Reinforcement Learning to Diversify Top-N Recommendation

Lixin Zou[1(✉)], Long Xia[2], Zhuoye Ding[2], Dawei Yin[2], Jiaxing Song[1], and Weidong Liu[1]

[1] Department of Computer Science and Technology, Tsinghua University, Beijing, China
{zoulx15,jxsong,liuwd}@mails.tsinghua.edu.cn
[2] Data Science Lab, JD.com, Beijing, China
{xialong,dingzhuoye}@jd.com, yindawei@acm.org

Abstract. In this paper, we study how to recommend both accurate and diverse top-N recommendation, which is a typical instance of the maximum coverage problem. Traditional approaches are to treat the process of constructing the recommendation list as a problem of greedy sequential items selection, which are inevitably sub-optimal. In this paper, we propose a reinforcement learning and neural networks based framework – **Div**ersify top-N Recommendation with **F**ast **M**onte **C**arlo **T**ree **S**earch (Div-FMCTS) – to optimize the diverse top-N recommendations in a global view. The learning of Div-FMCTS consists of two stages: (1) searching for better recommendation with MCTS; (2) generalizing those plans with the policy and value neural networks. Due to the difficulty of searching over extremely large item permutations, we propose two approaches to speeding up the training process. The first approach is pruning the branches of the search tree by the structure information of the optimal recommendations. The second approach is searching over a randomly chosen small subset of items to quickly harvest the fruits of searching in the generalization with neural networks. Its effectiveness has been proved both empirically and theoretically. Extensive experiments on four benchmark datasets have demonstrated the superiority of Div-FMCTS over state-of-the-art methods.

Keywords: Recommender system · Recommendation diversity · Monte Carlo Tree Search

1 Introduction

With the rapid growth of the Internet, the recommender system has become an indispensable part for many companies (e.g. Amazon[1], Netflix[2], JD.com[3]) to

L. Zou—Work performed during an internship at JD.com.
[1] https://www.amazon.com.
[2] https://www.netflix.com.
[3] https://www.jd.com.

G. Li et al. (Eds.): DASFAA 2019, LNCS 11447, pp. 104–120, 2019.
https://doi.org/10.1007/978-3-030-18579-4_7

overcome the information overload and to help customers find products. As a basic service, recommender system aims at filtering out the unattractive items and generating item recommendations in favor of user preferences. To cater to users, the recommender system is designed to find best-fitted products for users (e.g. highest rated movies, likest cellphone), which results in researches focusing on improving the accuracy of recommendations. However, the most accurate recommendations are sometimes not the recommendations that are most useful to users [22,33]. Fully exploiting learned user preferences without exploration of probably liking items might result in performance deterioration and falling into the cycle of recommending same items, such as always recommending cellphones if a new user only clicked a cellphone, which would disappoint users and cause the loss of users. Improving the recommendation diversity has been recognized as an effective way to alleviate this issue because it can broaden users' horizon and help users find new interesting items. Additionally, the platform can improve both their performance and users' satisfaction, which is a win-win situation.

During the last decade, various diversity-enhancing methods have been developed to increase diversity while maintaining the accuracy [1–3,5,41]. The previous methods for diversity improvement can be roughly divided into two categories: point-wise and list-wise approaches. The point-wise approaches usually involve two phases: generating the candidate items with the highest accuracy and re-ranking the items by heuristic methods [1], such as re-ranking with popularity [2]. For list-wise approach, it directly trains a supervised learning model to generate the recommendation lists with high accuracy and diversity [5]. However, the ground truth training labels are obtained by greedy selection. All in all, the existing approaches are treating the process of constructing the recommendation list as a problem of greedy sequential items selection. At each step, the algorithm iteratively selects the item with the highest marginal gain (which usually takes both accuracy and diversity into account) with respect to the selected items. However, the diversity of an item depends on the other recommended items, selecting an optimal ranking of items is a typical instance of the maximum coverage problem, a classical NP-hard problem in computational complexity theory [11]. Therefore, the recommendation lists produced by greedy items selection are inevitably sub-optimal. In general, the recommendation algorithm needs to explore the whole candidate item space, if the optimal recommendation list is mandatory. However, it is usually infeasible in real recommender systems as the huge space of possible recommendations: for selecting $N(N \sim 10^1)$ items from $K(K > 10^4)$ candidate items there exist $\frac{K!}{N!}(\gg 10^{10})$ different recommendation permutations. In this way, there is an urgent need for an algorithm suitable for recommendation diversity task that can efficiently search for optimal results in large spaces.

In this work, we will investigate to obtain accurate and diverse top-N recommendation in a sequential decision-making manner. To avoid local optimal solution and learn the optimal global policy efficiently, we propose a novel reinforcement learning and neural network based approach named **Div**ersify recommendation with **F**ast **M**onte **C**arlo **T**ree **S**earch (**Div-FMCTS**).

Specifically, Div-FMCTS formulates the generation of recommendation as a finite-horizon MDP and finds the optimal policy by iterating between two procedure: (1) searching the space of item permutations to find optimal top-N recommendations; (2) generalizing those searching results with neural networks. In searching stage, Monte Carlo Tree Search (MCTS) is proposed to heuristically search for the optimal recommendations. However, searching with MCTS is slow due to the extremely large size of candidate items. In this work, we propose two approaches to deal with large items set. The first approach, called structure pruning, uses the structure information of optimal recommendations to narrow down the search space of MCTS. The second approach, called problem decomposition, searches over a small randomly chosen subset of candidate items to quickly harvest the fruits of MCTS and generate more training sets for the imitation learning in the second stage. To prove its validity, we theoretically and empirically verify that diversifying recommendations can be equivalently solved by searching over many subset candidate item set. In the generalizing stage, a two-head GRU with a factorized similarity model [37] is proposed to approximate the searching results and estimate the goodness of current recommendations by tracking both user's temporary and intrinsic interests. Finally, extensive experiments on four benchmark MovieLens datasets show that Div-FMCTS can recommend diverse items at the same time maintaining high accuracy, which can not be achieved by either traditional methods or the state-of-the-art methods [2,3,9,17,25,26].

The rest of the paper is organized as follows. Section 2 discusses the formulation of diversify top-N recommendation and its difficulty. Section 3 describes our proposed model and learning algorithm in detail. In Sect. 4, we propose how to speed up the tree search process. We discuss related work on the diverse top-N recommendation in Sect. 5. Experimental results for the analysis and performance comparisons are presented in Sect. 6. We conclude with a summary of this work in Sect. 7.

