Knowledge-Based Systems 186 (2019) 104960

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Diverse personalized recommendations with uncertainty from implicit preference data with the Bayesian Mallows model☆

Qinghua Liu [a], Andrew Henry Reiner [b], Arnoldo Frigessi [b,c], Ida Scheel [a,*]

[a] Department of Mathematics, University of Oslo, P.O. Box 1053, Blindern, 0316 Oslo, Norway
[b] Oslo Center for Biostatistics and Epidemiology, Oslo University Hospital, Klaus Torgårds vei 3, 0372 Oslo, Norway
[c] Oslo Centre for Biostatistics and Epidemiology, University of Oslo, Sognsvannsveien 9, 0372 Oslo, Norway

ARTICLE INFO

Article history:
Received 11 February 2019
Received in revised form 8 August 2019
Accepted 15 August 2019
Available online 20 August 2019

Keywords:
Preference learning
Collaborative filtering
Clicking data
Probabilistic modeling

ABSTRACT

Clicking data, which exists in abundance and contains objective user preference information, is widely used to produce personalized recommendations in web-based applications. Current popular recommendation algorithms, typically based on matrix factorizations, often focus on achieving high accuracy. While achieving good clickthrough rates, diversity of the recommended items is often overlooked. Moreover, most algorithms do not produce interpretable uncertainty quantifications of the recommendations. In this work, we propose the Bayesian Mallows for Clicking Data (BMCD) method, which simultaneously considers accuracy and diversity. BMCD augments clicking data into compatible full ranking vectors by enforcing all the clicked items clicked by a user to be top-ranked regardless of their rarity. User preferences are learned using a Mallows ranking model. Bayesian inference leads to interpretable uncertainties of each individual recommendation, and we also propose a method to make personalized recommendations based on such uncertainties. With a simulation study and a real life data example, we demonstrate that compared to state-of-the-art matrix factorization, BMCD makes personalized recommendations with similar accuracy, while achieving much higher level of diversity, and producing interpretable and actionable uncertainty estimation.

1. Introduction

Personalized recommendations are widely used to help users and customers sort digital information for their purpose. From online streaming services to e-commerce websites, recommender systems can improve business efficiency, sort search results and enhance user experience.

Accuracy, reliability and diversity are some of the most important objectives for effective recommender systems. To help users find the items they prefer, recommendations should be accurate. At the same time, a recommender system should assess how reliable, or certain, the recommendations are, so that the users will not be disturbed by many irrelevant recommendations. To achieve this, uncertainty quantification of recommendations is essential. Furthermore, recommendations should be diverse. From a user's standpoint, a recommendation list is more interesting if it consists of a variety of items of different categories and genres; from a vendor's point of view, it is more cost efficient for rare and less popular items to have more exposure because the licensing of such items is less costly [1]. The importance of diversity is often overlooked by many recommender systems, and it is challenging to achieve both high accuracy and diversity. There is scarce information about the less popular items, therefore, it is much more risky to consider such items for recommendations. The trade-off between accuracy and diversity is referred to as the "accuracy–diversity dilemma" [2–4].

Personalized recommendations are based on the users' preference data, which can be explicit feedbacks such as ratings, and implicit feedbacks such as click stream data. Clicking data is easy to collect, exists in great abundance, and often better reflects user preferences compared to ratings. However, the interpretation of clicking data can be challenging, as there is no direct negative feedback from users [5], and the data naturally exhibits high sparsity [6].

The state-of-the-art approach using implicit feedback for personalized recommendation is the Collaborative Filtering for Implicit Data method developed by Hu et al. [5]. This method is based on matrix factorization (MF). It is effective and scalable,

☆ No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.knosys.2019.104960.

* Corresponding author.
E-mail addresses: qinghual@math.uio.no (Q. Liu), a.h.reiner@medisin.uio.no (A.H. Reiner), arnoldo.frigessi@medisin.uio.no (A. Frigessi), idasch@math.uio.no (I. Scheel).

https://doi.org/10.1016/j.knosys.2019.104960

and is commonly adopted by commercial applications [7]. However, there are some drawbacks. First of all, it does not provide interpretable uncertainty quantifications: when an item is recommended to a user, the method does not quantify the reliability of the recommendation. More importantly, the collaborative filtering framework does not consider diversity, and it has a tendency to favor the most popular items. While achieving high accuracy, these recommendations can sometimes be monotonous and lack diversity. Post-processing methods have been proposed to improve the diversity performance of collaborative filtering [8,9]. These methods usually contain a tuning parameter that balances the accuracy–diversity tradeoff. However, it is often impossible to achieve higher diversity without sacrificing the accuracy performance, and a poor choice of the tuning parameter can heavily impair the accuracy performance.

The method we introduce in this paper takes a more holistic approach. Instead of considering accuracy and diversity as separate objectives, our method embeds diversity considerations into the model for learning user preferences. In addition it provides interpretable uncertainty quantifications. Our method is called the Bayesian Mallows for Clicking Data (BMCD) method, and is constructed by further developing the approach introduced by Vitelli et al. [10]. We assume that users prefer clicked items to unclicked items, and individual clicking data is subsequently augmented to ranking vectors by enforcing all the clicked items to be top-ranked. This construction ensures that the uniqueness of each user, demonstrated especially by the rare items that are clicked, can be preserved and subsequently learned by the model. Our method inherently consider diversity in the model, achieving a good accuracy–diversity balance without the use of tuning parameters. Through a simulation study and an offline testing with a real life dataset provided by the Norwegian Broadcasting Corporation (NRK), we illustrate BMCD's effective accuracy–diversity performance, comparing to the state-of-the art Collaborative Filtering for Implicit Data method, and two popular post-processing diversity enhancements methods.

The main contribution of our work is a new method for making personalized recommendation based on implicit data, which inherently balances accuracy and diversity without the need of a tuning parameter. Contrasting with the current accuracy-driven methods such as Collaborative Filtering [5], our method holistically considers both accuracy and diversity. Compared to the diversity-driven post-processing methods [8,9], our method does not involve tuning parameters to achieve the accuracy–diversity balance, and does not sacrifice accuracy for diversity. Moreover, the Bayesian nature of our method ensures that each recommendation made is associated with a calibrated and interpretable uncertainty estimation.

The paper is organized as follows: We start by discussing related work in Section 2. In Section 3 we summarize the Bayesian Mallows Method, and then we introduce BMCD, and show how we can make personalized recommendations based on posterior probabilities. In Section 4, we introduce the evaluation metrics: accuracy and four diversity metrics. In Section 5.1 we explain the simulation study and demonstrate how BMCD makes recommendations with uncertainty quantification. In Section 5.2 we present a detailed comparison of BMCD's accuracy and diversity compared to Collaborative Filtering for Implicit Data for the simulation study. In Sections 5.3.2 and 5.3.4 we apply both methods to the NRK dataset, and compare their accuracy and diversity performances. It is followed by Section 5.4, which is dedicated to the comparison between BMCD and two post-processing methods in combination with Collaborative Filtering, based on the NRK dataset. We compare these methods' performances in terms of the accuracy–diversity balance. Last, a summary and further work are included in Section 6.

2. Related work

Collaborative filtering [11] is a framework utilizing user–item interaction data to make personalized recommendations through borrowing strength across the pool of users and items.

User-based collaborative filtering is an early method. For a particular user, the basic idea is first to discover other users who have similar preferences, often measured by cosine similarities or Pearson's correlation coefficient. After such neighbors are identified, recommendations are made based on an aggregation of the neighbors' preferences. User-based collaborative filtering is intuitive and easy to implement, however, it is often limited by the sparsity of the data as well as scalability. Instead, Sarwar et al. [12] proposed an item-based collaborative filtering algorithm. For a given user, her preference of an unknown item is predicted based on the users' past preferences of the k most similar items.

Matrix Factorization (MF)-based collaborative filtering methods are among the most successful [13]. The MF method proposed by Koren et al. [14] is developed for a user–item rating matrix. The data matrix $\mathbb{X}$ has dimensions $N \times n$, where N is the number of users and n is the number of items. Each entry x_{ij} is the rating given by a user j to an item i, or is empty. Assume that each user j has rated $\leq n$ items. MF obtains two reduced-dimension matrices $\boldsymbol{U}^{N\times L}$ and $\boldsymbol{V}^{n\times L}$, with $L < n$, so that their product will be a full matrix $\hat{\mathbb{X}}$ that approximates the original rating matrix $\mathbb{X}$. $\hat{\mathbb{X}}$ predicts, for each user, the ratings of the items that the user has not rated. Hu et al. [5] extended the method to implicit data. In this paper, BMCD is compared with the method in [5] since this is the widely adopted, state-of-the-art method. For more details on collaborative filtering, see [15].

