ENHANCING COLLABORATIVE FILTERING MUSIC RECOMMENDATION BY BALANCING EXPLORATION AND EXPLOITATION

Zhe Xing, Xinxi Wang, Ye Wang
School of Computing, National University of Singapore
{xing-zhe,wangxinxi,wangye}@comp.nus.edu.sg

ABSTRACT

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

1. INTRODUCTION

In the field of music recommendation, content-based approaches and collaborative filtering (CF) approaches have been the prevailing recommendation strategies. Content-based algorithms [1, 9] analyze acoustic features of the songs that the user has rated highly in the past and recommend only songs that have high degrees of acoustic similarity. On the other hand, collaborative filtering (CF) algorithms [7, 13] assume that people tend to get good recommendations from someone with similar preferences, and the user's ratings are predicted according to his neighbors' ratings. These two traditional recommendation approaches, however, share a weakness.

Working in a greedy way, they always generate "safe" recommendations by selecting songs with the highest predicted user ratings. Such a purely *exploitative* strategy may result in suboptimal performance over the long term due to the lack of *exploration*. The reason is that user preference is only estimated based on the current knowledge available in the recommender system. As a result, uncertainty always exists in the predicted user ratings and may give rise to a situation where some of the non-greedy options deemed almost as good as the greedy ones are actually better than them. Without exploration, however, we will never know which ones are better. With the appropriate amount of *exploration*, the recommender system could gain more knowledge about the user's true preferences before *exploiting* them.

Our previous work [12] tried to mitigate the greedy problem in content-based music recommendation, but no work has addressed this problem in the CF context. We thus aim to develop a CF-based music recommendation algorithm that can strike a balance between exploration and exploitation and enhance long-term recommendation performance. To do so, we introduce exploration into collaborative filtering by formulating the music recommendation problem as a reinforcement learning task called n-*armed bandit* problem. A Bayesian graphical model taking account of both collaborative filtering latent factors and recommendation novelty is proposed to learn the user preferences. The lack of efficiency becomes a major challenge, however, when we adopt an off-the-shelf Markov Chain Monte Carlo (MCMC) sampling algorithm for the Bayesian posterior estimation. We are thus prompted to design a much faster sampling algorithm for Bayesian inference. We carried out both simulation experiments and a user study to show the efficiency and effectiveness of the proposed approach. Contributions of this paper are summarized as follows:

- To the best of our knowledge, this is the first work in music recommendation to temper CF's greedy nature by investigating the exploration-exploitation trade-off using a reinforcement learning approach.
- Compared to an off-the-shelf MCMC algorithm, a much more efficient sampling algorithm is proposed to speed up Bayesian posterior estimation.
- Experimental results show that our proposed approach enhances the performance of CF-based music recommendation significantly.

2. RELATED WORK

Based on the assumption that people tend to receive good recommendations from others with similar preferences, col-

laborative filtering (CF) techniques come in two categories: memory-based CF and model-based CF. Memory-based CF algorithms [3, 8] first search for neighbors who have similar rating histories to the target user. Then the target user's ratings can be predicted according to his neighbors' ratings. Model-based CF algorithms [7, 14] use various models and machine learning techniques to discover latent factors that account for the observed ratings.

Our previous work [12] proposed a reinforcement learning approach to balance exploration and exploitation in music recommendation. However, this work is based on a content-based approach. One major drawback of the personalized user rating model is that low-level audio features are used to represent the content of songs. This purely content-based approach is not satisfactory due to the semantic gap between low-level audio features and high-level user preferences. Moreover, it is difficult to determine which underlying acoustic features are effective in music recommendation scenarios, as these features were not originally designed for music recommendation. Another shortcoming is that songs recommended by content-based methods often lack variety, because they are all acoustically similar to each other. Ideally, users should be provided with a range of genres rather than a homogeneous set.

