End-to-End Deep Reinforcement Learning based Recommendation with Supervised Embedding

Feng Liu[*][†][‡]
Harbin Institute of Technology, Shenzhen
fengliu@stu.hit.edu.cn

Huifeng Guo[*]
Noah's Ark Lab, Huawei
huifeng.guo@huawei.com

Xutao Li[*][‡]
Harbin Institute of Technology, Shenzhen
lixutao@hit.edu.cn

Ruiming Tang
Noah's Ark Lab, Huawei
tangruiming@huawei.com

Yunming Ye[‡][§]
Harbin Institute of Technology, Shenzhen
yeyunming@hit.edu.cn

Xiuqiang He
Noah's Ark Lab, Huawei
hexiuqiang1@huawei.com

ABSTRACT

The research of reinforcement learning (RL) based recommendation method has become a hot topic in recommendation community, due to the recent advance in interactive recommender systems. The existing RL recommendation approaches can be summarized into a unified framework with three components, namely *embedding component* (EC), *state representation component* (SRC) and *policy component* (PC). We find that EC cannot be nicely trained with the other two components simultaneously. Previous studies bypass the obstacle through a pre-training and fixing strategy, which makes their approaches unlike a real end-to-end fashion. More importantly, such pre-trained and fixed EC suffers from two inherent drawbacks: (1) Pre-trained and fixed embeddings are unable to model evolving preference of users and item correlations in the dynamic environment; (2) Pre-training is inconvenient in the industrial applications. To address the problem, in this paper, we propose an **E**nd-to-end **D**eep **R**einforcement learning based **R**ecommendation framework (EDRR). In this framework, a supervised learning signal is carefully designed for smoothing the update gradients to EC, and three incorporating ways are introduced and compared. To the best of our knowledge, we are the first to address the training compatibility between the three components in RL based recommendations. Extensive experiments are conducted on three real-world datasets, and the results demonstrate the proposed EDRR effectively achieves the end-to-end training purpose for both policy-based and value-based RL models, and delivers better performance than state-of-the-art methods.

KEYWORDS

Recommendation, Reinforcement Learning, End-to-End, Supervised Embedding

ACM Reference Format:
Feng Liu, Huifeng Guo, Xutao Li, Ruiming Tang, Yunming Ye, and Xiuqiang He. 2020. End-to-End Deep Reinforcement Learning based Recommendation with Supervised Embedding. In *The Thirteenth ACM International Conference on Web Search and Data Mining (WSDM '20), February 3–7, 2020, Houston, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3336191.3371858

1 INTRODUCTION

Recently, interactive recommender systems (IRS) attract much attention in personalization services, e.g., Pandora radio, musical.ly videos and Instagram feeds. In such platforms, items are sequentially delivered to users upon their previous feedbacks. Due to the new working mechanism, IRS faces two challenges. First, the recommendation-feedback interaction is a dynamic process, where the user preference and item correlations may evolve over time. Hence, it needs great dynamic modeling capability. Second, in addition to immediate user satisfactory, IRS has to take the long-term performance into account. Focusing merely on immediate rewards loses the chance to throughly exploit user preference in the long-run. However, conventional recommendation methods, from the static content-based filtering [25] and collaborative filtering (CF) [34, 42] to multi-armed bandit methods [2, 15, 33, 36] and recently deep learning models [5, 8, 26, 44], fail to handle above challenges.

Thanks to the inherent capabilities of dynamic modeling and long-term planning, Reinforcement Learning (RL) [32], which has shown great potential in various challenging sequential decision making scenarios, such as game playing [24, 30], robotics [12], real-time ads bidding [10], neural architecture searching [45], has been recently introduced to IRS [3, 4, 7, 19, 38–41, 43].

Though these RL approaches are developed in a very different manner and with their-own features, they can be unified into a framework with three components, namely *embedding component* (EC), *state representation component* (SRC) and *policy component* (PC). EC accounts for mapping each item and user into a low-dimensional embedding vector. Then, the environment state is

[*] Co-first authors with equal contributions.
[†] The work was done when Feng Liu worked as an intern in Noah's Ark Lab, Huawei.
[‡] Shenzhen Key Laboratory of Internet Information Collaboration, Harbin Institute of Technology, Shenzhen.
[§] Corresponding author.

WSDM '20, February 3–7, 2020, Houston, TX, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-6822-3/20/02…$15.00
https://doi.org/10.1145/3336191.3371858

modeled in the SRC, by different state representation strategies, e.g., fully-connected neural network [7, 43], recurrent neural network [3, 40], and carefully designed interaction strategy [19] between the embeddings of users and their consumed items, etc. Finally, the encoded state is utilized to learn the recommendation policy in PC.