Hu et al. [5] introduced the Collaborative Filtering for Implicit Data method (CF), which extends the classic matrix factorization method. It can be applied to datasets based on implicit user feedbacks, such as clicking data. We now denote the implicit user–item matrix as $\mathbb{X}$. The content of x_{ij} depends on the use case, for example, it can represent the number of times user i has clicked on item j. First, a binary matrix $\mathbb{W}$ is introduced by binarizing $\mathbb{X}$ such that w_{ij} is set to 1 if $x_{ij} > 0$, and 0 otherwise, i.e., w_{ij} is set to 1 if user j has clicked item i, and 0 otherwise. Second, a set of "confidence" variables c_{ij} is introduced. The rationale behind this variable is that different interactions indicate different levels of certainty that an item is preferred by the user. One choice for c_{ij} is: $c_{ij} = 1 + \beta x_{ij}$, $\beta \geq 0$. Finally, the factor matrices are obtained through minimizing the penalized loss function $\min_{U,V} \sum_{j,i} c_{ij}(w_{ij} - \boldsymbol{u}_j^T \boldsymbol{v}_i) + \theta(\sum_j \|\boldsymbol{u}_j\|^2 + \sum_i \|\boldsymbol{v}_i\|^2)$, where both $\boldsymbol{u}_j$ and $\boldsymbol{v}_i$ are L-dimensional column vectors. The last term in the loss function is a regularization term and is added to reduce overfitting. The parameters β, θ, and the reduced dimension of the factor matrices L are determined by cross-validation, while the minimization process is often achieved using algorithms such as alternating least square (ALS) [14]. In the later sections, the term "CF" refers exclusively to the method proposed by Hu et al. [5], and we use its implementation in Apache Spark [16]. BMCD will later be compared with this CF method, in terms of recommendation performances.

To address the accuracy–diversity dilemma, Zhou et al. [2] proposed a graph-based method inspired by the heat diffusion process. One accuracy-driven model and one diversity-driven model are combined with linear weighted average to balance accuracy and diversity. Karakaya and Tevfik [17] introduced a modification of Koren et al. [14]'s MF model for explicit feedback by penalizing popular items to improve diversity. However, the method has not been extended to implicit datasets.

Post-processing of recommendations can help enhance diversity. Antikacioglu et al. [18] proposed a bipartite graph-based

post-processing method. After a recommendation model is fitted, a score for each user–item pair is obtained, and then these scores serve as weights for the user–item edges. The recommendation process is modeled as a maximum-weight bipartite graph matching problem, and diversity is achieved by imposing diversity-related constraints to the optimization, which can be solved using algorithms for minimum cost flow problems. This method could post-process both BMCD and the Collaborative Filtering for Implicit Data Method, but we do not pursue this any further.

The most popular post-processing methods are proposed by Adomavicius et al. [9] and Ziegler et al. [8]. Both methods are used in combination with a model for recommendation, such as CF, which predicts a score for each user–item pair. To make k recommendations for each user, instead of using the traditional recommendation method, which simply recommends the k items with the highest such scores for each user, these methods impose special rules based on the predicted user–item scores. Adomavicius et al.'s [9] approach introduces a threshold: for each user, if there exists $l \geq k$ items whose scores have surpassed the threshold, the k least popular items among them are recommended to the user. If there exists $0 < l < k$ items whose scores surpassed the threshold, all the l items are recommended first, and the rest of the $k - l$ items are recommended using the traditional method. In the case where no item has surpassed the threshold, the traditional recommendation method is used to recommend the top k items to the user. The threshold is the tuning parameter that balances accuracy and diversity — a high threshold makes the recommendations more accuracy driven, while a low threshold results in more diverse but likely less accurate recommendations. Ziegler et al. [8] introduced an iterative approach instead. First, for each user, the top -1 item is added to the recommendation list using the traditional method. For each item that is not in the recommendation list, a similarity score is calculated between the item and the current recommendation list, and these items are ranked based on the similarity scores — from the least similar to the most similar. We denote this ranking as $rank_{sim}$. At the same time, another ranking is produced based on the ranking of the scores in descending order, denoted as $rank_{scores}$. The two rankings are combined in a linear weighted average, i.e. $rank_{combined} = \alpha\ rank_{sim} + (1-\alpha) rank_{score}$, $0 \leq \alpha \leq 1$, and the item with the lowest combined ranking (top-ranked) is added to the recommendation list. This procedure is repeated until k recommendations are reached. Clearly, α is the tuning parameter, for which a high value closer to 1 makes the algorithm more diversity-driven. We will use these two post-processing methods in combination with CF, and compare their accuracy–diversity performances with BMCD in Section 5.4.

3. Bayesian Mallows for Clicking Data (BMCD)

Consider a dataset of N users and n items $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$. Suppose first that each user j indicates her preferences with a ranking of all n items, $\boldsymbol{R}_j = \{R_{1j}, R_{2j}, \ldots, R_{nj}\}$, where R_{ij} is the rank assigned to item i by user j, $i = 1, \ldots, n, j = 1, \ldots, N$. The Mallows model is a probabilistic model on the space of permutations of n items $\mathcal{P}_n$. In the simplest case, assuming that all users share a common latent consensus $\boldsymbol{\rho} \in \mathcal{P}_n$, it has the form of $P(\boldsymbol{R}_j = \boldsymbol{r}|\alpha, \boldsymbol{\rho}) = \frac{1}{Z_n(\alpha,\boldsymbol{\rho})}\exp\{-\frac{\alpha}{n} d(\boldsymbol{r}, \boldsymbol{\rho})\}$, where α is a scale parameter, and $d(\boldsymbol{r}, \boldsymbol{\rho})$ is a distance between $\boldsymbol{r}$ and $\boldsymbol{\rho}$. Possible choices of distance functions include the footrule distance, the Spearman distance, and the Kendall distance. In this paper, we choose the footrule distance, defined as $d(\boldsymbol{r}, \boldsymbol{\rho}) = \sum_{i=1}^{n} |r_i - \rho_i|$, because of its effectiveness [19]. Other distances can also be used. Lastly, $Z_n(\alpha, \boldsymbol{\rho}) = \sum_{\boldsymbol{r}\in\mathcal{P}_n} \exp\{-\frac{\alpha}{n} d(\boldsymbol{r}, \boldsymbol{\rho})\}$ is the normalizing function. As the footrule distance is a right-invariant distance function, the partition function Z_n is independent of $\boldsymbol{\rho}$, and only depends on α, hence we denote it as $Z_n(\alpha)$. For $n < 50$, $Z_n(\alpha)$ has been computed [10], but is otherwise not analytically computable. When $n \geq 50$, the asymptotic approach introduced by Mukherjee et al. [20] and the importance sampling scheme introduced in [10] are available.

Realistically however, it is uncommon that all users are homogeneous. Assume that the N users are grouped in C clusters, and within each cluster, users share a common latent consensus. For each of the homogeneous clusters, we assume a Mallows distribution with parameters α_c, $\boldsymbol{\rho_c}$, $c = 1, \ldots, C$. The random variable denoted by $z_j \in \{1, \ldots, C\}$ assigns user j to cluster z_j. Assuming that users' preferences are conditionally independent given the Mallows parameters and their cluster assignments z_j, the likelihood function is hence

$$P(\boldsymbol{R}_1, \ldots, \boldsymbol{R}_N | \{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}, z_1, \ldots, z_N) = \prod_{j=1}^{N} [Z_n(\alpha_{z_j})]^{-1} \exp\{-\frac{\alpha_{z_j}}{n} d(\boldsymbol{R}_j, \boldsymbol{\rho}_{z_j})\}. \quad (1)$$

Vitelli et al. [10] introduce a Bayesian version of this model. The Mallows parameters $\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}$ are assumed a priori mutually independent. An exponential prior with hyperparameter λ is chosen for $\alpha_c, c = 1, \ldots, C$, i.e., $\pi(\alpha_1, .., \alpha_c|\lambda) = \lambda^C \exp\{-\lambda \sum_{c=1}^{C} \alpha_c\}$. For $\boldsymbol{\rho}_c, c = 1, \ldots, C$, the noninformative uniform prior $\pi(\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_C) = n!^{-C}$ is chosen. The prior for the cluster assignments $z_j, j = 1, \ldots, N$ is $p(z_1, \ldots, z_N|\tau_1, \ldots, \tau_C) = \prod_{j=1}^{N} \tau_{z_j}$, where the probabilities $\tau_1, \ldots, \tau_C$ follow a Dirichlet prior $\pi(\tau_1, \ldots, \tau_C) = \Gamma(\psi C)\Gamma(\psi^{-C}) \prod_{c=1}^{C} \tau_c^{\psi-1}$, $\psi > 0$. Hyperparameters ψ and λ are assumed to be fixed, see [10] for guidelines.

The posterior distribution of $\{\{\alpha_c, \boldsymbol{\rho_c}\}_{c=1,\ldots,C}, z_1, \ldots, z_N\}$ is therefore

$$\begin{aligned} &P(\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}, z_1, \ldots, z_N | \boldsymbol{R}_1, \ldots, \boldsymbol{R}_N) \\ &\propto \pi(\alpha_1, \ldots, \alpha_C|\lambda)\pi(\boldsymbol{\rho}_1, \ldots, \boldsymbol{\rho}_C) p(z_1, \ldots, z_N|\tau_1, \ldots, \tau_C) \\ &\cdot \pi(\tau_1, \ldots, \tau_C) P(\boldsymbol{R}_1, \ldots, \boldsymbol{R}_N | \{\alpha_c, \boldsymbol{\rho}_c\}, z_1, \ldots, z_N). \end{aligned} \quad (2)$$

We will now extend the Bayesian Mallows model to clicking data. For clicking data, the full ranking of the n items is not available and needs to be inferred from the clicking data. We denote the latent, full individual ranking vector for user j as $\tilde{\boldsymbol{R}}_j$. Suppose that each user j has clicked on a subset of the items $\mathcal{A}_j \subseteq \mathcal{A}$, with the number of clicks $|\mathcal{A}_j| = c_j$. It is common to assume that a clicked item is preferred by the user to any other un-clicked item [21]. This assumption also ensures that the uniqueness of each user is enforced, as all clicked items are forced to be more preferred to non-clicked items, regardless of the items' popularities. For each user j, the set of rankings compatible with this assumption is $\mathcal{S}_j(\mathcal{A}_j) = \{\tilde{\boldsymbol{R}}_j \in \mathcal{P}_n \text{ s.t. } \tilde{R}_{ij} < \tilde{R}_{kj} \text{ if } A_i \in \mathcal{A}_j \text{ and } A_k \in \mathcal{A}_j^c, \forall i, k, i \neq k\}$.