While no work has attempted to address the greedy problem of CF approaches in the music recommendation context, Karimi et al. tried to investigate it in other recommendation applications [4, 5]. However, their active learning approach merely explores items to optimize the prediction accuracy on a pre-determined test set [4]. No attention is paid to the exploration-exploitation trade-off problem. In their other work, the recommendation process is split into two steps [5]. In the exploration step, they select an item that brings maximum change to the user parameters, and then in the exploitation step, they pick the item based on the current parameters. The work takes balancing exploration and exploitation into consideration, but only in an ad hoc way. In addition, their approach is evaluated using only an offline and pre-determined dataset. In the end, their algorithm is not practical for deployment in online recommender systems due to its low efficiency.

3. PROPOSED APPROACH

We first present a simple matrix factorization model for collaborative filtering (CF) music recommendation. Then, we point out major limitations of this traditional CF algorithm and describe our proposed approach in detail.

3.1 Matrix Factorization for Collaborative Filtering

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

Matrix factorization characterizes users and songs by vectors of latent factors. Every user is associated with a user feature vector $\mathbf{u}_i \in \mathbb{R}^f, i = 1, 2, ..., m$, and every song a song feature vector $\mathbf{v}_j \in \mathbb{R}^f, j = 1, 2, ..., n$. For a given song j, $\mathbf{v}_j$ measures the extent to which the song contains the latent factors. For a given user i, $\mathbf{u}_i$ measures the extent to which he likes these latent factors. The user rating can thus be approximated by the inner product of the two vectors:

$$\hat{r}_{ij} = \mathbf{u}_i^T \mathbf{v}_j \tag{1}$$

To learn the latent feature vectors, the system minimizes the following regularized squared error on the training set:

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

where I is the index set of all known ratings, λ a regularization parameter, n_{u_i} the number of ratings by user i, and n_{v_j} the number of ratings of song j. We use the *alternating least squares* (ALS) [14] technique to minimize Eq. (2).

However, this traditional CF recommendation approach has two major drawbacks. (I) It fails to take recommendation *novelty* into consideration. For a user, the novelty of a song changes with each listening. (II) It works *greedily*, always recommending songs with the highest predicted mean ratings, while a better approach may be to actively explore a user's preferences rather than to merely exploit available rating information [12]. To address these drawbacks, we propose a reinforcement learning approach to CF-based music recommendation.

3.2 A Reinforcement Learning Approach

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

The n-armed bandit problem assumes a slot machine with n levers. Pulling a lever generates a payoff from the unknown probability distribution of the lever. The objective is to maximize the expected total payoff over a given number of action selections, say, over 1000 plays.

3.2.1 Modeling User Rating

To address drawback (I) in Section 3.1, we assume that a song's rating is affected by two factors: CF score, the extent to which the user likes the song in terms of each CF latent factor, and novelty score, the dynamically changing novelty of the song.

From Eq. (1), we define the CF score as:

$$U_{CF} = \boldsymbol{\theta}^T \mathbf{v} \tag{3}$$

where vector $\boldsymbol{\theta}$ indicates the user's preferences for different CF latent factors and $\mathbf{v}$ is the song feature vector

learned by the ALS CF algorithm. For the novelty score, we adopt the formula used in [12]:

$$U_N = 1 - e^{-t/s} \tag{4}$$

where t is the time elapsed since when the song was last heard, s the relative strength of the user's memory, and $e^{-t/s}$ the well-known forgetting curve. The formula assumes that a song's novelty decreases immediately when listened and gradually recovers with time. (For more details on the novelty definition, please refer to [12].) We thus model the final user rating by combining these two scores:

$$U = U_{CF}U_N = (\boldsymbol{\theta}^T\mathbf{v})(1 - e^{-t/s}) \tag{5}$$

Given the variability in musical taste and memory strength, each user is associated with a pair of parameters $\boldsymbol{\Omega} = (\boldsymbol{\theta}, s)$, to be learned from the user's rating history. More technical details will be explained in Section 3.2.2.