As the fundamental layer, embedding component (EC) plays a crucial role and its quality directly affects the final performance. However, with an extensive survey and analysis on the existing studies [3, 4, 7, 19, 38–41, 43], we find that majority of the studies leave the embedding component as pre-trained and fixed (Although a majority of studies do not explicitly mention the fact in their model introduction part, they implicitly confirm this setting in experiment part.). For instance, the authors in [3] and [19] state that they utilize a PMF model to pre-train the embeddings and keep them unchanging, and Zhao et al. [39, 40] adopt the word2vector [22] embeddings in a similar setting. Nevertheless, in IRS, the users consecutively interact with the items, such a fixed representation setting is not suitable to model the user's dynamic preference. Moreover, in the recent research of computer vision, He et al. [9] challenge the conventional wisdom of ImageNet "pretraining and fine-tuning", and find that training from scratch (random initialization) is robust and performs no worse than their ImageNet pre-training counterparts. Therefore, both these discoveries motivate us to rethink the current de facto paradigm of "pre-training and fixing" strategy for EC in RL recommendation methods.

After a comprehensive investigation on the EC training strategy, we find that simultaneously training the EC and the other two parts will result in unstable training and inferior performance. To bypass the obstacle, majority of the studies (as mentioned above) adopt pre-trained and fixed EC. However, such a bypass strategy, albeit usable, suffers from the following two inherent drawbacks:

(1) Pre-trained and fixed embeddings are unable to model evolving preference of users and item correlations in the dynamic environment. On one hand, as the users and items interact frequently, the user preference evolves rapidly. Therefore, the representation of the user should be updated accordingly to avoid being out-of-date. For instance, on Instagram, a user favors football, and the Instagram recommends the latest pictures and videos of the football matches. And after a few days, his preference may change to the popular NBA playoff. Obviously, such a pre-trained and fixed representation of user preference leads the Instegram fails to meet his latest demands. On the other hand, the item correlations also evolve during the interactions, e.g., the RAM and the Hard Drive or other bonus are binded for sale during the festivals, while the pre-trained and fixed embeddings fail to model such evolving item correlations. Therefore, the pre-trained and fixed embeddings setting is inappropriate for dynamic environment.

(2) Pre-training is inconvenient in the industrial applications. Although the pre-training technique is widely used in various domains, such as in computer vision and natural language processing [6, 9]. However, in real industrial IRS applications, pre-training is inconvenient. The reasons are two-fold: (i) The dynamic interactions between the users and items are extremely frequent so that their representations should also be adapted accordingly. The model for pre-training embeddings has to be re-trained with newly arrived data frequently and rapidly, which is not always feasible in a large scale real-world setting; (ii) For the cold-start scenario, a new user or item has no pre-trained embedding, therefore such strategy cannot be adapted.

In this paper, we investigate how to train the embedding component and the other two parts simultaneously, and propose an **E**nd-to-end **D**eep **R** reinforcement learning based **R**ecommendation framework (EDRR). First, we analyze and find that the embedding component cannot be trained with the other two parts directly as the gradients in RL have great fluctuations. To alleviate the fluctuations, we need another signal to guide the optimization direction. Hence, a supervised learning (SL) part is introduced in EDRR. We design three ways to incorporate the SL into RL, named as EDRR-V1, EDRR-V2 and EDRR-V3, and highlight EDRR-V3 that shows much more stable training procedure than the other two. According to extensive experimental results, we find the proposed EDRR-V3 not only enables the simultaneous training of embedding component in both the policy-based and value-based RL recommendation models, but also significantly outperforms state-of-the-art interactive recommendation methods. The main contributions in this paper can be summarized as follows:

- We find that the embedding component cannot be nicely trained with the state representation and policy components at the same time. To address the issue, we propose an End-to-End Deep Reinforcement learning based Recommendation framework (EDRR), which introduces a SL signal to enable the training stable. To the best of our knowledge, we are the first to develop a real end-to-end RL based recommendation method.
- We design and compare three strategies to incorporate the SL into RL, and find that EDRR-V3 achieves the most stable training procedure.
- Extensive experiments are conducted on three real-world datasets under both offline and simulated online evaluation settings. The results demonstrate the proposed EDRR-V3 indeed achieves the end-to-end training purpose in both policy-based and value-based RL recommendation models, and delivers better performance when compared with state-of-the-art methods.

2 RELATED WORK

Conventional Recommendation Techniques. Conventional recommendation methods develop from the static content-based filtering [25], matrix factorization based methods [13, 14, 23, 34], logistic regression, factorization machines and its variants [11, 16, 21, 28], and until recently deep learning models [5, 8, 18, 27, 44], to multi-armed bandit methods [2, 15, 33, 36].

