These systems cannot handle new users or new songs, because without annotations and ratings, these systems are not aware of anything about the particular user or song. This is the so-called *cold-start* problem [Schein *et al.*, 2002]. However, as we demonstrate in this chapter, with automated *music audio content analysis* (or, simply, music content analysis), it is possible to judge computationally whether or not a song is suitable for some daily activity. Moreover, with data from sensors on mobile phones such as acceleration, ambient noise, time of day, and so on, it is possible to infer automatically a user's current activity. Therefore, we expect that a system that *combines* activity inference with music content analysis can outperform existing systems when no rating or annotation exists, thus providing a solution to the cold-start problem.

Motivated by these observations, this chapter presents a ubiquitous system built using off-the-shelf mobile phones that infers automatically a user's activity from low-level, real-time sensor data and then recommends songs matching the inferred activity based on music content analysis. More specifically, we make the following contributions:

- *Automated activity classification*: We present the first system we are aware of that recommends songs explicitly for everyday user activities including *working, studying, running, sleeping, walking* and *shopping*. We present algorithms for classifying these contexts in real time from low-level data gathered from the sensors of users' mobile phones.
- *Automated music content analysis*: We present the results of a feasibility study demonstrating strong agreement among different people regarding songs that are suitable for particular daily activities. We then describe

how we use music content analysis to train a statistical model for predicting the activities for which a song is suitable. This analysis can operate offline since the predictions they produce are independent of individual user activities or listening behaviors.

- *Solution to the cold-start problem:* We present an efficient probabilistic model for *Adaptive Context-Aware Content Filtering* (ACACF) that seamlessly unifies the activity classification and music content analysis results. This model can be updated *on-the-fly* for each user to adapt to their ongoing listening behavior.
- *Implementation and evaluation:* We present a prototype mobile application that implements all parts of the ACACF model except music content analysis entirely on a mobile phone, and we present evaluation results demonstrating its accuracy and usability.

This chapter is organized as follows. Section 3.2 formulates the probabilistic model used to do context-aware recommendation based on context inference and music content analysis. Section 3.3 describes the system design and implementation. Section 3.4 describes evaluations of our model and system. Section 3.5 concludes this chapter.

3.2 Unified Probabilistic Model

In this section we present our Adaptive Context-Aware Content Filtering model, ACACF. The model uses a Bayesian framework to seamlessly integrate context-aware *activity classification* and *music content analysis*.

3.2.1 Problem Formulation

Let \mathcal{S} be a set of songs and \mathcal{C} a set of context categories.¹ For our model, the contexts are daily activities, with $\mathcal{C} = \{running, walking, sleeping, studying, working, shopping\}$. These activities are chosen because they may have impact on users' music preference and they can possibly be detected using sensor data collected from current mobile phones. We can extend the model to other activities with the access to richer sensors in the future. A user is assumed to be in exactly one context category $c \in \mathcal{C}$ at any time. We also assume the user always carries his/her mobile phone, and that a sensor data stream can be recorded continuously from the phone. For our model, the sensor data includes time of day, accelerometer data, and audio from a microphone. The sensor data stream is divided into a sequence of frames, possibly with overlap between adjacent frames. For each frame, a vector \mathbf{f} of *features* of the sensed data is extracted. The recommendation problem is then formulated as a two-step process: (1) infer the user's current context category $c \in \mathcal{C}$ from \mathbf{f} , and (2) find a song $s \in \mathcal{S}$ matching c the best. We call the first step *context inference* and the second step *music content analysis*.

3.2.2 Probability Models

Inferring a user's current context category c from the feature vector \mathbf{f} is not an easy task. In our early experience we found it difficult sometimes to differentiate working and studying by a mobile phone; as sensed activities they appear to be similar, but they need to be differentiated because people have different music preferences when working versus studying. In order to capture

¹In the notation we present, bold letters represent vectors, calligraphic upper case letters represent sets, and random variables and their values are indicated by italicized upper-case and lower-case letters respectively.

such uncertainty, instead of obtaining exactly one context category from \mathbf{f} , we obtain a probability distribution $p(c_i|\mathbf{f})$ over all categories. For instance, if there is complete uncertainty about whether a user is working or studying, then we can assign the probability 0.5 to both working and studying. Using Bayes's rule, $p(c|\mathbf{f})$ can be as in Equation (3.1):²

$$p(c|\mathbf{f}) = \frac{p(\mathbf{f}|c)p(c)}{p(\mathbf{f})} \propto p(\mathbf{f}|c)p(c) \quad (3.1)$$

We call this part of our model the *sensor-context* model, and we elaborate it further in Section 3.2.5.

To model whether a song s is suitable for a context category c , we introduce a random variable $R \in \{0, 1\}$. $R = 1$ means s is suitable for c , and $R = 0$ otherwise. Then we use the probability $p(R = 1|c, s)$ to indicate the user satisfaction degree of song s when he/she is in context c . We call this part of our model the *music-context* model, and we elaborate it further in Section 3.2.3.

Combining $p(\mathbf{f}|c)p(c)$ with $p(R = 1|c, s)$, we obtain the joint probability shown in Equation (3.2):

$$p(c, \mathbf{f}, R, s) \propto p(\mathbf{f}|c)p(R|c, s)p(c) \quad (3.2)$$

We assume that all songs share the same prior probability, so $p(s)$ can be omitted. The combined model can be represented by the graph depicted in Figure 3.1. The combined model is our ACACF model. Random variable Θ is a probability prior and will be explained in Section 3.2.3.1. The model is adaptive in that the component probabilities are updated dynamically as a result of evolving user behavior; the adaptive features of ACACF are presented

²In this and subsequent formulas, we indicate proportional equivalents where normalizing constants can be omitted, thereby improving computation efficiency.

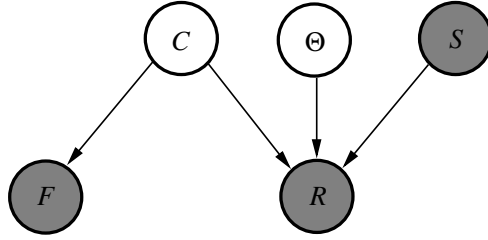


Figure 3.1: Graphical representation of the ACACF Model. Shaded and unshaded nodes represent observed and unobserved variables, respectively.

in Sections 3.2.3.1 and 3.2.5.

With this model, the recommendation task is defined as follows: Given the feature vector \mathbf{f} calculated from sensor data, find a song s that maximizes the user satisfaction $p(R = 1|s, \mathbf{f})$, which is calculated in Equation (3.3) as the sum of the joint probabilities for all possible context categories:

$$\begin{aligned}
 p(R = 1|s, \mathbf{f}) &\propto p(R = 1, s, \mathbf{f}) \\
 &= \sum_{i=1}^{|\mathcal{C}|} p(R = 1, s, \mathbf{f}, c_i)
 \end{aligned} \tag{3.3}$$

To calculate the joint probabilities of Equation (3.2), we compute estimates for the probabilities of the music-context model and the sensor-context model, as explained in Sections 3.2.3 and 3.2.5, respectively.

3.2.3 Music-Context Model

3.2.3.1 Modeling and Adaptation

As described later in Section 3.4.3, we have found that users agree on the *features* of music they prefer when they are doing a particular activity. However, in general, different users like different songs. Thus, to provide a more personalized recommendation, ACACF incorporates implicit user feedback. For

example, if a user listened to a song completely, the user probably likes the song; we call this *positive feedback*. If the user skipped a song after listening for only a few seconds, then the user probably dislikes the song; we call this *negative feedback*. Implicit feedback has been exploited by some researchers [Hu and Ogihara, 2011], and we integrate it seamlessly and efficiently in our own ACACF model.

To model implicit feedback, for each user we assign a *probability prior* (or simply a *prior*) $\Theta_{c,s} \sim \text{beta}(\theta_{c,s}; a_{c,s}, b_{c,s})$ to $p(R|c, s)$ for every pair (c, s) . $\text{beta}(\theta, a, b)$ indicates the beta distribution with shape parameters a, b . Here a, b can be interpreted as the total number of occurrences of negative and positive feedback, respectively, when the user is in context c and is recommended song s . Therefore, the prior captures the personal history of preferences of the user. The probability $p(R = 1|c, s)$ can be expressed as in Equation (3.4):

$$p(R = 1|c, s) = \frac{b}{a + b} \quad (3.4)$$

User feedback can be described as a triple $\mathbf{x} = (\mathbf{f}, s, r)$, where \mathbf{f} is a feature vector extracted from mobile phone sensors during the play of a recommended song, s is the recommended song, and r is the observed value of R , which is the user feedback. The value $r = 0$ indicates negative feedback, while $r = 1$ indicates positive feedback.

The user's true context category is unknown, and thus c is a *latent variable* during adaptation. In this situation, updating the beta prior by exact Bayesian learning is computation-intensive, and thus not suitable for mobile phones. Here approximate inference is used to reduce the computation. First the MAP

estimation \hat{c} of c is given by Equation (3.5):

$$\hat{c} = \arg \max_c p(c|\mathbf{f}) \quad (3.5)$$

Then the corresponding beta prior $\hat{\theta}$ for pair (\hat{c}, s) is updated as in Equation (3.6):

$$p(\hat{\theta}|\mathbf{x}) \approx \begin{cases} \text{beta}(\hat{\theta}; a + 1, b) & \text{if } r = 0 \\ \text{beta}(\hat{\theta}; a, b + 1) & \text{if } r = 1 \end{cases} \quad (3.6)$$

Finally, the corresponding $p(R = 1|\hat{c}, s, \mathbf{x})$ representing the user's preference is updated as in Equation (3.7):

$$p(R = 1|\hat{c}, \mathbf{x}) \approx \begin{cases} \frac{b}{a+b+1} & \text{if } r = 0 \\ \frac{b+1}{a+b+1} & \text{if } r = 1 \end{cases} \quad (3.7)$$

Comparing Equation (3.7) with Equation (3.4), we can see that when a user skips song s in context \hat{c} , $r = 0$ and the probability of that song $p(R = 1|\hat{c}, s)$ decreases. Thus, s will have a smaller chance of being recommended next time. Otherwise, if the user listened completely ($r = 1$), then the probability $p(R = 1|\hat{c}, s)$ increases, and thus s will be more likely to be recommended next time.

The whole updating process is very efficient: We first obtain the MAP estimation \hat{c} , and then the counters a, b and $p(R = 1|\hat{c}, s)$ are updated. The adaptation can be done directly on a mobile phone without the use of backend servers.

We next describe the use of music content analysis results to initialize the beta priors.

3.2.4 Initialization

We model the relationship between music and context by examining the music audio content. There are many existing works on music content classification, such as genre classification, mood classification, and so on. Classification usually assumes that the classes are mutually exclusive. The problem here is different, since one song can be suitable for many context categories. Thus, our problem is similar to the tagging problem: Given a song, we want to know the probability that a tag is suitable for the song. Therefore, we use a state-of-the-art music tagging method called Autotagger [Bertin-Mahieux *et al.*, 2008].

Autotagger estimates the probability $\pi_{c,s}$ that a song s is suitable for context c for *all* users. We use $\pi_{c,s}$ in our model to initialize the prior $beta(\theta; a, b)$ described in Section 3.2.3.1. First, the ratio of a and b is determined by Equation (3.8):

$$p(R = 1|c, s) = \frac{b}{a + b} = \pi_{c,s} \quad (3.8)$$

To further determine a and b , the equivalent sample size β is needed:

$$a + b = \beta$$

β is a free parameter of the system, balancing user feedback against music content analysis results. A large β indicates a belief that music content analysis results are good enough to provide a good recommendation, and that the adaptation (Equation (3.7)) will change $p(R = 1|c, s)$ very slowly. On the other hand, a small β indicates that music content analysis is relatively inaccurate, requiring more reliance on user feedback to perform recommendation. From our subjects' experiences, $\beta = 5$ is a reasonable setting.

	Prediction	Incremental training
AdaBoost	Fast	Slow and Non-trivial
C4.5	Fast	Slow and Non-trivial
LR	Fast	Not supported
NB	Fast	Fast
SVM	Fast	Slow and Non-trivial
KNN	Slow	Fast

Table 3.1: Comparison of Classifiers

After initialization, $p(R = 1|c, s)$ is adapted dynamically to the particular user according to Equation (3.7).

3.2.5 Sensor-Context Model

There are many ways to infer context categories from sensor data. Choosing a proper model is very important and requires careful consideration. First, since much of the computation is to be done on a mobile phone, energy consumption is critically important. Second, the model needs be accurate. Third, in order to adapt the model to a user on the fly as he/she is using the system, the model should support efficient incremental training.

We considered six popular methods used in activity recognition—AdaBoost, C4.5 decision trees, logistic regression (LR), Naive Bayes (NB), support vector machine (SVM) and K-nearest neighbors (KNN). We compared them from three perspectives: prediction accuracy, overhead of prediction computation, and incremental training. Table 3.1 compares the methods qualitatively in terms of prediction overhead and incremental training, and we present results on prediction accuracy in Section 3.4.4.3. We chose Naive Bayes because it offers very good incremental training and prediction overhead with just a small relative loss in accuracy.

Feature vectors extracted from sensor data are usually real-valued vectors.

Since Naive Bayes cannot handle real-valued feature attributes directly, we first discretize the attributes using the well known *equal frequency discretization* method. As a result, every feature vector \mathbf{f} becomes a vector of integers: (f_1, f_2, \dots, f_v) , and $1 \leq f_l \leq d_l$, where d_l is the number of bins of the l -th attribute. Using the Naive Bayes assumption (that the features are conditionally independent), the sensor-context model $p(\mathbf{f}|c)p(c)$ can be decomposed as follows:

$$p(\mathbf{f}|c)p(c) = \prod_{l=1}^v p(f_l|c)p(c) \quad (3.9)$$

To estimate the parameters $p(c)$ and $p(f_l|c)$ in Equation (3.9), training samples need to be collected, which are tuples of the form (\mathbf{f}^k, c^k) , where \mathbf{f}^k is the k -th observed feature vector, and c^k is the corresponding context category. Then, based on Maximum Likelihood Estimation, parameters are learned using Equations (3.10) and (3.11), where $n(c)$ indicates the number of times that category c occurs in the training samples, and $n(F_l = f, c)$ indicates the number of times that the l -th attribute of \mathbf{f} is f and the context category is c :

$$p(c) = \frac{n(c)}{\sum_{i=1}^{|C|} n(c_i)} \quad (3.10)$$

$$p(f_l|c) = \frac{n(F_l = f_l, c)}{\sum_{f=1}^{d_l} n(F_l = f, c)} \quad (3.11)$$

An alternative to incremental training for adaptation is to store all old training data in the mobile phone, and then newly arriving training data is combined with the old data and a new model trained again on the combined dataset. We argue that this is not suitable for a mobile application. First, storing all the training data in the mobile phone is too expensive due to the limited storage space. Second, re-training the model on the complete data

after each update would be too expensive computationally.

For these reasons we opt for incremental training. First, it trains a model on some training data and then discards that training data. When new data arrives, instead of training a completely new model, it uses the new training data to update the model *incrementally* and then again discards the new training data. In this way, no training data needs to be stored, and the model can be updated efficiently.

Incremental training in Naive Bayes is straightforward. According to Equation (3.10) and (3.11), the parameters are estimated using counters $n(c)$ and $n(F_l = f, c)$, which are the sufficient statistics of the sensor-context model. When new training data arrives, these counters are updated, and then parameters $p(c)$, $p(F_l = f|c)$ are updated via Equations (3.10) and (3.11). In this way, the sensor-context model can be efficiently updated to adapt to the user.

3.3 System Implementation

We have implemented the ACACF model in a prototype system, which comprises two components: (1) music audio content analysis on a remote server, and (2) a context-aware music recommender application on a mobile phone.

Music content analysis is done on a server since it is compute-intensive and needs to be performed just once per song. Doing it on a mobile phone would quickly drain the battery.

The mobile application is implemented on the Android SDK, and its interface is depicted in Figure 3.2. To stream music, the application connects to the server via a wireless connection (3G or WiFi). The application also can run without connecting to the server, but then songs and music content analysis results must be cached beforehand.

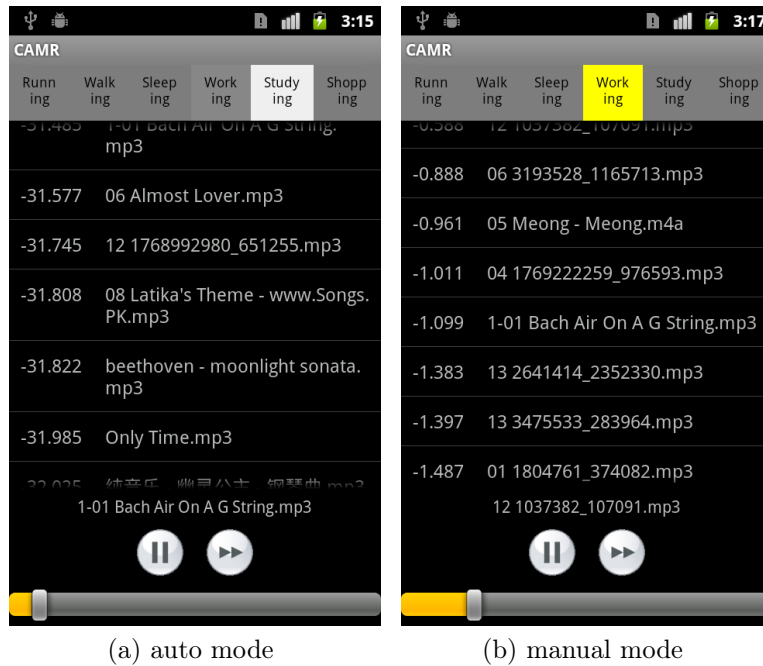


Figure 3.2: Context-aware mobile music recommender.

At the top of the user interface is a list of activities. Users can let the system infer his/her current activity automatically, which is called the *auto mode* and is shown as Figure 3.2a. The background intensity of the activity labels is adjusted according to the inferred probabilities. The whiter the background is, the higher the activity’s probability is. Users also can select a single category manually, which is called *manual mode* and is shown as Figure 3.2b. When an activity is selected manually, its background becomes yellow. To switch back to auto mode from manual model, the user just needs to tap the yellow label once.

When the application is in manual mode, the selected activity and sensor data are used to update the sensor-context model described in Section 3.2.5, which makes context inference increasingly accurate. Ideally, manual mode will not be needed after several days since auto mode should be accurate

enough by then.

The list in the middle of the user interface contains the recommended songs ranked by the probabilities described in Equation (3.3); logarithms of these probabilities are shown on the left side of the songs. At the bottom of the user interface are play/pause and skip buttons.

The adaptation described in Section 3.2.3.1 is performed whenever the user finishes listening to a song or skips a song. After adaptation, the probability of the song just listened to or skipped will be updated, and all songs will be re-ranked. This makes the list of recommended songs increasingly accurate, thereby adapting to the user’s personal preferences.

3.4 Experiments

In this section we describe results from our evaluation of the ACACF model and its prototype implementation. We have conducted preliminary experimental evaluations of both model accuracy and system usability, and the results demonstrate significant promise from both perspectives.

3.4.1 Datasets

3.4.1.1 Playlists Crawled from the Web

To build and evaluate the music-context model, we require a large number of songs with context labels, which we use as ground truth for activity prediction. One dataset we considered is the publicly available CAL500 dataset, which incorporates some usage annotations such as *driving* and *sleeping* [Turnbull *et al.*, 2007]. However, those annotations only partially cover our six categories. Furthermore, although the annotations were made by a large number of sub-

Context	Playlists	Distinct Songs	Observations
Running	393	3430	7810
Walking	197	3601	4123
Sleeping	195	3941	5318
Working	194	4533	4988
Studying	195	3405	4363
Shopping	77	3786	3847
Total	1251	22108	30449

Table 3.2: Summary of the Grooveshark Dataset. *Distinct Songs* indicates the number of distinct songs from the playlists for the specified context, while *Observations* indicates the total number of songs including duplicates.

jects (66 undergraduates), each subject annotated only a small portion of the dataset, and each song was annotated only by around three subjects, which is too few to obtain reliable results.

For these reasons, we constructed a new, larger dataset of 24224 songs crawled from Grooveshark³ and YouTube⁴. Grooveshark has numerous playlists created by users, titled with context information such as *Studying*, *running songs*, etc. From Grooveshark we therefore collected playlists that match our context categories. The audio tracks for the songs were then crawled from YouTube through YouTube’s open data API. Details of the dataset are presented in Table 3.2; the total number of 22108 distinct songs shown in the table is less than 24224, since latter number includes songs not associated with any of our six context categories.

3.4.1.2 Context Annotation of 1200 Songs

From the 22108 distinct songs shown in Table 3.2, we selected 1200 for annotation. It was necessary to consider fewer songs for two reasons. First, data

³<http://grooveshark.com>

⁴<http://www.youtube.com>

crawled from the Web is inevitably noisy, since some users may be careless in their creation of playlists. Second, in order to verify agreement between different users, we require songs labeled by multiple users. In the Grooveshark dataset, most songs exist in only a single playlist. For these reasons, it was necessary to carry out an additional phase of annotation in which *all* songs were annotated by *multiple* subjects to produce the ground truth classification for our study. 1200 songs provides a large sample size but not so large as to make the annotation effort unreasonable for our subjects. We randomly chose the 1200 songs so that there would be roughly an equal number of songs from each context category (as classified by the Grooveshark playlist titles).

We recruited 10 undergraduate students to annotate all 1200 songs through the school’s mailing list. There were equal numbers of males and females. All of them listen to music at least one hour a day, and exercise regularly (at least 3 hour-long sessions per week). They have different culture background and are from India, Malaysia, Singapore, Indonesia, China, and Vietnam. Every participant was rewarded with a small token payment for their time and effort. Participants were chosen with the requirement that they listen to music regularly for at least one hour per day. Annotation was performed through a Web site we set up that simply required clicking checkboxes. Because different parts of the same song can have very different styles, we required the subjects to listen to each song for at least 45 seconds. Subjects were allowed to advance or rewind the music playback. For each song, each subject selected one or more suitable context categories.

The resulting dataset thus contains 1200 songs, with each song annotated by all 10 subjects, and with each subject having selected one or more context categories per song.

3.4.2 Sensor Data Collection

To build and evaluate the sensor-context model, we had the same 10 subjects collect data from onboard sensors on their mobile phones. The sensors we used were gyroscopes, accelerometers, GPS receivers, microphones and ambient light sensors.

Sensor data was collected by a mobile application we designed, which will be offered to other interested researchers in the future. All the mobile phones used are based on Android OS. To make the trained model robust to different phone models, we provided our subjects with five different phone models we purchased from Samsung and HTC. The quality of these phones is also different. Some are expensive and have all the sensors mentioned above, while some are cheaper models having only accelerometer, a GPS receiver and a microphone. We imposed no restrictions on how the subjects held or carried or used their phones. To record a data session, a subject first selected their current context category from the application interface and then recorded 30 minutes of data. Each subject was required to record one session for every context category. The recorded sensor data and selected context category were stored together in a SQLite database on the mobile phone's SD card. The resulting 30-hour dataset contains 6 context categories, and every category has 0.5 hour sensor data collected by every of the 10 subjects.

3.4.3 Music-Context Model Evaluation

Demonstrating agreement on suitable songs for an activity is a very important foundation for music content analysis, because if there is no agreement among people, then a trained music-context model will work only for the users in the training set but will not reliably generalize to other users. Therefore,

Activity	Kappa Agreement	Percent Agreement
Running	0.27	0.35
Working	0.03	0.02
Sleeping	0.29	0.28
Walking	0.03	0.03
Shopping	0.07	0.17
Studying	0.09	0.11

Table 3.3: Inter-Subject Agreement on Music Preferences for Different Activities

we first studied inter-subject agreement. Fleiss’s Kappa [Landis and Koch, 1977] and percent agreement were calculated among the 10 subjects for every context category, and the results are presented as Table 3.3. We observe that all Kappa values are significantly higher than 0 (p-value < 0.0001) and are especially high for *running* and *sleeping*. The results therefore indicate that subjects have statistically significant agreement on context categories, indicating the feasibility of training generalizable statistical models.