Since the predicted user ratings always carry uncertainty, we assume them to be random variables rather than fixed numbers. Let R_j denote the rating of song j given by the target user, and R_j follows an unknown probability distribution. We assume that the expectation of R_j is the U_j defined in Eq. (5). Thus, the expected rating of song j can be estimated as:

$$\mathbb{E}[R_j] = U_j = (\boldsymbol{\theta}^T\mathbf{v}_j)(1 - e^{-t_j/s}) \tag{6}$$

Traditional recommendation strategy will first obtain the $\mathbf{v}_j$ and t_j of each song in the system to compute the expected rating using Eq. (6) and then recommend the song with the highest expected rating. We call this a *greedy* recommendation as the system is *exploiting* its current knowledge of the user ratings. By selecting one of the non-greedy recommendations and gathering more user feedback, the system *explores* further and gains more knowledge about the user preferences. A greedy recommendation may maximize the expected reward in the current iteration but would result in suboptimal performance over the long term. This is because several non-greedy recommendations may be deemed nearly as good but come with substantial *variance* (or uncertainty), and it is thus possible that some of them are actually better than the greedy recommendation. Without exploration, however, we will never know which ones they are.

Therefore, to counter the greedy nature of CF (drawback II), we introduce exploration into music recommendation to balance exploitation. To do so, we adopt one of the state-of-the-art algorithms called Bayesian Upper Confidence Bounds (Bayes-UCB) [6]. In Bayes-UCB, the expected reward U_j is a random variable rather than a fixed value. Given the target user's rating history $\mathcal{D}$, the posterior distribution of U_j, denoted as $p(U_j|\mathcal{D})$, needs to be estimated. Then the song with the highest fixed-level quantile value of $p(U_j|\mathcal{D})$ will be recommended to the target user.

3.2.2 Bayesian Graphical Model

To estimate the posterior distribution of U, we adopt the Bayesian model (Figure 1) used in [12]. The corresponding probability dependency is defined as follows:

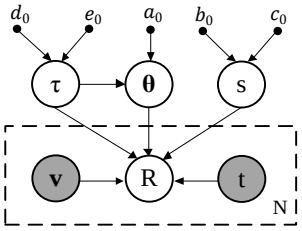

Figure 1*:* Bayesian Graphical Model.

$$R|\mathbf{v}, t, \boldsymbol{\theta}, s, \sigma^2 \sim \mathcal{N}(\boldsymbol{\theta}^T\mathbf{v}(1 - e^{-t/s}), \sigma^2) \tag{7}$$

$$\boldsymbol{\theta}|\sigma^2 \sim \mathcal{N}(\mathbf{0}, a_0\sigma^2\mathbf{I}) \tag{8}$$

$$s \sim Gamma(b_0, c_0) \tag{9}$$

$$\tau = 1/\sigma^2 \sim Gamma(d_0, e_0) \tag{10}$$

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

At current iteration $h+1$, we have gathered h observed recommendation history $\mathcal{D}_h = \{(\mathbf{v}_i, t_i, r_i)\}_{i=1}^h$. Given that each user in our model is described as $\boldsymbol{\Omega} = (\boldsymbol{\theta}, s)$, we have according to the Bayes theorem:

$$p(\boldsymbol{\Omega} \mid \mathcal{D}_h) \propto p(\boldsymbol{\Omega})p(\mathcal{D}_h \mid \boldsymbol{\Omega}) \tag{11}$$

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

$$p(U_j|\mathcal{D}_h) = \int p(U_j|\boldsymbol{\Omega})p(\boldsymbol{\Omega}|\mathcal{D}_h)d\boldsymbol{\Omega} \tag{12}$$

Since Eq. (11) has no closed form solution, we are unable to directly estimate the posterior PDF in Eq. (12). We thus turn to a Markov Chain Monte Carlo (MCMC) algorithm to adequately sample the parameters $\boldsymbol{\Omega} = (\boldsymbol{\theta}, s)$. We then substitute every parameter sample into Eq. (6) to obtain a sample of U_j. Finally, the posterior PDF in Eq. (12) can be approximated by the histogram of the samples of U_j.