2 Diverse Top-N Recommendation as MDP

2.1 Diverse Top-N Recommendation

Assuming that there are a set of users $\mathcal{U}$ and a set of items $\mathcal{I}$. For each item $i \in \mathcal{I}$, it lies on a set of topics/categories $\mathcal{Z}_i = \{\zeta_{i_j}\}_{j=1}^{Z}$, where ζ_{i_j} is one topic of item i and Z is the number of associated topics, usually ranging from 1 to 5. Given a user's rated history $\mathcal{O}_u = \{(i_{o_j}, r_{o_j})\}_{j=1}^{O}$ where r_{o_j} are numeric ratings, we are concerned with finding ranked N-items $\boldsymbol{i}_\nu = [i_{\nu_t}]_{t=1}^{N}$ from not being rated set $\mathcal{C}_u = \mathcal{I} - \{i_{o_j}\}_{j=1}^{O}$ that maximize the trade-off between accurate and diverse metrics as

$$\max_{\{i_{\nu_t}\}_{t=1}^{N} \subset \mathcal{C}_u} 2\frac{f(\boldsymbol{i}_\nu) \times g(\boldsymbol{i}_\nu)}{f(\boldsymbol{i}_\nu) + g(\boldsymbol{i}_\nu)}, \tag{1}$$

where $f, g : Y \to R$ are the metrics on recommendation accuracy and diversity respectively, e.g. NDCG, α-NDCG, Pairwise Accuracy Metric (PA), Normalized

Topic Coverage (NTC) [5,6,13]. $2\frac{f(\cdot)g(\cdot)}{f(\cdot)+g(\cdot)}$ is the F-measure between accuracy and diversity, which is being widely used as the evaluation metric on recommendation task [5].

Theoretically, recommendation diversity can be naturally stated as a bi-criterion optimization problem, and it is NP-hard [3]. To comprehend its difficulty, Fig. 1 shows a toy example, which includes the candidate set, the optimal ranking and greedy ranking of top-3 recommendations. The greedy solution, selecting the item that maximizing the relevance and diversity gain, is different with optimal recommendations. In practice, most previous approaches on recommendation diversity are based on greedy approximation, which sequentially selects a 'local-best' item from the remanent candidate set [8], which inevitably is the suboptimal recommendations. Therefore, it is urgent to view recommendation as a sequential selection process and to optimize the ranking in a global view.

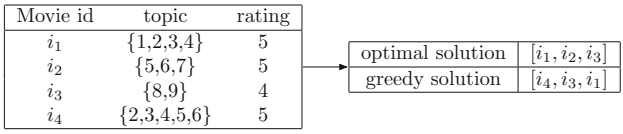

Fig. 1. A toy example of top-3 recommendation.

2.2 MDP Formulation of Diverse Recommendation

In this work, we formulate the diverse top-N recommendation as a Markov Decision Process (MDP) and optimize whole recommendations with reinforcement learning. A MDP is defined by $M = \langle S, A, P, R, \gamma \rangle$, where S is the state space, A is the action space, $P : S \times A \times S \rightarrow \mathbb{R}$ is the transition function with $p(s_{t+1}|s_t, a_t)$ being the probability of seeing state s_{t+1} after taking action a_t at s_t, $R : S \times A \rightarrow \mathbb{R}$ is the mean reward function with $r(s, a)$ being the immediate goodness of (s, a), and $\gamma \in [0, 1]$ is the discount factor. We design the state at time step t as a tuple $s_t = \{\boldsymbol{i}_{\nu_{<t}}, u, \mathcal{O}_u\}$ consisted of the user's rated history $\mathcal{O}_u$, user id u and previous $t-1$ recommendations $\boldsymbol{i}_{\nu_{<t}} = [i_{\nu_1}, i_{\nu_2}, \ldots, i_{\nu_{t-1}}]$. Given s_t, the recommendation agent chooses a action, e.g. the recommendation i_{ν_t}, and updates the state by appending i_{ν_t} to $\boldsymbol{i}_{\nu_{<t}}$. Then, a reward $r_t(s_t, i_{\nu_t})$ should be given for i_{ν_t}. To be consistent with the metrics defined in Eq. (1), we consider the reward function as the accuracy and diversity gain deriving from recommending item i_{ν_t} at position t as

$$r_t(s_t, i_{\nu_t}) = 2\frac{G(i_{\nu_t}) \times \alpha\text{-}G(i_{\nu_t})}{\log_2(t+1)(G(i_{\nu_t}) + \alpha\text{-}G(i_{\nu_t}))}, \tag{2}$$

where $r_t(s_t, i_{\nu_t})$ is the discounted F-measure of accuracy gain $G(i_{\nu_t})$ and diversity gain α-$G(i_{\nu_t})$ on t-th recommendation. $\frac{1}{\log_2(t+1)}$ is the discounted factor for the t-th recommendation, which has been widely used in metrics for recommendation evaluation [6,13]. Because of the discount factor and the fact that top-N

recommendation is a finite-horizon MDP problem, there is no need for additional discount on the reward function and we set the discounted factor as $\gamma = 1$.

To maximize the reward, we usually learn a policy $\pi : S \times A \rightarrow \mathbb{R}$, which maps the state s_t to the probability of recommending i_{ν_t} at time step t. With the policy π, the probability of recommending $\boldsymbol{i}_\nu$ can be modeled by the chain rule as,

$$p(\boldsymbol{i}_\nu) = \prod_{t=1}^{N} \pi(i_{\nu_t}|s_t). \tag{3}$$

We aim to learn a parameterized policy π_θ that assigns highest probability $p(\boldsymbol{i}_\nu)$ to the recommendation $\boldsymbol{i}_\nu^*$ with the biggest reward, e.g. maximizing the accuracy and diversity of top-N recommendations.

3 The Training Framework

To find the optimal recommendation policy π_*, we split the learning algorithm into two-stage iterative optimization problem: (1) given the parameterized policy π_θ, searching for better policies π_e by using π_θ as prior knowledge. In this work, we employ MCTS for finding the better policy π_e, which usually selects much stronger moves than the raw probabilities π_θ [30]; (2) updating the π_θ by minimizing the difference between π_e and π_θ. Then, the updated policy π_θ is used in the next iteration to make the π_e even stronger.