Next, the music-context model was trained. The videos we crawled from YouTube were first converted by *ffmpeg*⁵ into mono channel WAV files with a 16KHz sampling rate. Then feature vectors were extracted using a program we developed based on the MARSYAS library⁶, in which a window size of 512 was used without overlapping. The features we used and their dimensionalities are ZeroCrossing (1), Centroid (1), Rolloff (1), Flux (1), MFCC (13), Chroma (14), SCF (24) and SFM (24). To reduce the training set size, we used the mean and standard deviation of feature vectors computed from every 30-second period. Finally, we added the 1-dimensional feature *tempo* to the summarized feature vectors. So the resulting combined feature vector is $79 \times 2 + 1 = 159$ -dimensional.

⁵<http://ffmpeg.org>⁶<http://marsyas.sourceforge.net>

We split the Grooveshark dataset into three disjoint subsets: a training set of 16281 songs, a large test set of 6943 songs, and our annotated dataset of 1200 songs, which we also used as a test set. We used the Autotagger method for the training: Using the training set, one binary AdaBoost classifier was trained for every context category. The classifier for context c_i estimates the probability that a song s_j is suitable for context c_i , which is $p(R = 1|c_i, s_j)$.

To measure the accuracy of these classifiers, we simulated the following retrieval process: Given context c as the query, we use its corresponding classifier to compute the probability $p(R = 1|c, s_j)$ for every song s_j and then rank all songs in descending order according to the estimated probabilities. Then the top- K songs are returned. Suppose there are only L songs of the top- K are labeled with context c in our dataset. Then L/K is the Precision@ K for context c . The final Precision@ K is the average of all Precision@ K for the six categories.

We tested the classifiers on the three datasets. For our dataset of 1200 annotated songs, we used majority voting to determine the context for every song. For instance, if at least six of the 10 subjects annotated a song as being suitable for context c , then the song was labeled with c . The retrieval performance measured by Precision@ K depends on the test set size, because the more songs we have, the more likely that we can find good songs for a context category, and thus the Precision@ K will be higher. Therefore, in order to produce results that are comparable between our annotated song set and the large test set of 6943 songs, a set of 1200 songs was randomly sampled from the large test set; we refer to this as the *small test set* below.

We used a random estimator as our baseline, which ranks all songs randomly. The random estimator was also tested on the annotated dataset (baseline 1), the large test set (baseline 2) and the small test set (baseline 3). All

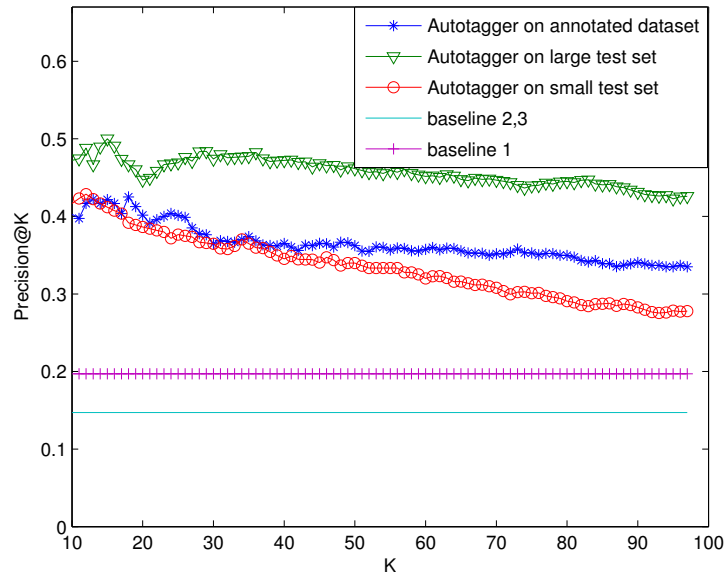


Figure 3.3: Retrieval performance of the music-context model.

results are presented in Figure 3.3. We observe that our trained models significantly outperformed the random estimator. Therefore, the models are able to associate a song accurately with daily activities by examining the song’s audio content.

3.4.4 Sensor-Context Model Evaluation

3.4.4.1 Sensor Selection

Time data were used with data from accelerometers and microphones in our sensor-context model. Although GPS data are used by much of the previous work in human activity recognition, we did not use it for our own work because our activity set is different from other work, plus GPS appears not to increase classification accuracy even though it consumes a great deal of power.

Additionally, we did not use the ambient light sensors and gyroscopes, for two reasons: First, gyroscopes do not improve accuracy very much since accelerometers already provide good motion data. Second, both kinds of sensors reside only in a small number of relatively expensive phones, while our aim is to build a model suitable for most available Android phones.

3.4.4.2 Feature Extraction from Sensor Data

Performing feature extraction from sensor data involves computing feature vectors from the sensor data stream.

Human daily activities have very strong time regularity. Most of us sleep at night and work during the day. Therefore, time is a very important feature for daily activity recognition, and we use the hour of the day in our feature set.

Window size in feature extraction is important. Generally, a larger window size can make inference more accurate because it captures more information. However, a larger window size also reduces system responsiveness, thus degrading the user experience. From our experience, a window size of five seconds appears to be a reasonable setting.

Each accelerometer data sample has three axes, x , y and z . From this data we use the magnitude $m = \sqrt{x^2 + y^2 + z^2}$, which is robust to the direction of the phone. Then the mean, standard deviation, minimum and maximum of all five-second samples of m are used in the final feature vectors.

For audio data from a microphone, we calculate the average amplitude of all samples as a measure of how noisy the environment of the phone is. The final feature vector therefore has $1 + 4 + 1 = 6$ dimensions.

	AdaBoost	C4.5	LR	NB	SVM	KNN
Running	0.974	0.976	0.975	0.841	0.974	0.97
Working	0.933	0.932	0.921	0.876	0.929	0.922
Sleeping	0.999	0.999	0.999	0.994	0.999	0.993
Walking	0.961	0.960	0.955	0.909	0.960	0.953
Shopping	0.972	0.972	0.948	0.953	0.965	0.955
Studying	0.854	0.867	0.835	0.694	0.860	0.855
<i>overall</i>	0.951	0.952	0.941	0.893	0.950	0.943

Table 3.4: Activity Classification Accuracy

3.4.4.3 Context Classification Accuracy

We evaluated AdaBoost, C4.5, LR, NB, SVM and KNN for context classification, and we used 10-fold cross-validation to compare their accuracy. The results are presented in Table 3.4, with a value of 1.0 representing perfect accuracy. We observe that while NB is not as accurate as other methods, it still produces very good results. The categories *studying* and *working* are not distinguished well by any of the methods, because the context information sensed during those two activities is very similar. In fact, sometimes even human beings cannot distinguish the two.

3.4.5 User Study

3.4.5.1 User Needs Study

To understand user needs for music recommendation, we conducted a survey (Questionnaire 1) with our 10 subjects. The questionnaire and survey results are presented in Table 3.5. All questions were answered on a 5-point Likert scale from “strongly disagree” (1) to “strongly agree” (5). Q1 helps in understanding user needs for a context-aware music experience; the results demonstrate that subjects generally prefer different music in different contexts.

	Questions	Mean	Stdev
Q1	I prefer different music (different genre, tempo, pitch, dynamics etc.) when I'm in different context (In different contexts means doing different things e.g. running, sleeping, or at different places e.g. school, home).	3.7	0.95
Q2	I usually listen to different sets of music when I'm in different context .	3.5	1.18
Q3	It is time consuming to create different lists of songs for different contexts with existing technologies.	4.4	0.97
Q4	It is not convenient to change music when I'm doing other things with existing technologies.	4.0	0.94
Q5	I want to have a mobile application that can accurately play suitable music to me according to my context automatically.	4.4	0.52

Table 3.5: Questionnaire 1

The results for *Q2*, *Q3* and *Q4* demonstrate that their requirements cannot be satisfied very well with the existing technologies. Finally, the results for *Q5* demonstrate that a context-aware mobile music recommender potentially can satisfy their needs better.

3.4.5.2 Evaluation of Recommendation Quality

Most existing music recommender systems, including context-aware ones, require user ratings or annotations. During the cold-start stage, these systems are able only to recommend songs randomly. Comparison with existing CAMRSs is impossible, for three reasons: First, we focus on daily activities, and none of the reported literature has used these before. Second, most existing CAMRSs do not infer context categories from mobile phone sensor data. Third, most existing CAMRSs do not use music content analysis. Therefore, for our evaluation we undertook a comparison between the following three

kinds of recommendations:

- (**R1**) recommending songs completely randomly. This simulates traditional recommender systems during the cold-start stage.
- (**R2**) recommending songs with context category inferred by the system automatically. This is the auto mode of our application.
- (**R3**) recommending songs with context category selected by subjects manually. This is the manual mode of our application.

The same 10 subjects participated in this evaluation. The subjects were divided into an experimental group and a control group of five subjects each. The experimental group tested R2 and R3, and the control group tested R1. The subjects did not know which group they were in. All phones were supplied with the music content analysis results and with an identical set of 800 songs chosen randomly from the large test set of 6943 songs described in Section 3.4.3. During evaluation, each subject did each of the six activities for about 20 minutes while listening to the activity’s top recommended songs. Each song was played for about a minute and then rated with the above 5-point Likert scale. Thus, the higher the rating for a song, the more the subject liked the song. Adaptation of both the music-context model and sensor-context model was turned off during this evaluation.

The average and standard deviation of the resulting ratings are presented in Figure 3.4. We observe that R2 performs significantly better than R1 (p-value = 0.0478), and R3 is much better than R1 (p-value = 0.0001). These results indicate that context-awareness combined with music content analysis can produce significantly better results than random recommendation. Therefore, our system provides a promising solution to the cold-start problem. R3 is

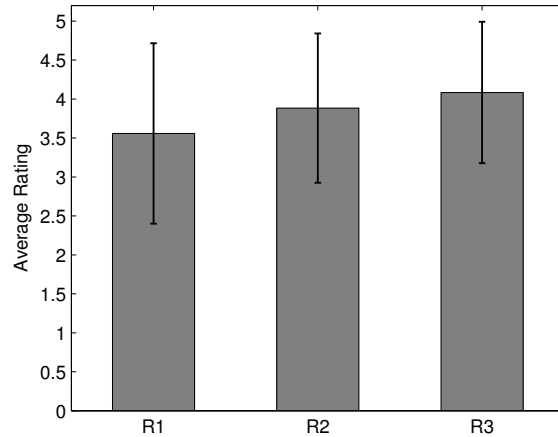


Figure 3.4: Average recommendation ratings. The error bars show the standard deviation of the ratings.

better than R2 but not significantly better (p -value=0.1374), demonstrating that auto mode is almost as good as manual mode, and further demonstrating the accuracy of automated context inference.

3.4.5.3 Adaptation Evaluation

Two of our 10 subjects participated in a one-week adaptation evaluation. The subjects used the application continuously every day for a week. Most of the time the application was used in auto mode. If a subject found that the recommended songs did not match his/her activity, he/she could switch the application to manual mode or skip the recommended song. The whole system was updated continuously and became more and more accurate over the one-week period with respect to the subject's preferences. We compared the accuracy of both context inference and recommendation quality, both before the one-week adaptation period and after the one-week adaptation period. Finer-grained investigation of the performance improvement w.r.t. the amount

	Before Adaptation	After Adaptation
Context Inference	0.87	0.96
Recommendation	0.68	0.93

Table 3.6: Context Inference and Recommendation Accuracy Before and After Adaptation

of adaptation data is possible but requires more times of user evaluations. To reduce the cost, we chose to leave that as future work.

Context inference: The trained Naive Bayes model described in Section 3.4.4.3 was used as the initial sensor-context model. Before and after adaptation, each subject’s sensor-context model was evaluated on sensor data collected by that subject. The average classification accuracy for the two subjects is presented in Table 3.6.

Recommendation: Each subject rated the top-20 recommended songs with “like”, and “dislike” for every context category, both before and after adaptation. Recommendation accuracy is defined as the proportion of liked songs. The results are presented as Table 3.6.

We observe that the accuracy of both context inference and recommendation increased after one week of adaptation although the results could be further confirmed by adding more subjects.

3.4.5.4 User Experience

All 10 subjects completed a second survey at the end of the study (Questionnaire 2). Two of the questions and the survey results are presented in Table 3.7. All questions were answered with a 5-point Likert scale as before. We observe that most of the subjects agree that the application is easy to use and are willing to use it. One subject commented, “I really like the idea, hope you can improve on it and sell it”. However, some of the subjects thought

	Questions	Mean	Stdev
Q6	I can fully understand the functionalities of the mobile application and it's easy to use.	4.6	0.51
Q7	I'm willing to use the application if I have a copy.	4.4	0.84

Table 3.7: Questionnaire 2

that more context categories should be added, such as: *entertaining* (playing games, surfing the Web), *cooking*, *traveling* (bicycle/bus/subway), *partying* and *just relaxing and enjoying music* (no work, no study). Two of the subjects thought the interface could be made more appealing.

3.5 Conclusion

We have proposed a context-aware music recommendation system that combines automated activity classification with automated music content analysis, with support for a rich set of activities and music content features. We collected three datasets—a set of playlists from the Web, a set of 1200 cleanly annotated songs, and a set of sensor data recorded from daily activities. These datasets will be offered eventually to researchers who want to carry out related research on context-aware music recommendation. Based on the set of 1200 annotated songs, we found that although context annotation can be subjective, people nevertheless often do agree on their annotations. Using the datasets, both the sensor-context model and the music-context model were evaluated, and the results are very promising. Based on the probabilistic model, we implemented a CAMRS for off-the-shelf mobile phones. The results from our user study demonstrate that the system is easy to use and can provide good recommendations even in the absence of pre-existing user ratings or annota-

tions, a situation in which traditional systems only can recommend songs randomly. Therefore, our system satisfies users' short-term needs better because of context-awareness, and also provides a solution to the cold-start problem. Evaluation results demonstrate that our system can update itself in real-time to adapt to a particular user.

Chapter 4

Content-Based and Hybrid Music Recommendation Using Deep Learning

4.1 Introduction

A music recommendation system automatically recommends songs that match a user's music preference from a large database. The quality of a match is influenced by many factors concerning the user (e.g., personality, emotional states, activities, social environment) and the song (e.g., music audio content, novelty, diversity).

Among song-related factors, *music audio content* is of great importance. In most cases, we like/dislike a song as a result of characteristics from its audio content, such as vocal, melody, rhythm, timbre, genre, instrument, or lyrics. Without listening to the content, we know almost nothing about the song's quality, let alone whether we would like it. Because music content largely

determines our preferences, it should be able to provide good predictive power for recommendation.

However, existing music recommenders relying on music audio content usually produce unsatisfactory recommendation performance. They all follow a *two-stage* approach: extracting traditional audio content features such as Mel-frequency cepstral coefficients (MFCC), then using these features to predict user preferences [Cano *et al.*, 2005; Yoshii *et al.*, 2006; Wang *et al.*, 2012a]. Traditional audio content features, however, were not created for music recommendation or music related tasks (For example, MFCC was originally used for speech recognition [Mermelstein, 1976]). They only became attached to music recommendation after the discovery that they can also describe high-level music concepts like genre, timbre, and melody. Using such features can result in poor recommendation performance in two ways. First, the high-level concepts cannot be described accurately due to the so-called *semantic gap* [Casey *et al.*, 2008]. Second, even if the feature descriptions are accurate, the high-level concepts may not be essential to the user’s music preferences. Therefore, traditional features could fail to take into account information relevant to music recommendation.

We believe that the key to an effective content-based music recommendation method is a set of good content features. *Manually* crafting such features is possible but time consuming and painstaking. A better approach is to combine the existing two-stage approach into a unified and automated process: features are *learnt* automatically and directly from audio content to maximize recommendation performance. Recent development in *deep learning* techniques [Bengio *et al.*, 2012] has made such a unified approach possible. In fact, people have already started using deep learning to learn features for other music tasks such as music genre classification [Hamel and Eck, 2010] and

music emotion prediction [Schmidt and Kim, 2011] with promising results.

Content methods also frequently combines *collaborative filtering* (CF), which recommends songs based on the interests of like-minded users. Most existing recommenders are based on CF because of its superior accuracy [McFee *et al.*, 2012b]. However, as it depends solely on usage data, CF is powerless when confronted with the new-song problem — it cannot recommend songs without prior usage history. Content-based methods do not suffer from this problem because they can predict based on a song’s audio content, which is usually available for online merchants. Therefore, content-based methods can rescue CF in the new-song scenario. Because CF and content-based methods take advantage of different dimensions of information, it is possible to combine them into a *hybrid method* for better predictions.

Thus motivated, we first develop a content-based model that automatically and simultaneously extracts features from audio content and makes personalized recommendations. We then develop a hybrid method to combine both CF and content features. Specifically, this work seeks to make the following contributions:

- *Content-based method*: We develop a novel content-based recommendation model based on probabilistic graphical model and the *deep belief network* (DBN) proposed by the deep learning community [Hinton *et al.*, 2006]. It unifies feature learning and recommendation. While it does not rely on collaborative filtering, it outperforms baseline content-based models relying on CF in both the *cold-start* stage and *warm-start* stage.
- *Hybrid method*: To combine CF and music content, we apply the automatically learnt audio features to an efficient hybrid model. Experimen-

Symbol	Description
u	User u
v	Song v
r_{uv}	The rating that user u gives to song v
γ_u	The latent features for u estimated by MF
y_v	The latent features for v estimated by MF
β_u	User u 's preference of content features
μ	All users' common preference
x_v	The learnt content features for song v
Ω	The parameters of DBN
\mathcal{U}, \mathcal{V}	User and song sets, respectively
U, V	The number of users and songs, respectively
I	All user, song pairs in the training dataset

Table 4.1: Frequently used symbols

tal results show that our method outperforms CF and the traditional feature based hybrid method.

The remainder of this chapter is organized as follows. Section 2 describes our content-based and hybrid recommendation model and discusses the baseline content-based model used in our experiments. Section 3 describes our extensive experimental evaluations. Section 4 concludes this work and discusses future research directions.

4.2 Recommendation Models

In this section, we will introduce our content-based model and hybrid model, as well as the two baseline content-based models with which to compare our models.

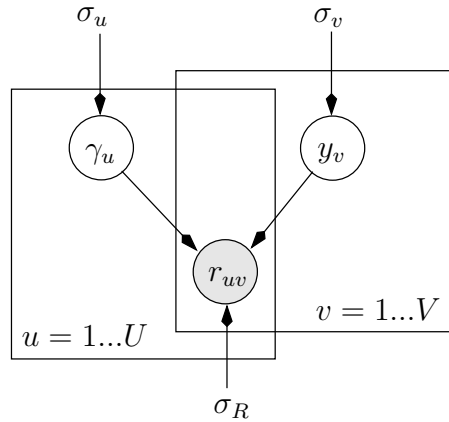


Figure 4.1: Probabilistic matrix factorization

4.2.1 Collaborative Filtering via Probabilistic Matrix Factorization

Collaborative filtering is a popular recommendation method. The state-of-the-art CF methods are based on matrix factorization (MF). A MF method named probabilistic matrix factorization (PMF) [Salakhutdinov and Mnih, 2008b] is used in this work for its simplicity, accuracy, and efficiency. In addition, PMF’s principled probabilistic interpretation enables it to be extended to incorporate content information more easily.

PMF assumes that each user $u \in \mathcal{U}$ and song $v \in \mathcal{V}$ can be represented as latent feature vectors γ_u and y_v , respectively. The rating that user u gives to song v is the inner product of γ_u and y_v . The training data is usually very sparse, and without regularization the model is crippled by severe overfitting. Therefore, Gaussian priors are used for both γ_u and y_v as regularization. Formally, the model is specified as the following¹ (see graphical representation in

¹ $\mathcal{N}(a, b)$ is the normal distribution with mean a and variance b . $x \sim p$ indicates that x satisfies the distribution p or x is drawn/generated from p .

Figure 4.1):

$$\begin{aligned} r_{uv} | \gamma_u, y_v, \sigma_R &\sim \mathcal{N}(\gamma'_u y_v, \sigma_R^2) \\ \gamma_u | \sigma_u &\sim \mathcal{N}(\mathbf{0}, \sigma_u^2 \mathbf{I}) \\ y_v | \sigma_v &\sim \mathcal{N}(\mathbf{0}, \sigma_v^2 \mathbf{I}) \end{aligned}$$

The negative log-likelihood of the model can be simplified as Equation (4.1), where I is the user-song pairs in the training set. λ_u and λ_v are usually tuned using a validation data set [Salakhutdinov and Mnih, 2008b].

$$L_{\text{MF}} = \sum_{u,v \in I} (r_{uv} - \gamma'_u y_v)^2 + \lambda_u \sum_u \|\gamma_u\|^2 + \lambda_v \sum_v \|y_v\|^2 \quad (4.1)$$

Since a new user/song without rating data has no vector representation in the model, their ratings cannot be predicted. This cold-start problem is endemic to all CF methods. In the following sections, we will introduce our solution to the new-song problem.

4.2.2 Content-Based Music Recommendation

4.2.2.1 Hierarchical Linear Model with Deep Belief Network (HLDBN)

We assume that the audio content of song v is f_v , and its automatically learnt feature vector is x_v . User u 's music preference is represented as a vector β_u . The rating that u gives to song v , denoted as r_{uv} , is the inner product of x_v and β_u . We use μ to represent all users' common music preference, which is the mean of all users' β_u -s. The model (Figure 4.2) is formulated as:

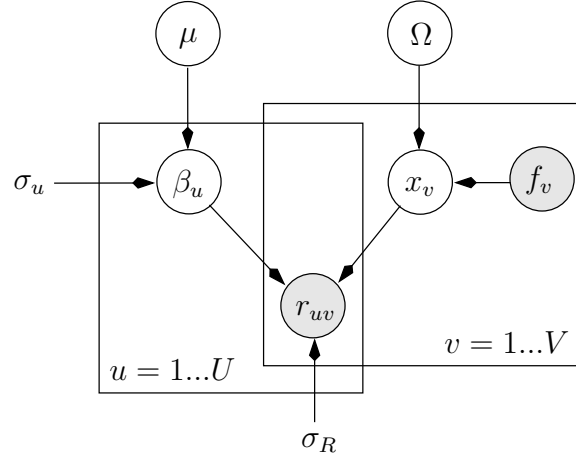


Figure 4.2: Hierarchical linear model with deep belief network

$$\begin{aligned}
 r_{uv} | \beta_u, x_v &\sim \mathcal{N}(\beta_u' x_v, \sigma_R^2) \\
 \beta_u &\sim \mathcal{N}(\mu, \sigma_u^2 \mathbf{I}) \\
 x_v &= \text{DBN}(f_v; \Omega)
 \end{aligned}$$

σ_u indicates the variance of user preferences. The smaller the σ_u , the more similar the user preferences are to the common preference μ and the more strongly β_u is regularized. The Gaussian prior for β_u models user common interests as *one* cluster. However, users of different genders, ages and different culture backgrounds could form different groups. To capture this grouping effect, we could change the single Gaussian prior to a mixture of Gaussians. We tried such a prior and used Monte Carlo Expectation Maximization to estimate the parameters, but it resulted in overfitting. Therefore, we chose a single Gaussian as the prior.

Automatic learning of features x_v from music content f_v is achieved by deep belief network (DBN), which is briefly introduced in Section 2.3.1 and 2.3.2. DBN can be treated as a very flexible deterministic function that maps f_v

to x_v . It has hundreds of thousands, perhaps even millions, of parameters (denoted as Ω) to be learnt from training data. We assume that r_{uv} follows a normal distribution to account for the noise in user ratings.²

Learning - Maximum Likelihood Estimation (MLE) is used to train the model. The negative log-likelihood of the model is shown in Equation (4.2), where irrelevant constants are omitted. The hyperparameter λ is the ratio σ_u^2/σ_R^2 , with a larger λ indicating stronger regularization.

$$L_{\text{HLDBN}} = \sum_{u,v \in I} (r_{uv} - \beta'_u \text{DBN}(f_v, \Omega))^2 + \lambda \sum_u \|\beta_u - \mu\|^2 \quad (4.2)$$

Since Ω consists of a large amount of parameters, directly optimizing L_{HLDBN} using gradient descent could easily overfit. Following the DBN training procedure established in [Hinton *et al.*, 2006], we first pre-train the DBN as stacked layers of Restricted Boltzmann Machines in an unsupervised fashion and then optimize L_{HLDBN} using mini-batch stochastic gradient descent, where the gradient descent part of DBN is implemented as back-propagation.³

Unlike the traditional two-stage methods, our model automatically and simultaneously optimizes audio features (x_v) and user preference parameters (β_u -s). This provides a unified and more principled method to content-based recommendation.