After estimating the posterior PDF of each song's expected rating, we follow the Bayes-UCB approach [6] to recommend song j^* that maximizes the quantile function:

$$j^* = \arg\max_{j=1,\ldots,|S|} Q\left(\alpha, p(U_j|\mathcal{D}_h)\right) \tag{13}$$

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

3.3 Efficient Sampling Algorithm

Bayesian inference is very slow with an off-the-shelf MCMC sampling algorithm because it takes a long time for the Markov chain to converge. In response, we previously proposed an approximate Bayesian model using piecewise linear approximation [12]. However, not only is the original

Algorithm 1 Exploration-Exploitation Balanced Music Recommendation

for $h = 1 \rightarrow N$ **do**
  if $h == 1$ **then**
    Recommend a song randomly;
  else
    Draw samples of $\boldsymbol{\theta}$ and s based on $p(\boldsymbol{\Omega} \mid \mathcal{D}_{h-1})$;
    for song $j = 1 \rightarrow |S|$ **do**
      Obtain $\mathbf{v}_j$ and t_j of song j and compute samples of U_j using Eq. (6);
      Estimate $p(U_j|\mathcal{D}_{h-1})$ using histogram of the samples of U_j;
      Compute quantile $q_j^h = Q\left(1 - \frac{1}{h}, p(U_j|\mathcal{D}_{h-1})\right)$;
    end for
    Recommend song $j^* = \arg\max_{j=1,\ldots,|S|} q_j^h$;
    Collect user rating r_h and update $p(\boldsymbol{\Omega} \mid \mathcal{D}_h)$;
  end if
end for

Bayesian model altered, tuning the numerous (hyper)parameters is also tedious. In this paper, we present a better way to improve efficiency. Since it is simple to sample from a conditional distribution, we develop a specific Gibbs sampling algorithm to hasten convergence.

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

$$p(\boldsymbol{\theta}|\mathcal{D}, \tau, s) \propto p(\tau)p(\boldsymbol{\theta}|\tau)p(s)\prod_{i=1}^{N} p(r_i|\mathbf{v}_i, t_i, \boldsymbol{\theta}, s, \tau)$$

$$\propto p(\boldsymbol{\theta}|\tau)\prod_{i=1}^{N} p(r_i|\mathbf{v}_i, t_i, \boldsymbol{\theta}, s, \tau) \propto \exp\left(-\frac{1}{2}\boldsymbol{\theta}^T(a_0\sigma^2\mathbf{I})^{-1}\boldsymbol{\theta}\right)$$

$$\times\exp\left(\sum_{i=1}^{N} -\frac{1}{2\sigma^2}\left(r_i - \boldsymbol{\theta}^T\mathbf{v}_i(1 - e^{-t_i/s})\right)^2\right)$$

$$\propto \exp\left(-\frac{1}{2}\boldsymbol{\theta}^T\boldsymbol{\Lambda}\boldsymbol{\theta} + \boldsymbol{\eta}^T\boldsymbol{\theta}\right) \propto \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad (14)$$

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

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Lambda} = \tau\left(\frac{1}{a_0}\mathbf{I} + \sum_{i=1}^{N}(1 - e^{-t_i/s})^2\mathbf{v}_i\mathbf{v}_i^T\right) \quad (15)$$

$$\boldsymbol{\mu}^T\boldsymbol{\Sigma}^{-1} = \boldsymbol{\eta}^T = \tau\left(\sum_{i=1}^{N} r_i(1 - e^{-t_i/s})\mathbf{v}_i^T\right) \quad (16)$$