3.1 The Architecture of Policy and Value Neural Networks

For the success of training a good policy, the architecture of policy function is important. Apart from the policy function, the value function is also parameterized in our neural network, which evaluates the goodness of following current policy. We employ a two-head architecture for reinforcement learning, which has two advantages: (1) training with the information from the expert policy and estimating value can effectively avoid the overfitting; (2) the parameterized value function is helpful for fast estimation of leaf nodes' value in the tree search phases. In the following, we first discuss how to model user's preferences in states. Then, based on user's preferences, the policy and value functions have been built.

Preference Embedding. The state $s_t = \{\boldsymbol{i}_{\nu_{<t}}, u, \mathcal{O}_u\}$ contains user's historical information. We first computes state embedding as,

$$\boldsymbol{s}_t = \left[\boldsymbol{u}^\top, GRU\left(\mathcal{O}_u, \boldsymbol{i}_{\nu_{<t}}\right)^\top\right]^\top$$

where $\boldsymbol{u}^\top \in \mathcal{R}^{d_U}$ is a latent factor used to embed user's intrinsic preference. $GRU\left(\mathcal{O}_u, \boldsymbol{i}_{\nu<t}\right)$ is a GRU model to summary user's temporary preference on

first $t-1$ recommendation. The initial hidden state $\boldsymbol{h}_0 = GRU(\mathcal{O}_u)$ is set by a factorized similarity model [37] as,

$$\boldsymbol{h}_0 = \sum_{(i_{o_j}, r_{o_j}) \in \mathcal{O}_u} r_{o_j} \boldsymbol{i}_{o_j} + \boldsymbol{b}_u$$

where $\boldsymbol{b}_u \in \mathcal{R}^{d_H}$ is the bias term, $\boldsymbol{i}_{o_j} \in R^{d_H}$ is the embedding vector for item i_{o_j}.

The Policy and Value Function. Given the state embedding $\boldsymbol{s}_t$, the recommendation policy π_θ and value function v_θ are defined as

$$\begin{aligned} \pi_\theta\left(i_{\nu_t} | \boldsymbol{i}_{\nu_{<t}}, u, \mathcal{O}_u\right) &= \frac{\exp\left(\Gamma\left(i_{\nu_t} | s_t\right)\right)}{\sum_{i_j \in \mathcal{I}} \exp\left(\Gamma\left(i_j | s_t\right)\right)} \\ \Gamma\left(i_1, \ldots, i_I | \boldsymbol{s}_t\right) &= W_s \boldsymbol{s}_t + \boldsymbol{b}_s \\ v_\theta\left(s_t\right) &= \boldsymbol{V}^\top \boldsymbol{s}_t + b_v \end{aligned}$$

where $W_s \in \mathcal{R}^{(d_U + d_H) \times d_I}$, $\boldsymbol{V} \in \mathcal{R}^{d_U + d_H}$, $\boldsymbol{b}_s \in \mathcal{R}^{d_I}$ and b_v are the weight and bias terms. $\Gamma\left(i_j | s_t\right)$ is the score for recommending item i_j. π_θ is set as the softmax of score $\Gamma\left(i_j | s_t\right)$.

3.2 Searching for the Expert Policy

In the first phases, we employ MCTS to search for an improved policy compared with current π_θ. In the executing process, the MCTS planning process incrementally builds an asymmetric search tree that is guided in the most promising direction by an exploratory action-selection policy. This process usually consists of four phases: **selection**, **evaluation**, **expansion** and **backup** as

- **Selection:** At each node s_t, MCTS uses a selection policy to balance of exploitation of actions with high value estimates and exploration of actions with uncertain value estimates. To balance the trade-off of exploitation and exploration, a variant of PUCT is employed [30]. Specifically, an action i_{ν_t} is selected at node s_t so as to maximize action value plus uncertain estimate:

$$i_{\nu_t} = \arg\max_i (Q(s_t, i) + c_{puct} P(s_t, i) \frac{\sqrt{\sum_{b \in \mathcal{C}_u} N(s_t, b)}}{1 + N(s_t, i)})$$

  where $Q(s_t, a)$ is the mean action-value, $P(s_t, i) \frac{\sqrt{\sum_{b \in \mathcal{C}_u} N(s_t, b)}}{1 + N(s_t, i)}$ is the uncertain value estimate, $P(s_t, i)$ is the prior probability of selecting that edge, $N(s_t, i)$ is the visit count and c_{puct} is the constant determining the level of exploration.
- **Evaluation:** When the iteration reaches a leaf node s_t, the node $v(s_t)$ is evaluated either with the value function $v_\theta(s_t)$ or with the predefined performance measure if the node is the end of an episode and the human labels are available.

- **Expansion:** The leaf node s_t may be expanded. Each edge from the leaf position s_t is initialized as: $P(s_t, i_{\nu_t}) = p(i_{\nu_t}|s_t)$, $Q(s_t, i_{\nu_t}) = 0$, and $N(s_t, i_{\nu_t}) = 0$. In this paper all of the available actions of s_t are expanded.
- **Backup:** Finally, we recursively back-up the results in the tree nodes. Denoting the current forward trace in the tree as $\{s_1, i_1, s_2, .., s_t\}$. For $\forall(s_j, i_j)\, j < t$, we recursively update the $Q(s_j, i_j)$ as

$$Q(s_j, i_j) \leftarrow \frac{N(s_j, i_j) \times Q(s_j, i_j) + v(s_j)}{N(s_j) + 1}$$
$$v(s_j) \leftarrow v(s_{j+1}) + r_j(s_j, i_j).$$

This cycle of selection, evaluation, expansion and backup is repeated until the maximum iteration number has been reached. At this point, the best action is been chosen by selecting the action that leads to the most visited state (robust child) as follow:

$$i_{\nu_t} = \arg\max_{i_{\nu_t}} \pi_e(i_\nu|s_t)$$
$$\pi_e(i_\nu|s_t) = \frac{N(s_t, i_\nu)}{\sum_{b \in \mathcal{C}_u} N(s_t, b)}, \tag{4}$$

where $\boldsymbol{\pi}_e(s_t) = [\pi_e(i|s_t)]^\top_{i \in \mathcal{C}_{s_u}}$ is the improved policy after MCTS.