Prediction - After the learning phase, the rating that user u gives to song v can be estimated as $\hat{r}_{uv} = \gamma'_u \text{DBN}(f_v, \Omega)$. As the predictions are based on audio content, new songs can be recommended accurately as well.

²The normal distribution may be replaced with a softmax or probit model. In this work, we follow PMF and use the normal distribution to keep the model clean.

³For the following DBN-based models, the same training approach is used.

4.2.2.2 Baseline Models

We now turn our attention to the two content-based approaches proposed in Oord *et al.* and hereby used as our baseline methods [van den Oord *et al.*, 2013]. The models are based on convolutional neural network (CNN), another popular deep learning method. To make their approach directly comparable with ours, we replace CNN with DBN while keeping the other parts unchanged.

Content-based baseline model 1 (CB1) - This model first uses PMF to learn latent features γ_u and y_v for all users and songs and then trains a DBN to map from audio content to the latent features y_v . Formally, the objective can be formulated as:

$$\min_{\Omega} \sum_v (y_v - \text{DBN}(f_v, \Omega))^2 \quad (4.3)$$

Let $x_v = \text{DBN}(f_v, \Omega)$; the rating that user u gives to song v can be predicted as $\hat{r}_{uv} = \gamma'_u x_v$. This model, however, fails because of a fundamental flaw shown in Theorem 1.

Theorem 1. *Model CB1 does not minimize the sum of squared errors of predicted ratings.*

Proof. Let $\epsilon_v = y_v - x_v$. The optimization objective Equation (4.3) is equivalent to

$$\min_{\Omega} \sum_v \|\epsilon_v\|^2 \quad (4.4)$$

Instead of predicting the latent features, our *true objective* is to predict ratings, so we actually need to minimize the sum of squared errors of predicted ratings:

$$\min_{\Omega} \sum_{u,v \in I} (r_{uv} - \gamma'_u x_v)^2 \quad (4.5)$$

which can be transformed as:

$$\begin{aligned}
 & \min_{\Omega} \sum_{u,v \in I} (r_{uv} - \gamma'_u x_v)^2 \\
 &= \min_{\Omega} \sum_{u,v \in I} (r_{uv} - \gamma'_u (y_v - \epsilon_v))^2 \\
 &= \min_{\Omega} \sum_v \epsilon_v \left(\sum_{u:u,v \in I} \gamma_u \gamma'_u \right) \epsilon'_v + 2 \sum_{u:u,v \in I} \gamma'_u (r_{uv} - \gamma'_u y_v) \epsilon_v \quad (4.6)
 \end{aligned}$$

Since ϵ_v is not constrained because MLPs are universal approximators [Hornik *et al.*, 1989], we can see that Equation (4.6) and (4.4) have different optimal solutions. \square

The original model in Oord *et al.* [van den Oord *et al.*, 2013] uses weighted sum of squared errors. Following the same approach, we can prove that CB1 does not minimize the weighted version, either.

Content-based baseline model 2 (CB2) - This is the other model proposed by Oord *et al.* [van den Oord *et al.*, 2013]. It is presented as the following,

$$\min_{\Omega} \sum_{u,v \in I} (r_{uv} - \gamma'_u \text{DBN}(f_v, \Omega))^2 \quad (4.7)$$

where γ_u is obtained from MF beforehand. Rating r_{uv} is predicted as $\gamma'_u x_v$, where $x_v = \text{DBN}(f_v, \Omega)$.

This model uses the correct objective and thus does not have the issue of CB1 discussed in Theorem 1. However, it lacks regularization on the parameters, which may cause overfitting. We will show this empirically in Section 4.3.5.

Another issue of both CB1 and CB2 is that they are directly based on the

results of MF and thus their prediction results are strongly correlated with the collaborative filtering (CF) results. As we will show in Section 4.3.5 and 4.3.6, this hinders us from combining CB1 or CB2 with CF to form an effective hybrid approach.

4.2.3 Hybrid CF and Content-Based Music Recommendation

Collaborative filtering and content-based methods use different information. To fuse all the information available for more accurate predictions, we can combine the two in a hybrid method.

Information fusion has been studied extensively in other domains such as sensor fusion and multimedia information fusion. There are mainly two approaches for our problem. Decision fusion combines the prediction results from existing CF and content-based methods. On the other hand, data fusion develops a new unified model to incorporate both CF and audio content. Our hybrid method is based on the latter, but it also uses the features learnt by HLDBN.

In our hybrid model (Figure 4.3), we assume that the audio features x_v for every song is already known. γ_u , y_v and β_u are not directly adopted from the results of PMF and HLDBN but need to be jointly learnt from data. Rating r_{uv} is predicted by the sum of the CF part $\gamma'_u y_v$ and the content part $\beta'_u x_v$. The priors for γ_u and y_v are set following the PMF model, and β_u the HLDBN model.

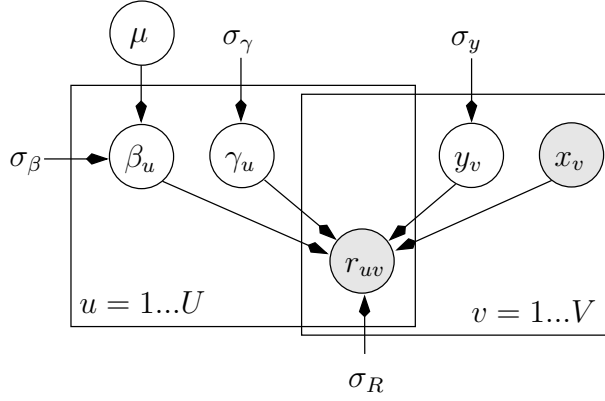


Figure 4.3: Hybrid recommendation

$$\begin{aligned}
 r_{uv} | \beta_u, x_v, \gamma_u, y_v, \sigma_R &\sim \mathcal{N}(\beta_u' x_v + \gamma_u' y_v, \sigma_R^2) \\
 \beta_u | \sigma_\beta &\sim \mathcal{N}(\mu, \sigma_\beta^2 \mathbf{I}) \\
 \gamma_u | \sigma_\gamma &\sim \mathcal{N}(\mathbf{0}, \sigma_\gamma^2) \\
 y_v | \sigma_y &\sim \mathcal{N}(\mathbf{0}, \sigma_y^2)
 \end{aligned} \tag{4.8}$$

The negative log-likelihood can be simplified as the following, where $\lambda_\beta = \sigma_R^2 / \sigma_\beta^2$, $\lambda_\gamma = \sigma_R^2 / \sigma_\gamma^2$, and $\lambda_y = \sigma_R^2 / \sigma_y^2$.

$$\begin{aligned}
 L_{\text{Hybrid}} &= \sum_{u,v \in I} (r_{uv} - \beta_u' x_v - \gamma_u' y_v)^2 + \lambda_\beta \|\beta - \mu\|_F^2 \\
 &\quad + \lambda_\gamma \|\gamma\|_F^2 + \lambda_y \|y\|_F^2
 \end{aligned}$$

L_{Hybrid} is not a convex function, but if we fix any three of β_u , μ , γ_u , and y_v , it is convex and the optimal solution can be obtained in closed form. We thus optimize L_{Hybrid} using the alternative least square (ALS) algorithm: we first set the derivatives of L_{Hybrid} with respect to each of the four parameters to zero and solve the equations, which results in the following four updating formulas. We then iterate them until L_{Hybrid} converges or until the prediction

performance on a validation set reaches the highest point.

$$\begin{aligned}\beta_u &\leftarrow \left(\sum_{v:u,v \in I} x_v x'_v + \lambda_\beta \mathbf{I} \right)^{-1} \left(\lambda_\beta \mu + \sum_{v:u,v \in I} (r_{uv} - \gamma'_u y_v) x_v \right) \\ \gamma_u &\leftarrow \left(\sum_{v:u,v \in I} y_v y'_v + \lambda_\gamma \mathbf{I} \right)^{-1} \left(\sum_{v:u,v \in I} (r_{uv} - \beta'_u x_v) y_v \right) \\ y_v &\leftarrow \left(\sum_{u:u,v \in I} \gamma_u \gamma'_u + \lambda_y \mathbf{I} \right)^{-1} \left(\sum_{u:u,v \in I} (r_{uv} - \beta'_u x_v) \gamma_u \right) \\ \mu &\leftarrow \frac{1}{U} \sum_{u \in \mathcal{U}} \beta_u\end{aligned}$$

To achieve faster convergence, γ_u , y_v are first initialized using PMF.

We could create a pure data fusion model by adding $x_v = \text{DBN}(f_v, \Omega)$ after Equation 4.8, and optimizing Ω jointly with other parameters. However, we found its performance inferior to the one above.

4.3 Experiments

4.3.1 Dataset

Deep belief network has a large number of parameters, and a large amount of data is required to adequately train such a model. We chose The Echo Nest Taste Profile Subset [McFee *et al.*, 2012b] because it is the largest publicly available music recommendation dataset as far as we know. The original dataset has 1,019,318 users, 384,546 songs, and 48,373,586 listening histories. We were able to crawl preview audio clips with length of about half a minute from 7digital⁴ for 282,508 of the songs. We selected the top 100,000 users mainly to reduce the training time.

Implicit feedback - From the Taste Profile Subset, we know the songs

⁴<http://7digital.com>

that a user has listened to, so the dataset can be presented as a set of (user, song) pairs. We assign a rating of 1 to each pair and use them as the positive samples. To generate negative samples, we use the well-established User-Oriented Sampling method built in [Pan *et al.*, 2008]: for user u who listened to songs \mathcal{V}_u , we randomly sampled $|\mathcal{V}_u|$ songs from $\mathcal{V} \setminus \mathcal{V}_u$ and assign a rating 0 to each generated (user, song) pair. We now have equal number of positive and negative samples for every user.

Instead of using the sample method described above, HLDBN could theoretically use the weighted matrix factorization method proposed in Hu *et al.* [Hu *et al.*, 2008] to directly handle the implicit feedback. While the method may be more accurate, the computational overhead is prohibitive: there will be about 2.83×10^{10} rating data points, which can make all algorithms about 1000 times slower and take years to finish.

Table 4.2 gives the statistics of the final dataset. The density of the rating matrix is only 0.1%.

Splitting the dataset - The dataset was then split into 5 disjoint sets: the training set, warm-start validation/test sets, and cold-start validation/test sets. All users and songs in the warm-start sets need to be in the training set. To simulate the new-song problem, songs in the cold-start validation/test sets cannot exist in the training set, while all users in the cold-start sets still need to be in the training set because the new user problem is not our focus. The statistics of the five datasets are shown in Table 4.2, where WS and CS stand for warm-start and cold-start, respectively.

Audio content preprocessing - We first converted all audio clips to WAV files with mono channel, 8kHz sampling rate, and 16 bit depth. We then randomly sampled a 5-second continuous segment from each audio clip, because directly using the half-minute clips requires too much memory and

	# of users	# of songs	# of ratings
Total	100,000	282,508	28,258,926
Train	100,000	262,508	18,382,954
WS Valid	100,000	262,454	3,939,204
WS Test	100,000	262,457	3,939,206
CS Valid	99,963	10,000	1,025,654
CS Test	99,933	10,000	971,908

Table 4.2: Dataset statistics

computation while segments shorter than 5 seconds may lose too much information. We next converted each 5-second segment into a 166×120 spectrogram (30ms window, no overlap). PCA was then used to transform the spectrograms into vectors whose dimensions were ranked according to their significance. The top- K dimensions were finally normalized to have zero mean and unit variance and fed into DBN. The normalization step is required because we use Gaussian-Bernoulli RBM for the DBN’s input layer [Hinton, 2012]. K , the dimensionality of f_v and the number of nodes of the DBN’s input layer, is determined by a validation step.

4.3.2 Implementation and Training of deep belief network

We implemented our DBN using Theano⁵, because it supports convenient GPU programming and automatic symbolic differentiation. Since our input for DBN is continuous, we used the Gaussian-Bernoulli RBM for the input layer and binary RBMs for the rest [Hinton, 2012].

Training DBN on CPUs is extremely slow. GPUs with large memory are thus indispensable in the deep learning experiments. Training and testing of

⁵<http://deeplearning.net/software/theano/>

every model takes three to four days using a single GPU with 6GB GPU memory. Since DBN has many hyperparameters that could have great impact on the prediction performance, tuning seems unavoidable at this stage. Sequentially trying each configuration of the hyperparameters on a single GPU is too time consuming. We thus utilized a GPU cluster with 15 computing nodes, each of which containing two Tesla M2090 GPU cards.

Mini-batch stochastic gradient descent was used as the training algorithm. We cannot transfer all data into one GPU because of its memory limit. Sequentially transferring one batch after computing the previous batch is slow because of the low bandwidth of the bus between the GPU memory and main memory. Our solution is to use multithreading to enable computing and transferring next batch at the same time.

4.3.3 Evaluation Metrics

The Root Mean Square Error (RMSE) metric was used to evaluate most models in this work. It is defined as:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^N (\hat{r}_i - r_i)^2}{N}}$$

where r_i , \hat{r}_i are the true and predicted ratings, respectively. We prefer RMSE to the truncated mAP used in the million song dataset challenge [McFee *et al.*, 2012b] because our models are regression models, for which RMSE is a more accurate and sensible metric [McFee *et al.*, 2012b]. Moreover, RMSE is feasible in our case because the sampling step in the preprocessing of the dataset described in Section 4.3.1 have converted all implicit feedback into explicit ratings.

For models which rank songs instead of predicting ratings, we still resort to truncated mAP. mAP was originally widely used in information retrieval to measure the ranking quality of search results. Suppose the system recommends user u a list of songs $l_{u,1}, l_{u,2} \dots l_{u,M}$, we first define the precision-at- k (P_k) metric as:

$$P_k(u, l_u) = \frac{1}{k} \sum_{i=1}^k r_{u,l_{u,i}}$$

Then we define the average precision as the following,

$$\text{AP}(u, l_u) = \frac{1}{n_u} \sum_{k=1}^M r_{u,l_{u,k}} P_k(u, l_u)$$

where n_u is the number of songs preferred by user u , i.e. $n_u = \sum_{v=1}^V r_{u,v}$.

Finally, mAP is defined as:

$$\text{mAP} = \frac{1}{U} \sum_{u=1}^U \text{AP}(u, l_u)$$

4.3.4 Probabilistic Matrix Factorization

Because CB1, CB2 and the hybrid method all depend on the results of PMF, we trained several PMF models with different configurations using the Alternative Least Square algorithm. The training procedure was stopped when the performance on the warm-start validation set converged. We found that the best performance on the validation set was achieved when the dimensionality of the latent features was 100 and $\lambda_u = \lambda_v = 4$. Further increasing the dimensionality of the latent features brought little improvement.

The results for CF are shown in Table 4.4. We should note that a rating

predictor which randomly generates 0s or 1s have $RMSE = 0.707$, and a mean predictor which constantly gives 0.5s has $RMSE = 0.5$.

There is no result for PMF on the cold-start validation/test sets, as PMF cannot recommend during the cold-start stage (see Section 4.2.1).

4.3.5 Content-Based Music Recommendation

Comparisons between deep learning based methods and traditional features (e.g. MFCC) based methods have been conducted in [van den Oord *et al.*, 2013]. We avoid repeating those comparisons and only compare HLDBN with CB1 and CB2.

Because the objectives for the warm-start and cold-start scenarios are different, we discuss them separately in the following two sections.

4.3.5.1 Warm-Start

Evaluating the performance of a content-based model in the warm-start stage is important for two reasons. First, the warm-start stage is a crucial stage to a recommender. Second, a content-based model performing well in the warm-start stage would serve as a better building block for a good hybrid model, whose performance in the warm-start stage is determined by both the collaborative filtering part and the content part.

In the warm-start stage, all songs in the validation/test sets are in the training set. Therefore, whether the content-based model generalizes to new songs or not is not of our focus.

To determine the structure of DBN, we tried different number of layers as well as different number of nodes for each layer on the validation set. For HLDBN, we finally used DBN with four layers (the input layer included),

each containing 500 nodes because this setting performs the best on the validation set. Increasing the number of nodes of each layer does not produce better results. Unsupervised pre-training was conducted for 200 iterations. The mini-batch size for both pre-training and finetune is 5000. The learning rate for the Gaussian-Bernoulli RBM is 5×10^{-5} and binary RBM 10^{-2} . The finetune learning rate of the supervised training stage is 0.5, and the regularization parameter $\lambda = 0.1$. The finetune process was stopped when the model's predictive performance on the warm-start validation set started to drop.

For CB1 and CB2, the number of nodes of their output layer is determined by the dimensionality of the PMF's latent features, i.e. 100 (see Section 4.3.4). Other layers use the same configuration as HLDBN.

The results of HLDBN, CB1, and CB2 are shown in Table 4.3. Although HLDBN only slightly outperforms CB2, we should notice that while CB2 is trained based on the results of PMF, HLDBN is a unified model and does not rely on PMF, which makes it easier to train and more principled.

The results also show that CB1 has RMSE larger than 0.5, which is worse than the trivial mean predictor. In fact, the RMSE of CB1 on the training set is also as large as 0.7, and increasing the size of DBN does not lead to improvement. These observations support our assertion in Section 4.2.2.2 that CB1 used the incorrect objective.

4.3.5.2 Cold-Start

The major practical advantage of content-based methods over CF is that content-based methods work even in the new-song scenario. Thus an effective content-based model should generalize well to new songs.

Most experimental settings are the same as those in the warm-start evaluation except the configuration of DBN. Even with pre-training, large DBN

	WS Valid	WS Test	CS Valid	CS Test
PMF	0.270	0.270	-	-
HLDBN	0.323	0.323	0.477	0.478
CB1	0.679	0.679	0.688	0.669
CB2	0.325	0.325	0.495	0.495
Mean	0.500	0.500	0.500	0.500

Table 4.3: Predictive performance of CF and content-based methods using DBN (Root Mean Squared Error). WS and CS stand for warm-start and cold-start, respectively.

	WS Valid	WS Test
Hybrid w/ HLDBN	0.255	0.255
Hybrid w/ CB2	0.270	0.270

Table 4.4: Predictive performance of our hybrid method with the features learnt by our HLDBN model and the baseline CB2 model (Root Mean Squared Error)

is prone to overfitting. We tried many configurations and decided on using four layers, each of which contains 300 nodes. Increasing the number of input nodes makes the model overfit.

The results are shown in Table 4.3. We can see that HLDBN outperforms CB1 and CB2 significantly. CB1 has very poor results due to its incorrect objective function. CB2 performs only slightly better than the mean predictor. We tried to reduce the size of its DBN and also applied the deep convolutional neural network directly on the spectrogram following the same settings as [van den Oord *et al.*, 2013], but there were no significant changes, which suggests that CB2 does not generalize well. This could be due to the lack of proper regularization. Therefore, among the three models, only HLDBN generalizes to new songs.

4.3.6 Hybrid Music Recommendation

The focus of our hybrid method is to boost the recommendation performance in the warm-start stage instead of solving the new-song problem, for which we can simply fall back to HLDBN. It is also possible to build a model to handle both scenarios seamlessly, and we leave it as future work.

As CB1 performs worse than the trivial mean predictor, it makes little sense to train a hybrid model based on its learnt features. We thus only consider the hybrid methods based on the features learnt by HLDBN and CB2. Table 4.4 shows that the hybrid approach based on CB2’s features does not bring any improvement over PMF’s results shown in Table 4.3. This is because the content features learnt by CB2 are highly correlated with the latent features from PMF and do not provide much new information. On the other hand, HLDBN does not rely on PMF, and its learnt features have incorporated audio content information that PMF fails to take into account. The results show that HLDBN performs significantly better than PMF.

To show that the learnt features are better than traditional features in music recommendation, the aspect model (AM, introduced in Section 2.2.4), one of the two existing model-based hybrid music recommenders (Section 2.2.4), was chosen as the baseline. Because AM ranks songs instead of predicting ratings, we switched our evaluation metric from RMSE to truncated mAP, for which the top-500 recommended songs were considered.

For AM, we first extracted a rich set of traditional features. Marsyas [Tzanetakis and Cook, 1999] was used on the 30-second WAV files described in Section 4.3.1. A window size of 512 was used without overlapping. Descriptions about the features are shown in Table 4.6. Because AM cannot handle continuous features directly, we built a codebook using k-means based on 8 million

	WS Valid	WS Test
PMF	0.0109	0.0110
Hybrid w/ HLDBN	0.0132	0.0131
AM w/ traditional features	0.0108	0.0108
AM w/ features from HLDBN	0.0123	0.0120

Table 4.5: Comparison between hybrid methods using features learnt from HLDBN and traditional features (mean Average Precision)

feature vectors from 10,000 randomly sampled songs. K-means was used instead of GMM in [Yoshii *et al.*, 2006] mainly for better efficiency. Hadoop⁶ was used to handle the large amount of data and computation. Finally, feature vectors of each song were quantized as codewords, which were further aggregated into a vector with each element representing occurrences of the corresponding codeword. Other parts remain the same as [Yoshii *et al.*, 2006]. To use HLDBN’s learnt features in AM, we also quantized the learnt features and aggregated each song’s codewords into one vector.

The results of our hybrid method and AM are shown in Table 4.5. We can see that AM with traditional features performs slightly worse than PMF. However, AM performs significantly better with the features learnt by HLDBN. This suggests that the automatically learnt features are more effective than the traditional features in hybrid music recommendation.

In addition, our hybrid method performs significantly better than AM. The possible reasons could be: (1) our model has regularization terms but AM does not; (2) our method can directly use the features, but AM has to quantize feature vectors, which results in information loss.

⁶<http://hadoop.apache.org/>

Feature name	Description
MFCC	Mel-Frequency Cepstral Coefficients. It models the auditory perception system and is widely used in speech and music domain.
Zero Crossings	The rate of sign-changes along a signal.
Spectral Centroid	The “center of mass” of the spectrum. Measures the brightness of the sound.
Spectral Flux	The squared change in normalized amplitude between two consecutive time frames. It measures how much the sound changes between frames.
Spectral Rolloff	Measures the amount of the right-skewness of the power spectrum.
Spectral Flatness Measure	Measures how much the audio signal sounds like a tone instead of noise.
Spectral Crest Factor	Another measure of noisiness. Similar to Spectral Flatness Measure.
Chroma	Pitch based feature. It projects the spectrum into 12 bins, representing the 12 distinct pitches of the chromatic musical scale.
Tempo	Beats per minute

Table 4.6: Traditional audio features used

4.4 Conclusion

In this chapter, we have described a novel model for content-based music recommendation based on deep belief network and probabilistic graphical model. Instead of splitting feature extraction and recommendation into separate steps, our model unifies them in an automated process. Compared with existing deep learning based models, our model outperforms them in both the warm-start and cold-start stages without relying on collaborative filtering. Based on the automatically learnt features, we created a hybrid collaborative filtering and content-based recommendation model, which not only significantly improves the performance of CF but also outperforms the traditional feature based hybrid method.

Chapter 5

Interactive Music Recommendation

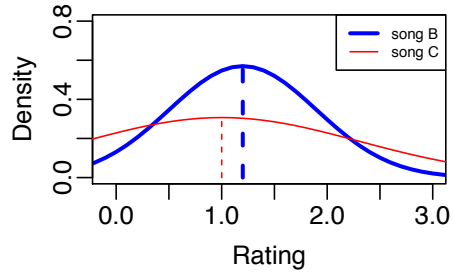
5.1 Introduction

A music recommendation system recommends songs from a large database by matching songs with a user's preferences. An *interactive* recommender system adapts to the user's preferences online by incorporating user feedback into recommendations. Each recommendation thus serves two objectives: (i) satisfy the user's current musical need, and (ii) elicit user feedback in order to improve future recommendations.