At the beginning of this century, collaborative filtering (CF) is put forward and extensively studied. However, the conventional CF based methods suffer from the data scarcity because the similarity calculated from sparse data might be unreliable. Hence, the Matrix factorization (MF) models [13, 14, 23, 34] are proposed, which characterize both items and users by vectors in the same space and infer from the observed user-item interactions. Moreover, regarding the recommendation as a binary classification problem, logistic regression (LR) and its variants [21] are also utilized in recommender systems. However, LR based models are hard to generalize to the feature interactions that never or rarely appear in the training data.

Therefore, factorization machines [28] and its variants [11] are proposed to model pairwise feature interactions as inner product of latent vectors between features and show promising results.

Recently, deep learning models [5, 8, 26, 44] improve the performance of recommender systems by using deep neural networks to model the complicated feature interactions, which enhance the model capability greatly. Cheng et al. [5] propose an interesting hybrid network structure (Wide & Deep) that combines a linear (Wide) model and a deep model for recommendation. Moreover, the authors in [8] impose a factorization machines as 'wide' module in Wide & Deep [5] with no need of feature engineering. Zhou et al. [44] propose a novel model, Deep Interest Network (DIN), where a local activation unit is designed to adaptively learn the representation of user interests from historical behaviors with respect to a certain ad.

As a distinguished direction, contextual multi-armed bandits (MAB) are also utilized to model the interactive nature of recommender systems [2, 15, 33, 36]. Li et al. apply Thompson Sampling (TS) and Upper Confident Bound (UCB) to balance the trade-off between exploration and exploitation in [2] and [15], respectively. The authors in [33] further learn hidden features for each arm to model the potential reward based on [15]. Moreover, Zeng et al. [36] propose a context drift model to address the time varying problem.

However, all these methods assume that the user preference keeps unchanged during the recommendation process, and do not explicitly model the long-term rewards that the recommendations can make.

RL based Recommendation Techniques. Recently, the research of RL based recommendation techniques has become a hot topic, and several approaches have been proposed [3, 4, 7, 19, 38–41, 43], which can be divided into two categories: *policy-based methods* [3, 4, 7, 19, 38, 39, 41] and *value-based methods* [40, 43].

Policy-based approaches [3, 4, 7, 19, 38, 41] aim to generate a policy, of which the input is a state, and the output is an action. (1) One type of studies [7, 19, 38] employ deterministic policies [17, 31], which generates an action directly. Dulac-Arnold et al. [7] resolves the large action space problem by modeling the state in a continuous item embedding space and selecting the items via a neighborhood method. Zhao et al. [38] propose to employ a Deep Deterministic Policy Gradient framework (DDPG) [17] with a page-display approach for page-wise recommendation. And the authors in [19] comprehensively study the state representation issue in RL based recommendation framework by explicitly modeling the user-item interactions, as the state representation is crucial for reinforcement learning [20]. (2) Another type of studies [3, 4] employ the stochastic policies [32], which output a distribution of the actions, and the action is sampled from such distribution. The authors in [3] propose to utilize a balanced hierarchical clustering tree to tackle the large discrete action space issue with REINFORCE algorithm [32]. Chen et al. [4] present an off-policy correction framework to address the data bias issue in a top-K recommender system at YouTube.

For value-based approaches [40, 43], the action with maximum Q-value over all the possible actions is selected as the best action. Zhao et al. [40] take both user's positive feedback and negative feedback into consideration when modeling user state. Dueling Q-network is utilized in [43], to model Q-value of a state-action pair. Moreover, a minor update with exploration by dueling bandit gradient descent is proposed. However, such value-based approaches need to evaluate the Q-values of all the actions under a specific state, which is very inefficient when the action space is large.

To the best of our knowledge, most of the above methods needs to pre-train embeddings and keep them fixed for stable learning. However, the pre-training and fixing strategy probably leads to suboptimal solution and is costly in industrial applications. To break such limitation, in this paper, we propose to further investigate on how to train RL recommendation models in an end-to-end fashion.

3 PRELIMINARY

3.1 Markov Decision Process.

The interactive recommendation task can be modeled as a Markov Decision Process (MDP). At each time step, the recommender agent takes an action $a \in \mathcal{A}$ according to the user state $s \in \mathcal{S}$, and receives a reward $R(s, a)$. Consequently, the user state is updated to s' with transition $p(s'|s, a)$. The target of the recommender agent is to find an optimal policy ($\pi_\theta : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$) which maximizes the expected cumulative rewards for the recommender system. Specifically, the elements of an MDP $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ in recommendation scenario are defined as follows:

- **States $\mathcal{S}$.** A state s is the representation of user's positive interaction history with recommender, as well as her demographic information (if it exists in the datasets).
- **Actions $\mathcal{A}$.** An action a is to recommend items available in the dataset. In this paper, we assume that the agent only recommends one item to the user each time. And it is straightforward to extend to recommend a list of items.
- **Transitions $\mathcal{P}$.** The state is modeled as the representation of user's positive interaction history. Hence, once the user's feedback is collected, the state transition $p(s'|s, a)$ is determined.
- **Reward $\mathcal{R}$.** Given the recommendation based on the action a and the user state s, the user will provide her feedback, i.e., click, not click, or rating, etc. The recommender receives immediate reward $R(s, a)$ according to the user's feedback.
- **Discount rate γ.** $\gamma \in [0, 1]$ is a factor measuring the present value of long-term rewards.