Given the clicking data, the goal is hence, to sample from the posterior distribution

$$\begin{aligned} &P(\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}, z_1, \ldots, z_N | \mathcal{A}_1, \ldots, \mathcal{A}_N) \\ &= \sum_{\tilde{\boldsymbol{R}}_1 \in \mathcal{S}_1(\mathcal{A}_1)} \cdots \sum_{\tilde{\boldsymbol{R}}_N \in \mathcal{S}_N(\mathcal{A}_N)} P(\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}, z_1, \ldots, z_N, \\ &\quad \tilde{\boldsymbol{R}}_1, .. \tilde{\boldsymbol{R}}_N | \mathcal{A}_1, \ldots, \mathcal{A}_N) \end{aligned} \quad (3)$$

To make inference, we follow a Markov Chain Monte Carlo (MCMC) scheme similar to the one in [10]. Each iteration of the algorithm consists of three major steps:

(i) Update the parameters $\{\alpha_c, \boldsymbol{\rho}_c\}$ within each cluster $c = 1, .., C$, given the current values of the individual rankings $\tilde{\boldsymbol{R}}_j$, and the cluster assignments $z_j, j = 1, \ldots, N$

(ii) Re-assign users to clusters based on the current values of the parameters $\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}$ and the individual ranking vectors $\tilde{\boldsymbol{R}}_j$, $j = 1, \ldots, N$

(iii) Update $\tilde{\boldsymbol{R}}_j$ for each user j given the current values of $z_1, \ldots, z_N$ and $\{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}$

As in [10], we use a Metropolis–Hasting algorithm for step (i), and a Gibbs sampler for step (ii). For step (iii), we sample from $P(\tilde{\boldsymbol{R}}_1, \ldots, \tilde{\boldsymbol{R}}_N | \{\alpha_c, \boldsymbol{\rho}_c\}_{c=1,\ldots,C}, z_1, \ldots, z_N, \mathcal{A}_1, \ldots, \mathcal{A}_N)$. Given the cluster assignments and the Mallows parameters, the individual rankings are conditionally independent. Therefore, for each user j, we can independently sample from the posterior $P(\tilde{\boldsymbol{R}}_j | \alpha_{z_j}, \boldsymbol{\rho}_{z_j}, z_j, \mathcal{A}_j)$ using a Metropolis–Hasting algorithm, where a new $\tilde{\boldsymbol{R}}_j'$ for each user must be proposed. One convenient way is to choose two items i, k such that $\{i, k\} \in \mathcal{A}_j$ or $\{i, k\} \in \mathcal{A}_j^c$, and then swap the rankings of the two items for each user j. This proposal is symmetric, and each proposed latent full individual ranking vectors $\tilde{\boldsymbol{R}}_j'$ is accepted with probability $\min\{1, \exp[-\frac{\alpha_{z_j}}{n}(d(\tilde{\boldsymbol{R}}_j', \rho_{z_j}) - d(\tilde{\boldsymbol{R}}_j, \rho_{z_j}))]\}$. Another way of proposing a new $\tilde{\boldsymbol{R}}_j'$ is to treat each $\tilde{\boldsymbol{R}}_j$ as two parts: the clicked part and the un-clicked part. The "leap-and-shift" algorithm in [10] can then be used separately for the two parts.

To make personal recommendations, the variables of interest are the latent augmented full ranking $\tilde{\boldsymbol{R}}_j$ for each user. For a given user j that has clicked on c_j items, the objective of making $k \geq 1$ recommendations is equivalent to inferring which items are to be ranked as the user's $c_j + 1$th, …, $c_j + k$th items. We therefore calculate for each user j and each item i the posterior probability to be ranked between $c_j + 1, \ldots, c_j + k$, which we refer to as the "next top - k" items. That is, we estimate for each user j and each item i

$$\begin{aligned} P_{ij} &= P(c_j + 1 \leq \tilde{R}_{ij} \leq c_j + k | \mathcal{A}_1, \ldots, \mathcal{A}_N) \\ &= P(\tilde{R}_{ij} \leq c_j + k | \mathcal{A}_1, \ldots, \mathcal{A}_N). \end{aligned} \tag{4}$$

Once estimated, these posterior probabilities are later ranked for each user j in descending order, and the k items with the highest such probabilities are recommended to the user. The estimated top-k probabilities are referred to as the top posterior probabilities (TPP), and the set of k recommended items for user j is denoted as $Rec_{j,k}$. Section 6 in the supplement contains a structured description of our algorithms.

4. Recommendation evaluation—accuracy and diversity

In this section, we introduce the assessment metrics that will be used in order to assess the accuracy and diversity performances of the recommendation methods.

To assess recommendation accuracy, the next $k \geq 1$ recommendations are made for each user. In simulations, the recommendations are later compared with the truth. In offline experiments, based on a train-test split of the dataset, accuracy is measured as the percentage of the recommended items that are clicked in the test set. For online experiments, the truth is obtained by experimentation. The drawback of offline experiments compared to online experiments is that the recommendations are not actually given to the users, and hence the truth defined by the test set is not a response to the recommendations. This might be problematic for the recommendation of less popular items, since the users might not even be aware of these items, and hence could not have clicked them in the test set. Offline training-test experimentation is often the only and best alternative for assessing accuracy.

Despite being an important measure of performance, accuracy is not the only factor that defines successful recommendations [22,23]. User experience can be greatly enhanced when recommendations are diverse, and hence has the potential to be novel and surprising. To assess the diversity of recommendations, we adopt the following four metrics: coverage [24], correct coverage, intra-list similarity [8,25] and novelty [2].

4.1. Coverage

Ge et al. [24] introduced the metric "coverage", defined as

$$\text{coverage} = \frac{\#\text{ distinct items recommended to users}}{\#\text{ distinct items eligible for recommendation}},$$

the percentage of the distinct items ever recommended to the users. A recommender system with a high coverage has exploited its pool of items more efficiently, and their users, collectively, are exposed to a wider spectrum of items.

This coverage metric has one major limitation. For a highly inaccurate recommender system, in the extreme case, when recommendations are made randomly, the coverage can be very high while the recommendation accuracy is extremely low. Therefore, we also introduce the "correct coverage" metric, defined as:

$$\text{correct coverage} = \frac{\#\text{ distinct items recommended and clicked by at least one user}}{\#\text{ distinct items eligible for recommendation}}$$

4.2. Intra-list similarity

Ziegler et al. [8] introduced the "Intra-list similarity" metric to assess diversity on an individual level. The rationale behind this metric is that, on an individual level, each user tends to prefer recommendations from various categories. A recommendation list that contains items from only one or a few specific categories (for example, a list of only Harry Potter movies), are far less exciting compared to a good mixture of very different items (even for Harry Potter fans). Similarity between two items a, b is measured by binary cosine similarity [25] based on the training data, defined as

$$CosSim(a, b) = \frac{\#\text{ users clicked both a and b}}{\sqrt{\#\text{ users that clicked } a} \times \sqrt{\#\text{ users that clicked } b}},$$

and the intra-list similarity metric is hence defined as

$$\text{Intra-list similarity} = \frac{1}{N}\sum_{j=1}^{N} \sum_{a,b \in Rec_{j,k}, b<a} CosSim(a, b).$$

It is desirable for a recommender to have a low intra-list similarity.

4.3. Novelty

The novelty measure, introduced by Zhou et al. [2], assesses the recommender's ability to recommend items less explored by the users. It is defined as

$$\text{novelty} = \frac{1}{N}\sum_{j=1}^{N} \sum_{i \in Rec_{j,k}} \frac{|\log_2 pop_i|}{k},$$

in which pop_i refers to the popularity of item i, in this case, the fraction of all clicks attributed to item i in the training data. A recommender that recommends rare and less popular items, and hence makes novel recommendations, will have a high novelty score. It is desirable to make novel recommendations because a recommendation list consisting of only the most popular items lacks personalization. In addition, the less popular items are challenging to be recommended due to lack of data [23], and are often valuable to the business [1].

5. Experiment and results

In this section, we conducted a detailed comparison between the recommendation performances by BMCD and the Apache Spark [16] implementation of CF through a simulation study, as well as an offline case study with a dataset provided by the Norwegian Broadcasting Corporation (NRK). Each method will be assessed in terms of recommendation accuracy as well as diversity. Using the NRK dataset, we also compare the accuracy–diversity performances between BMCD and the post-processed versions of CF in Section 5.4.