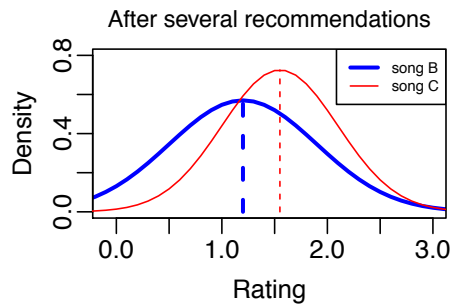
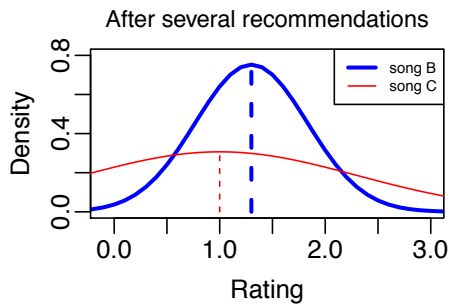
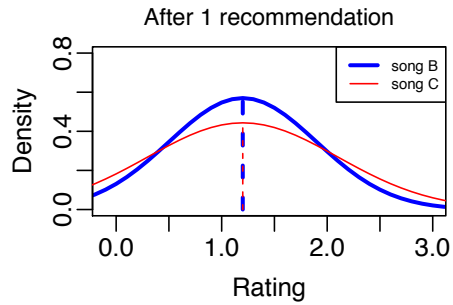
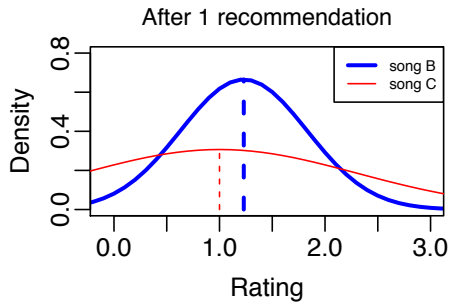
Current recommender systems typically focus on the first objective, while completely ignoring the other. They recommend songs with the highest user ratings. Such a greedy strategy, which does not actively seek user feedback, often results in suboptimal recommendations over the long term. Consider the simple example in Figure 5.1. The table contains the ratings for three songs by four users (Figure 5.1a), with 3 being the highest and 1 being the lowest.

		user			
		1	2	3	4
song	A	1, 2	1	1	1
	B	2	1	1, 1	
	C		1		

(a) Rating table. A song could be rated multiple times.



(b) Predicted rating distributions



(c) Refining the predictions using the greedy strategy (pure exploitation)

(d) Refining the predictions using multi-armed bandit (exploration/exploitation tradeoff)

Figure 5.1: Uncertainty in recommendation

For simplicity, let us assume that the recommender chooses between two songs B and C only. The target user is 4, whose true ratings for B and C are 1.3 and 1.6, respectively. The true rating is the expected rating of a song by the user. It is a real number, because a user may give the same song different ratings as a result of external factors. In this case, a good recommender should choose C . Since the true user ratings are unknown to the recommender, it may approximate the rating distributions for B and C as Gaussians, P_B and P_C (Figure 5.1b), respectively, using the data in Figure 5.1a. The distribution P_B has mean 1.2. The distribution P_C has mean 1. P_B has much lower variance than P_C , because B has more rating data. A greedy recommender (including the highly successful collaborative filtering (CF) approach) would recommend B , the song with the highest mean rating. In response to this recommendation, user 4 gives a rating, whose expected value is 1.3. The net effect is that the mean of P_B likely shifts towards 1.3 and its variance further reduces (Figure 5.1c). Consequently the greedy recommender is even more convinced that user 4 favors B and will always choose B for all future recommendations. It will never choose C and find out its true rating, resulting in clearly suboptimal performance.

To overcome this difficulty, the recommender must take into account *uncertainty* in the mean ratings. If it considers both the mean and the variance of the rating distribution, the recommendation will change. Consider again Figure 5.1b. Although P_C has slightly lower mean than P_B , it has very high variance. It may be worthwhile to recommend C and gather additional user feedback in order to reduce the variance. User 4's rating on C has expected value 1.6. Therefore, after one recommendation, the mean of P_C will likely shift towards 1.6 (Figure 5.1d). By recommending C several times and gathering user feedback, we will then find out user 4's true preference C .

This example illustrates that a good interactive music recommender system must *explore* user preferences actively rather than merely *exploit* the rating information available. Balancing exploration and exploitation is critical, especially when the system is faced with a *cold start*, i.e., when a new user or a new song appears.

Another crucial issue for music recommendation is playlist generation. People often listen to a group of related songs together and may repeat the same song multiple times. This is unique to music recommendation and rarely occurs in other recommendation domains such as newspaper articles or movies. A playlist is a group of songs arranged in a suitable order. The songs in a playlist have strong interdependencies. For example, they share the same genre [Chen *et al.*, 2012] or have a consistent mood [Logan, 2002], but are diversified at the same time [Zhang *et al.*, 2012]. They may repeat, but are not repetitive. Existing recommender systems based on CF or audio content analysis typically recommend one song at a time and do not consider their interdependencies during the recommendation process. They divide playlist generation into two distinct steps [Chen *et al.*, 2012]. First, choose a set of favored songs through CF or content analysis. Next, arrange the songs into a suitable order in a process called *automatic playlist generation* (APG).

In this work, we formulate interactive, personalized music recommendation as a reinforcement learning task called the *multi-armed bandit* [Sutton and Barto, 1998] and address both exploration-exploitation trade-off and playlist generation with a single unified model:

- Our bandit approach systematically balances exploration and exploitation, a central issue well studied in reinforcement learning. Experimental results show that our recommender system mitigates the difficulty of

cold start and improves recommendation performance compared to the greedy approach.

- We build a single rating model that captures both user preference over audio content and the novelty of recommendations. It seamlessly integrates music recommendation and playlist generation.
- We also present an approximation to the rating model and new probabilistic inference algorithms in order to achieve real-time recommendation performance.
- Although our approach is designed specifically for music recommendation, it is possible to be generalized to other media types as well.

In the following, Section 5.2 formulates the rating model and our multi-armed bandit approach to music recommendation. Section 5.3 presents the approximate Bayesian models and inference algorithms. Section 5.4 presents evaluation of our models and algorithms. Section 5.5 discusses the possible generalization directions of the approach to other media types. Section 5.6 summarizes the main results and provides directions for future research.

5.2 A Bandit Approach to Music Recommendation

5.2.1 Personalized User Rating Model

Music preference is the combined effect of many factors including music audio content, novelty, diversity, moods and genres of songs, user emotional states, and user context information [Wang *et al.*, 2012a]. As it is unrealistic to cover

Table 5.1: Table of symbols

Symbol	Explanation
\mathcal{D}	Training data set. The first l training samples: $\mathcal{D}_l = \{(\mathbf{x}_i, t_i, r_i)\}_{i=1}^l$
\mathcal{S}	Song set
U_c, U_n	User preference of music content and novelty, respectively
U_k	Mean rating for song k
Ω	Parameter set
θ	User preference of different music features
β^t	Piecewise linear approximation of novelty $U_n(t)$
\mathbf{x}	Music audio feature vector
s	Novelty recovery speed
t	Time elapsed since the last listening of the song
$\sigma^2, 1/\tau$	Variance of the residuals in user ratings
$\mu_{\theta 0}, \sigma^2 \mathbf{D}_0$	Mean and covariance matrix for the prior of θ in the approximate model, respectively
$\mu_{\beta 0}, \sigma^2 \mathbf{E}_0$	Mean and covariance matrix for the prior of β in the approximate model, respectively
$\Lambda_{\theta N}, \Lambda_{\beta N}$	Precision matrices for the posterior distributions of θ and β in the approximate model, respectively
c_0, d_0, e_0, f_0, g_0	Parameters of the prior distributions for the exact Bayesian model

all the factors in this thesis due to time constraints, we focus on audio content and novelty.

Music Audio Content - Whether a user likes or dislikes a song is highly related to the audio content of the song. We assume that the music audio content of a song can be described as a feature vector \mathbf{x}^1 . Without considering other factors, a user's overall preference in this song can be represented as a

¹Please refer to Table 5.1 for a summary of the notations used in this chapter.

linear function of \mathbf{x} as:

$$U_c = \boldsymbol{\theta}'\mathbf{x} \tag{5.1}$$

where the parameter vector $\boldsymbol{\theta}$ represents user preference in different music features. Different users may have different preference and thus different values of $\boldsymbol{\theta}$. To keep the problem simple, we assume that a user's preference is invariant, i.e. $\boldsymbol{\theta}$ remains constant over time, and we will address the case of changing $\boldsymbol{\theta}$ in future work.

Although the idea of exploration/exploitation tradeoff can be applied to collaborative filtering (CF) as long as the rating distribution can be estimated as shown in Figure 5.1, we choose to work on the content-based approach instead of CF for a number of reasons. First, we need a posterior distribution of U_c in order to use Bayes-UCB as introduced in Section 2.4.1, so non-Bayesian CF methods cannot be used. Second, existing Bayesian methods for matrix factorization [Salakhutdinov and Mnih, 2008a; Silva and Carin, 2012] are much more complicated than our linear model and also require large amounts of training data. These render the user study costly and cumbersome. Third, our bandit approach requires the model to be updated once a new rating is obtained, but existing Bayesian matrix factorization methods are inefficient for online updating [Salakhutdinov and Mnih, 2008a; Silva and Carin, 2012]. Fourth, CF suffers from the new-song problem while the content-based method does not. Fifth, CF captures correlation instead of causality and thus does not explain why a user likes a song. In contrast, the content-based approach captures one important aspect of the causality, i.e. music content.

Novelty - Inspired by [Gunawardana and Shani, 2009], we seek to measure novelty by first examining the repetition distributions of 1000 users' listening

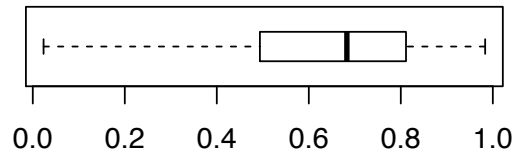


Figure 5.2: Proportion of repetitions in users' listen history. This boxplot shows a five-number summary: minimum, first quartile, median, third quartile, and maximum.

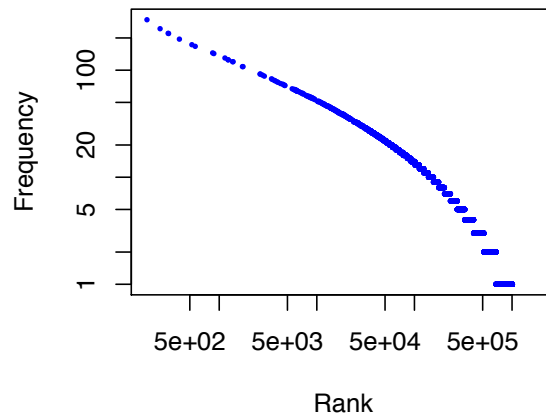


Figure 5.3: Zipf's law of song repetition frequency

histories collected from Last.fm². The box plot in Figure 5.2 shows the proportion of repetitions, which is defined as: $1 - \frac{\text{number of of unique songs}}{\text{listening history length}}$. Note that since Last.fm does not record the user's listening histories outside Last.fm, the actual proportion is expected to be even larger than the median 68.3% shown here. Thus, most of the songs a user listens to are repeats. We also studied the song repetition frequency distribution of every individual user's listening history. The frequencies of songs were first computed for every user. Then, all users' frequencies were ranked in decreasing order. Finally, we plotted frequencies versus ranks on a log-log scale (Figure 5.3). The distribution approximately follows the Zipf's law [Newman, 2005]—only a small set

²<http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html>

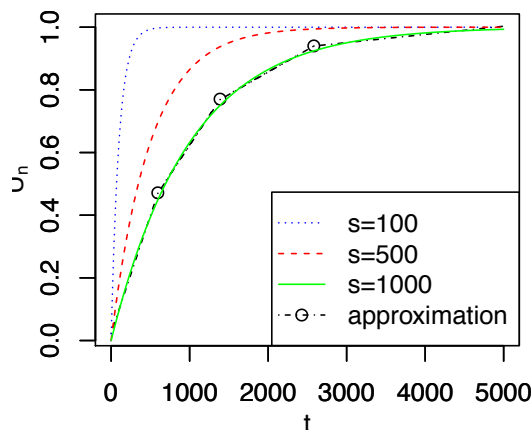


Figure 5.4: Examples of $U_n = 1 - e^{-t/s}$. The line marked with circles is a 4-segment piecewise linear approximation.

of songs are repeated most of the time while all the rest are repeated much less often. Most other types of recommenders do not follow Zipf’s law. For instance, recommending books that have been bought or movies that have been watched makes little sense. In music recommendation, however, it is critically important to repeat songs appropriately.

One problem with existing novelty models is that they do not take the time elapsed since previous listening into consideration [Gunawardana and Shani, 2009; Lathia *et al.*, 2010; Castells *et al.*, 2011; Zhang *et al.*, 2012]. As a result, songs listened to a year ago and just now have the same likelihood to be recommended. Inspired by [Hu and Ogihara, 2011], we address this issue by assuming that the novelty of a song decays immediately after it is listened to it and then gradually recovers. Let t be the time elapsed since the last listening of the song, the novelty recovers according to the function:

$$U_n = 1 - e^{-t/s} \quad (5.2)$$

where s is a parameter indicating the recovery speed. The higher the value of

s , the slower the recovery of novelty. Figure 5.4 shows a few examples of U_n with different values of s . Note that the second term of Equation (5.2), $e^{-t/s}$, is the well-established forgetting curve proposed by [Ebbinghaus *et al.*, 1913], which measures the user’s memory retention of a song. Novel songs are thus assumed to be those of which a user has little or no memory.

Different users can have different recovery rates s . As can be seen from the widespread distribution in Figure 5.2, some users may repeatedly listen to their favorite songs, while the others may be keen to exploring songs they have not listened to previously. Therefore, s is an unknown user parameter to be learnt from user interactions.

Combined Model - A user’s preference of a recommendation can be represented as a rating; the higher the rating, the more the user likes the recommendation. Unlike traditional recommenders, which assume a user’s ratings are static, we assume that a rating is the combined effect of the user’s preference of the song’s content and the dynamically changing novelty. Thus, a song rated as 5 last time could be rated as 2 later because the novelty has decreased. Finally, we define the combined rating model as:

$$U = U_c U_n = \boldsymbol{\theta}' \mathbf{x} (1 - e^{-t/s}). \quad (5.3)$$

In this model, the more the user likes a particular song, the more likely it will be repeated due to the larger U_c value. Also, given that the user’s favorites comprise a small subset of his/her library, the U model behaves in accordance with Zipf’s Law and ensures that only a small proportion of songs will be repeated frequently. This property of the model will be verified in Section 5.4.3.2.

In Section 5.4.3.1, we will show that the product form of Equation (5.3)

leads to significantly better performance than the alternative linear combination $U = aU_c + bU_n$.

Other factors - We note that, besides novelty, the repetition of songs can also be affected in other ways. When a user comes to a song of great excitement, he may listen to it again and again. When his interest changes, he may discard songs that he has been frequently repeating. Sometimes, the user finds a song boring initially but repeats it frequently later, while in other cases he may stop repeating a song because he is bored. Understanding and modeling all these factors and precisely predicting when to repeat a song for a particular user would make very interesting follow-up studies.

5.2.2 Interactive Music Recommendation

Under our rating model, each user is represented by a set of parameters $\Omega = \{\theta, s\}$. If we knew the values of Ω of a user, we could simply recommend the songs with the highest rating according to Equation (5.3). However, Ω is hidden and needs to be estimated from historical data, and thus uncertainty always exists. In this case, the greedy strategy used by traditional systems is suboptimal, and it is necessary to take the uncertainty into account and balance exploration and exploitation.

The multi-armed bandit approach introduced in Section 2.4.1 offers a way to do so for the interactive music recommendation process. As illustrated in Figure 5.5, we treat songs as arms and user ratings as payoffs³. The music recommendation problem is then transformed into a multi-armed bandit problem, and the objective of the music recommender is also changed to maximizing the

³Although in reality users usually do not give explicit feedback (i.e. ratings) to every recommended song, implicit feedback (e.g. skipping a song, listening to a song fully) can be obtained much more easily. In this work, we focus on explicit feedback to keep the problem simple.

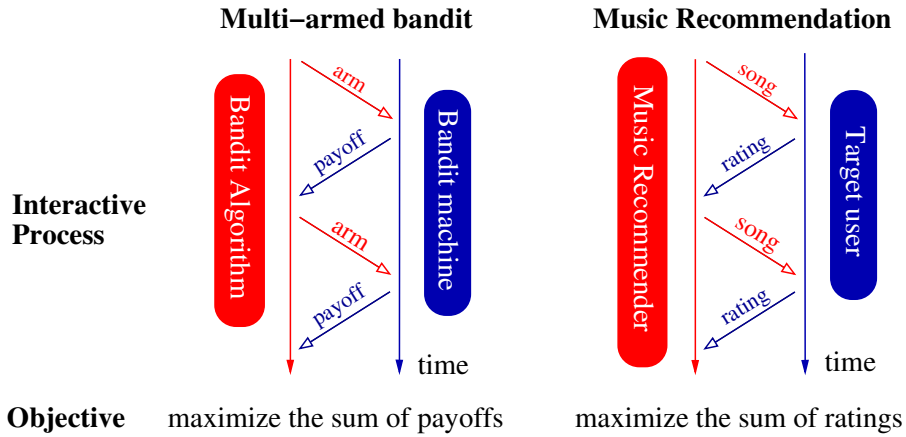


Figure 5.5: Relationship between the multi-armed bandit problem and music recommendation

sum of the ratings given by the target user *over the long term*. We argue that the cumulative rating is a more realistic objective than the *myopic* predictive accuracy used by traditional music recommenders, because users usually listen to songs for a long time instead of focusing on one individual song.

We adopt the Bayes-UCB algorithm introduced in Section 2.4.1 for our recommendation task. We denote the rating given by the target user to a recommendation i as a random variable R_i , and the expectation of R_i is U given the feature vector (\mathbf{x}_i, t_i) :

$$\mathbb{E}[R_i] = U_i = \boldsymbol{\theta}'\mathbf{x}_i (1 - e^{-t_i/s}) \quad (5.4)$$

Then, we develop Bayesian models to estimate the posterior distribution of U given the history recommendation and user ratings. We sketch the framework here and explain it in greater detail in Section 5.3. We assume that the prior distribution of $\boldsymbol{\Omega}$ is $p(\boldsymbol{\Omega})$ and that, at the $(l + 1)$ th recommendation, we have accumulated l history recommendations $\mathcal{D}_l = \{(\mathbf{x}_i, t_i, r_i)\}_{i=1}^l$ as training

ALGORITHM 1: Recommendation using Bayes-UCB

```

for  $l = 1$  to  $n$  do
  for all song  $k = 1, \dots, |\mathcal{S}|$  do
    compute  $q_k^l = Q(1 - 1/(l + 1), \lambda_k^{l-1})$ 
  end for
  recommend song  $k^* = \arg \max_{k=1 \dots |\mathcal{S}|} q_k^l$  and gather rating  $r_l$ ; update
   $p(\boldsymbol{\Omega}|\mathcal{D}_l)$  and  $\lambda_k^l$ 
end for

```

samples, where r_i is the rating given by the user to the i -th recommendation. The posterior distribution of $\boldsymbol{\Omega}$ can then be obtained based on the Bayes' rule:

$$p(\boldsymbol{\Omega}|\mathcal{D}_l) \propto p(\boldsymbol{\Omega})p(\mathcal{D}_l|\boldsymbol{\Omega}) \quad (5.5)$$

Consequently, the expected rating of song k , denoted as U_k can be predicted as:

$$p(U_k|\mathcal{D}_l) = \int p(U_k|\boldsymbol{\Omega})p(\boldsymbol{\Omega}|\mathcal{D}_l)d\boldsymbol{\Omega} \quad (5.6)$$

Henceforth, we will use λ_k^l to denote $p(U_k|\mathcal{D}_l)$ for simplicity.

Finally, to balance exploration and exploitation, Bayes-UCB recommends song k^* , which maximizes the quantile function: $k^* = \arg \max_{k=1 \dots |\mathcal{S}|} Q(\alpha, \lambda_k^l)$ where Q satisfies $\mathbb{P}[U_k \leq Q(\alpha, \lambda_k^l)] = \alpha$ and \mathcal{S} is all the songs in the database. We set $\alpha = 1 - \frac{1}{l+1}$. The detailed recommendation algorithm is described in Algorithm 1.

The cold-start problem is caused by the lack of information required for making good recommendations. There are many ways for mitigating the cold-start problem, most of which rely on additional information about the users or songs, e.g., popularity/metadata information about the songs [Hariri *et al.*, 2012], context/demographic information about the users [Wang *et al.*, 2012a].

Although music audio content is required by U_c , it is usually easy to obtain from industry. Our bandit approach addresses the cold-start problem without relying on additional information about users and songs. Instead, it seeks to appropriately explore and exploit information during the whole interactive process. Thus, the bandit approach presents a fundamentally different solution to the cold-start problem, yet it can be used in conjunction with the existing methods.

There are other Bayesian multi-arm bandit algorithms such as Thompson sampling [Agrawal and Goyal, 2012] and optimistic Bayesian sampling [May et al., 2012]. Thompson sampling is based on the probability matching idea, i.e. selecting the song according to its probability of being optimal. Optimistic Bayesian sampling uses an exploitative function on top of Thompson sampling. Which of the three is superior? Theoretically, this remains an open question. However, they have been shown to perform comparably well in practice [May et al., 2012]. Existing studies provide little guidance on our selection between them. In this work, we focus on the exploration/exploitation tradeoff principle and simply choose the most recent Bayes-UCB in our implementation. Nevertheless, our system could easily adapt to the other two algorithms, as they are also based on the posterior rating distributions.

5.3 Bayesian Models and Inference

5.3.1 Exact Bayesian model

To compute Equations (5.5) and (5.6), we develop the following Bayesian model (see Figure 5.6a for graphical representation).

$$\begin{aligned}R|\mathbf{x}, t, \boldsymbol{\theta}, s, \sigma^2 &\sim \mathcal{N}(\boldsymbol{\theta}'\mathbf{x}(1 - e^{-t/s}), \sigma^2) \\ \boldsymbol{\theta}|\sigma^2 &\sim \mathcal{N}(\mathbf{0}, c_0\sigma^2\mathbf{I}) \\ s &\sim \mathcal{G}(d_0, e_0) \\ \tau = 1/\sigma^2 &\sim \mathcal{G}(f_0, g_0)\end{aligned}$$

Every part of the model defines a probabilistic dependency between the random variables. $\mathcal{N}(\cdot, \cdot)$ is a (multivariate) Gaussian distribution with the mean and (co)variance parameters, and $\mathcal{G}(\cdot, \cdot)$ is a Gamma distribution with the shape and rate parameters. The rating R is assumed to be normally distributed following the convention of recommender systems. A gamma prior is put on s because s takes positive values. Following the conventions of Bayesian regression models, a normal prior is put on $\boldsymbol{\theta}$ and a Gamma prior on τ . We assume that $\boldsymbol{\theta}$ depends on σ^2 because it leads to better convergence in the simulation study.

Since there is no closed-form solution to Equation (5.5) under this model, Markov Chain Monte Carlo (MCMC) is used as the approximate inference procedure. Directly evaluating Equation (5.6) is also infeasible. Thus we use Monte Carlo simulation to obtain λ_k^l : for every sample obtained from the MCMC procedure, we substitute it into Equation (5.4) to obtain a sample of U_i , and then use the histogram of the samples of U_i to approximate λ_k^l .

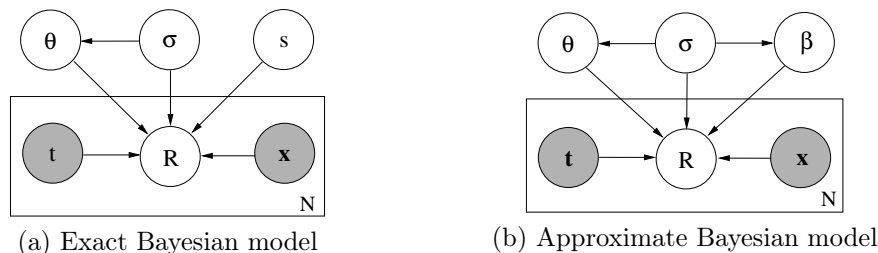


Figure 5.6: Graphical representation of the Bayesian models. Shaded nodes represent observable random variables, while white nodes represent hidden ones. The rectangle (plate) indicates that the nodes and arcs inside are replicated for N times.

This approach is easy to understand and implement. However, it is very slow, and users could wait for up to a minute until the Markov chain converges. To make the algorithm more responsive, we develop an approximate Bayesian model and a highly efficient variational inference algorithm in the following sections.

5.3.2 Approximate Bayesian Model

5.3.2.1 Piecewise Linear Approximation

It is very difficult to develop better inference algorithms for the exact Bayesian model because of the irregular form of the function $U_n(t)$. Fortunately, U_n can be well approximated by a piecewise linear function (as shown in Figure 5.4), which enables us to develop an efficient model.