3.2 Unified RL Recommendation Framework.

With an extensive survey and analysis on the existing studies [3, 4, 7, 19, 38, 40, 41, 43], we find though the RL recommendation models are developed from different methodologies, their architecture can be summarized into a unified framework. As shown in Figure 2 (a), the unified framework consists of three components, namely *embedding component* (EC), *state representation component* (SRC) and *policy component* (PC).

EC: At the bottom, EC maps the items together with the user demographic information from a high dimensional sparse vector to a low dimensional dense one: $\mathbf{e}_t = [\mathbf{u}, \mathbf{H}_t]$, where $\mathbf{H}_t = \{\mathbf{I}_1, ..., \mathbf{I}_n\}$ denotes the embeddings of n latest positive interacted items at time t; $\mathbf{u}$ and $\{\mathbf{I}_1, ..., \mathbf{I}_n\}$ denote the latent representation of the user demographics and the embeddings of the items, respectively. The interaction history $\mathbf{H}$ works in a sliding window manner. When the recommender agent recommends an item $\mathbf{I}_t$, we have $\mathbf{H}_{t+1} = \{\mathbf{I}_2, ..., \mathbf{I}_n, \mathbf{I}_t\}$ if the user provides positive feedback, otherwise $\mathbf{H}_{t+1} = \mathbf{H}_t$.

SRC: In the middle, a carefully designed SRC, which can be a fully-connected network [43] or recurrent neural network [40], is to model the environment state. For simplicity, we term the function as $f(\cdot)$. Hence, the current state is represented as $\mathbf{s}_t = f(\mathbf{e}_t)$.

PC: PC is appended to SRC, whose input is state $\mathbf{s}_t$ and output is Q-values for value-based models and policy for policy-based models. Upon the output, the action $a = \pi_\theta(\mathbf{s})$ is determined by recommending the item with highest Q-value for value-based models; on the other hand, policy-based models decide the action directly from the output policy. Then, the reward $R(\mathbf{s}, a)$ is obtained based on the user feedback, and the related parameters are updated according to the temporal-difference learning approach [32][1], i.e., minimizing the mean squared error as

$$L_{RL} = \frac{1}{N} \sum_i (y_i - Q_{\pi_\theta}(\mathbf{s}_i, a_i))^2, \tag{1}$$

where $y_i = r_i + \gamma Q_{\pi_{\theta'}}(\mathbf{s}_{i+1}, \pi_{\theta'}(\mathbf{s}_{i+1}))$ and N is the batch size. The target network technique [24] is always adopted, where θ' is the set of parameters of the target Deep Q-network.

4 EC TRAINING STRATEGY STUDY

The **EC** is the fundamental layer and its quality directly affects the recommendation performance. Therefore, in this section, we investigate and compare four types of training strategies for **EC**, then reveal and analyze how different training strategies affect the RL recommendation models.

- **RU:** Randomly initialize the embeddings and Update them together with the other components of RL model in an end-to-end manner;
- **RF:** Randomly initialize the embeddings and Fix them when training the RL model;
- **PU:** Pre-train the embeddings[2] and Update them with the other components of RL model;
- **PF:** Pre-train the embeddings and Fix them unchanged.

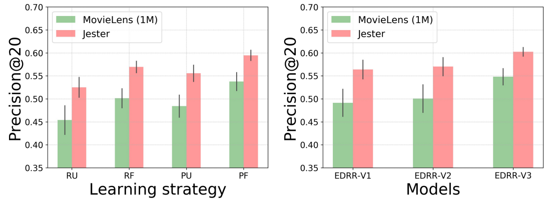
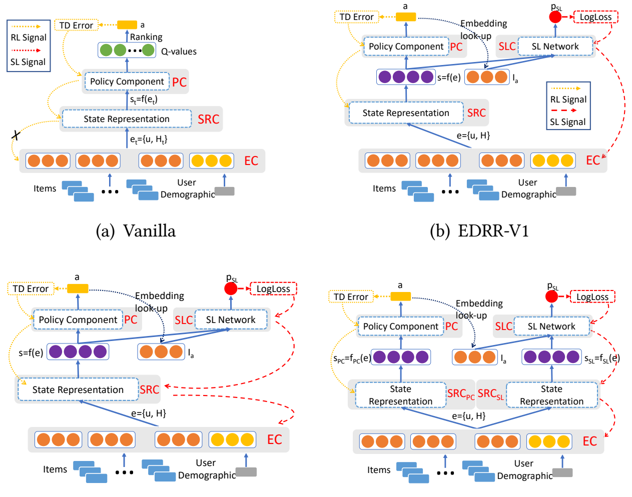


Figure 1: The left figure denotes the performance comparison on the four learning strategy on EC; The right figure reports the results compared with EDRR-V1, EDRR-V2 and EDRR-V3.