3.3 Generalizing with Policy and Value Neural Networks

The policy and value neural network is learned to mimic the searching policy π_e and predicts the expected return v_{π_e} of the following π_e. Specifically, the parameters θ are adjusted by gradient descent on a loss ℓ over the cross-entropy loss between $\boldsymbol{\pi}_\theta$ and the search policy $\boldsymbol{\pi}_e$, and the mean-squared error between the predicted value $v_\theta(s_t)$ and the sample discount reward z_t by following π_e in training stage as:

$$\ell(\theta) = \big\{(z_t - v_\theta(s_t))^2 - \boldsymbol{\pi}_e(s_t)^\top \log \boldsymbol{\pi}_\theta(s_t)$$
$$-(1 - \boldsymbol{\pi}_e(s_t))^\top \log(1 - \boldsymbol{\pi}_\theta(s_t))\big\} + \rho\|\boldsymbol{\theta}\|^2, \tag{5}$$

where $\boldsymbol{\pi}_\theta(s_t) = [\pi_\theta(i|s_t)]^\top_{i \in \mathcal{C}_{s_u}}$, and ρ is a parameter controlling the level of ℓ_2 weight regularization.

4 Dealing with Large-Scale Datasets

One disadvantage of DIV-MCTS is that the heuristic search over a large candidate item set is time-cost especially when dealing with massive datasets. For example, when the candidate item set contains 1000 items (not a very big item set), they will be nearly 1000 nodes to be evaluated at every expansion, which means that every node has nearly 1000 child nodes. Additionally, the expansion operation will be repeated over and over again until the maximum iteration is reached. Searching over such a broad tree would be problematic. To alleviate this problem, we propose two approaches to solve this problem.

4.1 MCTS with Structure Pruning

The first improvement is to prune the branches that seem impossible to be the optimal recommendations before expanding the leaf node. To reduce meaningless search, the structure of optimal recommendations has been incorporated into the expansion of the search tree to efficiently narrow down the searching space.

Lemma 1. *Assuming the undiscounted accuracy and diversity gain of item i at t is $h_t(i) = 2\frac{G(i)\times\alpha\text{-}G(i)}{G(i)+\alpha\text{-}G(i)}$, and the optimal ranking for $s_t = \{u, \mathcal{O}_u, \boldsymbol{i}_{\nu_{<t}}\}$ with $\mathcal{C}_u$ is $\boldsymbol{i}^*_{\nu_{\geq t}} = [i^*_{\nu_t}, i^*_{\nu_{t+1}}, \ldots, i^*_{\nu_N}]$, then we have $h_t(i^*_{\nu_t}) \geq h_t(i^*_{\nu_{t+1}}) \geq \cdots \geq h_t(i^*_{\nu_N})$.*

Proof. $\because$ the accuracy and diversity gain is discounted with t.
$\therefore$ the total reward is proportional to $\sum_{j=t}^{N} \frac{h_j(i^*_{\nu_j})}{\log_2(j+1)}$.
$\forall t_1, t_2 \in [t, N], t_1 < t_2$, if $h_{t_1}(i^*_{\nu_{t_1}}) < h_{t_1}(i^*_{\nu_{t_2}})$, then $[i^*_{\nu_t}, \ldots, i^*_{\nu_{t_2}}, \ldots, i^*_{\nu_{t_1}} \ldots, i^*_{\nu_N}]$ would be the optimal ranking, which is conflict with the assumption that $\boldsymbol{i}^*_{\nu_{\geq t}}$ is the optimal ranking. So, we have $h_t(i^*_{\nu_t}) \geq h_t(i^*_{\nu_{t+1}}) \geq \cdots \geq h_t(i^*_{\nu_N})$. □

Lemma 1 shows that the optimal recommendations must be descend on the undiscounted accuracy and diversity gains, which is a very valuable property for pruning unpromising leaf node in tree search. Combined with naive MCTS, we stop the expanding of a leaf node if the undiscounted accuracy and diversity gain is bigger than the parent node. Specifically, in FMCTS, the action node in the tree stores $\{N(s,i), P(s,i), Q(s,i), H(s,i)\}$, where $H(s,i)$ is the undiscounted immediate reward. During the expansion, the leaf node is expanded only when the undiscounted reward of edge $h(s_l, i_{\nu_l})$ is smaller than the parent node's $H(s_{l-1}, i_{\nu_{l-1}})$. Finally, the new edge is initialized with $H(s_l, i_{\nu_l}) = h(s_l, i_{\nu_l})$. Figure 2 plots the pruning expansion process.

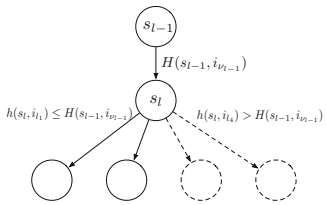

Fig. 2. The structure pruning of node-expansion on MCTS.

4.2 Acceleration with Problem Decomposition

Although the efficiency problem can be alleviated by pruning branches of the search tree, training over such large candidate item set can still be problematic. The intuitive idea is to reduce the size of the candidate item set, which is helpful for harvesting the fruits of tree search more frequently and sharing this knowledge over different tree search. To reduce the search space, we propose to decompose the top-N recommendation problem into many sub-problems

(e.g. searching over a small subset of candidate item set $\mathcal{C}_u$). Then, sharing the searching results by the generalization of the neural network. In Theorem 1, we prove that searching over a randomly chosen subset $\mathcal{C}_{s_u} \subset \mathcal{C}_u$ will discovery the optimal policy in the limits under a mild condition.

Theorem 1. *Assuming that the optimal policy for* $s_t = \{u, \mathcal{O}_u, \boldsymbol{i}_{\nu_{<t}}\}$ *under candidate item set* $\mathcal{C}_u$ *is* π_e. *If the searching over a randomly chosen subset* $\mathcal{C}_{s_u} \subset \mathcal{C}_u$ *satisfying* $(1 + \frac{M}{K})\frac{(M-1)!(K-N)!}{(K-1)!(M-N)!} \geq 1$, *where* $M = |\mathcal{C}_{s_u}|$, $K = |\mathcal{C}_u|$, $K, M >> N$. $\forall p, q \in \mathcal{C}_u$ *with* $\pi_e(i^p_{\nu_t}|s) \geq \pi_e(i^q_{\nu_t}|s)$, *we have* $\mathbb{E}[\hat{\pi}_e(i^p_{\nu_t}|s)] \geq \mathbb{E}[\hat{\pi}_e(i^q_{\nu_t}|s)]$, *where* $\hat{\pi}_e$ *is the optimal planning policy under* $\mathcal{C}_{s_u}$.