For simplicity, we discretize time t into K intervals: $[0, \xi_1), [\xi_1, \xi_2), \dots, [\xi_{K-1}, +\infty)$, and only consider the class of piecewise linear functions whose consecutive line segments intersect at the boundaries of these intervals. This class of functions can be compactly represented as a linear function [Hastie *et al.*, 2009]. We first map t into a vector $\mathbf{t} = [(t - \xi_1)_+, \dots, (t - \xi_{K-1})_+, t, 1]$, where $(t - \xi)_+ = \max(t - \xi, 0)$, and then approximate $U_n(t)$ as $U_n(t) \approx \beta' \mathbf{t}$, where

$\boldsymbol{\beta} = [\beta_1, \dots, \beta_{K+1}]'$ is a vector of parameters to be learnt from training data. Now, we can represent U as the product of two linear functions: $U = U_c U_n \approx \boldsymbol{\theta}' \mathbf{x} \boldsymbol{\beta}' \mathbf{t}$.

Based on this approximation, we approximate the distributions of R and the parameters of the exact Bayesian model as follows:

$$\begin{aligned} R | \mathbf{x}, \mathbf{t}, \boldsymbol{\theta}, \boldsymbol{\beta}, \sigma^2 &\sim \mathcal{N}(\boldsymbol{\theta}' \mathbf{x} \boldsymbol{\beta}' \mathbf{t}, \sigma^2), \\ \boldsymbol{\theta} | \sigma^2 &\sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}0}, \sigma^2 \mathbf{D}_0), \\ \boldsymbol{\beta} | \sigma^2 &\sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\beta}0}, \sigma^2 \mathbf{E}_0), \\ \tau &= 1/\sigma^2 \sim \mathcal{G}(a_0, b_0) \end{aligned} \tag{5.7}$$

where $\boldsymbol{\theta}, \boldsymbol{\beta}, \tau$ are parameters. $\mathbf{D}_0, \mathbf{E}_0, \boldsymbol{\mu}_{\boldsymbol{\theta}0}, \boldsymbol{\mu}_{\boldsymbol{\beta}0}, a_0, b_0$ are the hyperparameters of the priors to be specified beforehand; \mathbf{D}_0 and \mathbf{E}_0 are positive definite matrices. The graphical representation of the model is shown in Figure 5.6b. We use conjugate priors for $\boldsymbol{\theta}, \boldsymbol{\beta}, \tau$, which make the variational inference algorithm described later very efficient.

5.3.2.2 Variational Inference

Recall that our objective is to compute the posterior distribution of parameters $\boldsymbol{\Omega}$ (now it is $\{\boldsymbol{\theta}, \boldsymbol{\beta}, \tau\}$) given the history data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{t}_i, r_i)\}_{i=1}^N$, i.e., $p(\boldsymbol{\theta}, \boldsymbol{\beta}, \tau | \mathcal{D})$. Using piecewise linear approximation, we now develop an efficient variational inference algorithm.

Following the convention of the mean-field approximation [Friedman and Koller, 2009], we assume that the joint posterior distribution can be approximated by a restricted distribution $q(\boldsymbol{\theta}, \boldsymbol{\beta}, \tau)$, which consists of three indepen-

dent factors [Friedman and Koller, 2009]:

$$p(\boldsymbol{\Omega}|\mathcal{D}) = p(\boldsymbol{\theta}, \boldsymbol{\beta}, \tau|\mathcal{D}) \approx q(\boldsymbol{\theta}, \boldsymbol{\beta}, \tau) = q(\boldsymbol{\theta})q(\boldsymbol{\beta})q(\tau).$$

Because of the choice of the conjugate priors, it is easy to show that the restricted distributions $q(\boldsymbol{\theta})$, $q(\boldsymbol{\beta})$, and $q(\tau)$ take the same parametric forms as the prior distributions. Specifically,

$$\begin{aligned} q(\boldsymbol{\theta}) &\propto \exp\left(-\frac{1}{2}\boldsymbol{\theta}'\boldsymbol{\Lambda}_{\boldsymbol{\theta}N}\boldsymbol{\theta} + \boldsymbol{\eta}'_{\boldsymbol{\theta}N}\boldsymbol{\theta}\right) \\ q(\boldsymbol{\beta}) &\propto \exp\left(-\frac{1}{2}\boldsymbol{\beta}'\boldsymbol{\Lambda}_{\boldsymbol{\beta}N}\boldsymbol{\beta} + \boldsymbol{\eta}'_{\boldsymbol{\beta}N}\boldsymbol{\beta}\right) \\ q(\tau) &\propto \tau^{a_N-1} \exp(-b_N\tau). \end{aligned}$$

To find the values that minimize the KL-divergence between $q(\boldsymbol{\theta}, \boldsymbol{\beta}, \tau)$ and the true posterior $p(\boldsymbol{\theta}, \boldsymbol{\beta}, \tau|\mathcal{D})$ for the parameters $\boldsymbol{\Lambda}_{\boldsymbol{\theta}N}$, $\boldsymbol{\eta}_{\boldsymbol{\theta}N}$, $\boldsymbol{\Lambda}_{\boldsymbol{\beta}N}$, $\boldsymbol{\eta}_{\boldsymbol{\beta}N}$, a_N , and b_N , we use the coordinate descent method. Specifically, we first initialize the parameters of $q(\boldsymbol{\theta})$, $q(\boldsymbol{\beta})$, and $q(\tau)$, and then iteratively update $q(\boldsymbol{\theta})$, $q(\boldsymbol{\beta})$, and $q(\tau)$ until the variational lower bound \mathcal{L} (elaborated in the Appendix A.3) converges. Further explanation about the principle can be found in [Friedman and Koller, 2009]. The detailed steps are described in Algorithm (2), where p and K are the dimensionalities of \mathbf{x} and \mathbf{t} , respectively; the moments of $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, τ are derived in the Appendix A.2.

ALGORITHM 2: Variational inference**input:** $\mathcal{D}, \mathbf{D}_0, \mathbf{E}_0, \boldsymbol{\mu}_{\theta 0}, \boldsymbol{\mu}_{\beta 0}, a_0, b_0$ initialize $\boldsymbol{\Lambda}_{\theta N}, \boldsymbol{\eta}_{\theta N}, \boldsymbol{\Lambda}_{\beta N}, \boldsymbol{\eta}_{\beta N}, a_N, b_N$ **repeat** update $q(\boldsymbol{\theta})$:

$$\boldsymbol{\Lambda}_{\theta N} \leftarrow \mathbb{E}[\tau] \left(\mathbf{D}_0^{-1} + \sum_{i=1}^N \mathbf{x}_i \mathbf{t}_i' \mathbb{E}[\boldsymbol{\beta} \boldsymbol{\beta}'] \mathbf{t}_i \mathbf{x}_i' \right),$$

$$\boldsymbol{\eta}_{\theta N} \leftarrow \mathbb{E}[\tau] \left(\mathbf{D}_0^{-1} \boldsymbol{\mu}_{\theta 0} + \sum_{i=1}^N r_i \mathbf{x}_i \mathbf{t}_i' \mathbb{E}[\boldsymbol{\beta}] \right)$$

 update $q(\boldsymbol{\beta})$:

$$\boldsymbol{\Lambda}_{\beta N} \leftarrow \mathbb{E}[\tau] \left(\mathbf{E}_0^{-1} + \sum_{i=1}^N \mathbf{t}_i \mathbf{x}_i' \mathbb{E}[\boldsymbol{\theta} \boldsymbol{\theta}'] \mathbf{x}_i \mathbf{t}_i' \right)$$

$$\boldsymbol{\eta}_{\beta N} \leftarrow \mathbb{E}[\tau] \left(\mathbf{E}_0^{-1} \boldsymbol{\mu}_{\beta 0} + \sum_{i=1}^N r_i \mathbf{t}_i \mathbf{x}_i' \mathbb{E}[\boldsymbol{\theta}] \right)$$

 update $q(\tau)$:

$$a_N \leftarrow \frac{p+K+N}{2} + a_0$$

$$\begin{aligned} b_N &\leftarrow \frac{1}{2} [\text{tr} [\mathbf{D}_0^{-1} (\mathbb{E}[\boldsymbol{\theta} \boldsymbol{\theta}'])] + (\boldsymbol{\mu}'_{\theta 0} - 2\mathbb{E}[\boldsymbol{\theta}']) \mathbf{D}_0^{-1} \boldsymbol{\mu}_{\theta 0}] \\ &\quad + \frac{1}{2} [\text{tr} [\mathbf{E}_0^{-1} (\mathbb{E}[\boldsymbol{\beta} \boldsymbol{\beta}'])] + (\boldsymbol{\mu}'_{\beta 0} - 2\mathbb{E}[\boldsymbol{\beta}']) \mathbf{E}_0^{-1} \boldsymbol{\mu}_{\beta 0}] \\ &\quad + \frac{1}{2} \sum_{i=1}^N (r_i^2 + \mathbf{x}_i' \mathbb{E}[\boldsymbol{\theta} \boldsymbol{\theta}'] \mathbf{x}_i \mathbf{t}_i' \mathbb{E}[\boldsymbol{\beta} \boldsymbol{\beta}'] \mathbf{t}_i) - \sum_{i=1}^N r_i \mathbf{x}_i' \mathbb{E}[\boldsymbol{\theta}] \mathbf{t}_i' \mathbb{E}[\boldsymbol{\beta}] + b_0 \end{aligned}$$

until \mathcal{L} converges**return** $\boldsymbol{\Lambda}_{\theta N}, \boldsymbol{\eta}_{\theta N}, \boldsymbol{\Lambda}_{\beta N}, \boldsymbol{\eta}_{\beta N}, a_N, b_N$ **5.3.2.3 Predict the Posterior Distribution** $p(U|\mathcal{D})$

Because $q(\boldsymbol{\theta})$ and $q(\boldsymbol{\beta})$ are normal distributions, $\boldsymbol{\theta}'\mathbf{x}$ and $\boldsymbol{\beta}'\mathbf{t}$ are also normally distributed:

$$p(\boldsymbol{\theta}'\mathbf{x}|\mathbf{x}, \mathbf{t}, \mathcal{D}) \approx \mathcal{N}(\mathbf{x}' \boldsymbol{\Lambda}_{\theta N}^{-1} \boldsymbol{\eta}_{\theta N}, \mathbf{x}' \boldsymbol{\Lambda}_{\theta N}^{-1} \mathbf{x})$$

$$p(\boldsymbol{\beta}'\mathbf{t}|\mathbf{x}, \mathbf{t}, \mathcal{D}) \approx \mathcal{N}(\mathbf{t}' \boldsymbol{\Lambda}_{\beta N}^{-1} \boldsymbol{\eta}_{\beta N}, \mathbf{t}' \boldsymbol{\Lambda}_{\beta N}^{-1} \mathbf{t})$$

and the posterior distribution of U in Equation (5.6) can be computed as:

$$p(U|\mathbf{x}, \mathbf{t}, \mathcal{D}) = p(\boldsymbol{\theta}'\mathbf{x}\boldsymbol{\beta}'\mathbf{t}|\mathbf{x}, \mathbf{t}, \mathcal{D}) = \int p(\boldsymbol{\theta}'\mathbf{x} = a|\mathbf{x}, \mathbf{t}, \mathcal{D})p(\boldsymbol{\beta}'\mathbf{t} = \frac{U}{a}|\mathbf{x}, \mathbf{t}, \mathcal{D})da.$$

Since there is no closed-form solution to the above integration, we use Monte Carlo simulation. Namely, we first obtain one set of samples for each of $\boldsymbol{\theta}'\mathbf{x}$ and $\boldsymbol{\beta}'\mathbf{t}$ and then use the element-wise products of the two group of samples to approximate the distribution of U . Because $\boldsymbol{\theta}'\mathbf{x}$ and $\boldsymbol{\beta}'\mathbf{t}$ are normally distributed univariate random variables, the sampling can be done very efficiently. Moreover, the prediction for different songs is trivially parallelizable and is thus scalable.

5.3.2.4 Integration of Other Factors

Although the approximate model considers music audio content and novelty only, it is easy to incorporate other factors as long as they can be approximated by linear functions. For instance, diversity is another important factor for a playlist. We could measure the diversity that a song contributes to a playlist as d and assume user preference of d follows a function that can be approximated by a piecewise linear function. Following the method in Section 5.3.2.1, we can map d into a vector \mathbf{d} and modify the approximate Bayesian model in Section (5.3.2.1) by adding an additional term $\boldsymbol{\gamma}'\mathbf{d}$ to Equation (5.7) and putting a prior on $\boldsymbol{\gamma}$. As shown in the following,

$$R|\mathbf{x}, \mathbf{t}, \mathbf{d}, \sigma^2, \boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\gamma} \sim \mathcal{N}(\boldsymbol{\theta}'\mathbf{x}\boldsymbol{\beta}'\mathbf{t}\boldsymbol{\gamma}'\mathbf{d}, \sigma^2)$$
$$\boldsymbol{\gamma}|\sigma^2 \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\gamma}0}, \sigma^2\mathbf{F}_0).$$

Given the symmetry between \mathbf{x} , \mathbf{t} , and \mathbf{d} , we can modify Algorithm 2 without further derivation.

Similarly, we could incorporate more factors such as coherence of mood and genre into the model. Moreover, although the model is designed for music recommendation, it can also be applied to other regression tasks as long as the regression function can be factorized into the product of a few linear functions.

5.4 Experiments

We compare the results from our evaluations of 6 recommendation algorithms in this section. Extensive experimental evaluations of both efficiency and effectiveness of the algorithms and models have been conducted, and the results show significant promise from both aspects.

5.4.1 Experiment setup

5.4.1.1 Compared Recommendation Algorithms

To study the effectiveness of the exploration/exploitation tradeoff, we introduced two baselines, Random and Greedy. The Random approach represents pure exploration and recommends songs uniformly at random. The Greedy approach represents pure exploitation and always recommends the song with the highest predicted rating. Therefore, the Greedy approach simulates the strategy used by the traditional recommenders, where the parameters $\{\boldsymbol{\theta}, s\}$ were estimated by minimizing the mean square error using the L-BFGS-B algorithm [Byrd *et al.*, 1995].

To study the effectiveness of the rating model, we introduced a baseline using LinUCB, a bandit algorithm which assumes that the expected rating

is a linear function of the feature vector [Li *et al.*, 2010]. In LinUCB, ridge regression served as the regression method, and upper confidence bound is used to balance exploration and exploitation.

Two combinations of the factors U_c and U_n were evaluated: U_c and U_cU_n . We denote them as C and CN for short, where C and N indicate content and novelty, respectively. For example, Bayes-UCB-CN contains both content and novelty. Furthermore, Bayes-UCB-CN corresponds to the exact Bayesian model with the MCMC inference algorithm (Section 5.3.1), and Bayes-UCB-CN-V the approximate model with the variational inference algorithm (Section 5.3.2).

We evaluated 6 recommendation algorithms, which were combinations of the four approaches and three factors: Random, LinUCB-C, LinUCB-CN, Bayes-UCB-CN, Bayes-UCB-CN-V, and Greedy-CN. Because LinUCB-CN cannot handle nonlinearity and thus cannot directly model U_cU_n , we combined the feature vector \mathbf{x} in U_c and the time variable t in U_n as one vector and treated the expected rating as a linear function of the combined vector. Greedy-C was not included because it was not related to our objective. As discussed in Section 5.2.2, the bandit approach can also combine with existing methods to solve the cold-start problem. We plan to study the effectiveness of such combinations in future works.

5.4.1.2 Songs and Features

Ten thousand songs from different genres were used in the experiments. Videos of the songs were crawled from YouTube and converted by ffmpeg⁴ into mono channel WAV files with a 16kHz sampling rate. For every song, a 30-second audio clip was used [Wang *et al.*, 2012a]. Feature vectors were then extracted

⁴<http://ffmpeg.org>

using a program developed based on the MARSYAS library⁵, in which a window size of 512 was used without overlapping. The features used (and their dimensionalities) are Zero Crossing Rate (1), Spectral Centroid (1), Spectral Rolloff (1), Spectral Flux (1), MFCC (13), Chroma (14), Spectral Crest Factor (24) and Spectral Flatness Measure (24). Detailed descriptions of these features are given in Table 5.2. The features have been commonly used in the music retrieval/recommendation domain [Cano *et al.*, 2005; Yoshii *et al.*, 2006; Wang *et al.*, 2012a]. To represent a 30-second clip in one feature vector, we used the mean and standard deviation of all feature vectors from the clip. Next, we added the 1-dimensional feature *tempo* to the summarized feature vectors, and the resulting feature dimensionality is $79 \times 2 + 1 = 159$. Directly using the 159-dimensional features requires a large amount of data to train the models and makes user studies very expensive and time-consuming. To reduce the dimensionality, we conducted Principal Component Analysis (PCA) with 90% of variance reserved. The final feature dimensionality is thus reduced to 91.

The performance of these features in music recommendation was checked based on a dataset that we built. We did not use existing music recommendation datasets because they lack explicit ratings, and dealing with implicit feedbacks is not our focus. Fifty-two undergraduate students with various cultural backgrounds contributed to the dataset, with each student annotating 400 songs with a 5-point Likert scale from “very bad” (1) to “very good” (5). We computed the 10-fold cross-validation RMSE of U_c for each user and averaged the accuracy over all users. The resulting RMSE is 1.10, significantly lower than the RMSE (1.61) of the random baseline with the same distribution as the data. Therefore, these audio features indeed provide useful information

⁵<http://marsyas.sourceforge.net>

Table 5.2: Music Content Features

Feature name	Description
Zero Crossings	The rate of sign-changes along a signal.
Spectral Centroid	The “center of mass” of the spectrum. Measures the brightness of the sound.
Spectral Flux	The squared change in normalized amplitude between two consecutive time frames. It measures how much the sound changes between frames.
Spectral Rolloff	Measures the amount of the right-skewedness of the power spectrum.
MFCC	Mel-Frequency Cepstral Coefficients. It models the auditory perception system and is widely used in speech and music domain.
Spectral Flatness Measure	Measures how much the audio signal sounds like a tone instead of noise.
Spectral Crest Factor	Another measure of noisiness. Similar to Spectral Flatness Measure.
Chroma	Pitch based feature. It projects the spectrum into 12 bins, representing the 12 distinct pitches of the chromatic musical scale.

for recommendation. The accuracy can be further improved by feature engineering [van den Oord *et al.*, 2013], which we reserve for future work.

5.4.1.3 Evaluation Protocol

In [Li *et al.*, 2011], an offline approach is proposed for evaluating contextual-bandit approaches with the assumption that the context (including the audio features and the elapsed time of songs) at different iterations is identically independently distributed. Unfortunately, this is not true in our case because when a song is not recommended, its elapsed time t keeps increasing and is thus strongly correlated. Therefore, an online user study is the most reliable means of evaluation.

To reduce the cost of the user study, we first conducted a comprehensive simulation study to verify the approaches. We then proceeded to user study for further verification only if they passed the simulations. The whole process underwent a few iterations, during which the models and algorithms were continually refined. The results hereby presented are from the final iteration, and intermediate results are either referred to as preliminary study whenever necessary or omitted due to page limitation.

5.4.2 Simulations

5.4.2.1 Effectiveness Study

$U = U_c U_n$ was used as the true model because the preliminary user studies showed that this resulted in better performance, which will be verified in Section 5.4.3 again. Because our model considers the impact of time, to make the simulations close to real situations, songs were rated about 50 seconds after being recommended. We treated every 20 recommendations as a recommen-

dation session, and the sessions were separated by 4-minute gaps.

Priors for the Bayesian models were set as uninformative ones or chosen based on preliminary simulation and user studies. For the exact Bayesian model, they are: $c_0 = 10$, $d_0 = 3$, $e_0 = 10^{-2}$, $f_0 = 10^{-3}$, $g_0 = 10^{-3}$, where f_0, g_0 are uninformative and c_0, d_0, e_0 are based on preliminary studies. For the approximate Bayesian model, they are: $\mathbf{D}_0 = \mathbf{E}_0 = 10^{-2}\mathbf{I}$, $\boldsymbol{\mu}_{\theta_0} = \boldsymbol{\mu}_{\beta_0} = \mathbf{0}$, $a_0 = 2$, $b_0 = 2 \times 10^{-8}$, where $\boldsymbol{\mu}_{\theta_0}, \boldsymbol{\mu}_{\beta_0}, a_0, b_0$ are uninformative and $\mathbf{D}_0, \mathbf{E}_0$ are based on preliminary studies; \mathbf{I} is the identity matrix.

U_n was discretized into the following intervals (in minutes) according to the exponentially decaying characteristics of human memory [Ebbinghaus *et al.*, 1913]: $[0, 2^{-3}), [2^{-3}, 2^{-2}), \dots, [2^{10}, 2^{11}), [2^{11}, +\infty)$. We defined the smallest interval as $[0, 2^{-3})$ because people usually do not listen to a song for less than 2^{-3} minutes (7.5 seconds). The largest interval was defined as $[2^{11}, +\infty)$ because our preliminary user study showed that evaluating one algorithm takes no more than 1.4 day, i.e., approximately 2^{11} minutes. Further discretization of $[2^{11}, +\infty)$ should be easy. For songs that had not been listened to by the target user, the elapsed time t was set as one month to ensure that U_n is close to 1.

We compared the recommendation performance of the six recommendation algorithms in terms of regret, which is a widely used metric in RL literatures. First we define that for the l -th recommendation, the difference between the maximum expected rating $\mathbb{E}[\hat{R}^l] = \max_{k=1\dots|\mathcal{S}|} U_k$ and the expected rating of the recommended song is $\Delta_l = \mathbb{E}[\hat{R}^l] - \mathbb{E}[R^l]$. Then, the cumulative regret for the n -th recommendation is: $\mathfrak{R}_n = \sum_{l=1\dots n} \Delta_l = \sum_{l=1\dots n} \mathbb{E}[\hat{R}^l] - \mathbb{E}[R^l]$, where a smaller \mathfrak{R}_n indicates better performance.

Different values of parameters $\{\boldsymbol{\theta}, s\}$ were tested. Elements of $\boldsymbol{\theta}$ were sampled from the standard normal distribution and s was sampled from the

uniform distribution with the range (100, 1000), where the range was determined based on the preliminary user study. We conducted 10 runs of the simulation study. Figure 5.7 shows the means and standard errors of the regret of different algorithms at different number of recommendations n . From the figure, we see that the algorithm Random (pure exploration) performs the worst. The two LinUCB-based algorithms are worse than Greedy-CN because LinUCB-C does not capture the novelty and LinUCB-CN does not capture the nonlinearity within U_c and U_n although both LinUCB-C and LinUCB-CN balance exploration and exploitation.

Bayes-UCB-based algorithms performed better than Greedy-CN because Bayes-UCB balances exploration and exploitation. In addition, the difference between Bayes-UCB and Greedy increases very fast when n is small. This is because small n means a small number of training samples and results in high uncertainty, i.e., the cold-start stage. Greedy algorithms, which are used by most existing recommendation systems, do not handle the uncertainty well, while Bayes-UCB can reduce the uncertainty quickly and thus improves the recommendation performance. The good performance of Bayes-UCB-CN-V also indicates that the piecewise linear approximation and variational inference is accurate.