With similar experimental settings of [40], we conduct an offline evaluation on generalized RL recommendation model to compare the performance of the four mentioned EC training strategies on the MovieLens (1M) and Jester datasets. For simplicity, we adopt a fully-connected network structure for the SRC, which is different from the proposed SRC in Section 5.3.2.

The evaluation results are presented in Figure 1 (a), where the evaluation metric is **precision@20**. From the Figure 1 (a), we obtain two important observations: (1) The strategies of updating EC with the other components (namely, PU and RU strategies) deliver the worst results. This is because in the offline log, the majority of the feedbacks of the items are unobservable as the users never rate the items before. Such feedbacks and the exploration of the RL agent lead to highly biased and inconsistent rewards, which further result in unstable and severely fluctuated gradients when updating EC; (2) RF and PF outperform RU and PU respectively, and PF performs the best. This is because both RF and PF keep EC unchanged, which alleviates the unstable training difficulty in RL. This is also the reason why most previous studies adopt PF [3, 19, 39, 40].

Although PF is usable, the unchanged EC fails to model the evolving preference of users and item correlations in the dynamic environment. Moreover, pre-training is inconvenient in the industrial applications, especially when the recommender would like to capture the evolving properties of users and items, pre-training has to be frequently applied. The facts motivate us to develop a real end-to-end RL recommendation model, which is trained from scratch stably and with good performance.



(a) Vanilla

(b) EDRR-V1

(c) EDRR-V2

(d) EDRR-V3

Figure 2: (a) represents the unified RL recommendation framework; (b), (c) and (d) denote the proposed EDRR framework, termed as EDRR-V1, EDRR-V2 and EDRR-V3, respectively.

5 PROPOSED FRAMEWORK

5.1 Overview of Our Notion

According to the analysis in Section 4, in the offline training process, the rewards are highly biased and inconsistent, and further result in unstable and severely fluctuated gradients when updating EC, which is a severe obstacle to a real end-to-end framework. Hence, to

[1] To illustrate our idea more clear, we focus on value-based models for now (more specifically, Deep Q-network [24] as an example, unless otherwise stated). Later, we will elaborate details on policy-based models.

[2] We utilize MF [23] as the pre-training technique unless otherwise stated, due to the superior pre-training performance reported by [3].

address the problem, another signal is needed to guide the update direction of EC and smooth the fluctuations. However, rather than positive or negative feedbacks from users, no other supervision information exists. This motivates us to introduce a supervised learning signal in RL recommendation model.

Following this idea, we propose an **E**nd-to-end **D**eep **R**einforcement learning based **R**ecommendation framework (**EDRR**). In EDRR, a *Supervised Learning Component* (SLC) is introduced, as a binary classifier, to predict whether the user offers positive feedback to the recommended item based on the learned embedding. Specifically, we have the following cross entropy loss function:

$$L_{SL}(\theta) = \frac{1}{N} \sum_{i=1}^{N} y_i \log(y'_i) + (1 - y_i) \log(1 - y'_i), \tag{2}$$

where y_i is the ground truth indicating whether the item in i-th instance receives positive feedback from user. In i-th instance, the action (namely, the recommended item) selected by RL model is a_i, with embedding $\mathbf{I}_{a_i}$. And $y'_i = f(s_i, \mathbf{I}_{a_i})$ is a predictor (a neural network in our model), estimating the probability of item $\mathbf{I}_{a_i}$ that will be consumed by the user with state s_i. Next, we will discuss different ways to design SLC.

5.2 Three Ways to Incorporate SL Signals

We follow two guidelines to leverage the SL signals: (i) EC cannot be updated directly by RL signal (as observed in Section 4) but can be guided by SL signal; (ii) In SRC, RL signal has to be incorporated to model dynamic adaptation and long-term planning, while SL signal is optional.