Proof. Assuming that the optimal ranking under $\mathcal{C}_u$ starts with $i^p_{\nu_t}$ is $\boldsymbol{i}^p_{\nu_{\geq t}} = [i^p_{\nu_t}, i^p_{\nu_{t+1}}, \ldots, i^p_{\nu_N}]$. From Lemma 1, we can infer that $\mathbb{E}[\hat{\pi}_e(i^p_{\nu_t}|s)] \geq \mathbb{E}[\hat{\pi}_e(i^p_{\nu_{t+i}}|s)], \forall i \geq 1$.

For a item $i^q_{\nu_t}(i^q_{\nu_t} \notin \boldsymbol{i}^p_{\nu_{\geq t}})$, the probability of $\boldsymbol{i}^p_{\nu_{\geq t}} \subset \mathcal{C}_{s_u}$ is $p(\boldsymbol{i}^p_{\nu_{\geq t}} \subset \mathcal{C}_{s_u}) = \frac{C^{M+t-(N+1)}_{K+t-(N+1)}}{C^M_K}$ and $p(\boldsymbol{i}^p_{\nu_{\geq t}} \not\subset \mathcal{C}_{s_u}$ and $i^q_{\nu_t} \in \mathcal{C}_{s_u}) = (1 - \frac{C^{M+t-(N+1)}_{K+t-(N+1)}}{C^M_K})\frac{C^{M-1}_{K-1}}{C^M_K}$.

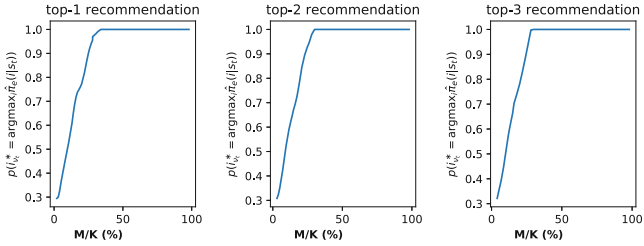

Fig. 3. The influence of $\frac{M}{K}$ on the mismatch of $\hat{\pi}_e$ and π_e.

Then,

$$
\begin{aligned}
&\mathbb{E}[\hat{\pi}_e(i^p_{\nu_t}|s)] \geq \mathbb{E}[\hat{\pi}_e(i^q_{\nu_t}|s)] \\
\Leftarrow\ & p(\boldsymbol{i}^p_{\nu_{\geq t}} \subset \mathcal{C}_{s_u}) \geq p(\boldsymbol{i}^p_{\nu_{\geq t}} \not\subset \mathcal{C}_{s_u} \text{ and } i^q_{\nu_t} \in \mathcal{C}_{s_u}) \\
\Leftarrow\ & \frac{C^{M+t-(N+1)}_{K+t-(N+1)}}{C^M_K} \geq (1 - \frac{C^{M+t-(N+1)}_{K+t-(N+1)}}{C^M_K})\frac{C^{M-1}_{K-1}}{C^M_K} \\
\Leftrightarrow\ & 1 \leq (1 + \frac{M}{K})\frac{(M-1)!(K-N)!}{(K-1)!(M-N)!}
\end{aligned} \tag{6}
$$

So, $\forall p, q \in \mathcal{C}_u$ with $\pi_e(i^p_{\nu_t}|s) \geq \pi_e(i^q_{\nu_t}|s)$, we have $\mathbb{E}[\hat{\pi}_e(i^p_{\nu_t}|s)] \geq \mathbb{E}[\hat{\pi}_e(i^q_{\nu_t}|s)]$. □

The condition in Eq. (6) is a very restrictive, since we assume that $i^q_{\nu_t}$ would be the optimal solution with $\boldsymbol{i}^p_{\nu_t} \not\subset \mathcal{C}_{s_u}$. To get a loosen condition, we study the influence of $\frac{M}{K}$ on $\mathbb{E}[\hat{\pi}_e]$ by simulating on a synthetic dataset. For the construction of synthetic dataset, we randomly create multi-candidate item sets $\mathcal{C}_{s_u}$ with

100 items, 20 topics, and 5 ratings, and searching over a randomly chosen subset $\mathcal{C}_{s_u}$ with a fixed size M. Then, MCTS is employed to generate expert policy $\hat{\pi}_e$. Finally, we study the difference between $\hat{\pi}_e$ and π_e by comparing the recommendation chosen by π_e and $\hat{\pi}_e$, because it determines the policies' performances. The simulation code is available on https://github.com/zoulixin93/FMCTS. In Fig. 3, we show the difference between $\hat{\pi}_e$ and π_e by comparing the recommendation made by $\hat{\pi}_e$ and π_e. As we can see, the bigger $\frac{M}{K}$ means greater possibility of choosing optimal recommendation $i^*_{\nu_t}$ with $\hat{\pi}_e$. For $M/K \geq 0.25$, the recommendation made by π_e and $\hat{\pi}_e$ would be same, which is much more loosen than the condition in Eq. (6). Combining these two techniques with the training framework, the resulting detailed training procedure is provided in Algorithm 1.

5 Related Work

In this section, we briefly review works related to our study. In general, the related work can be mainly grouped into the following three categories: recommendation diversity, deep learning based methods, and reinforcement learning based methods.

ALGORITHM 1. Training of Div-FMCTS

Input: Training dataset $\mathcal{D} = \{(u^{(j)}, \mathcal{O}_u^{(j)}, \mathcal{C}_u^{(j)})\}_{j=1}^{D}$, learning rate δ, maximum iteration number Λ, trade-off parameter c_{puct}, reward function r, subset size M, buffer size B.
Output: Well-trained π_θ.

Randomly initialize parameters $\boldsymbol{\theta} \leftarrow \mathcal{U}(-0.1, 0.1)$
Initialize a queue Ψ to capacity B
repeat
  for $\{(u, \mathcal{O}_u, \mathcal{C}_u)\} \in \mathcal{D}$ **do**
    Randomly chose M items from $\mathcal{C}_u$ as the candidate set $\mathcal{C}_{s_u}$
    for *t from 1 to N* **do**
      Set $s_t \leftarrow \{u, \mathcal{O}_u, \boldsymbol{i}_{\nu_{<t}}\}$
      Do MCTS under s_t until the maximum iteration number Λ is reached
      Choose $i_{\nu_t} = \arg\max_{i \in \mathcal{C}_{s_u}} \pi_e(i|s_t)$
      Calculate the reward r_t with Equation (2)
    end
    $z = 0$
    for *t from N to 1* **do**
      $z \leftarrow r_t + \gamma z$
      Store the tuple $\{u, \mathcal{O}_u, \boldsymbol{i}_{\nu_{<t}}, z, \boldsymbol{\pi}_e\}$ in Ψ
    end
    Sample a mini-batch of $\{u, \mathcal{O}_u, \boldsymbol{i}_{\nu_{<t}}, z, \boldsymbol{\pi}_e\}$ from Ψ
    Update parameters $\theta \leftarrow \theta - \delta \frac{\partial \ell(\theta)}{\partial \theta}$ with Equation (5)
  end
until *converge*;

5.1 Recommendation Diversity

Recommendation diversity refers to aggregate diversity and individual diversity. **Aggregate diversity** means the overall diversity of all users and is being measured by using absolute and relative long-tail metrics [1,4]. Adomavicius et al. [1] firstly proposed a graph-theoretic approach for maximizing aggregate

recommendation diversity based on maximum flow or maximum bipartite matching computations. **Individual diversity** aims at maximizing the recommendation diversity for each user and is usually being measured by topic coverage or α-$NDCG$ [19,41]. In this study, the focus is on individual diversity.