5.4.2.2 Efficiency Study

A theoretical efficiency study of MCMC and variational inference algorithms is difficult to analyze due to their iterative nature and deserve future work. Instead, we conducted empirical efficiency study of the training algorithms for Bayes-UCB-CN (MCMC), Bayes-UCB-CN-V (variational inference), Greedy-CN (L-BFGS-B). In addition, the variational inference algorithm for the 3-factor model describe in Section 5.3.2.4 was also studied. LinUCB and Random

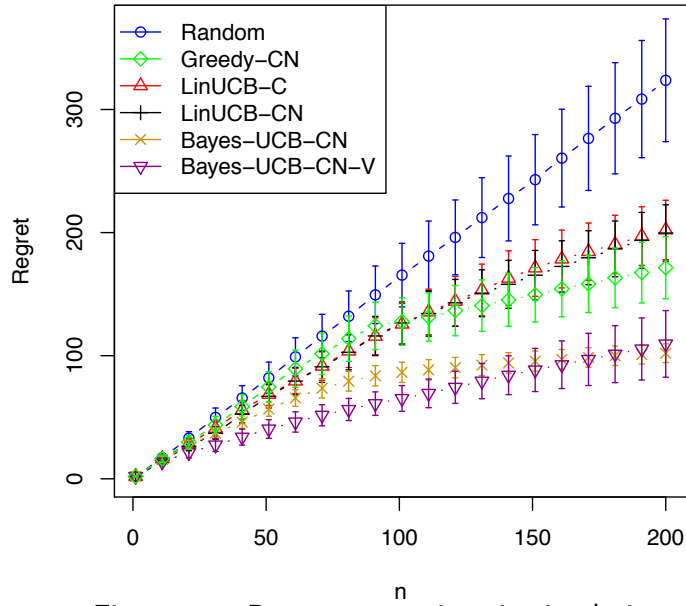


Figure 5.7: Regret comparison in simulation

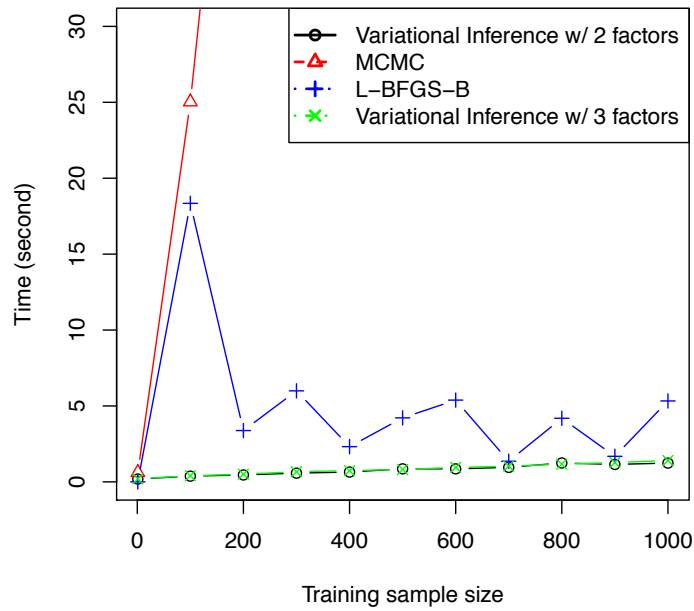


Figure 5.8: Time efficiency comparison

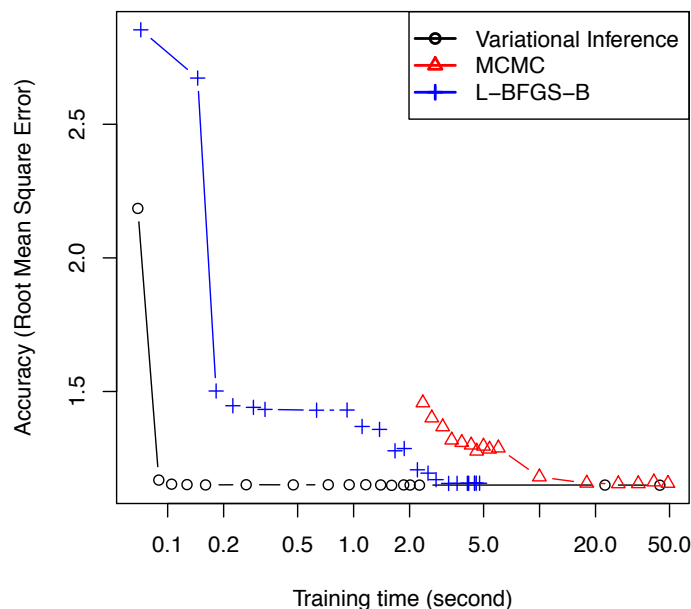


Figure 5.9: Accuracy versus training time

were not included because the algorithms are much simpler and thus faster (but also perform much worse). Experiments were conducted on a computer with an Intel Xeon CPU (L5520 @ 2.27GHz) and 32GB main memory. No multi-threading or GP-GPU were used in the comparisons. The programming language R was used to implement all the six algorithms.

From the results in Figure 5.8, we can see that time consumed by both MCMC and variational inference grows linearly with the training set size. However, variational inference is more than 100 times faster than the MCMC, and significantly faster than the L-BFGS-B algorithm. Comparing the variational inference algorithm with two or three factors, we find that adding another factor to the approximate Bayesian model only slightly slows down the variational inference algorithm. Moreover, when the sample size is less than 1000, the variational inference algorithm can finish in 2 seconds, which

makes online updating practical and meets the user requirement well. Implementing the algorithms in more efficient languages like C/C++ can result in even better efficiency.

The training time of all the algorithms is also affected by the accuracy that we want to achieve. To study this, we generated a training set (350 samples) and a test set (150 samples). For each algorithm, we ran it on the training set for multiple times, and every time we used different number of training iterations and collected both the training time and prediction accuracy on the test set. The whole process was repeated 10 times. For each algorithm and each number of iterations, the 10 training times and prediction accuracies were averaged. From the results shown in Figure 5.9, we can see that variational inference (VI) converges very fast; L-BFGS-B takes much longer time than VI to converge; MCMC is more than 100 times slower than VI.

Time consumed in the prediction phase of the Bayesian methods is larger than that of Greedy and LinUCB-based methods because of the sampling process. However, for the two factors model Bayes-UCB-CN-V, prediction can be accelerated significantly by the PRODCLIN algorithm without sacrificing the accuracy [MacKinnon *et al.*, 2007]. In addition, since prediction for different songs is trivially parallelizable, scaling variational inference to large music databases should be easy.

We also conducted sample efficiency study of the exact Bayesian model, the approximate Bayesian model, and the minimum mean squared error based frequentist model used for Greedy-CN. We first generated a test set (300 samples), and then tested all the models with different size of training samples. The whole process was repeated 10 times, and the average accuracies are shown in Figure 5.10. We can see that the exact Bayesian model and the frequentist model have almost identical sample efficiency, which confirms that the only

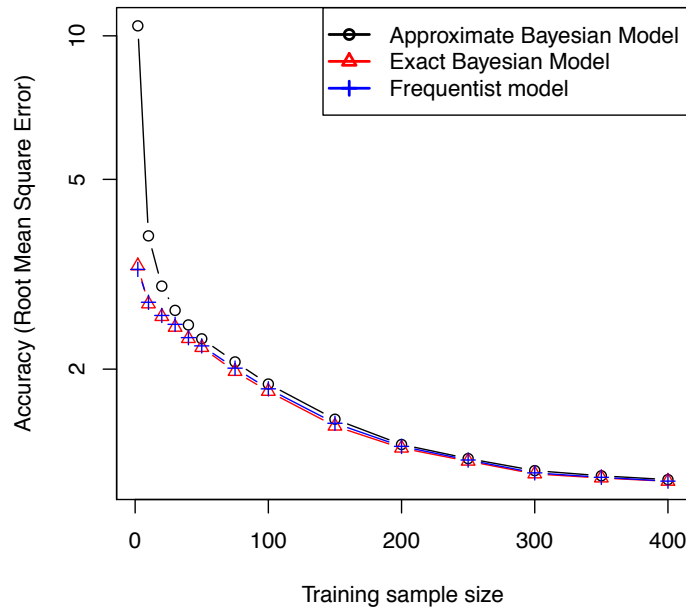


Figure 5.10: Sample efficiency comparison

difference between Bayes-UCB and Greedy-CN is whether uncertainty is considered or not. The approximate Bayesian model performs slightly worse than the others because of the piecewise linear approximation and the variational inference algorithm.

5.4.3 User Study

Undergraduate students aged 17-25 years were chosen as our study target. It would be interesting to study the impact of occupations and ages on our method in the future. Most applicants were females, and we selected 15 from them with approximately equal number of males (6) and females (9). Their cultural backgrounds were diversified to include Chinese, Malay, Indian and Indonesian. They all listen to music regularly (at least 3 hours per week). To reduce the number of subjects needed, the within-subject experiment design

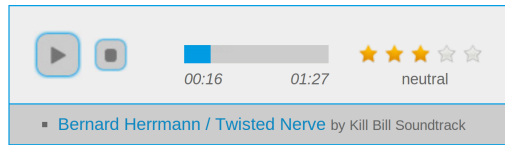


Figure 5.11: User evaluation interface

was used, i.e. every subject evaluated all recommendation algorithms. Every subject was rewarded with a small token payment for their time and effort. For each of the 6 algorithms, a subject evaluated 200 recommendations, a number more than sufficient to cover the cold-start stage. Every recommended song was listened to for at least 30 seconds (except when the subject was very familiar with the song *a priori*) and rated based on a 5-point Likert-scale as before. Subjects were required to rest for at least 4 minutes after listening to 20 songs to ensure the quality of the ratings and simulate recommendation sessions. To minimize the carryover effect of the within-subject design, subjects were not allowed to evaluate more than two algorithms within one day, and there must be a gap of more than 6 hours between two algorithms. The user study lasted one week. Every subject spent more than 14 hours in total. During the evaluation, the recommendation models were updated immediately whenever a new rating was obtained. The main interface used for evaluation is in Figure 5.11.

5.4.3.1 The Overall Recommendation Performance

Because the true model is not known in user study, the regret used in simulations cannot be used here. We thus choose average rating as the evaluation metric, which is also popular in evaluations of RL algorithms. Figure 5.12 shows the average ratings and standard errors of every algorithm from the beginning to the n -th recommendation.

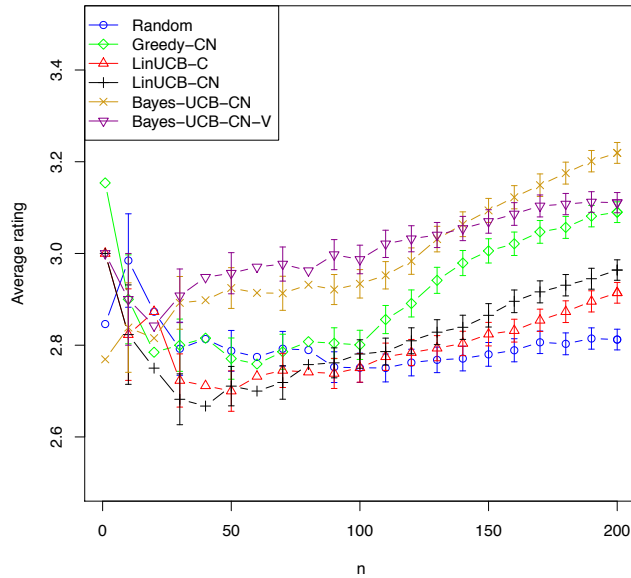


Figure 5.12: Performance comparison in user study

T-tests at different iterations show Bayes-UCB-CN outperforms Greedy-CN since the 45th iteration with p -values < 0.039 . Bayes-UCB-CN-V outperforms Greedy-CN from the 42th to the 141th iteration with p -values < 0.05 , and afterwards with p -values < 0.1 . Bayes-UCB-CN and Greedy-CN share the same rating model and the only difference between them is that Bayes-UCB-CN balances exploration/exploitation while Greedy-CN only exploits. Therefore, the improvement of Bayes-UCB-CN against Greedy-CN is solely contributed by the exploration/exploitation tradeoff.

More interestingly, when $n \leq 100$ (cold-start stage) the differences between Bayes-UCB-CN and Greedy-CN are even more significant. This is because during the cold-start stage, the uncertainty is very high; Bayes-UCB explores and thus reduces the uncertainty quickly while Greedy-CN always exploits and thus cannot reduce the uncertainty as efficiently as Bayes-UCB-CN. To verify this point, we first define a metric for uncertainty as $\frac{1}{|S|} \sum_{k=1}^{|S|} \text{SD} [p(U_k | \mathcal{D}_n)]$,

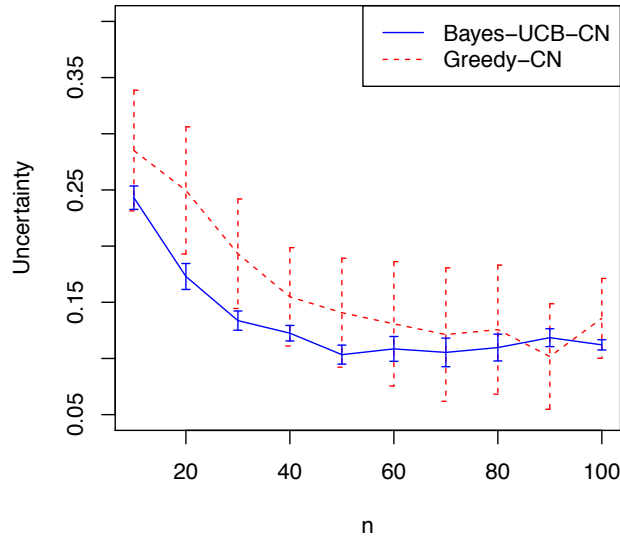


Figure 5.13: The uncertainty of Bayes-UCB decreases faster than that of Greedy

which is the mean of the standard deviations of all song’s posterior distributions $p(U_k|D_n)$ estimated using the exact Bayesian model. Larger standard deviation means larger uncertainty as illustrated in Figure 5.1. Given the iteration n , we calculate an uncertainty measure based on each user’s recommendation history. The means and standard errors of the uncertainties among all users at different iterations are shown in Figure 5.13. When the number of training data points increases, the uncertainty decreases. Also as expected, the uncertainty of Bayes-UCB-CN decreases faster than Greedy-CN when n is small, and later the two remain comparable because both have obtained enough training samples to fully train the models. Therefore, this verifies that our bandit approach handles uncertainty better during the initial stage, and thus mitigate the cold-start problem.

Results in Figure 5.12 also show that all algorithms involving CN outperforms LinUCB-C, indicating that the novelty factor of the rating model improves recommendation performance. In addition, Bayes-UCB-CN outper-

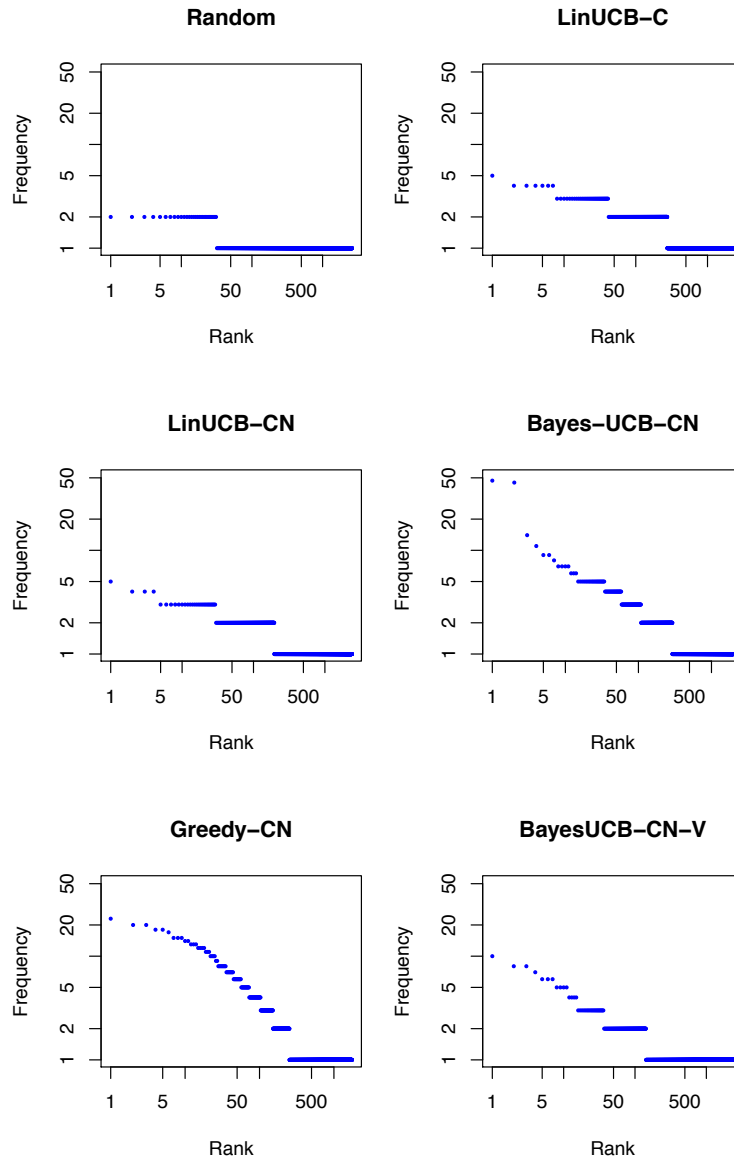


Figure 5.14: Distributions of song repetition frequency

forms LinUCB-CN significantly, suggesting that multiplying U_c and U_n together works better than linearly combining them.

5.4.3.2 Playlist Generation

As discussed in Section 5.2.1, repeating songs following the Zipf's law is important for playlist generation. Therefore, we evaluated the playlists generated during the recommendation process by examining the distribution of songs repetition frequencies for every user. We generated the plots of the distributions in the same way we generated Figure 5.3 for the six algorithms. Ideal algorithms should reproduce repetition distributions of Figure 5.3.

The results of the six algorithms are shown in Figure 5.14. As we can see all algorithms with U_c and U_n multiplied together (i.e. Bayes-UCB-CN, Greedy-CN, BayesUCB-CN-V) reproduce the Zipf's law pattern well, while the algorithms without U_c (Random, LinUCB-C) or with U_c and U_n added together (LinUCB-CN) do not. This confirms that our model $U = U_c U_n$ can effectively reproduce the Zipf's law distribution. Thus, we successfully modeled an important part for combining music recommendation and playlist generation.

5.4.3.3 Piecewise Linear Approximation

In addition to the studies detailed above, the piecewise linear approximation of the novelty model is tested again by randomly selecting four users and showing in Figure 5.15 their novelty models learnt by Bayes-UCB-CN-V. Specifically, the posterior distributions of $\beta' \mathbf{t}$ for $t \in (0, 2^{11})$ are presented. The lines represent the mean values of $\beta' \mathbf{t}$ and the regions around the lines the confidence bands of one standard deviation. The scale of $\beta' \mathbf{t}$ is not important because

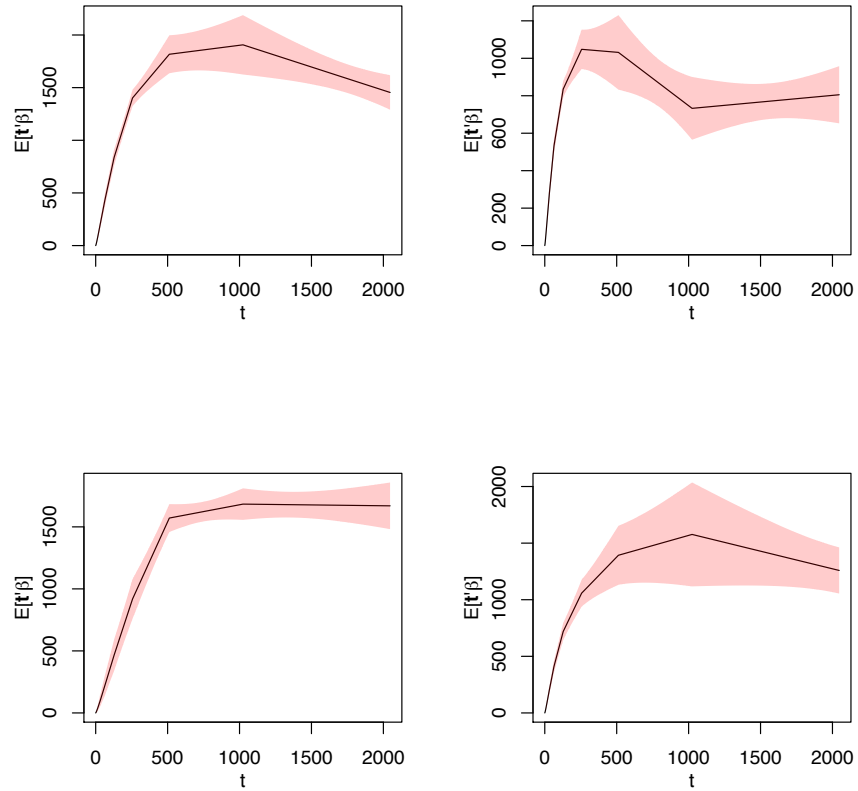


Figure 5.15: Four users' diversity factors learnt from the approximate Bayesian model

$\beta't$ is multiplied together with the content factor, and any constant scaling of one factor can be compensated by the scaling of the other one. Comparing Figure 5.15 and Figure 5.4, we can see that the learnt piecewise linear novelty factors match our analytic form U_n well. This again confirms the accuracy of the piecewise linear approximation.

5.5 Discussion

Exploring user preferences is a central issue for recommendation systems, regardless of the specific media types. Under uncertainty, the greedy approach

usually produces suboptimal results, and balancing exploration/exploitation is important. One successful example of exploration/exploitation tradeoff is the news recommender [Li *et al.*, 2010]. Our work has shown its effectiveness in music recommendation. Given that uncertainty exists universally in all kinds of recommenders, it will be interesting to examine its effectiveness in recommenders for other media types e.g., video and image.

Our models and algorithms could be generalized to other recommenders. First, the mathematical form of the approximate Bayesian model is general enough to cover a family of rating functions that can be factorized as the product of a few linear functions (Section 5.3.2.4). Moreover, we can often approximate nonlinear functions with linear ones. For instance, we can use a feature mapping function $\phi(\mathbf{x})$ and make $U_c = \boldsymbol{\theta}'\phi(\mathbf{x})$ to capture the non-linearity in our content model. Therefore, it will be interesting to explore our approximate Bayesian model and the variational inference algorithm in other recommendation systems. Second, the proposed novelty model may not be suitable for movie recommendation due to different consumption patterns in music and movie—users may listen to their favorites songs for many times, but repetitions are relatively rare for movies. However, the novelty model may suit recommenders which repeat items (e.g. food or makeup recommenders [Liu *et al.*, 2013]). If their repetition patterns also follow the Zipf’s law, both the exact and approximate Bayesian models can be used; otherwise, the approximate Bayesian model can be used at least.

5.6 Conclusion

In this chapter, we described a multi-armed bandit approach to interactive music recommendation that balances exploration and exploitation, mitigates

the cold-start problem, and improves recommendation performance. We described a rating model including music audio content and novelty to integrate music recommendation and playlist generation. To jointly learn the parameters of the rating model, a Bayesian regression model together with a MCMC inference procedure were developed. To make the Bayesian inference efficient enough for online updating and generalize the model for more factors such as diversity, a piecewise linear approximate Bayesian regression model and a variational inference algorithm were built. The results from simulation demonstrate that our models and algorithms are accurate and highly efficient. User study results show that (1) the bandit approach mitigates the cold-start problem and improves recommendation performance, and (2) the novelty model together with the content model capture the Zipf's law of repetitions in recommendations.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The ultimate goal of a music recommender is to satisfy users' music needs. We pressed on toward this goal by developing methods from different aspects. First, we demonstrated the first mobile-based context-aware mobile music recommender that recommends songs to match the target user's activity. Second, since existing recommenders based on traditional music audio content features are not accurate, we then presented a deep learning based method to automatically learn a set of features. Experiment results show that methods with the learnt features are significantly more accurate than those with traditional features for both content-based and hybrid music recommendation. Finally, we consider music recommendation as an interactive process and optimize the whole process in a holistic way together with playlist generation.

While good performance has been achieved for all these methods, there is still much room for improvement. In the following section, we list a few possible directions as future work.

6.2 Future Work

Context-aware recommendation has become increasingly popular because of the advent of powerful and sensor-rich smartphones. In the future, more context categories can be added when related sensors are available. The recent prevailingness of wearable healthcare devices also provides a source of contextual data, which could be integrated into the system. For example, with a watch that accurately measures users' heart rates¹, we could develop a recommender that controls the tempo and style of the music to better accompany or even guide the users through their jogging journey. Similarly, with a wearable sleep monitor, the system could then plan the music sequence to promote sleep.

As deep learning strives to provide a model for human cognition, it has the potential to reveal many secrets behind our preferences for music. Our study that uses deep learning serves as a mere starting point to tap into that potential. One practical future direction could be to further improve the recommendation performance by explicitly modeling the temporal structure of music content using deep recurrent neural network [Hermans and Schrauwen, 2013]. Another interesting direction could be to interpret the automatically learnt features to discover interesting characteristics of music.

The exploration/exploitation tradeoff idea can also be adopted to boost CF's performance. A simple approach is to use the latent features learnt by existing matrix factorization methods to replace the audio features in our methods, and keep other parts of our methods unchanged [Xing *et al.*, 2014]. To generate even better playlists, an interesting direction is to consider more factors such as diversity, mood, and genres.