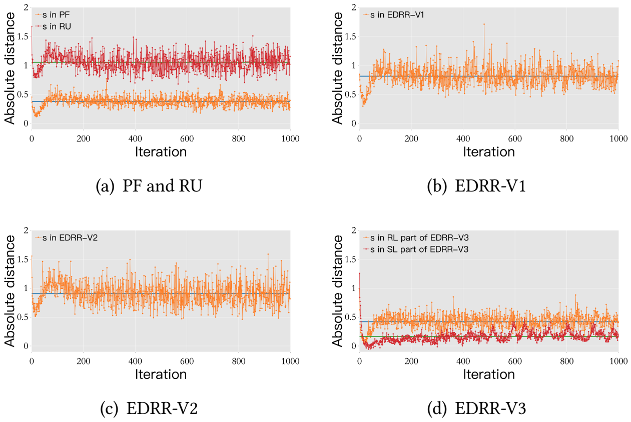
Following the two guidelines, we design three architectures in Figure 2 (b), 2 (c) and 2 (d), which are named as EDRR-V1, EDRR-V2 and EDRR-V3, respectively. The major difference of the three variants is whether SL signal is leveraged to update SRC.

- **EDRR-V1**: The input of the SLC is the state **s** and the embedding of the current action $\mathbf{I}_a$, and the SRC is updated only by the RL signal. EC is updated by SL signal.
- **EDRR-V2**: The only difference from EDRR-V1 is that the SRC in EDRR-V2 is updated by the combination of RL and SL signals.
- **EDRR-V3**: In EDRR-V3, two SRCs with the same structure are designed, namely SRC$_{PC}$ and SRC$_{SL}$: SRC$_{PC}$ connects to PC and is updated by RL signal, while SRC$_{SL}$ connects to SLC and is renewed by SL signal.

Note that all the SRCs in V1, V2 and V3 share the same structure.

Next, we compare the three variants of EDRR in terms of their training stability on MoiveLens (1M) dataset. We measure the variation of state vectors, defined as $\sum_{k=1}^{K} |s_t^k - s_{t+1}^k|$, in consecutive iterations to reflect the stability[3], where K is the dimension of the state s, and s_t^k is the value of the k-th position of the state s during the t-th update.

Figure 3 depicts the results, from which two important conclusions are made: (1) Sharing SRC between SL and RL (in EDRR-V1 and EDRR-V2) leads to severe perturbations, whose variance trend is similar to that of RU setting, as shown in Figure 3 (a), (b), (c); (2) When separating the SRC into two counterparts, one for SL and the other for RL, the training becomes much more stable. The variance trend is similar to that of PF setting, as shown in Figure 3 (a), (d). Note RU and PF are defined in Section 4 and they are performed on the architecture in Figure 2 (a).



(a) PF and RU

(b) EDRR-V1

(c) EDRR-V2

(d) EDRR-V3

Figure 3: Training stability analysis of the proposed models.

The state analysis suggests the training of EDRR-V3 is more stable than EDRR-V1 and EDRR-V2. Moreover, we also perform accuracy evaluation on the three variants, and the results are reported in Figure 1 (b). We observe that EDRR-V3 outperforms the other two in accuracy as well. Hence, next we elaborate the details of EDRR based on the V3 architecture. In the rest of this paper, it is V3 version when we mention EDRR unless otherwise stated.

5.3 EDRR Framework

In this subsection, the detailed pipeline of the EDRR framework is introduced.

5.3.1 EC. For EC layer, its output will be fed to SRC$_{PC}$ and SRC$_{SL}$ and it is only updated by SL signal. Note that except for the user demographic information and the interacted items, other types of side information also can be utilized (e.g., the corresponding rewards of the interacted items [3]).

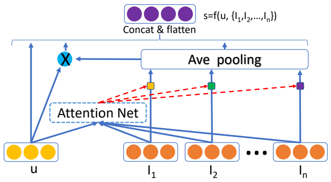


Figure 4: State Representation Component in EDRR.

5.3.2 SRC. In the SRC of EDRR, in addition to the widely utilized fully-connected neural network [7, 43] and recurrent neural network [3, 40], other types of state representation learning methods [19] are also suitable for EDRR.

To pursuit better performance, we design a product based neural network for **SRC** to model the feature interactions (ID features) explicitly, which is inspired by [8, 26], and the specifications are

[3] The experimental settings here are the same as when we get result in Figure 1.

shown in Figure 4. SRC$_{SL}$ and SRC$_{PC}$ share the same network structure but with different parameters, and the state represents as $s = [u, u \otimes g(H_t), g(H_t)]$, where $\otimes$ denotes the element-wise product, a_{uj} is a scalar indicating the importance of item I_j to user u, and $g(\cdot)$ represents the weighted average pooling layer. In addition, an attention network is introduced so as to generate the user-dependent weights a_{uj} for each item, i.e., we utilize the attention score to represent the weight of each item, which is specified in Eq. (3).

$$
\begin{aligned}
a'_{uj} &= w^1 ReLU(w^2([u, I_j]) + b^2) + b^1, \\
a_{uj} &= \frac{exp(a'_{uj})}{\sum_{p=1}^{n} exp(a'_{up})}.
\end{aligned}
\tag{3}
$$

5.3.3 PC. The proposed EDRR is a unified framework which can be applied to both value-based and policy-based RL models. Therefore, we take value-based Deep Q-Network (DQN) and policy-based Deep Deterministic Policy Gradient (DDPG) as examples to specify how PC works in EDRR framework, respectively.