The methods for individual diversity improvement can be divided into two categories. The first category is the point-wise methods, which involves two phases: generating the accuracy estimates of items and heuristically ranking the items considering both the accuracy and diversity. For example, a popular heuristic re-ranking approach has been proposed in [2]. Ziegler et al. [41] came up with a novel approach named topic diversification to balance and diversify personalized recommendations. Azin et al. [3] proposed a greedy selection method for diversifying individual recommendation. The second category is the list-wise approach, which directly generates a subset accurate and diverse recommendation. For example, Cheng et al. [5] presented a supervised learning method, which combined parameterized matrix factorization and structural learning to seek significant improvement in diversity while maintaining accuracy. However, point-wise methods require fine-tuning features for heuristic methods. Additionally, leveraging heuristic method possibly only find the suboptimal solution to the problem. For list-wise approach, ground truth labels for training is needed, which is usually obtained by heuristic methods and can not scale for the large datasets.

5.2 Deep Learning Based Recommendation

With the booming success of deep learning in computer vision and natural language processing [15], there are many works trying to apply neural networks for recommender systems. The first work is proposed in [27], which uses a variant of the restricted boltzmann machine for collaborative filtering (RBM-CF). Similar to RBM, Autoencoders and the stacked-denoising autoencoders based recommender systems have been analyzed in [18,31], which are compact and efficiently trainable model. Recently, Zheng et al. [40] introduced a neural autoregressive approach for collaborative filtering (CF-NADE), which has beat the state-of-the-art methods on MovieLens and Netflix datasets. Xue et al. [12] proposed to use DNN to learn the latent factors from the sparse user-item matrix and make recommendations based on the relevance measure of user-item latent factors. There are also many hybrid methods, which combine matrix factorization (MF) with DNN to solve the problem of cold start, such as [34,35]. Apart from the traditional recommender, recurrent neural networks have been widely employed to sequential recommender system, such as GRU4Rec [10], NARM [17]. In spite of its success, all these methods are mainly concerned with increasing the recommendation accuracy in traditional user-item setting and none of these works could directly optimize the diversity of recommendation.

5.3 Reinforcement Learning Based Methods

Since the great success in playing Atari 2600 video games and Go at a superhuman level directly from image pixels [23,24,29], deep reinforcement learning has attracted enormous research interests. However, due to the difficulty of dealing with larger action space, there are just a few works applying the techniques of deep reinforcement learning to the recommender system. For example, in [28], an MDP based recommender system has been proposed, which recommends most likely purchasing items by using a sample-based transition distribution. It requires huge samples to learn the transition distribution and is limited to solve the problem with small state space. To maximize the accumulative rewards, Mahmood et al. [21] adopted the reinforcement learning technique to optimize the responses of users in a conversational recommender. Sunehag et al. [32] introduced slate-MDPs and successfully addressed sequential decision problems with high-dimensional combinatorial slate-action spaces. Arnold et al. [7] proposed a actor-critic based model to deal with the problem of large discrete action spaces. It leverages the policy gradient to learn a policy mapping from state to a continuous action and chooses the action with the biggest Q-value in candidate actions, which is generated by finding the k-nearest neighbors of continuous action in discrete action space. In [20], it proposed to use the latent factors learned from probability matrix factorization (PMF) as the belief states and approximate the Q-value by using a neural network with the latent factors as input.

Apart from the recommendation scene, reinforcement learning has also been applied to information retrieval. For example, in [38], MDP has been adapted to solve the search result diversification by using policy gradient. Yisong et al. formulated [39] the real-time learning in the search engine as a dueling bandit.

6 Experiments

In this section, we conduct experiments to demonstrate the effectiveness of our proposed method. We also do some extensive experiments to analysis the effectiveness of structure pruning and the decomposition of diversity recommendation.

6.1 Experimental Settings

Dataset. Since we mainly focus on the recommendation diversity, the chosen datasets have to satisfy a fundamental requirement: each item should be associated with topics for generating training instances and evaluating the predictions. In this study, fours benchmark datasets MovieLens 100k, 1M, 10M and 20M[4] have been chosen to evaluate Div-FMCTS. The statistics of datasets are shown in Table 1.

[4] https://grouplens.org/datasets/movielens/.

Table 1. Summary statistics of datasets.

Dataset	#user	#item	#rating	#topic	average topics
MovieLens100k	943	1,682	100,000	19	1.72
MovieLens1M	6,040	3,900	1,000,209	19	1.67
MovieLens10M	71,567	10,681	10,000,054	18	2.02
MovieLens20M	138,493	26,774	20,000,263	20	2.00

Baselines. First of all, we choose DCF [5] as our main baseline, which is a strong baseline on diverse top-N recommendation. Besides, since Div-FMCTS is a reinforcement learning based algorithm, we also choose REINFORCE [36] as our baseline. Additionally, we also compare Div-FMCTS with a RNN-based recommendation model NARM [17] and the most popular and widely used matrix factorization (MF) [16].

Evaluation. Like most diverse recommender systems [5], we choose $NDCG@N$ and α-$NDCG@N$ as the metrics to evaluate the effectiveness on recommendation accuracy and diversity. Furthermore, we also utilize the F-measure, named $F_{NDCG}@N$, to assess the trade-off performance with consideration of both accuracy and diversity.