5.1. Simulation study design

In this simulation study, we consider a group of $N = 3000$ users and $n = 50$ items. The users are partitioned in $C = 3$ equally sized and distinct clusters. The users in each cluster are given full ranking vectors $\boldsymbol{R}_j$ sampled from the Mallows model using the sampler in [10]. For each cluster c, the parameters are chosen to be $(\alpha = 3, \boldsymbol{\rho_c})$, with $\boldsymbol{\rho_1} = \{1, 2, .., 50\}$, $\boldsymbol{\rho_2} = \{50, 49, \ldots, 1\}$, and $\boldsymbol{\rho_3} = \{39, 36, 11, 1, 13, 12, 8, 48, 20, 49, 29, 32, 22, 28, 19, 5, 42, 18, 15, 7, 6, 27, 24, 16, 46, 4, 21, 26, 34, 44, 25, 43, 41, 38, 35, 37, 45, 2, 14, 50, 40, 47, 9, 23, 30, 31, 3, 10, 33\}$. The 3 consensuses are chosen since they will produce 3 distinct clusters that separate well. Hence, a dense $N \times n$ ranking dataset is obtained, which will later serve as the ground truth for checking recommendation accuracy, and from which we will build the incomplete clicking dataset.

To simulate clicking data, the full ranking dataset is converted to a binary dataset in the following way. For each user $j = 1, \ldots, N$, we draw the number of clicks c_j from a truncated Poisson distribution with parameter $\lambda = 5$, truncated to a minimum of 1. Thereafter, the top ranked c_j items are considered "clicked", while the rest of the items considered "unclicked". In other words, for each user j, we obtain $\mathcal{A}_j = \{A_i : R_{ij} \leq c_j\}$.

We generate independently 20 such datasets, and use both CF and BMCD to recommend $k = 5$ and $k = 10$ items for each user, i.e., to predict for each user j which items are ranked among $c_j + 1, \ldots, c_j + k$. The parameters for CF are determined through 10-fold cross validation. Although the ground truth dataset is generated from a Mallows model, which can impose some bias towards BMCD in terms of accuracy checking, the dataset is converted to a binary clicking dataset for model fitting. The binarization adds great sparsity to the dataset, and converts ranking vectors into binary vectors, which the Mallows model is not defined for. BMCD's advantages in inference are considerably reduced due to the binarization.

To use BMCD, the number of clusters C needs to be determined first. We run Algorithm 1 and 2 in the supplementary material with random initialization and varying numbers of clusters $C = 2, 3, \ldots, 8$. For each value of C, we estimate the posterior mean of the sum of within cluster footrule distances (MWCD), defined as

$$\text{MWCD} = \mathbb{E}[\sum_{c=1}^{C}\sum_{j=1}^{N} d(\tilde{\boldsymbol{R}}_j, \boldsymbol{\rho}_{z_j})|\mathcal{A}_1, \ldots, \mathcal{A}_N],$$

by the natural Monte Carlo mean. For each of the 20 simulated datasets, the number of clusters C with the smallest MWCD is chosen. It turns out that $C = 3$ is chosen for all runs, except for run numbers 5, 8, and 20, for which $C = 4$. Fig. 1 in the supplementary material Section 1 shows the boxplots of the posterior sum of within cluster distances for 3 selected runs (1, 5, 10).

The MCMC is run for 1 million iterations, with the first 500 000 iterations discarded as burn-in. Similar to the set up in [10], parameters $\{\alpha_1, \ldots \alpha_C\}$ are only proposed for update every 10 iterations. The trace plots of α_c for run 1, $c = 1, 2, 3$, after the burn-in period is included in the supplementary material Section 3. Convergence was checked by using multiple starting points of the MCMC chains. Recommendations of the Bayesian Mallows method are made based on TPP. The recommendation procedure is described in Section 3 in the supplementary material.

5.2. Simulation results and discussion

5.2.1. Recommendation accuracy

After recommendations are made, we refer to the ground truth full ranking vectors $\boldsymbol{R}_j$ to check whether the recommended items are truly among each user's next-k items.

Table 1 shows the recommendation accuracy using CF and BMCD to predict each user's next $k = 5$ and $k = 10$ items. It can be observed that BMCD makes slightly more accurate recommendations compared to CF in predicting both the next 5 and next 10 items. The accuracy advantage over CF is more significant in the next 10 case. In addition, BMCD's accuracy performance is less varied than that of CF's. BMCD's high accuracy demonstrates that the Bayesian Mallows model is a good fit for the recommendation problem, and that the recommendation process, which is based on posterior probability estimations, can accurately reflect users' true preferences.

We have also discovered that the number of clusters C chosen has little effect on the overall recommendation accuracy, as long as the number of clusters chosen is not too small. As shown in Fig. 1, the overall recommendation accuracy stabilizes after $C \geq 3$. It is apparent that if the number of clusters C is too small, users that do not share a consensus, i.e., users of very different tastes, are grouped together. Hence, the estimated ρ will not be able to represent all the cluster members' tastes, which will negatively affect the estimation accuracy. On the other hand, when a too larger C value is chosen, some of the clusters break down into smaller clusters. As long as the cluster size is not too small, the estimated consensus parameter still fairly represents a consensus of the cluster members' tastes. Hence, we can effectively learn each individual's preferences, borrowing strength from other users within the cluster. In our simulation, given that $C \geq 3$, the accuracy performance is quite stable as long as each cluster has roughly more than 60 members, as shown in the supplementary material.

5.2.2. Recommendation uncertainty quantification

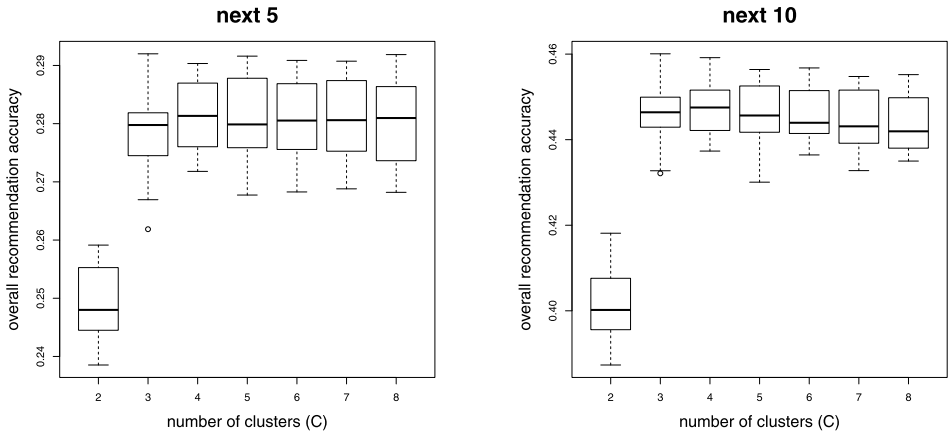
BMCD also estimates the uncertainty associated with each recommendation through the TPP. Such uncertainties can help assess the reliability of the recommendations by predicting the actual "hit rates" of the recommended items. For each user j and each recommended item i, we can use the binary indicator t to indicate whether the recommended item i is truly among user j's next top-k: $t_{ij} = 1$ if $R_{ij} \leq c_j + k$, and 0 otherwise.

At the same time, we can bin the TPPs by putting them into M intervals of equal width. In this case, we choose 0.01 as the bin size. For all TPPs that belong to interval m, the associated indicators t are averaged to $\bar{t}_m$, indicating the average "hit rate" of the recommended items associated with the corresponding level of certainty.

In Fig. 2 we plot $\bar{t}_m$ against the binned TPPs for the next $k = 5$ and $k = 10$ cases. Run number 10 is shown here, but other runs demonstrate similar trends. The blue dotted $x = y$ line indicates perfect calibration. The red dotted line in the figure represents the percentage of correct recommendations made by CF for this run. From the next-5 case, we can clearly observe excellent calibration, especially when the TPPs are in the range

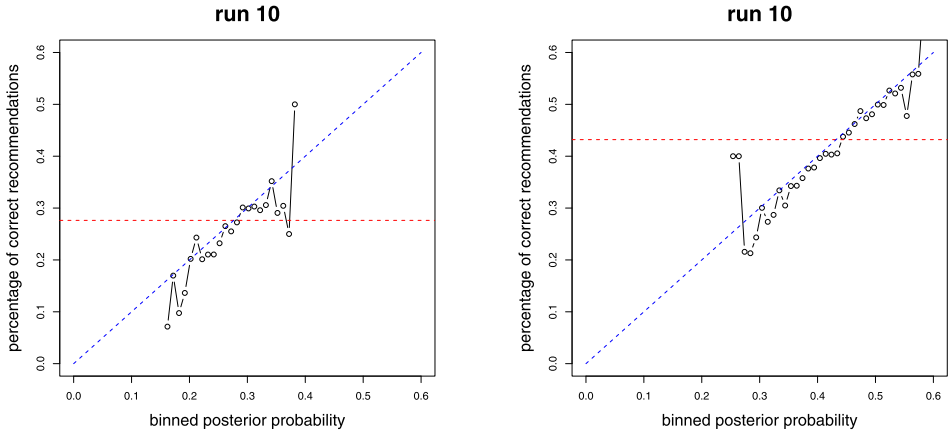
Table 1
Comparison of accuracies of BMCD and CF, summary of 20 runs.

Method	Min	25%	Median	75%	Max	Mean	Std dev
CF Next-5	25.02%	26.59%	27.34%	27.66%	28.65%	27.16%	0.89%
BMCD Next-5	26.69%	27.54%	27.98%	28.26%	29.20%	27.92%	0.69%
CF Next-10	40.54%	41.96%	42.70%	43.64%	44.47%	42.79%	1.05%
BMCD Next-10	43.21%	44.33%	44.64%	45.04%	46.01%	44.67%	0.72%



(a) Recommend next 5 (b) Recommend next 10

Fig. 1. BMCD's recommendation accuracy vs number of clusters chosen.