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

$$p(\tau|\mathcal{D}, \boldsymbol{\theta}, s) \propto p(\tau)p(\boldsymbol{\theta}|\tau)p(s)\prod_{i=1}^{N} p(r_i|\mathbf{v}_i, t_i, \boldsymbol{\theta}, s, \tau)$$

$$\propto p(\tau)p(\boldsymbol{\theta}|\tau)\prod_{i=1}^{N} p(r_i|\mathbf{v}_i, t_i, \boldsymbol{\theta}, s, \tau)$$

$$\propto \tau^{d_0-1}\exp(-e_0\tau) \times \exp\left(-\frac{1}{2}\boldsymbol{\theta}^T(a_0\sigma^2\mathbf{I})^{-1}\boldsymbol{\theta}\right) \times$$

$$\left(\sigma\sqrt{2\pi}\right)^{-N}\exp\left(\sum_{i=1}^{N} -\frac{1}{2\sigma^2}\left(r_i - \boldsymbol{\theta}^T\mathbf{v}_i(1 - e^{-t_i/s})\right)^2\right)$$

$$\propto \tau^{\alpha-1}\exp(-\beta\tau) \propto Gamma(\alpha, \beta) \quad (17)$$

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

Table 1: Size of the dataset. *Density* is the percentage of entries in the user-song matrix that have observations.

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

$$\alpha = d_0 + \frac{f + N}{2} \quad (18)$$

$$\beta = e_0 + \frac{\boldsymbol{\theta}^T\boldsymbol{\theta}}{2a_0} + \frac{1}{2}\sum_{i=1}^{N}\left(r_i - \boldsymbol{\theta}^T\mathbf{v}_i(1 - e^{-t_i/s})\right)^2 \quad (19)$$

The conditional distribution $p(s|\mathcal{D}, \boldsymbol{\theta}, \tau)$ has no closed form expression. We thus adopt the Metropolis-Hastings (MH) algorithm [2] with a proposal distribution $q(s_{t+1}|s_t) = \mathcal{N}(s_t, 1)$ to draw samples of s. Our detailed Gibbs sampling process is presented in Algorithm 2.

Algorithm 2 Gibbs Sampling for Bayesian Inference

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

4. EVALUATION

4.1 Dataset

The Taste Profile Subset [1] used in the Million Song Dataset Challenge [10] has over 48 million triplets (user, song, count) describing the listening history of over 1 million users and 380,000 songs. We select 20,000 songs with top listening counts and 100,000 users who have listened to the most songs. Since this collection of listening history is a form of implicit feedback data, we use the approach proposed in [11] to perform negative sampling. The detailed statistics of the final dataset are shown in Table 1.

4.2 Learning CF Latent Factors

First, we determine the optimal value of λ, the regularization parameter, and f, the dimensionality of the latent feature vectors. We randomly split the dataset into three disjoint parts: training set (80%), validation set (10%), and test set (10%). Training set is used to learn the CF latent factors, and the convergence criteria of the ALS algorithm is achieved when the change in root mean square

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

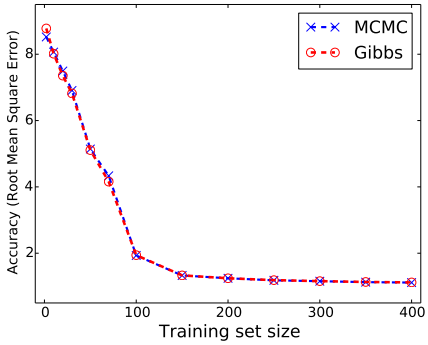

Figure 2: Prediction accuracy of sampling algorithms.

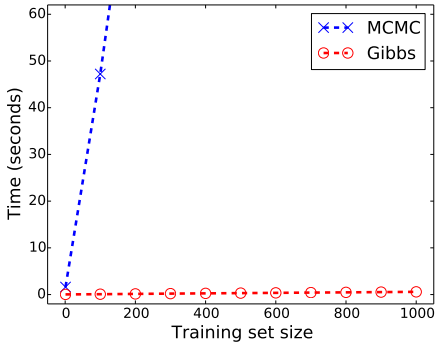

Figure 3: Efficiency comparison of sampling algorithms. ($Time_{MCMC} = 538.762s$ and $Time_{Gibbs} = 0.579s$ when $TrainingSetSize = 1000$).