¹<http://www.polar.com/>

While the methods developed in this thesis can be applied in practice separately, it is possible to build a unified system to take advantage of all them. For example, features learnt through the deep belief network can be used in the content part of the Bayesian model for exploration/exploitation tradeoff. In context-aware recommendation, uncertainty also exists, and thus balancing exploration/exploitation can reduce the amount of data required for adaptation and improve performance. The deep learning method can also be used to improve activity classification accuracy for context-aware recommendation.

Bibliography

- [Adomavicius and Tuzhilin, 2005] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the State-of-the-Art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.
- [Adomavicius and Tuzhilin, 2008] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Proceedings of the 2008 ACM conference on Recommender systems*, RecSys '08, pages 335–336, New York, NY, USA, 2008. ACM.
- [Agarwal and Chen, 2009] Deepak Agarwal and Bee C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '09, pages 19–28, New York, NY, USA, 2009. ACM.
- [Agrawal and Goyal, 2012] Shipra Agrawal and Navin Goyal. Analysis of Thompson Sampling for the multi-armed bandit problem, April 2012.
- [Anderson, 2006] Chris Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006.
- [Auer and Long, 2002] Peter Auer and M. Long. Using confidence bounds for exploitation-exploration trade-offs. In *Journal of Machine Learning Research*, 2002.
- [Baltrunas and Amatriain, 2009] Linas Baltrunas and Xavier Amatriain. Towards Time-Dependant Recommendation based on Implicit Feedback. In *CARS*, 2009.
- [Baltrunas *et al.*, 2010] Linas Baltrunas, Marius Kaminskas, Francesco Ricci, L. Rokach, B. Shapira, and K. H. Luke. Best Usage Context Prediction for Music Tracks. In *CARS*, September 2010.

BIBLIOGRAPHY

- [Baltrunas *et al.*, 2011] Linas Baltrunas, Marius Kaminskas, Bernd Ludwig, Omar Moling, Francesco Ricci, Aykan Aydin, Karl-Heinz Lüke, and Roland Schwaiger. InCarMusic: Context-Aware Music Recommendations in a Car E-Commerce and Web Technologies. In *LNBIP*. 2011.
- [Bell *et al.*, 2009] Robert M. Bell, Yehuda Koren, and Chris Volinsky. The BellKor solution to the netflix prize, 2009.
- [Bengio *et al.*, 2012] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives, October 2012.
- [Bengio, 2009] Yoshua Bengio. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, January 2009.
- [Berchtold *et al.*, 2010] M. Berchtold, M. Budde, D. Gordon, H. R. Schmidtke, and M. Beigl. ActiServ: Activity Recognition Service for mobile phones. In *ISWC*, pages 1–8, October 2010.
- [Bertin-Mahieux *et al.*, 2008] Thierry Bertin-Mahieux, Douglas Eck, François Maillet, and Paul Lamere. Autotagger: A Model for Predicting Social Tags from Acoustic Features on Large Music Databases. *JNMR*, 37(2):115–135, June 2008.
- [Bogdanov *et al.*, 2010] Dmitry Bogdanov, Martín Haro, Ferdinand Fuhrmann, Emilia Gómez, and Perfecto Herrera. Content-based music recommendation based on user preference examples. In *The 4th ACM Conference on Recommender Systems. Workshop on Music Recommendation and Discovery (Womrad 2010)*, 2010.
- [Bogdanov *et al.*, 2013] Dmitry Bogdanov, Martín Haro, Ferdinand Fuhrmann, Anna Xambó, Emilia Gómez, and Perfecto Herrera. Semantic audio content-based music recommendation and visualization based on user preference examples. *Inf. Process. Manage.*, 49(1):13–33, January 2013.
- [Boström, 2008] Fredrik Boström. AndroMedia - Towards a Context-aware Mobile Music Recommender. Master’s thesis, 2008.
- [Brezmes *et al.*, 2009] Tomas Brezmes, Juan-Luis Gorricho, and Josep Cotrina. Activity Recognition from Accelerometer Data on a Mobile Phone. In *IWANN*. 2009.

BIBLIOGRAPHY

- [Bu *et al.*, 2010] Jiajun Bu, Shulong Tan, Chun Chen, Can Wang, Hao Wu, Lijun Zhang, and Xiaofei He. Music recommendation by unified hypergraph: Combining social media information and music content. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 391–400, New York, NY, USA, 2010. ACM.
- [Byrd *et al.*, 1995] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, September 1995.
- [Camurri *et al.*, 2010] Antonio Camurri, Gualtiero Volpe, Hugues Vinet, Roberto Bresin, Marco Fabiani, Gaël Dubus, Esteban Maestre, Jordi Llop, Jari Kleimola, Sami Oksanen, Vesa Välimäki, and Jarno Seppanen. User-Centric Context-Aware Mobile Applications for Embodied Music Listening User Centric Media. In *LNICST*, pages 21–30. 2010.
- [Cano *et al.*, 2005] Pedro Cano, Markus Koppenberger, and Nicolas Wack. Content-based music audio recommendation. In *Proceedings of the 13th annual ACM international conference on Multimedia*, MULTIMEDIA '05, pages 211–212, New York, NY, USA, 2005. ACM.
- [Casey *et al.*, 2008] M. A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-Based Music Information Retrieval: Current Directions and Future Challenges. *Proceedings of the IEEE*, 96(4):668–696, March 2008.
- [Castells *et al.*, 2011] 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 ECIR 2011*, April 2011.
- [Chen and Chen, 2001] Hung C. Chen and Arbee L. P. Chen. A music recommendation system based on music data grouping and user interests. In *Proceedings of CIKM*, CIKM '01, pages 231–238, New York, NY, USA, 2001. ACM.
- [Chen *et al.*, 2012] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *KDD '12*, 2012.
- [Chen *et al.*, 2013] Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In

BIBLIOGRAPHY

- Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 193–202, New York, NY, USA, 2013. ACM.
- [Chi *et al.*, 2010] Chung Y. Chi, Richard Tzong Han Tsai, Jeng Y. Lai, and Jane Yung jen Hsu. A Reinforcement Learning Approach to Emotion-based Automatic Playlist Generation. *Technologies and Applications of Artificial Intelligence, International Conference on*, 0:60–65, 2010.
- [Cunningham *et al.*, 2008] Stuart Cunningham, Stephen Caulder, and Vic Grout. Saturday Night or Fever? Context-Aware Music Playlists. In *AM '08*, 2008.
- [Das *et al.*, 2007] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.
- [de Campos *et al.*, 2010] Luis M. de Campos, Juan M. Fernández-Luna, Juan F. Huete, and Miguel A. Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785–799, September 2010.
- [Del Castillo, 2007] Hugo S. Del Castillo. Hybrid Content-Based Collaborative-Filtering music recommendations. Master's thesis, University of Twente, The Netherlands, 2007.
- [Domingues *et al.*, 2012] Marcos A. Domingues, Fabien Gouyon, Al'ipio M. Jorge, José P. Leal, Jo . Vinagre, Lu'is Lemos, and Mohamed Sordo. Combining usage and content in an online music recommendation system for music in the long-tail. In *Proceedings of the 21st International Conference Companion on World Wide Web*, WWW '12 Companion, pages 925–930, New York, NY, USA, 2012. ACM.
- [Dornbush *et al.*, 2007] Sandor Dornbush, Anupam Joshi, Zary Segall, and Tim Oates. A Human Activity Aware Learning Mobile Music Player. In *Proc. of the 2007 conference on Advances in Ambient Intelligence*, 2007.
- [Ebbinghaus *et al.*, 1913] H. Ebbinghaus, H.A. Ruger, and C.E. Bussenius. *Memory: a contribution to experimental psychology*. Educational reprints. Teachers College, Columbia University, 1913.

BIBLIOGRAPHY

- [Elliott and Tomlinson, 2006] Greg T. Elliott and Bill Tomlinson. Personal-Soundtrack: context-aware playlists that adapt to user pace. In *SIGCHI*, 2006.
- [Erhan *et al.*, 2010] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre A. Manzagol, Pascal Vincent, and Samy Bengio. Why Does Unsupervised Pre-training Help Deep Learning? *J. Mach. Learn. Res.*, 11:625–660, March 2010.
- [Friedman and Koller, 2009] Nir Friedman and Daphne Koller. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, 1 edition, July 2009.
- [Golovin and Rahm, 2004] N. Golovin and E. Rahm. Reinforcement learning architecture for Web recommendations. In *Proceedings of International Conference on Information Technology: Coding and Computing*, volume 1, pages 398–402 Vol.1. IEEE, April 2004.
- [Gunawardana and Shani, 2009] Asela Gunawardana and Guy Shani. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *JMLR*, 10, December 2009.
- [Hamel and Eck, 2010] Philippe Hamel and Douglas Eck. Learning features from music audio with deep belief networks. In *11th International Society for Music Information Retrieval Conference*, 2010.
- [Han *et al.*, 2010] Byeong J. Han, Seungmin Rho, Sanghoon Jun, and Eenjun Hwang. Music emotion classification and context-based music recommendation. *Multimedia Tools Appl.*, 47(3):433–460, May 2010.
- [Hariri *et al.*, 2012] Negar Hariri, Bamshad Mobasher, and Robin Burke. Context-aware music recommendation based on latenttopic sequential patterns. In *Proceedings of the sixth ACM conference on Recommender systems*, RecSys '12, pages 131–138. ACM Press, 2012.
- [Hastie *et al.*, 2009] T.J. Hastie, R.J. Tibshirani, and J.J.H. Friedman. *The Elements of Statistical Learning*. Springer-Verlag New York, 2009.
- [Herlocker *et al.*, 1999] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John 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*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

BIBLIOGRAPHY

- [Hermans and Schrauwen, 2013] Michiel Hermans and Benjamin Schrauwen. Training and analyzing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2013.
- [Hinton *et al.*, 2006] Geoffrey E. Hinton, Simon Osindero, and Yee W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.
- [Hinton, 2012] Geoffrey E. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 599–619. Springer Berlin Heidelberg, 2012.
- [Hornik *et al.*, 1989] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989.
- [Hu and Ogihara, 2011] Yajie Hu and Mitsunori Ogihara. Nexttone player: A music recommendation system based on user behavior. In *ISMIR*, 2011.
- [Hu *et al.*, 2008] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative Filtering for Implicit Feedback Datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, volume 0 of *ICDM '08*, pages 263–272, Washington, DC, USA, December 2008. IEEE Computer Society.
- [Huang *et al.*, 2008] Thomas S. Huang, Charlie K. Dagli, Shyamsundar Rajaram, Edward Y. Chang, Michael Mandel, Graham E. Poliner, and Daniel P. W. Ellis. Active learning for interactive multimedia retrieval. *Proceedings of the IEEE*, 96(4):648–667, April 2008.
- [Humphrey *et al.*, 2012] Eric J. Humphrey, Juan P. Bello, and Yann LeCun. Moving Beyond Feature Design: Deep Architectures and Automatic Feature Learning in Music Informatics. In *13th International Society for Music Information Retrieval Conference*, 2012.
- [Humphrey *et al.*, 2013] Eric J. Humphrey, Juan P. Bello, and Yann LeCun. Feature learning and deep architectures: new directions for music informatics. *Journal of Intelligent Information Systems*, 41(3):461–481, 2013.
- [Joachims *et al.*, 1997] Thorsten Joachims, Dayne Freitag, and Tom Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *Proceedings of the*

BIBLIOGRAPHY

- Fifteenth International Joint Conference on Artificial Intelligence, IJCAI '97*, pages 770–777, 1997.
- [Kaminskas and Ricci, 2011] Marius Kaminskas and Francesco Ricci. Location-Adapted Music Recommendation Using Tags. In *UMAP*, 2011.
- [Karimi *et al.*, 2011] R. Karimi, C. Freudenthaler, A. Nanopoulos, and L. Schmidt-Thieme. Towards optimal active learning for matrix factorization in recommender systems. In *ICTAI*, pages 1069–1076. IEEE, November 2011.
- [Kaufmann *et al.*, 2012] Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On Bayesian Upper Confidence Bounds for Bandit Problems. *JMLR - Proceedings Track*, 22:592–600, 2012.
- [Khan *et al.*, 2011] Mridul Khan, Sheikh I. Ahamed, Miftahur Rahman, and Roger O. Smith. A Feature Extraction Method for Real time Human Activity Recognition on Cell Phones. In *isQoLT*, 2011.
- [Kim *et al.*, 2006] Jong-Hun Kim, Chang-Woo Song, Kee-Wook Lim, and Jung-Hyun Lee. Design of Music Recommendation System Using Context Information. In *LNCS*, volume 4088, chapter 83, pages 708–713. 2006.
- [Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix Factorization Techniques for Recommender Systems. *Computer*, 42(8):30–37, August 2009.
- [Kwapisz *et al.*, 2011] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, 12:74–82, March 2011.
- [Landis and Koch, 1977] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, March 1977.
- [Lathia *et al.*, 2010] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd international ACM SIGIR conference, SIGIR '10*, pages 210–217, New York, NY, USA, 2010. ACM.
- [Leake *et al.*, 2006] David Leake, Ana Maguitman, and Thomas Reichherzer. Cases, Context, and Comfort: Opportunities for Case-Based Reasoning in Smart Homes. In *Designing Smart Homes*, LNCS. 2006.

BIBLIOGRAPHY

- [Lee and Cho, 2011] Young S. Lee and Sung B. Cho. Activity recognition using hierarchical hidden markov models on a smartphone with 3D accelerometer. In *H AIS*, pages 460–467, 2011.
- [Lee and Lee, 2008] Jae Lee and Jin Lee. Context Awareness by Case-Based Reasoning in a Music Recommendation System. pages 45–58. 2008.
- [Lee *et al.*, 2009] Honglak Lee, Yan Largman, Peter Pham, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *NIPS*, 2009.
- [Lehtiniemi, 2008] Arto Lehtiniemi. Evaluating SuperMusic: streaming context-aware mobile music service. In *ACE*, 2008.
- [Levitin and McGill, 2007] Daniel J. Levitin and James McGill. Life Soundtracks: The uses of music in everyday life. In *unpublished*, 2007.
- [Li *et al.*, 2007] Qing Li, Sung H. Myaeng, and Byeong M. Kim. A probabilistic music recommender considering user opinions and audio features. *Information Processing and Management*, 43(2):473–487, March 2007.
- [Li *et al.*, 2010] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, WWW ’10, pages 661–670, New York, NY, USA, 2010. ACM.
- [Li *et al.*, 2011] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of WSDM*, WSDM ’11, pages 297–306, New York, NY, USA, 2011. ACM.
- [Liebman and Stone, 2014] Elad Liebman and Peter Stone. Dj-mc: A reinforcement-learning agent for music playlist recommendation. *CoRR*, abs/1401.1880, 2014.
- [Liu *et al.*, 2009] Hao Liu, Jun Hu, and Matthias Rauterberg. Music Playlist Recommendation Based on User Heartbeat and Music Preference. In *IC-CTD*, 2009.
- [Liu *et al.*, 2013] Luoqi Liu, Hui Xu, Junliang Xing, Si Liu, Xi Zhou, and Shuicheng Yan. Wow! you are so beautiful today! In *ACM MM ’13*, October 2013.

BIBLIOGRAPHY

- [Liu, 2013] Ning-Han Liu. Comparison of content-based music recommendation using different distance estimation methods. *Applied Intelligence*, 38(2):160–174, June 2013.
- [Logan and Salomon, 2001] Beth Logan and Ariel Salomon. A Content-Based music similarity function. Technical report, Cambridge Research Laboratory, 2001.
- [Logan, 2002] Beth Logan. Content-Based Playlist Generation: Exploratory Experiments. 2002.
- [MacKinnon *et al.*, 2007] David P. MacKinnon, Matthew S. Fritz, Jason Williams, and Chondra M. Lockwood. Distribution of the product confidence limits for the indirect effect: program PRODCLIN. *Behavior research methods*, 39(3):384–389, August 2007.
- [May *et al.*, 2012] Benedict C. May, Nathan Korda, Anthony Lee, and David S. Leslie. Optimistic Bayesian Sampling in Contextual-Bandit Problems. *JMLR*, 2012:2069–2106, June 2012.
- [McFee *et al.*, 2012a] B. McFee, L. Barrington, and G. Lanckriet. Learning content similarity for music recommendation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(8):2207–2218, October 2012.
- [McFee *et al.*, 2012b] Brian McFee, Thierry Bertin-Mahieux, Daniel P. W. Ellis, and Gert R. G. Lanckriet. The million song dataset challenge. In *21st International Conference Companion on World Wide Web*, pages 909–916, 2012.
- [Mermelstein, 1976] Paul Mermelstein. Distance measures for speech Recognition—Psychological and instrumental. In *Joint Workshop on Pattern Recognition and Artificial Intelligence*, 1976.
- [Mooney and Roy, 2000] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, DL '00, pages 195–204, New York, NY, USA, 2000. ACM.
- [Newman, 2005] M. E. J. Newman. Power laws, pareto distributions and zipf’s law. *Contemporary Physics*, 46(5):323–351, September 2005.
- [North *et al.*, 2004] Adrian C. North, David J. Hargreaves, and Jon J. Hargreaves. Uses of Music in Everyday Life. *Music Perception: An Interdisciplinary Journal*, 22(1), 2004.

BIBLIOGRAPHY

- [Oliveira and Oliver, 2008] Rodrigo D. Oliveira and Nuria Oliver. TripleBeat: enhancing exercise performance with persuasion. In *MobileHCI*, 2008.
- [Pan *et al.*, 2008] Rong Pan, Yunhong Zhou, Bin Cao, N. N. Liu, R. Lukose, M. Scholz, and Qiang Yang. One-Class Collaborative Filtering. In *Data Mining, 2008. ICDM. Eighth IEEE International Conference on*, volume 0, pages 502–511, Los Alamitos, CA, USA, December 2008. IEEE.
- [Park *et al.*, 2006] Han-Saem Park, Ji-Oh Yoo, and Sung-Bae Cho. A Context-Aware Music Recommendation System Using Fuzzy Bayesian Networks with Utility Theory. In *FSKD*. 2006.
- [Park *et al.*, 2013] Sunho Park, Yong D. Kim, and Seungjin Choi. Hierarchical Bayesian Matrix Factorization with Side Information. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI'13*, pages 1593–1599. AAAI Press, 2013.
- [Pikrakis, 2013] Aggelos Pikrakis. A deep learning approach to rhythm modelling with applications. In *6th International Workshop on Machine Learning and Music (MML13)*, 2013.
- [Popescul and Ungar, 2001] Rin Popescul and Lyle H. Ungar. Probabilistic models for unified collaborative and content-based recommendation in sparsedata environments. In *UAI 2011*, 2001.
- [Porteous *et al.*,] Ian Porteous, Arthur Asuncion, and Max Welling. Bayesian Matrix Factorization with Side Information and Dirichlet Process Mixtures.
- [Reddy and Mascia, 2006] Sasank Reddy and Jeff Mascia. Lifetrak: music in tune with your life. In *HCM*, 2006.
- [Resa, 2010] Z. Resa. Towards Time-aware Contextual Music Recommendation: An Exploration of Temporal Patterns of Music Listening Using Circular Statistics. Master’s thesis, 2010.
- [Resnick *et al.*, 1994] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW*, 1994.
- [Reynolds *et al.*, 2008] G. Reynolds, D. Barry, T. Burke, and E. Coyle. Interacting with large music collections: Towards the use of environmental metadata. In *ICME*, June 2008.

BIBLIOGRAPHY

- [Rho *et al.*, 2009] Seungmin Rho, Byeong J. Han, and Eenjun Hwang. SVR-based music mood classification and context-based music recommendation. In *ACM MM*, 2009.
- [Salakhutdinov and Mnih, 2008a] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning, ICML '08*, pages 880–887, New York, NY, USA, 2008. ACM.
- [Salakhutdinov and Mnih, 2008b] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. 2008.
- [Saponas *et al.*,] T. Scott Saponas, Jonathan Lester, Jon Froehlich, James Fogarty, and James Landay. iLearn on the iPhone: Real-Time Human Activity Classification on Commodity Mobile Phones.
- [Schein *et al.*, 2002] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, 2002.
- [Schmidt and Kim, 2011] Erik M. Schmidt and Youngmoo E. Kim. Learning emotion-based acoustic features with deep belief networks. In *ISMIR*, 2011.
- [Schmidt and Kim, 2013] Erik M. Schmidt and Youngmoo E. Kim. Learning rhythm and melody features with deep belief networks. In *ISMIR*, 2013.
- [Seppänen and Huopaniemi, 2008] Jarno Seppänen and Jyri Huopaniemi. Interactive and context-aware mobile music experiences. September 2008.
- [Shan and Banerjee, 2010] Hanhuai Shan and Arindam Banerjee. Generalized Probabilistic Matrix Factorizations for Collaborative Filtering. pages 1025–1030, December 2010.
- [Shani *et al.*, 2005] Guy Shani, David Heckerman, and Ronen I. Brafman. An MDP-Based Recommender System. *JMLR*, 6:1265–1295, 2005.
- [Shao *et al.*, 2009] Bo Shao, Dingding Wang, Tao Li, and Mitsunori Ogihara. Music recommendation based on acoustic features and user access patterns. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(8):1602–1611, November 2009.
- [Shruthi *et al.*,] J. Shruthi, S. Sneha, Uttarika R. Shetty, and D. Jayalakshmi. A hybrid music recommender system.

BIBLIOGRAPHY

- [Silva and Carin, 2012] Jorge Silva and Lawrence Carin. Active learning for online bayesian matrix factorization. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '12*, pages 325–333, New York, NY, USA, 2012. ACM.
- [Song *et al.*, 2012] Yading Song, Simon Dixon, and Marcus Pearce. A Survey of Music Recommendation Systems and Future Perspectives. In *Proceedings of the 9th International Symposium on Computer Music Modelling and Retrieval, CMMR '12*, June 2012.
- [Srivihok and Sukonmanee, 2005] Anongnart Srivihok and Pisit Sukonmanee. E-commerce intelligent agent: personalization travel support agent using Q Learning. In *Proceedings of the 7th International Conference on Electronic Commerce, ICEC '05*, pages 287–292. ACM Press, 2005.
- [Su *et al.*, 2010] Ja-Hwung Su, Hsin-Ho Yeh, P. S. Yu, and V. S. Tseng. Music Recommendation Using Content and Context Information Mining. *Intelligent Systems, IEEE*, 25(1):16–26, January 2010.
- [Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, March 1998.
- [Szepesvári, 2010] Csaba Szepesvári. *Algorithms for Reinforcement Learning*, volume 4 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool, San Rafael, CA, USA, January 2010.
- [Taghipour and Kardan, 2008] Nima Taghipour and Ahmad Kardan. A hybrid web recommender system based on Q-learning. In *SAC '08*, pages 1164–1168, 2008.
- [Tiemann and Pauws, 2007] Marco Tiemann and Steffen Pauws. Towards ensemble learning for hybrid music recommendation. In *Proceedings of the 2007 ACM conference on Recommender systems, RecSys '07*, pages 177–178, New York, NY, USA, 2007. ACM.
- [Turnbull *et al.*, 2007] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Towards musical query-by-semantic-description using the CAL500 data set. In *SIGIR*, 2007.
- [Turnbull *et al.*, 2009] Douglas Turnbull, Luke Barrington, Mehrdad Yazdani, and Gert Lanckriet. Combining audio content and social context for semantic music discovery. In *SIGIR*, 2009.

BIBLIOGRAPHY

- [Tzanetakis and Cook, 1999] George Tzanetakis and Perry Cook. MARSYAS: a framework for audio analysis. *Org. Sound*, 4(3):169–175, December 1999.
- [van den Oord *et al.*, 2013] Aaron van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *NIPS*, 2013.
- [Wang and Wang, 2014] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the ACM International Conference on Multimedia*, MM ’14, pages 627–636, New York, NY, USA, 2014. ACM.
- [Wang *et al.*, 2012a] Xinxi Wang, David Rosenblum, and Ye Wang. Context-aware mobile music recommendation for daily activities. In *Proceedings of the 20th ACM international conference on Multimedia*, MM ’12, pages 99–108, New York, NY, USA, 2012. ACM Press.
- [Wang *et al.*, 2012b] Xinxi Wang, Ye Wang, and David Rosenblum. A daily, activity-aware, mobile music recommender system. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM ’12, pages 1313–1314, New York, NY, USA, 2012. ACM.
- [Wang *et al.*, 2014] Xinxi Wang, Yi Wang, David Hsu, and Ye Wang. Exploration in interactive personalized music recommendation: A reinforcement learning approach. *ACM Trans. Multimedia Comput. Commun. Appl.*, 11(1), September 2014.
- [Wijnalda *et al.*, 2005] Gertjan Wijnalda, Steffen Pauws, Fabio Vignoli, and Heiner Stuckenschmidt. A Personalized Music System for Motivation in Sport Performance. *IEEE Pervasive Computing*, 4:26–32, July 2005.
- [Xing *et al.*, 2014] Zhe Xing, Xinxi Wang, and Ye Wang. Enhancing collaborative filtering music recommendation by balancing exploration and exploitation. In *ISMIR*, 2014.
- [Yoshii *et al.*, 2006] Kazuyoshi Yoshii, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *The International Society for Music Information Retrieval*, pages 296–301, 2006.
- [Zhang and Seo, 2001] Byoung-Tak Zhang and Young-Woo Seo. Personalized Web-Document Filtering Using Reinforcement Learning. In *Proceedings of Applied Artificial Intelligence*, pages 665–685, 2001.