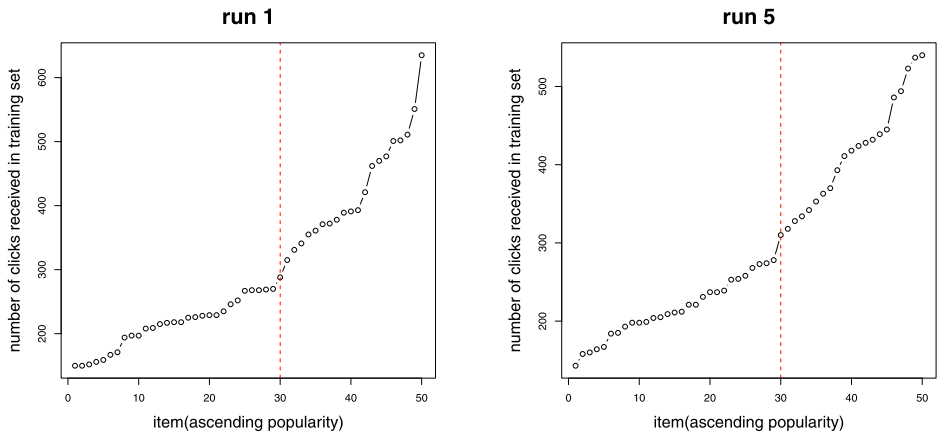
(a) Run 10, next-5 (b) Run 10, next-10

Fig. 2. Percentage of correct recommendations vs. binned TPPs of one selected run. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

between 0.25 and 0.35, where the majority of the recommendations lie within. The uncertainty is not as well calibrated when the TPPs are higher than 0.35 and below 0.25, since there are very few recommendations made with these TPPs. We observe a similar trend in the next-10 case, also in Fig. 2, with overall higher TPPs, and higher accuracy.

The TPP calculations make it possible for BMCD to identify which recommendations are more reliable than others, because the posterior probabilities are precise and interpretable, and hence can be further exploited. CF on the other hand, produces scores useful for ranking the items but are not easily interpretable. One usage of BMCD's TPPs is introducing a nearly calibrated cut off in order to achieve a higher overall recommendation accuracy. That is to say, we can decide to only make recommendations whose posterior probabilities of being in the next top k has surpassed a threshold and can be expected to be at least the threshold value as hit rate. This will inevitably reduce the number of recommendations made to the users, however, overall accuracy can be expected to be higher.

Tables 2 and 3 show how the recommendation accuracies improve when cut off TPPs are used, for the next-5 and next-10 case, respectively. For the next-5 case, it can be observed from Table 2 that all of the recommendations made with BMCD have TPPs above 0.1. Setting a TPP cut off of 0.25 can increase the overall recommendation accuracy by 1.7% points compared to not having a cut off, while retaining more than 70% of the recommendations. Likewise, all TPPs for the next-10 case are above 0.2, but fewer than 100 recommendations have a TPP of 0.6 or higher. When the cut off TPP is set at 0.45, the number of



(a) Run 1 (b) Run 5

Fig. 3. Item popularity of selected runs. The 20 most clicked items are considered as popular.

Table 2
Cut off TPPs and the corresponding recommendation accuracies for predicting next 5 items, summary of 20 runs. CF average accuracy: 27.2%.

Cut off	Number of recommendations	Recommendation accuracy
0.10	15 000 ± 0	28.0 ± 0.3%
0.15	14 997 ± 3.1	28.0 ± 0.3%
0.20	14 136 ± 148.9	28.6 ± 0.3%
0.25	11 196 ± 395.1	29.7 ± 0.3%
0.30	5 425 ± 592.6	31.0 ± 0.4%
0.35	577 ± 211.1	32.4 ± 1.5%
0.40	145 ± 9.3	32.5 ± 8.4%

Table 3
Cut off TPPs and the corresponding recommendation accuracies for predicting next 10 items, summary of 20 runs. CF average: 42.8%.

Cut off	Number of recommendations	Recommendation accuracy
0.20	30 000 ± 0.0	44.6 ± 0.3%
0.25	29 997 ± 1.7	44.6 ± 0.3%
0.30	29 254 ± 144.8	45.0 ± 0.3%
0.35	26 308 ± 460.5	46.4 ± 0.3%
0.40	21 215 ± 694.9	48.0 ± 0.3%
0.45	14 468 ± 762.1	50.0 ± 0.3%
0.50	7 104 ± 800.0	51.0 ± 0.4%
0.55	1 467 ± 438.4	52.0 ± 1.2%
0.60	59 ± 44.0	61.0 ± 8.2%

recommendations are reduced to roughly 50%, while increasing the overall recommendation accuracy by 5.4% points to 50%.

To summarize, BMCD makes recommendations with similar or slightly higher recommendation accuracies compared to CF in this simulation study. Moreover, the posterior probabilities associated with the recommendations are well calibrated and can be further exploited to assess the reliability of the recommendations. Overall recommendation accuracy can be improved by setting a cut off posterior probability.

5.2.3. Diversity

In this section, we assess both CF and BMCD's abilities to fully exploit the item collection, by making novel and diverse recommendations for each user. We will use the four metrics described in Section 4.

Table 4 summarizes the diversity performances of BMCD and CF. It is desirable to have high values of the coverage, correct coverage and novelty metrics, and a low value of intra-list similarity. We see that recommendations made with BMCD are more diverse and novel compared to CF. BMCD outperforms CF especially on

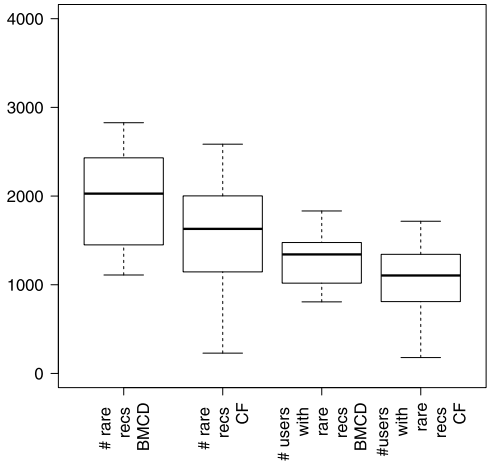


Fig. 4. Comparisons of number of rare item recommendations and number of users with at least 1 recommendations, summary of 20 runs.

the coverage metric, suggesting that BMCD has stronger ability to discover the less popular items.

If we rank all n items according to the number of clicks received by each item (popularity) in the training data in ascending order, and plot the corresponding number of clicks, as shown in Fig. 3, it can be observed that the majority of the clicks are received by a small fraction of items. If we define the 20 most clicked items as "popular", and the rest of the 30 items as less popular, or "rare", we can take a closer look at how often BMCD and CF recommend these "rare" items, and how many users have received at least one such rare recommendation.

From Fig. 4, it can be seen that BMCD recommends many more rare items, and out of $N = 3000$ users, more than 1000 users have received at least one rare recommendation for all runs, outperforming CF in its ability to explore rare items.

Unsurprisingly, BMCD outperforms CF in its diversity performance since the model construction inherently considers diversity by enforcing all clicked items to be top-ranked regardless of how rare these items are. CF on the other hand, is an accuracy-driven method that does not consider diversity as part of its objectives. Instead, the rare items clicked by the users might be sacrificed in the optimization process.

Table 4
Comparison of diversity performances of BMCD and CF.

Metric	Min	25%	Median	75%	Max	Mean
Coverage	CF:0.540 BMCD: 0.680	CF: 0.575 BMCD: 0.720	CF:0.720 BMCD: 0.740	CF: 0.720 BMCD: 0.740	CF:0.760 BMCD: 0.780	CF:0.672 BMCD: 0.731
Corr covg	CF:0.520 BMCD: 0.620	CF: 0.560 BMCD: 0.640	CF:0.660 BMCD: 0.680	CF: 0.680 BMCD: 0.700	CF:0.700 BMCD: 0.720	CF:0.629 BMCD: 0.676
Intra-list similarity	CF:1.92 BMCD: 1.70	CF: 2.05 BMCD: 1.78	CF:2.11 BMCD:1.91	CF: 2.16 BMCD:2.06	CF:2.23 BMCD: 2.11	CF:2.10 BMCD: 1.91
Novelty	CF:5.05 BMCD: 5.14	CF: 5.11 BMCD: 5.15	CF:5.18 BMCD: 5.29	CF: 5.20 BMCD: 5.43	CF:5.23 BMCD: 5.44	CF:5.16 BMCD: 5.29

Table 5
Description of the two NRK training datasets.

Dataset	# of items (n)	# of users (N)	Min clicks	Median clicks	Max clicks	Sparsity
1	200	7872	3	8	93	5.26%
2	200	2143	3	9	83	5.98%

5.3. Case study: A clicking dataset from the Norwegian Broadcasting Corporation (NRK)

In this section, we study a dataset containing anonymous users' clicks on movies, TV-series and news programs that are available on the NRK TV website as well as the apps for mobile phones, tablets and other streaming devices such as AppleTV. The data was collected when no personalized recommendation was implemented. We consider only the 200 most popular items. Here for simplification, a whole season of TV series, or a daily news program (which consists of more than one episode), is considered as one single item. Each user–item click is only recorded once, that is, multiple clicks on one item by one user are treated as one click.