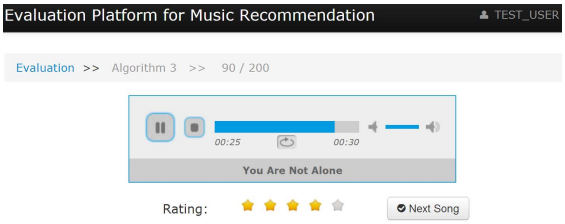

Figure 4: Online evaluation platform.

error (RMSE) on the validation set is less than 10^{-4}. Then we use the learned latent factors to predict the ratings on the test set. We first fix $f = 55$ and vary λ from 0.005 to 0.1; minimal RMSE is achieved at $\lambda = 0.025$. We then fix $\lambda = 0.025$ and vary f from 10 to 80, and $f = 75$ yields minimal RMSE. Therefore, we adopt the optimal value $\lambda = 0.025$ and $f = 75$ to perform the final ALS CF algorithm and obtain the learned latent feature vector of each song in our dataset. These vectors will later be used for reinforcement learning.

4.3 Efficiency Study

To show that our Gibbs sampling algorithm makes Bayesian inference significantly more efficient, we conduct simulation experiments to compare it with an off-the-shelf MCMC algorithm developed in JAGS [2] . We implemented the Gibbs algorithm in C++, which JAGS uses, for a fair comparison.

For each data point $d_i \in \{(\mathbf{v}_i, t_i, r_i)\}_{i=1}^n$ in the simulation experiments, $\mathbf{v}_i$ is randomly chosen from the latent feature vectors learned by the ALS CF algorithm. t_i is randomly sampled from $uniform(50, 2592000)$, i.e. between a time gap of 50 seconds and one month. r_i is calculated using Eq. (6) where elements of $\boldsymbol{\theta}$ are sampled from $\mathcal{N}(0, 1)$ and s from $uniform(100, 1000)$.

To determine the burn-in and sample size of the two algorithms and to ensure they draw samples equally effectively, we first check to see if they converge to a similar level. We generate a test set of 300 data points and vary the size of the training set to gauge the prediction accuracy. We set $K = 5$ in the MH step of our Gibbs algorithm. While our Gibbs algorithm achieves reasonable accuracy with burn-in = 20 and sample size = 100, the MCMC algorithm gives comparable results only when both parameters are 10000. Figure 2 shows their prediction accuracies averaged over 10 trials. With burn-in and sample size determined, we then conduct an efficiency study of the two algorithms. We vary the training set size from 1 to 1000 and record the time they take to finish the sampling process. We use a computer with Intel Core i7-2600 CPU @ 3.40Ghz and 8GB RAM. The efficiency comparison result is shown in Figure 3. We can see that computation time of both two sampling algorithms grows linearly with the training set size. However, our proposed Gibbs sampling algorithm is hundreds of times faster than MCMC, suggesting that our proposed approach is practical for deployment in online recommender systems.

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

4.4 User Study

In an online user study, we compare the effectiveness of our proposed recommendation algorithm, Bayes-UCB-CF, with that of two baseline algorithms: (1) Greedy algorithm, representing the traditional recommendation strategy without exploration-exploitation trade-off. (2) Bayes-UCB-Content algorithm [12], which also adopts the Bayes-UCB technique but is content-based instead of CF-based. We do not perform offline evaluation because it cannot capture the effect of the elapsed time t in our rating model and the interactiveness of our approach.