Parameter Settings. For each datasets, 10% of the ratings are randomly selected as the test set, leaving the remaining 90% of the ratings as the training set. Among the ratings in the training set, 5% are used as validation set, which is used to find the optimal hyperparameters. For neural network, we randomly initialized model parameters with a uniform distribution ranging from -0.01 to 0.01, optimizing the model with mini-batch Adam [14]. In MCTS, the trade-off parameter for exploitation and exploration is set as $c_{puct} = 2000$. The $\frac{M}{K}$ is set as 0.2 for $K \geq 100$. The maximum iteration number of MCTS is 1000 on experiments. Following the setting of DCF [5], we mainly consider top-3 recommendation problem, where a list of 3 items is recommended for each user. The models are defined and trained in Tensorflow on a Nvidia Tesla P40 GPU. The source code for Div-FMCTS can be found at Github: https://github.com/zoulixin93/FMCTS.

6.2 Experimental Results

Comparison Against Baselines. We compared Div-FMCTS with state-of-the-art methods. The results of all methods over the benchmark datasets in terms of three metrics are shown in Table 2. From the results we can see that, Div-FMCTS outperformed all of the baseline methods on α-$NDCG@3$ and $F_{NDCG}@3$ by a large margin. And also, even if it is not the best result, it is not much different from the best result in term of α-$NDCG@3$. We conducted significance testing (t-test) on the improvements of our approaches over the all baselines. ▼ denotes

strong significant divergence with p-value<0.05. The improvements are significant, in terms of α-$NDCG$@3 and F_{NDCG}@3. The results indicate that our model Div-FMCTS can indeed improve recommendation diversity while maintaining recommendation accuracy.

Table 2. Comparison between Div-FMCTS and state-of-the-art methods.

Methods	MovieLens100k			MovieLens1M		
	$NDCG$@3	α-$NDCG$@3	F_{NDCG}@3	$NDCG$@3	α-$NDCG$@3	F_{NDCG}@3
Div-FMCTS	0.8080	**0.8364**	**0.8222**	0.8288	**0.8277**	**0.8282**
REINFORCE	0.7523$^{\blacktriangledown}$	0.6291$^{\blacktriangledown}$	0.6888$^{\blacktriangledown}$	0.7546$^{\blacktriangledown}$	0.6075$^{\blacktriangledown}$	0.6731$^{\blacktriangledown}$
DCF[a]	0.7070	0.7414	0.7238	0.7415	0.6735	0.7059
NARM	**0.8213**	0.5208$^{\blacktriangledown}$	0.6374$^{\blacktriangledown}$	**0.8305**	0.4935$^{\blacktriangledown}$	0.6191$^{\blacktriangledown}$
MF	0.8041$^{\blacktriangledown}$	0.5346$^{\blacktriangledown}$	0.6422$^{\blacktriangledown}$	0.8023$^{\blacktriangledown}$	0.5011$^{\blacktriangledown}$	0.6169$^{\blacktriangledown}$
	MovieLens10M			MovieLens20M		
Div-FMCTS	**0.8274**	**0.8201**	**0.8237**	0.8053	**0.8278**	**0.8165**
REINFORCE	0.7277$^{\blacktriangledown}$	0.6442$^{\blacktriangledown}$	0.6834$^{\blacktriangledown}$	0.7325$^{\blacktriangledown}$	0.6232$^{\blacktriangledown}$	0.6734$^{\blacktriangledown}$
NARM	0.8206	0.5098$^{\blacktriangledown}$	0.6289$^{\blacktriangledown}$	**0.8310**	0.4782$^{\blacktriangledown}$	0.6071$^{\blacktriangledown}$
MF	0.8047$^{\blacktriangledown}$	0.4774$^{\blacktriangledown}$	0.5992$^{\blacktriangledown}$	0.8179$^{\blacktriangledown}$	0.4702$^{\blacktriangledown}$	0.5971$^{\blacktriangledown}$

[a]The experimental results of DCF is taken from [5].

Comparison Between FMCTS and Naive MCTS. It is a key question that how much impact on the learned policy $\hat{\pi}_\theta$ through MCTS over a subset of candidate item set. To investigate its influence, we train a policy $\hat{\pi}_\theta$ by MCTS over the complete candidate set $\mathcal{C}_u$, named Div-MCTS. For the sake of fairness, Div-FMCTS and Div-MCTS are trained with same hyperparameters. In Fig. 4, we show the F_{NDCG}@3 w.r.t. the training iteration. The blue one is the performance of training with problem decomposition and the orange one is training with the complete candidate item set $\mathcal{C}_u$. From Fig. 4, the F_{NDCG}@3 is nearly same on these four benchmark dataset, which means that training over a subset of candidate set does not influence the performance. Meanwhile, Div-FMCTS converges faster than Div-MCTS, which indicates that MCTS over a subset of candidate item set can do the optimization more efficiently.

The Advantage of FMCTS on Searching Time. To examine the advantage of Div-MCTS, we compared average searching time of four different MCTS: **(1)** Div-MCTS is searching over the whole candidate item set without structure pruning; **(2)** Pruning the MCTS with the descend immediate reward – Div-PMCTS; **(3)** MCTS with problem decomposition, named Div-DMCTS; **(4)** The proposed Div-FMCTS, searching with a smaller candidate set and structure pruning. In our experiments, the tree searches are running on a server with 64 GB memory and an INTEL Core i7-7800X CPU, the maximum iteration number of MCTS are 1000. Figure 5 plots the average searching time over these four datasets. We can see that the searching time of Div-FMCTS and Div-DMCTS do not increase too much with respect to the increasing number of items. In contrast, the average searching time of Div-MCTS and Div-PMCTS grow fast w.r.t the increasing size of datasets. Additionally, using structure pruning will decrease planning time.

From the results, we can see that searching over a subset of candidate item set and pruning branches with descend immediate reward is helpful on saving the training time.

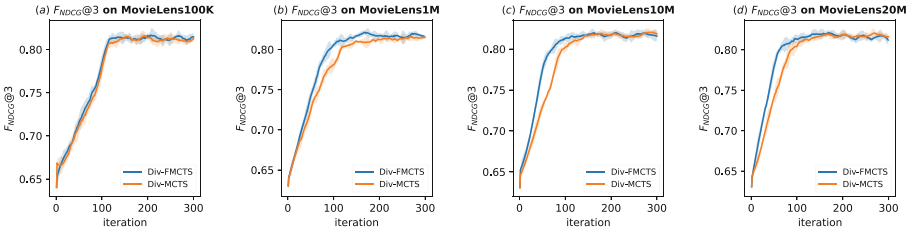

Fig. 4. The influence of planning over a subset of candidate item set.