We created two relatively sparse training datasets for model fitting, each with 3 or more clicks per user, as seen in Table 5. They were created by finding the users with 13 (dataset 1) and 23 (dataset 2) or more clicks per user, then randomly removing $k = 10$ (dataset 1) and $k = 20$ clicks per user. These k clicks were hence not part of the training data, but reserved for evaluating the ability of the fitted models to make accurate recommendations. After fitting a model, k recommendations were made by the model, and then these recommendations were compared to the k clicks not part of the training data. The fraction of overlap between the recommendations and the actual k clicks gives the recommendation accuracy. We will also assess both methods' diversity with the metrics introduced in Section 4.

5.3.1. MCMC set up and initialization

First, the number of clusters C needs to be determined. Alternative to the approach shown in Section 5.1, we used K-means clustering on the NRK binary datasets $\{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$ with different values of C, and plot the within cluster sum of square against the value of C, see Fig. 5 in the supplement. Combining the elbow method and a preference towards a slightly larger number of clusters, which we showed was important in Section 5.1, $C = 17$ is chosen for dataset 1 and $C = 12$ for dataset 2.

While there are many ways of initializing the MCMC, we use the following procedures in order to achieve faster convergence. To initialize the augmented individual ranking vectors $\tilde{\boldsymbol{R}}_j^0$, we first suppose that all users belong to the same cluster, and estimate very roughly a consensus for all users $\boldsymbol{\rho}^0$ based on item popularity. We obtain $\boldsymbol{\rho}^0$ by ranking the n items according to the number of clicks each of them has received, and randomize the ties if there are any. Next, we initialize the augmented individual ranking vectors $\tilde{\boldsymbol{R}}_j^0$ based on $\boldsymbol{\rho}^0$. First, $\tilde{\boldsymbol{R}}_j^0$ needs to be compatible with the restriction that the clicked items are top-ranked, i.e., $\tilde{R}_{ij}^0 \leq c_j \forall A_i \in \mathcal{A}_j$, and $\tilde{R}_{ij}^0 > c_j \forall A_i \in \mathcal{A}_j^c$. Second, while satisfying this restriction, we want to inherit the pairwise comparisons represented in the group consensus $\boldsymbol{\rho}^0$. That is to say, for each user j,

$$\forall p, q \in \mathcal{A}_j \text{ and } \forall a, b \in \mathcal{A}_j^c$$
$$\tilde{R}_{pj}^0 < \tilde{R}_{qj}^0 \leq c_j, \text{ if } \rho_p^0 < \rho_q^0$$
$$c_j < \tilde{R}_{aj}^0 < \tilde{R}_{bj}^0, \text{ if } \rho_a^0 < \rho_b^0.$$

For example, if we have a 5-item set {A, B, C, D, E}, $\boldsymbol{\rho}^0 = \{1, 2, 3, 4, 5\}$, and user j has clicked on item A, C, and E, the initialization of the augmented vector $\tilde{\boldsymbol{R}}_j^0$ is therefore, {1, 4, 2, 5, 3}. This initialization speeds up the MCMC convergence significantly.

The cluster assignment z_j for $j = 1, \ldots, N$, is initialized randomly. Within each cluster c, $\boldsymbol{\rho}_c^0$ is initialized in a similar manner as $\boldsymbol{\rho}^0$, however, the item popularity is calculated only based on the clicks by the users that belong to cluster c. The parameters $\{\alpha_c^0\}_{c=1,\ldots,C}$ are initialized as $a_1^0 = \cdots = a_C^0 = 3$, other values can also be chosen.

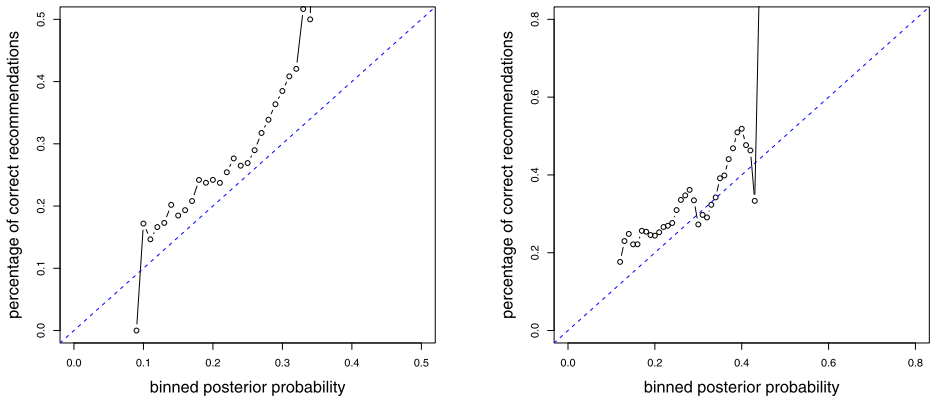
We run the MCMC for 5 million iterations and 7 million iterations, for dataset 1 and dataset 2, respectively. It takes longer for dataset 2 to reach convergence, presumably since there are more users that swing between different clusters. Only the last 1 million iterations are used for subsequent analyses. The MCMC is thinned at every 100 iterations while $\{\alpha_1, \ldots \alpha_C\}$ is proposed every 10 iterations. The trace plots of $\{\alpha_1, \ldots \alpha_C\}$ after the burn-in period are shown in Fig. 4 in the supplement.

Table 6
Comparison of next-k recommendation accuracies for the NRK datasets.

Method	Dataset 1 (next 10)	Dataset 2 (next 20)
BMCD	26.4%	35.0%
CF	29.9%	34.9%

5.3.2. Recommendation accuracy

Table 6 shows the overall recommendation accuracy for predicting the next-k items for both datasets using BMCD and CF respectively. It can be observed that CF in this case outperforms BMCD in terms of accuracy for dataset 1, while for dataset 2, the two methods' accuracies are almost identical. The NRK dataset is collected when no personalized recommendation is rolled out. In this situation, all users' clicks are quite concentrated on the popular items. As will be discussed in more detail in Section 5.3.4, BMCD's tendency to recommend a more diverse set of items and the inclusion of less popular items, compared to CF, presumably contributes to the slightly inferior recommendation accuracy for dataset 1.



(a) Dataset 1 (b) Dataset 2

Fig. 5. Recommendation accuracies vs. binned TPPs for BMCD.

Table 7
Comparisons of coverage and correct coverage.

Dataset	Coverage	Corr coverage
Dataset 1	CF:0.61 (122/200) BMCD: 0.865(173/200)	CF: 0.545 (109/200) BMCD: 0.720(144/200)
Dataset 2	CF: 0.38 (76/200) BMCD: 0.82 (164/200)	CF: 0.280 (56/200) BMCD: 0.685(137/200)

5.3.3. Uncertainty quantification of BMCD recommendations

Similar to Fig. 2, Fig. 5 shows the recommendation accuracy plotted against the binned TPPs. A clearly increasing trend can be observed. BMCD in this case, tends to underestimate the certainty of each recommendation made, or in other words, the TPPs estimated are slightly lower than the actual hit rates of the recommendations as the blue line is slightly above the dotted line. This can be explained by a slight misfit of the Mallows model. We can exploit the uncertainty to identify reliable recommendations as well as introducing cut off TPPs to improve overall accuracy of BMCD.

We see from Fig. 6 that, as the cut off TPP increases, the number of recommendations strictly decreases while the overall recommendation accuracy improves. For dataset 1, when the cut off posterior probability is 0.23 or above, BMCD's overall recommendation accuracy exceeds 30%, making it identical to CF, while retaining 60% of the recommendations.

5.3.4. Diversity

The coverage metric is especially important for NRK. As a national broadcaster, NRK has a large collection of valuable historical contents and non-mainstream programs that may be rarely discovered by its users; however, these programs have high quality and should be promoted.

Table 7 summarizes the comparisons of coverage and correct coverage of BMCD and CF. Both methods cover a broader range of items for dataset 1, as the dataset contains more users, leading to more diverse preferences. Dataset 2 is a more difficult scenario where the users' preferences are more homogeneous, and it is therefore more challenging to make diverse recommendations. It is clear that BMCD outperforms CF in terms of coverage, and the advantage is especially significant for dataset 2. This suggests that, consistent with the simulation, CF tends to recommend more popular items while BMCD has a stronger ability to explore the rare items. In addition, BMCD does not sacrifice much accuracy for diversity, as it also outperforms CF in the correct coverage metric.

Table 8
Comparison of rare items recommendations and number of users with at least 1 rare recommendation.

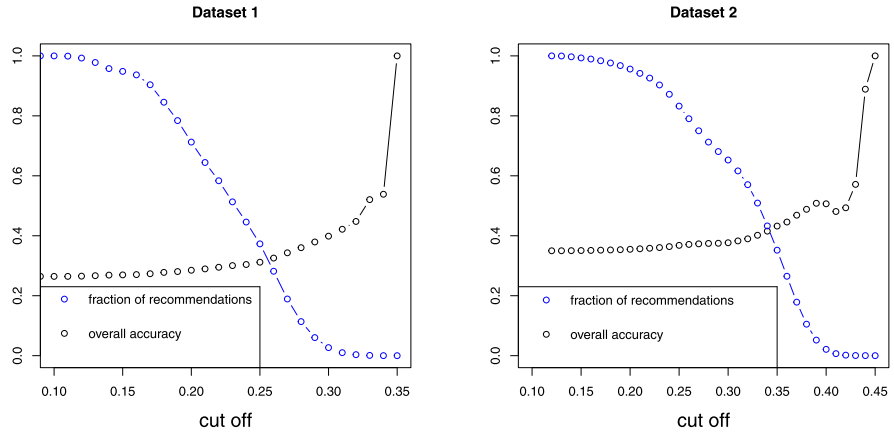
Dataset	# rare recs	# users w. $\geq$1 rare recs
Dataset 1	CF:4388 (5.6%) BMCD: 10071 (12.8%)	CF:929 (11.8%) BMCD: 3454 (43.9%)
Dataset 2	CF: 171 (4.0%) BMCD:8770 (20.5%)	CF: 58 (2.7%) BMCD: 1667 (77.8%)

Table 9
Comparison of intra-list similarity and novelty.