DQN. For value-based DQN, the input of PC is s_{PC} and output is the action a with the highest Q-value. Based on the action a, we can get the reward $r = R(s_{PC}, a)$. To train the model, the temporal difference learning strategy can be utilized (specified in section 3.2). And the RL signal can be calculated as follows:

$$
\nabla_\theta J(\pi_\theta) = \Sigma_{\pi_\theta}[(r + \gamma \max_{a'} Q(s', a') - Q(s, a)) \nabla_\theta Q(s, a)], \tag{4}
$$

where θ denotes the parameters of PC.

DDPG. For policy-based DDPG, PC stands for the Actor part, and we omit the Critic part here. PC outputs a continuous ranking vector [19]. The items are recommended according to the ranking scores by the inner product operation between the generated ranking vector and the item embeddings. And the RL signal is computed according to the sampled Policy Gradient [17]:

$$
\nabla_\theta J(\pi_\theta) \approx \frac{1}{N} \sum_t \nabla_a Q_\omega(s, a)|_{s=s_t, a=\pi_\theta(s_t)} \nabla_\theta \pi_\theta(s)|_{s=s_t}. \tag{5}
$$

where θ denotes the parameters of PC, ω indicates the parameters of Critic network.

5.3.4 SLC. SLC is a fully-connected neural network, the input of which is s_{SL} as well as the embedding of generated action a. The output of SLC is a probability of the user favoring the recommended item. As mentioned in Section 5.1, the widely used cross entropy loss is adopted as SL objective and utilized to update the parameters in SLC, SRC$_{SL}$ and EC. In addition to fully-connected neural network, other advanced network structures (such as [8, 26]) can be employed into our proposed framework as SLC.

With such a carefully designed framework, the EC layer can be nicely trained simultaneously with SRC and PC.

6 EXPERIMENT

6.1 Experimental Setup

6.1.1 Datasets and Evaluation Metrics. Following [19, 38–40], we conduct both offline and simulated online evaluation on three publicly available datasets, which are **MovieLens (100k)**, **MovieLens (1M)**, and **Jester (2)**. The statistic information of the datasets is given in Table 1. The MovieLens (100k) and MovieLens (1M) are abbreviated as ML (100k) and ML (1M) respectively. In the offline evaluation, the widely used **Precision@k**, **NDCG@k** and **MAP**[4] are adopted as the metrics. In the simulated online evaluation, the average reward (i.e., divide the total rewards by the total number of recommendations) is leveraged.

Table 1: Statistic information of the datasets

	ML (100k)	ML (1M)	Jester
# user	943	6,040	63,978
# item	1,682	3,952	150
# ratings	100,000	1,000,209	1,761,439

6.1.2 Compared Methods. We conduct experiments to compare several representative baselines. For easy to follow, we summarize the compared methods as three categories, namely:

(1) ***Only for offline***: **Popularity** [1] recommends the most popular item, i.e., the item with the highest average rating, and we remove few top most popular items following [1]; **PMF** [23] makes a matrix decomposition as SVD, while it only takes into account the non zero elements; **SVD++** [13] mixes strengths of the latent model as well as the neighborhood model; **DeepFM** [8] is a state-of-the-art deep learning based recommendation model, which explicitly models the low-order and high-order feature interactions.

(2) ***Only for online***: **LinUCB** [15] selects a contextual arm (i.e., recommend an item) according to the estimated upper confidence bound of the potential reward; **HLinUCB** [33] is another state-of-the-art bandit-based method for recommendation, which further learns hidden features for each arm to model the potential reward based on LinUCB.

(3) ***For both offline and online***: **DDPG-R** [41] denotes the DDPG-based recommendation method, which learns a ranking action and picks the item with the highest ranking score; **DRR** [19] explicitly models the user-item interactions to represent the state to better study recommendation policy under the Actor-Critic framework (for a fair comparison, we change the state representation in DRR to the one proposed in EDRR framework, named as **DRR-att**); **DQN-R** [43] denotes the DQN-based recommendation method; **DEERS** [40] is a state-of-the-art RL recommendation approach, which represents the state with both the positive and negative feedbacks by Recurrent Neural Network (RNN) under the Deep Q-Network (DQN) framework; **DQN-att** denotes the DQN based model with the same state representation component proposed in EDRR framework.

Models Equiped with EDRR Framework: To verify the effectiveness of the proposed **EDRR** framework on both value-based and policy-based RL models, for simplicity, we empirically adapt two models **DRR-att** and **DQN-att** into **EDRR** framework, which are then named **DRR-att (EDRR)**, and **DQN-att (EDRR)**. The reasons for choosing the two models are two-fold: (i) For a fair comparison, the only differences between the two chosen models and the corresponding EDRR adaption are the introduced SL signal and the dynamically updated EC; (ii) They outperform the

[4] As we recommend only one item per time step, we calculate the metrics after the end of the whole recommendation procedure.

other value-based and policy-based RL recommendation models, respectively.