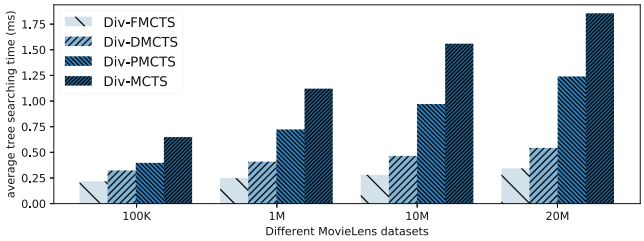

Fig. 5. The average tree searching time of four different MCTS.

7 Conclusion

In this paper, we have proposed a novel MDP based method – Div-FMCTS – to directly maximize the trade-off between accuracy and diversity for the top-N recommendation. The learning of Div-FMCTS is decomposed into iteration between searching with MCTS and generalizing those plans with a policy-value neural network. To faster the tree search process, a novel structure pruning technique has been incorporated into the node expansion to narrow down searching space for MCTS. Additionally, we theoretically and empirically verify that the diversity recommendation can be equivalently solved by planning under the sub-problems. Extensive experiments on four benchmark datasets have demonstrated the effectiveness of our proposed method.

References

1. Adomavicius, G., Kwon, Y.: Maximizing aggregate recommendation diversity: a graph-theoretic approach. In: RecSys, pp. 3–10 (2011)
2. Adomavicius, G., Kwon, Y.: Improving aggregate recommendation diversity using ranking-based techniques. TKDE **24**(5), 896–911 (2012)
3. Ashkan, A., Kveton, B., Berkovsky, S., Wen, Z.: Optimal greedy diversity for recommendation. In: IJCAI, pp. 173–182 (2015)

4. Brynjolfsson, E., Hu, Y., Smith, M.D.: Research commentary-long tails vs. superstars: the effect of information technology on product variety and sales concentration patterns. Inf. Syst. Res. **21**(4), 736–747 (2010)
5. Cheng, P., Wang, S., Ma, J., Sun, J., Xiong, H.: Learning to recommend accurate and diverse items. In: WWW, pp. 183–192 (2017)
6. Clarke, C.L.A., et al.: Novelty and diversity in information retrieval evaluation. In: SIGIR, pp. 659–666 (2008)
7. Dulac-Arnold, G.: Deep reinforcement learning in large discrete action spaces. arXiv preprint arXiv:1512.07679 (2015)
8. Feige, U.: A threshold of ln n for approximating set cover. JACM **45**(4), 634–652 (1998)
9. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborative filtering. In: WWW, pp. 173–182 (2017)
10. Hidasi, B., Karatzoglou, A., Baltrunas, L., Tikk, D.: Session-based recommendations with recurrent neural networks. arXiv (2015)
11. Hochba, D.S.: Approximation algorithms for NP-hard problems. ACM SIGACT News **28**(2), 40–52 (1997)
12. Xue, H.-J., Dai, X.-Y., Zhang, J., Huang, S., Chen, J.: Deep matrix factorization models for recommender systems. In: IJCAI, pp. 764–773 (2017)
13. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Trans. Inf. Syst. (TOIS) **20**(4), 422–446 (2002)
14. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
15. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
16. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: NIPS 2001, pp. 556–562 (2001)
17. Li, J., Ren, P., Chen, Z., Ren, Z., Ma, J.: Neural attentive session-based recommendation. arXiv preprint arXiv:1511.06939 (2015)
18. Li, S., Kawale, J., Fu, Y.: Deep collaborative filtering via marginalized denoising auto-encoder. In: CIKM, pp. 811–820 (2015)
19. Liu, T.-Y., et al.: Learning to rank for information retrieval. Found. Trends® Inf. Retr. **3**(3), 225–331 (2009)
20. Lu, Z., Yang, Q.: Partially observable Markov decision process for recommender systems. arXiv preprint arXiv:1608.07793 (2016)
21. Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies. In: HT 2009, pp. 73–82 (2009)
22. McNee, S.M., Riedl, J., Konstan, J.A.: Being accurate is not enough: how accuracy metrics have hurt recommender systems. In: CHI, pp. 1097–1101 (2013)
23. Mnih, V., et al.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
24. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
25. Rendle, S.: Factorization machines. In: ICDM, pp. 995–1000 (2007)
26. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: NIPS, pp. 1257–1264 (2007)
27. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted Boltzmann machines for collaborative filtering. In: ICML, pp. 791–798 (2007)
28. Shani, G., Heckerman, D., Brafman, R.I.: An MDP-based recommender system. JMLR **6**, 1265–1295 (2005)

29. Silver, D., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
30. Silver, D., et al.: Mastering the game of go without human knowledge. Nature **550**, 354–360 (2017)
31. Strub, F., Mary, J.: Collaborative filtering with stacked denoising autoencoders and sparse inputs. In: NIPS, pp. 111–112 (2015)
32. Sunehag, P., Evans, R., Dulac-Arnold, G., Zwols, Y., Visentin, D., Coppin, B.: Deep reinforcement learning with attention for slate Markov decision processes with high-dimensional states and actions. arXiv preprint arXiv:1512.01124 (2015)
33. Szpektor, I., Maarek, Y., Pelleg, D.: When relevance is not enough: promoting diversity and freshness in personalized question recommendation. In: WWW, pp. 173–182 (2013)
34. den Oord, A.V., Dieleman, S., Schrauwen, B.: Deep content-based music recommendation. In: NIPS, pp. 2643–2651 (2015)
35. Wang, H., Wang, N., Yeung, D.-Y.: Collaborative deep learning for recommender systems. In: SIGKDD, pp. 1235–1244 (2015)
36. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. **8**(3–4), 229–256 (1992)
37. Wu, Y., DuBois, C., Zheng, A.X., Ester, M.: Collaborative denoising auto-encoders for top-n recommender systems. In: WSDM, pp. 153–162 (2016)
38. Xia, L., Xu, J., Lan, Y., Guo, J., Zeng, W., Cheng, X.: Adapting Markov decision process for search result diversification. In: SIGIR, pp. 535–544 (2017)
39. Yue, Y., Joachims, T.: Interactively optimizing information retrieval systems as a dueling bandits problem. In: ICML, pp. 1201–1208 (2009)
40. Zheng, Y., Tang, B., Ding, W., Zhou, H.: A neural autoregressive approach to collaborative filtering. In: ICML, pp. 764–773 (2016)
41. Ziegler, C.-N., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. In: WWW, pp. 22–32 (2005)