Eighteen undergraduate and graduate students (9 females and 9 males, age 19 to 29) are invited to participate in the user study. The subject pool covers a variety of majors of study and nationalities, including American, Chinese, Korean, Malaysian, Singaporean and Iranian. Subjects receive a small payment for their participation. The user study takes place over the course of two weeks in April 2014 on a user evaluation website we constructed (Figure 4). The three algorithms evaluated are randomly assigned to numbers 1-3 to avoid bias. For each algorithm, 200 recommendations are evaluated using a rating scale from 1 to 5. Subjects are reminded to take breaks frequently to avoid fatigue. To minimize the carryover effect, subjects cannot evaluate two different algorithms in one day. For the user study, Bayes-UCB-CF's hyperparameters are set as: $a_0 = 10$, $b_0 = 3$, $c_0 = 0.01$, $d_0 = 0.001$ and $e_0 = 0.001$.

Since maximizing the total expected rating is the main objective of a music recommender system, we thus compare the cumulative average rating of the three algorithms. Figure 5 shows the average rating and standard error of

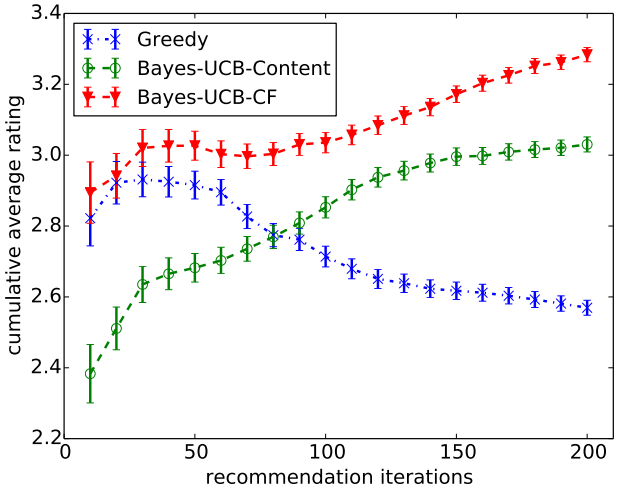

Figure 5: Recommendation performance comparison.

each algorithm from the beginning till the n-th recommendation iteration. We can see that our proposed Bayes-UCB-CF algorithm significantly outperforms Bayes-UCB-Content, suggesting that the latter still fails to bridge the semantic gap between high-level user preferences and low-level audio features.

T-tests show that Bayes-UCB-CF starts to significantly outperform the Greedy baseline after the 46th iteration (p-value < 0.0472). In fact, Greedy's performance decays rapidly after the 60th iteration while others continue to improve. Because Greedy solely exploits, it is quickly trapped at a local optima, repeatedly recommending the few songs with initial good ratings. As a result, the novelty of those songs plummets, and users become bored. Greedy will introduce new songs after collecting many low ratings, only to be soon trapped into a new local optima. By contrast, our Bayes-UCB-CF algorithm balances exploration and exploitation and thus significantly improves the recommendation performance.

5. CONCLUSION

We present a novel reinforcement learning approach to music recommendation that remedies the greedy nature of the collaborative filtering approaches by balancing exploitation with exploration. A Bayesian graphical model incorporating both the CF latent factors and novelty is used to learn user preferences. We also develop an efficient sampling algorithm to speed up Bayesian inference. In music recommendation, our work is the first attempt to investigate the exploration-exploitation trade-off and to address the greedy problem in CF-based approaches. Results from simulation experiments and user study have shown that our proposed algorithm significantly improves recommendation performance over the long term. To further improve recommendation performance, we plan to deploy a hybrid model that combines content-based and CF-based approaches in the proposed framework.

6. ACKNOWLEDGEMENT

We thank the subjects in our user study for their participation. We are also grateful to Haotian "Sam" Fang for proofreading the manuscript. This project is funded by the National Research Foundation (NRF) and managed through the multi-agency Interactive & Digital Media Programme Office (IDMPO) hosted by the Media Development Authority of Singapore (MDA) under Centre of Social Media Innovations for Communities (COSMIC).

7. REFERENCES

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

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

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

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

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

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

[7] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 165–172. ACM, 2011.

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

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

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

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

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

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

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