BIBLIOGRAPHY

- [Zhang *et al.*, 2006] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from Incomplete Ratings Using Non-negative Matrix Factorization. In Joydeep Ghosh, Diane Lambert, David B. Skillicorn, Jaideep Srivastava, Joydeep Ghosh, Diane Lambert, David B. Skillicorn, and Jaideep Srivastava, editors, *SDM*. SIAM, 2006.
- [Zhang *et al.*, 2009] Bingjun Zhang, Jialie Shen, Qiaoliang Xiang, and Ye Wang. CompositeMap: a Novel Framework for Music Similarity Measure. *SIGIR*, 2009.
- [Zhang *et al.*, 2012] Yuan C. Zhang, Diarmuid, Daniele Quercia, and Tamas Jambor. Auralist: introducing serendipity into music recommendation. In *Proceedings of the fifth ACM international conference on Web search and data mining*, WSDM '12, pages 13–22. ACM Press, 2012.
- [Zhao *et al.*, 2010a] Wei Zhao, Xinxu Wang, and Ye Wang. Automated sleep quality measurement using EEG signal: First step towards a domain specific music recommendation system. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 1079–1082, New York, NY, USA, 2010. ACM.
- [Zhao *et al.*, 2010b] Zhendong Zhao, Xinxu Wang, Qiaoliang Xiang, Andy M. Sarroff, Zhonghua Li, and Ye Wang. Large-scale music tag recommendation with explicit multiple attributes. In *Proceedings of the International Conference on Multimedia*, MM '10, pages 401–410, New York, NY, USA, 2010. ACM.

Appendices

Appendix A

A.1 Conditional distributions for the approximate Bayesian model

Given N training samples $\mathcal{D} = \{r_i, \mathbf{x}_i, \mathbf{t}_i\}_{i=1}^N$, the conditional distribution $p(\boldsymbol{\theta}|\mathcal{D}, \tau, \boldsymbol{\beta})$ remains a normal distribution:

$$\begin{aligned}
 & p(\boldsymbol{\theta}|\mathcal{D}, \tau, \boldsymbol{\beta}) \\
 & \propto p(\tau)p(\boldsymbol{\theta}|\tau)p(\boldsymbol{\beta}|\tau) \prod_{i=1}^N p(r_i|\mathbf{x}_i, \mathbf{t}_i, \boldsymbol{\theta}, \boldsymbol{\beta}, \tau) \\
 & \propto p(\boldsymbol{\theta}|\tau) \prod_{i=1}^N p(r_i|\mathbf{x}_i, \mathbf{t}_i, \boldsymbol{\theta}, \boldsymbol{\beta}, \tau) \\
 & \propto \exp\left(-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\mu}_{\theta 0})'(\sigma^2\mathbf{D}_0)^{-1}(\boldsymbol{\theta} - \boldsymbol{\mu}_{\theta 0})\right) \exp\left(\sum_{i=1}^N -\frac{1}{2}(r_i - \mathbf{x}'_i\boldsymbol{\theta}\mathbf{t}'_i\boldsymbol{\beta})^2(\sigma^2)^{-1}\right) \\
 & \propto \exp\left[-\frac{1}{2}\boldsymbol{\theta}'\left(\tau\mathbf{D}_0^{-1} + \tau\sum_{i=1}^N \mathbf{x}_i\mathbf{t}'_i\boldsymbol{\beta}\boldsymbol{\beta}'\mathbf{t}_i\mathbf{x}'_i\right)\boldsymbol{\theta} + \tau\left(\boldsymbol{\mu}'_{\theta 0}\mathbf{D}_0^{-1} + \sum_{i=1}^N r_i\boldsymbol{\beta}'\mathbf{t}_i\mathbf{x}'_i\right)\boldsymbol{\theta}\right] \\
 & \propto \exp\left(-\frac{1}{2}\boldsymbol{\theta}'\boldsymbol{\Lambda}_{\theta N}\boldsymbol{\theta} + \boldsymbol{\eta}'_{\theta N}\boldsymbol{\theta}\right)
 \end{aligned}$$

where

$$\begin{aligned}\mathbf{\Lambda}_{\theta N} &= \tau \left(\mathbf{D}_0^{-1} + \sum_{i=1}^N \mathbf{x}_i \mathbf{t}'_i \boldsymbol{\beta} \boldsymbol{\beta}' \mathbf{t}_i \mathbf{x}'_i \right) \\ \boldsymbol{\eta}'_{\theta N} &= \tau \left(\boldsymbol{\mu}'_{\theta 0} \mathbf{D}_0^{-1} + \sum_{i=1}^N r_i \boldsymbol{\beta}' \mathbf{t}_i \mathbf{x}'_i \right)\end{aligned}$$

Due to the symmetry between $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$, we can easily obtain

$$p(\boldsymbol{\beta} | \mathcal{D}, \tau, \boldsymbol{\theta}) \propto \exp \left(-\frac{1}{2} \boldsymbol{\beta}' \boldsymbol{\Lambda}_{\beta N} \boldsymbol{\beta} + \boldsymbol{\eta}'_{\beta N} \boldsymbol{\beta} \right),$$

where

$$\begin{aligned}\mathbf{\Lambda}_{\beta N} &= \tau \left(\mathbf{E}_0^{-1} + \sum_{i=1}^N \mathbf{t}_i \mathbf{x}'_i \boldsymbol{\theta} \boldsymbol{\theta}' \mathbf{x}_i \mathbf{t}'_i \right) \\ \boldsymbol{\eta}'_{\beta N} &= \tau \left(\boldsymbol{\mu}'_{\beta 0} \mathbf{E}_0^{-1} + \sum_{i=1}^N r_i \boldsymbol{\theta}' \mathbf{x}_i \mathbf{t}'_i \right).\end{aligned}$$

The conditional distribution $p(\tau | \mathcal{D}, \boldsymbol{\theta}, \boldsymbol{\beta})$ also remains a Gamma distribu-

tion:

$$\begin{aligned}
p(\tau|\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{\beta}) &\propto \tau^{a_N-1} \exp(-b_N\tau) \\
p(\tau|\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{\beta}) &\propto p(\tau)p(\boldsymbol{\theta}|\tau)p(\boldsymbol{\beta}|\tau) \prod_{i=1}^N p(r_i|\mathbf{x}_i, \mathbf{t}_i, \boldsymbol{\theta}, \boldsymbol{\beta}, \tau) \\
&= b_0^{a_0} \frac{1}{\Gamma(a_0)} \tau^{a_0-1} \exp(-b_0\tau) \times \\
&\quad \frac{1}{(2\pi)^{p/2} |\sigma^2 \mathbf{D}_0|^{1/2}} \exp\left(-\frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\mu}_{\theta_0})' (\sigma^2 \mathbf{D}_0)^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}_{\theta_0})\right) \\
&\quad \times \frac{1}{(2\pi)^{K/2} |\sigma^2 \mathbf{E}_0|^{1/2}} \exp\left(-\frac{1}{2} (\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta_0})' (\sigma^2 \mathbf{E}_0)^{-1} (\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta_0})\right) \\
&\quad \times \left(\frac{1}{(2\pi)^{1/2} |\sigma^2|^{1/2}}\right)^N \exp\left(\sum_{i=1}^N -\frac{1}{2} (r_i - \mathbf{x}_i' \boldsymbol{\theta} \mathbf{t}_i' \boldsymbol{\beta})^2 (\sigma^2)^{-1}\right) \\
&\propto \tau^{a_N-1} \exp(-b_N\tau)
\end{aligned}$$

where a_N and b_N are the parameters of the Gamma distribution, and they are

$$\begin{aligned}
a_N &= \frac{p + K + N}{2} + a_0 \\
b_N &= b_0 + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\mu}_{\theta_0})' \mathbf{D}_0^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}_{\theta_0}) + \frac{1}{2} (\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta_0})' \mathbf{E}_0^{-1} (\boldsymbol{\beta} - \boldsymbol{\mu}_{\beta_0}) \\
&\quad + \frac{1}{2} \sum_{i=1}^N (r_i - \mathbf{x}_i' \boldsymbol{\theta} \mathbf{t}_i' \boldsymbol{\beta})^2
\end{aligned}$$

A.2 Variational inference

To calculate the joint posterior distribution $p(\boldsymbol{\theta}, \tau, \boldsymbol{\beta}|\mathcal{D})$, we can use Gibbs sampling based on the conditional distributions. However, this is slow too, and therefore, we resort to variational inference (mean field approximation specifically). We assume that $p(\boldsymbol{\theta}, \tau, \boldsymbol{\beta}|\mathcal{D}) \approx q(\boldsymbol{\theta}, \boldsymbol{\beta}, \tau) = q(\boldsymbol{\theta})q(\boldsymbol{\beta})q(\tau)$. In the restricted distribution $q(\boldsymbol{\theta}, \boldsymbol{\beta}, \tau)$, every variable is assumed independent from

the other variables. Because all the conditional distributions $p(\boldsymbol{\theta}|\mathcal{D}, \tau, \boldsymbol{\beta})$, $p(\tau|\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{\beta})$, and $p(\boldsymbol{\beta}|\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{\beta})$ are in the exponential families, their restricted distributions $q(\boldsymbol{\theta})$, $q(\boldsymbol{\beta})$, $q(\tau)$ lie in the same exponential families as their conditional distributions. We then obtain the restricted distributions and update rules as in Section 5.3.2.2.

The expectation of b_N with respect to $q(\boldsymbol{\theta})$ and $q(\boldsymbol{\beta})$ might be a bit tricky to derive. We thus show it as the following:

$$\begin{aligned}
b_N &= b_0 + \frac{1}{2} \mathbb{E} [(\boldsymbol{\theta} - \boldsymbol{\mu}_{\boldsymbol{\theta}})' \mathbf{D}_0^{-1} (\boldsymbol{\theta} - \boldsymbol{\mu}_{\boldsymbol{\theta}})] + \frac{1}{2} \mathbb{E} [(\boldsymbol{\beta} - \boldsymbol{\mu}_{\boldsymbol{\beta}})' \mathbf{E}_0^{-1} (\boldsymbol{\beta} - \boldsymbol{\mu}_{\boldsymbol{\beta}})] \\
&\quad + \frac{1}{2} \mathbb{E} \left[\sum_{i=1}^N (r_i - \mathbf{x}'_i \boldsymbol{\theta} \mathbf{t}'_i \boldsymbol{\beta})^2 \right] \\
&= b_0 + \frac{1}{2} [\text{tr} [\mathbf{D}_0^{-1} (\mathbb{E}[\boldsymbol{\theta} \boldsymbol{\theta}'])] + (\boldsymbol{\mu}'_{\boldsymbol{\theta}} - 2\mathbb{E}[\boldsymbol{\theta}']) \mathbf{D}_0^{-1} \boldsymbol{\mu}_{\boldsymbol{\theta}}] \\
&\quad + \frac{1}{2} [\text{tr} [\mathbf{E}_0^{-1} (\mathbb{E}[\boldsymbol{\beta} \boldsymbol{\beta}'])] + (\boldsymbol{\mu}'_{\boldsymbol{\beta}} - 2\mathbb{E}[\boldsymbol{\beta}']) \mathbf{E}_0^{-1} \boldsymbol{\mu}_{\boldsymbol{\beta}}] + \frac{1}{2} \sum_{i=1}^N \mathbb{E} [(r_i - \mathbf{x}'_i \boldsymbol{\theta} \mathbf{t}'_i \boldsymbol{\beta})^2].
\end{aligned}$$

Since $\boldsymbol{\theta}$ and $\boldsymbol{\beta}$ are assumed independent, we have

$$\begin{aligned}
\mathbb{E} [(r_i - \mathbf{x}'_i \boldsymbol{\theta} \mathbf{t}'_i \boldsymbol{\beta})^2] &= \mathbb{E} [r_i^2 - 2r_i \mathbf{x}'_i \boldsymbol{\theta} \mathbf{t}'_i \boldsymbol{\beta} + \mathbf{x}'_i \boldsymbol{\theta} \mathbf{t}'_i \boldsymbol{\beta} \mathbf{x}'_i \boldsymbol{\theta} \mathbf{t}'_i \boldsymbol{\beta}] \\
&= r_i^2 - 2r_i \mathbf{x}'_i \mathbb{E}[\boldsymbol{\theta}] \mathbf{t}'_i \mathbb{E}[\boldsymbol{\beta}] + \mathbf{x}'_i \mathbb{E}[\boldsymbol{\theta} \boldsymbol{\theta}'] \mathbf{x}_i \mathbf{t}'_i \mathbb{E}[\boldsymbol{\beta} \boldsymbol{\beta}'] \mathbf{t}_i.
\end{aligned}$$

Therefore b_N can be calculated as

$$\begin{aligned}
 b_N &= b_0 + \frac{1}{2} \left[\text{tr} [\mathbf{D}_0^{-1} (\mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}'])] + (\boldsymbol{\mu}'_{\boldsymbol{\theta}_0} - 2\mathbb{E}[\boldsymbol{\theta}']) \mathbf{D}_0^{-1} \boldsymbol{\mu}_{\boldsymbol{\theta}_0} \right] \\
 &+ \frac{1}{2} \left[\text{tr} [\mathbf{E}_0^{-1} (\mathbb{E}[\boldsymbol{\beta}\boldsymbol{\beta}'])] + (\boldsymbol{\mu}'_{\boldsymbol{\beta}_0} - 2\mathbb{E}[\boldsymbol{\beta}']) \mathbf{E}_0^{-1} \boldsymbol{\mu}_{\boldsymbol{\beta}_0} \right] \\
 &+ \frac{1}{2} \left[\sum_{i=1}^N r_i^2 - 2r_i \mathbf{x}'_i \mathbb{E}[\boldsymbol{\theta}] \mathbf{t}'_i \mathbb{E}[\boldsymbol{\beta}] + \mathbf{x}'_i \mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}'] \mathbf{x}_i \mathbf{t}'_i \mathbb{E}[\boldsymbol{\beta}\boldsymbol{\beta}'] \mathbf{t}_i \right].
 \end{aligned}$$

The moments of $\boldsymbol{\theta}$, $\boldsymbol{\beta}$, and τ :

$$\mathbb{E}[\boldsymbol{\beta}\boldsymbol{\beta}'] = \boldsymbol{\Lambda}_{\boldsymbol{\beta}N}^{-1} + \mathbb{E}[\boldsymbol{\beta}]\mathbb{E}[\boldsymbol{\beta}']$$

$$\mathbb{E}[\boldsymbol{\beta}] = \boldsymbol{\Lambda}_{\boldsymbol{\beta}N}^{-1} \boldsymbol{\eta}_{\boldsymbol{\beta}N}$$

$$\mathbb{E}[\boldsymbol{\theta}\boldsymbol{\theta}'] = \boldsymbol{\Lambda}_{\boldsymbol{\theta}N}^{-1} + \mathbb{E}[\boldsymbol{\theta}]\mathbb{E}[\boldsymbol{\theta}']$$

$$\mathbb{E}[\boldsymbol{\theta}] = \boldsymbol{\Lambda}_{\boldsymbol{\theta}N}^{-1} \boldsymbol{\eta}_{\boldsymbol{\theta}N}$$

$$\mathbb{E}[\tau] = \frac{a_N}{b_N}$$

A.3 Variational lower bound

The following is the variational lower bound, where $\psi(\cdot)$ is the digamma function.

$$\begin{aligned}
\mathcal{L} &= \mathbb{E}[\ln(\mathcal{D}, \tau, \boldsymbol{\theta}, \boldsymbol{\beta})] - \mathbb{E}[\ln q(\boldsymbol{\theta}, \tau, \boldsymbol{\beta})] \\
&= \mathbb{E}[\ln p(\tau)] + \mathbb{E}[\ln p(\boldsymbol{\theta}|\tau)] + \mathbb{E}[\ln p(\boldsymbol{\beta}|\tau)] + \sum_{i=1}^N \mathbb{E}[\ln p(r_i|\mathbf{x}_i, \mathbf{t}_i, \boldsymbol{\theta}, \boldsymbol{\beta}, \tau)] \\
&\quad - \mathbb{E}[\ln q(\boldsymbol{\theta})] - \mathbb{E}[\ln q(\boldsymbol{\beta})] - \mathbb{E}[\ln q(\tau)] \\
&= a_0 \ln b_0 + (a_0 - 1) [\psi(a_N) - \ln b_N] - b_0 \frac{a_N}{b_N} - \frac{p}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{D}_0| + \frac{p}{2} (\psi(a_N) - \ln(b_N)) \\
&\quad - \frac{a_N}{2b_N} [\text{tr}(\mathbf{D}_0 \boldsymbol{\Lambda}_{\boldsymbol{\theta}N}^{-1}) + (\boldsymbol{\mu}_{\boldsymbol{\theta}0} - \mathbb{E}[\boldsymbol{\theta}])' \mathbf{D}_0^{-1} (\boldsymbol{\mu}_{\boldsymbol{\theta}0} - \mathbb{E}[\boldsymbol{\theta}])] - \frac{K}{2} \ln(2\pi) \\
&\quad - \frac{1}{2} \ln |\mathbf{E}_0| + \frac{K}{2} (\psi(a_N) - \ln(b_N)) \\
&\quad - \frac{a_N}{2b_N} [\text{tr}(\mathbf{E}_0 \boldsymbol{\Lambda}_{\boldsymbol{\beta}N}^{-1}) + (\boldsymbol{\mu}_{\boldsymbol{\beta}0} - \mathbb{E}[\boldsymbol{\beta}])' \mathbf{E}_0^{-1} (\boldsymbol{\mu}_{\boldsymbol{\beta}0} - \mathbb{E}[\boldsymbol{\beta}])] - \frac{1}{2} \ln(2\pi) + \frac{1}{2} (\psi(a_N) - \ln b_N) \\
&\quad - \frac{a_N}{2b_N} \sum_{i=1}^N (r_i^2 + \mathbf{x}_i' \mathbb{E}[\boldsymbol{\theta} \boldsymbol{\theta}'] \mathbf{x}_i + \mathbf{t}_i' \mathbb{E}[\boldsymbol{\beta} \boldsymbol{\beta}'] \mathbf{t}_i) + \frac{a_N}{b_N} \sum_{i=1}^N r_i \mathbf{x}_i' \mathbb{E}[\boldsymbol{\theta}] \mathbf{t}_i' \mathbb{E}[\boldsymbol{\beta}] \\
&\quad + \frac{K}{2} [1 + \ln(2\pi)] + \frac{1}{2} \ln |\boldsymbol{\Lambda}_{\boldsymbol{\beta}N}^{-1}| + \frac{p}{2} [1 + \ln(2\pi)] \\
&\quad + \frac{1}{2} \ln |\boldsymbol{\Lambda}_{\boldsymbol{\theta}N}^{-1}| - (a_N - 1) \psi(a_N) - \ln b_N + a_N
\end{aligned}$$

It might be a bit tricky to derive

$$\mathbb{E}[\ln p(\boldsymbol{\theta}|\tau)] = \iint p(\boldsymbol{\theta}|\tau) q(\boldsymbol{\theta}) d\boldsymbol{\theta} q(\tau) d\tau$$

which is part of the lower bound \mathcal{L} . We assume that $P = p(\boldsymbol{\theta}|\tau)$, and $Q = q(\boldsymbol{\theta})$, and we have $\int p(\boldsymbol{\theta}|\tau) q(\boldsymbol{\theta}) d\boldsymbol{\theta} = -H(Q, P)$, where $H(Q, P)$ is the cross entropy between Q and P . Given Q and P are multivariate normal distributions, the

KL-divergence between Q and P and the entropy of Q are

$$\begin{aligned}
 D_{KL}(Q||P) &= \frac{1}{2} \left[\text{tr}(\Sigma_P^{-1}\Sigma_Q) + (\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q)' \Sigma_P^{-1} (\boldsymbol{\mu}_P - \boldsymbol{\mu}_Q) - \ln \frac{|\Sigma_Q|}{|\Sigma_P|} - p \right] \\
 &= \frac{1}{2} \left[\text{tr}(\tau \mathbf{D}_0 \boldsymbol{\Lambda}_{\boldsymbol{\theta}_N}^{-1}) + (\boldsymbol{\mu}_{\boldsymbol{\theta}_0} - \boldsymbol{\mu}_{\boldsymbol{\theta}_N})' \tau \mathbf{D}_0^{-1} (\boldsymbol{\mu}_{\boldsymbol{\theta}_0} - \boldsymbol{\mu}_{\boldsymbol{\theta}_N}) - \ln |\boldsymbol{\Lambda}_{\boldsymbol{\theta}_N}^{-1}| + \ln \left| \frac{1}{\tau} \mathbf{D}_0 \right| - p \right] \\
 H(Q) &= \frac{1}{2} (p + p \ln(2\pi) + \ln |\boldsymbol{\Lambda}_{\boldsymbol{\theta}_N}^{-1}|).
 \end{aligned}$$

Therefore

$$\begin{aligned}
 \int p(\boldsymbol{\theta}|\tau)q(\boldsymbol{\theta})d\boldsymbol{\theta} &= -H(Q, P) \\
 &= -H(Q) - D_{KL}(Q||P) \\
 &= -\frac{p}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{D}_0| + \frac{p}{2} \ln \tau \\
 &\quad - \frac{1}{2} \left[\text{tr}(\tau \mathbf{D}_0 \boldsymbol{\Lambda}_{\boldsymbol{\theta}_N}^{-1}) + (\boldsymbol{\mu}_{\boldsymbol{\theta}_0} - \boldsymbol{\mu}_{\boldsymbol{\theta}_N})' \tau \mathbf{D}_0^{-1} (\boldsymbol{\mu}_{\boldsymbol{\theta}_0} - \boldsymbol{\mu}_{\boldsymbol{\theta}_N}) \right],
 \end{aligned}$$

and

$$\begin{aligned}
 \mathbb{E}[\ln p(\boldsymbol{\theta}|\tau)] &= \iint p(\boldsymbol{\theta}|\tau)q(\boldsymbol{\theta})d\boldsymbol{\theta}q(\tau)d\tau \\
 &= -\frac{p}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{D}_0| + \frac{p}{2} \mathbb{E}[\ln \tau] \\
 &\quad - \frac{1}{2} \left[\text{tr}(\mathbf{D}_0 \boldsymbol{\Lambda}_{\boldsymbol{\theta}_N}^{-1}) + (\boldsymbol{\mu}_{\boldsymbol{\theta}_0} - \boldsymbol{\mu}_{\boldsymbol{\theta}_N})' \mathbf{D}_0^{-1} (\boldsymbol{\mu}_{\boldsymbol{\theta}_0} - \boldsymbol{\mu}_{\boldsymbol{\theta}_N}) \right] \mathbb{E}[\tau] \\
 &= -\frac{p}{2} \ln(2\pi) - \frac{1}{2} \ln |\mathbf{D}_0| + \frac{p}{2} (\psi(a_N) - \ln(b_N)) \\
 &\quad - \frac{a_N}{2b_N} \left[\text{tr}(\mathbf{D}_0 \boldsymbol{\Lambda}_{\boldsymbol{\theta}_N}^{-1}) + (\boldsymbol{\mu}_{\boldsymbol{\theta}_0} - \boldsymbol{\mu}_{\boldsymbol{\theta}_N})' \mathbf{D}_0^{-1} (\boldsymbol{\mu}_{\boldsymbol{\theta}_0} - \boldsymbol{\mu}_{\boldsymbol{\theta}_N}) \right]
 \end{aligned}$$