Dataset	Intra-list similarity	Novelty
Dataset 1	CF:11.78 BMCD: 10.75	CF:5.97 BMCD: 6.18
Dataset 2	CF:42.69 BMCD:39.10	CF: 6.31 BMCD: 6.60

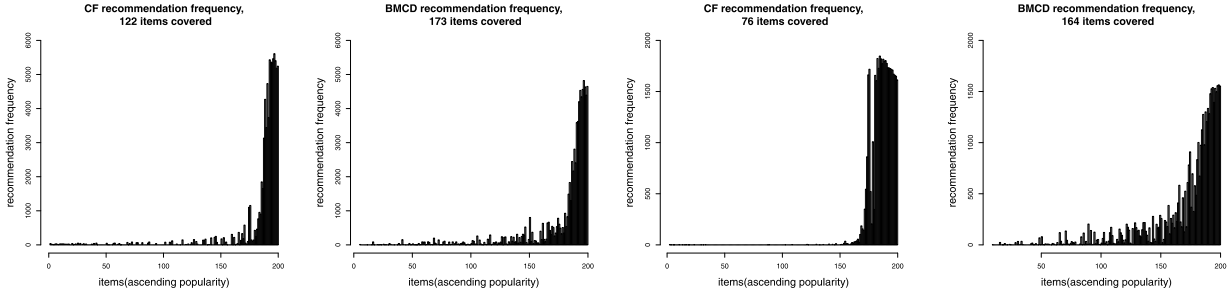
Fig. 7 shows the recommendation frequency of the items being recommended. On the *x*-axis, the items are ranked according to their popularity in ascending order. For both datasets, CF's recommendations are much more concentrated on the more popular items. BMCD in comparison, recommends much fewer popular items compared to CF.

To give a clearer definition of "popular" items, Fig. 8 shows the number of clicks received by each item in the training dataset, with the *x*-axis arranged from the least clicked item to the most clicked item. In both datasets, most of the clicks are attributed to roughly the 40 most popular items, which we define as "popular" items, while the rest we defined as "rare" items. Based on this definition, the number of "rare" items recommended by BMCD and CF, and the number of users receiving at at least one "rare" recommendations are shown in Table 8. It clearly shows that BMCD makes significantly more recommendations that are less popular compared to CF. In particular, for dataset 1, more than 12.8% of all recommendations made with BMCD involves rare items and more than 40% of all users receive at least 1 rare recommendation. For CF, only 5.6% of all recommendations are rare, while 11.8% of the users receive 1 or more rare recommendations. The contrast between CF and BMCD is even more obvious for dataset 2, where only 4% of all recommendations made with CF involves rare items, compared to BMCD's 20.5%. Given that CF's and BMCD's recommendation accuracies are similar in this case, and that BMCD makes more rare recommendations, it follows that BMCD also has a higher recommendation accuracy when recommending popular items compared to CF.



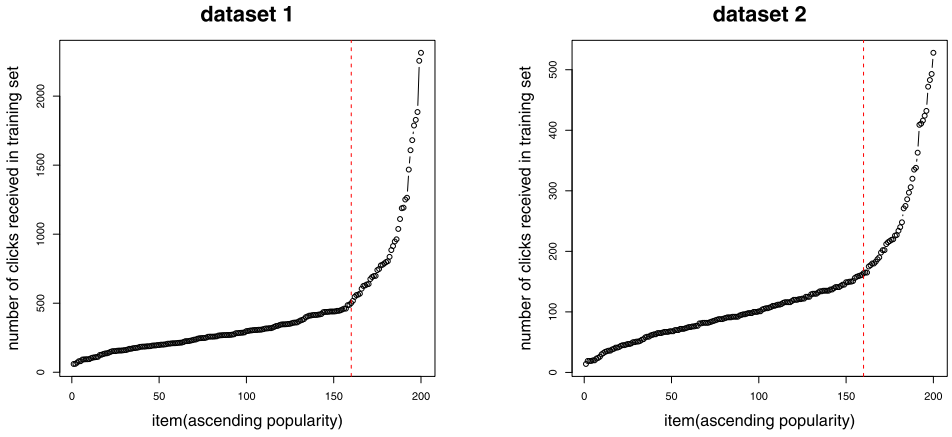
(a) Dataset 1 (b) Dataset 2

Fig. 6. Overall recommendation accuracies and fraction of recommendations performed vs. cut off posterior probabilities, blue line: fraction of recommendations, black line: overall recommendation accuracies. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Dataset 1, CF (b) Dataset 1, BMCD (c) Dataset 2, CF (d) Dataset 2, BMCD

Fig. 7. Histogram of correct recommendations. X-axis: item labels, arranged according to ascending item popularity (measured by the number of clicks received in the training set). Y-axis: recommendation frequency.



(a) Dataset 1 (b) Dataset 2

Fig. 8. Item popularity in the training set. X-axis: item labels, arranged according to ascending item popularity (measured by the number of clicks received in the training set). Y-axis: item popularity.

A comparison of intra-list similarity and novelty is shown in Table 9. Consistent with the simulation, BMCD recommends to each user a list of more diverse items, obtaining a lower intra-list similarity score compared to CF. At the same time, BMCD has a stronger ability to recommend more rare and novel items to the users.

5.4. Accuracy–diversity performances of BMCD and post-processing methods for CF

CF is not intrinsically a diversity-driven method. However, CF's diversity performance can be enhanced through post-processing. It is of interest to see if BMCD's accuracy–diversity performance

is effective compared to CF in combination with such post-processing methods. In this section, we choose two popular post-processing methods to combine with CF, one proposed by Adomavicius et al. [11] and one proposed by Ziegler et al. [8], and compare their accuracy–diversity performances with that of BMCD's. We refer to the two methods as CF+A and CF+Z respectively. We continue with the NRK datasets. Coverage is chosen as the diversity metric in this section, as it is a good indicator of both aggregated diversity of the recommender system, and level of personalization [9].

In Fig. 9, we plot the accuracy–diversity curve for the CF+A (blue) and CF+Z (black) methods, resulting from using a grid of tuning parameters. We can observe a clear trade-off between accuracy and diversity for both post-processing methods. BMCD's accuracy–diversity performance is represented in red and is a single point in the plots. For dataset 1, BMCD clearly outperforms CF+A: while fixing the accuracy level to that of BMCD, illustrated by the vertical red dotted line, BMCD achieves a higher diversity level, and vice versa. BMCD's accuracy–diversity performance is similar to the "elbow" of CF+Z's accuracy–diversity curve, at which point diversity is improved significantly without compromising much accuracy. However, beyond this point, poor choices of the tuning parameter can jeopardize CF+Z's accuracy without making much diversity improvement. BMCD achieves this accuracy–diversity balance without a tuning parameter, while maintaining a slight margin, as shown in the figure that the red point is positioned slightly above the black line. For dataset 2, the users are more similar to each other and therefore, it is a more challenging scenario to achieve diversity. We can clearly observe from Fig. 9(b) that BMCD outperforms both CF+A and CF+Z methods in the combined accuracy–diversity performance.

6. Further discussions and future works

In this paper, we have introduced and applied BMCD to make personalized recommendations based on clicking data. We have also compared the recommendation performances of BMCD with the popular Collaborative Filtering, in terms of accuracy and diversity.

Through a simulation study and an offline testing of a dataset, we have observed that BMCD and CF make recommendations with similar level of accuracy. BMCD, in addition, produces interpretable uncertainty estimation for each recommendation made. We showed that the uncertainty can be further exploited to improve the overall accuracy.

We have also assessed the recommendation diversity of both methods through measures of coverage, correct coverage, intra-list similarity and novelty. We have found that compared to CF, BMCD has a stronger ability to recommend diverse and rare items to the users, and considers more items for recommendations. We have also discovered that BMCD can achieve a similar or higher level of diversity compared to the post-processing methods, when a reasonable level of accuracy is required. BMCD however, does not require tuning, nor does it compromise its accuracy performance for higher diversity.

There are several reasons that explain BMCD's excellent diversity performance. First, BMCD, by construction, follows the restriction that all items clicked by a user, need to be among the user's top-ranked items, regardless of how unusual the clicked items are. This restriction enforces every user's uniqueness, and helps capture and preserve each individual user's "peculiar" behavior. CF on the other hand, often sacrifices the "unusual" items in the matrix factorization process, since the unusual items contribute less to the cost function.

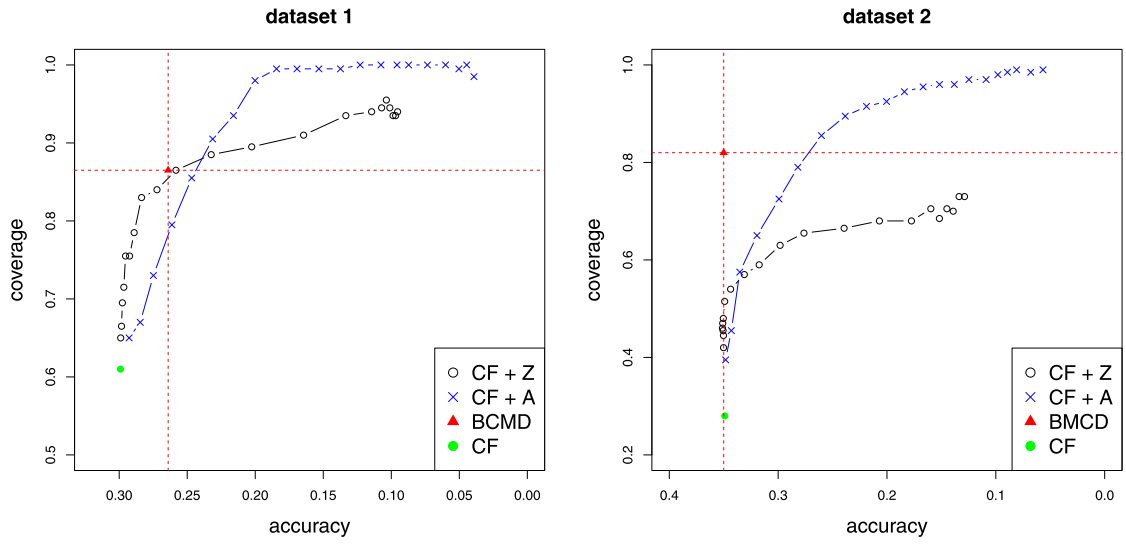
Second, BMCD is sensitive to the clicks in the sparse part of the dataset. BMCD contains the consensus parameter $\boldsymbol{\rho}$, and for the highly sparse part of the dataset, when an item receives a few clicks, these clicks will impact the distribution of the consensus parameter $\boldsymbol{\rho}$, and even more so, certain summary statistics such as the Maximum A Posteriori. The consensus, in turn, has an impact on the distribution and summary statistics of the individual users' latent full ranking vectors $\tilde{\boldsymbol{R}}_j$. However, it can also be a double-edge sword: when the sparse information turns out to be inaccurate or unrepresentative, it can decrease the method's recommendation accuracy.

It is therefore not surprising to observe that recommendations using BMCD are often more diverse, involving more rare items, even when user behaviors are rather homogeneous. BMCD's strong ability to capture the peculiarity of the users and its tendency to recommend less popular items partly explains why it was marginally outperformed in terms of accuracy by CF in the offline testing scenario, where the ground truth is limited by what the users have already seen and clicked. Rare items are often not yet discovered by the users, and it is almost impossible to verify the success of such recommendations in an offline testing. We are currently planning online testing of BMCD.

One of the biggest drawbacks of BMCD, which is based on MCMC, is scaling. BMCD does not scale well due to the huge amount of parameters to be estimated. The computing time required is dependent on the number of users N, the number of clusters C, as well as the number of iterations required to reach convergence. It takes 53 h to compute for 1 million iterations for the NRK dataset 1, and 14 h for dataset 2 using one core of the Intel Xeon e-8890 processor, running at 2.5 GHz. The iterative nature of MCMC also makes efficient parallelization more challenging since the computational overhead is very heavy. The Spark implementation of CF on the other hand, is very efficient. However, it can also be computationally costly if a thorough cross-validation is to be performed. For BMCD, in practice, it often happens that the cluster assignments for each user, $z_1, \ldots, z_N$ converge quite quickly. In the situation that most users do not switch cluster memberships often, after the cluster assignments have converged, we can split the dataset into C different segments, and compute BMCD algorithm without the clustering steps independently and in parallel. The reduction in the number of users, and the number of parameters needed to be estimated, can reduce computing time to at least 1/C of its original required computing time if the C clusters are similar in size. Another way to speed up the computation is by choosing smart starting points for the MCMC such that convergence can be reached in fewer iterations. We suggest, for example, that instead of randomly initializing the cluster assignment for each user, $z_1, \ldots, z_N$ can be initialized based on a K-means clustering. We are currently working on variational Monte Carlo versions of our algorithm, which we expect to reduce computational time very significantly.

Recently, Mao et al. [26] proposed a new approach to multiobjective recommendations using hypergraph rankings, aiming at recommending items that optimize a combination of objective functions, for example, price and quality. Such multiobjective methods could be expanded to a combination of accuracy and diversity. Exploiting social network relations between users, as done in [27], might also be an interesting avenue for further research, for example by giving neighboring users diverse recommendations.

In conclusion, BMCD can be considered as a valid alternative to traditional state-of-the-art collaborative filtering and its post-process enhancements when diversity in the recommendation is an important objective.



(a) Dataset 1 (b) Dataset 2

Fig. 9. Comparison of accuracy–diversity performances of BMCD and post-processing methods combined with CF.

Acknowledgments

We give special thanks to Linn Cecilie Solbergersen and the Norwegian Broadcasting Corporation for generously providing us research data and kind collaborations. We also thank Øystein Sørensen, Elja Arjas and Valeria Vitelli for fruitful discussions.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.knosys.2019.104960.

References

[1] D.G. Goldstein, D.C. Goldstein, Profiting from the long tail, Harv. Bus. Rev. 84 (6) (2006) 24–28.
[2] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J.R. Wakeling, Y.-C. Zhang, Solving the apparent diversity-accuracy dilemma of recommender systems, Proc. Natl. Acad. Sci. 107 (10) (2010) 4511–4515.
[3] J.-G. Liu, K. Shi, Q. Guo, Solving the accuracy–diversity dilemma via directed random walks, Phys. Rev. E 85 (1) (2012) 016118.
[4] L. Hou, K. Liu, J. Liu, R. Zhang, Solving the stability–accuracy–diversity dilemma of recommender systems, Physica A 468 (2017) 415–424.
[5] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on, IEEE, 2008, pp. 263–272.
[6] Z. Huang, H. Chen, D. Zeng, Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering, ACM Trans. Inf. Syst. 22 (1) (2004) 116–142.
[7] C.A. Gomez-Uribe, N. Hunt, The netflix recommender system: Algorithms, business value, and innovation, ACM Trans. Manag. Inf. Syst. 6 (4) (2016) 13.
[8] C.-N. Ziegler, S.M. McNee, J.A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: Proceedings of the 14th International Conference on World Wide Web, ACM, 2005, pp. 22–32.
[9] G. Adomavicius, Y. Kwon, Improving aggregate recommendation diversity using ranking-based techniques, IEEE Trans. Knowl. Data Eng. 24 (5) (2012) 896–911.
[10] V. Vitelli, Ø. Sørensen, M. Crispino, A. Frigessi, E. Arjas, Probabilistic preference learning with the Mallows rank model, J. Mach. Learn. Res. 18 (158) (2018) 1–49.
[11] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, IEEE Trans. Knowl. Data Eng. (6) (2005) 734–749.
[12] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, ACM, 2001, pp. 285–295.
[13] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2008, pp. 426–434.
[14] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer (8) (2009) 30–37.
[15] Y. Koren, R. Bell, Advances in collaborative filtering, in: Recommender Systems Handbook, Springer, 2015, pp. 77–118.
[16] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, et al., Mllib: Machine learning in apache spark, J. Mach. Learn. Res. 17 (1) (2016) 1235–1241.
[17] M.Ö. Karakaya, T. Aytekin, Effective methods for increasing aggregate diversity in recommender systems, Knowl. Inf. Syst. 56 (2) (2018) 355–372.
[18] A. Antikacioglu, R. Ravi, Post processing recommender systems for diversity, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2017, pp. 707–716.
[19] Q. Liu, M. Crispino, I. Scheel, V. Vitelli, A. Frigessi, Model-based learning from preference data, Annu. Rev. Stat. Appl. (6) (2019).
[20] S. Mukherjee, et al., Estimation in exponential families on permutations, Ann. Statist. 44 (2) (2016) 853–875.
[21] T. Joachims, Optimizing search engines using clickthrough data, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2002, pp. 133–142.
[22] S.M. McNee, J. Riedl, J.A. Konstan, Being accurate is not enough: how accuracy metrics have hurt recommender systems, in: CHI'06 Extended Abstracts on Human Factors in Computing Systems, ACM, 2006, pp. 1097–1101.
[23] G. Adomavicius, Y. Kwon, Overcoming accuracy–diversity tradeoff in recommender systems: A variance-based approach, in: Proceedings of WITS, Vol. 8, Citeseer, 2008.
[24] M. Ge, C. Delgado-Battenfeld, D. Jannach, Beyond accuracy: evaluating recommender systems by coverage and serendipity, in: Proceedings of the Fourth ACM Conference on Recommender Systems, ACM, 2010, pp. 257–260.
[25] Y.C. Zhang, D.Ó. Séaghdha, D. Quercia, T. Jambor, Auralist: introducing serendipity into music recommendation, in: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, ACM, 2012, pp. 13–22.
[26] M. Mao, J. Lu, J. Han, G. Zhang, Multiobjective e-commerce recommendations based on hypergraph ranking, Inform. Sci. 471 (2019) 269–287.
[27] M. Mao, J. Lu, G. Zhang, J. Zhang, Multirelational social recommendations via multigraph ranking, IEEE Trans. Cybern. 47 (12) (2016) 4049–4061.