6.1.3 Reward Shaping. As pointed in [3, 37], direct online evaluation and optimization is expensive and risky. Thus, in this paper, we concentrate on modeling and evaluating the proposed framework by building up an environment simulator to mimic online environment. Specifically, we feed the current state s_t and the recommended item a_t into the simulator, and receive immediate reward. The reward function $R(s, a)$ is considered as an evaluation of the quality of the recommended item. Furthermore, we define the reward function as: $R(s, a) = R_0(s, a) + \alpha\phi(s, a)$, where $R_0(s, a)$ is the original reward function, for which we leverage PMF as a supervision to the feedbacks of the recommended items that the users never rate before[5]. All the original rewards are empirically normalized into range $[-1, 1]$. The $\phi(s, a)$ is the potential reward function for better evaluating the actions. More precisely, we utilize the **promotion** of NDCG caused by the ranking action a. Here, α is a hyper-parameter to balance the two reward functions.

6.1.4 Settings. For each dataset, we choose 80% of the interactions in each user session as the training set, and leave the rest as the testing set. Moreover, on ML (100k) and ML (1M), the ratings with 4 and 5 scores are treated as positive feedbacks, while on Jester, those higher than 0 are treated as positive. The number of latest positively rated items n, is empirically set to 5. We perform PMF to pre-train the 100-dimensional embeddings of the users and items for RL based baselines. When applying them, the pre-trained embeddings are kept fixed. Moreover, in each episode, we do not recommend repeated items. The discount rate γ is 0.9, and batch size is 64. We utilize Adam optimizer for all the RL based methods with L_2-norm regularization to prevent overfitting. All the baseline methods are carefully tuned for a fair comparison.

6.2 Overall Performance Comparison

Following the evaluation settings in [19, 38–40], we compare the overall performance of the proposed EDRR framework and other baselines in offline and simulated online experiments.

Offline and Simulated Online Evaluation Results and Analysis. The offline and simulated online evaluation results are summarized in Table 2 and 3. For each type of RL methods, the best results are marked in bold type, and the second best ones are underlined. $*$ and $**$ indicate the statistically significant improvement [29] (i.e., two-sided t-test) with $p < 1e-5$ and $p < 1e-3$ over the best baselines, respectively. Here are four observations:

(1) We observe that the DRR-att (EDRR) and DQN-att (EDRR) models, which are equipped with the proposed EDRR framework, outperform all the baselines with corresponding p-values for significant test [29] in most case on the three datasets, which demonstrates the superiority of EDRR.

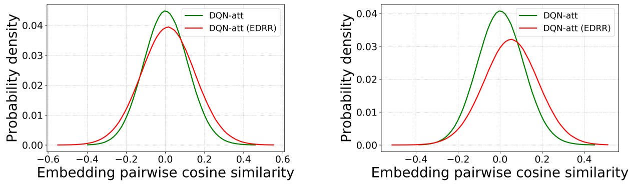
(2) When comparing the RL based methods with the conventional baselines, we observe that RL based methods yield better performance in most cases, as they have the ability of long-term planning and dynamic adaptation, which is lacking in the conventional baselines. DQN-R and DDPG-R provide inferior performance than DeepFM, the reason is that they don’t carefully study the user-item interaction, which is important for recommender system [8].

(3) Incorporating with the proposed SRC, the performance is improved. The reason is that the proposed SRC can capture the user-item interaction more reasonable through the product operation and the attention mechanism. This observation is from the fact that DQN-att and DRR-att outperform DQN and DRR respectively, and their performance is better than other RL based baselines.

(4) We also observe that the DQN-att (EDRR) and DRR-att (EDRR) models outperform DQN-att and DRR-att, respectively. There are two conclusions: (i) Training the embedding component with the other components simultaneously and properly like EDRR is able to improve the performance of RL-based baselines; (ii) The proposed EDRR indeed has the ability to generalize to both policy-based and value-based RL recommendation models. The reasons for such conclusions are as follows: On one hand, the introduced SL signal in EDRR can stabilize the learning process of embedding component as well as the other two components; On the other hand, the dynamically updated embeddings in EDRR do have the ability to model the evolving user preference and item correlations.

6.3 Analysis on Learned Embeddings.

Since the training strategies of embedding component is the main difference between EDRR and other RL based recommendation methods, it is meaningful to compare the embeddings of EDRR and the other RL models (whose embeddings are pre-trained with PMF). As shown in [35], an important indication of the effectiveness of the embedding is whether the distances between random pairs of item embeddings are widely distributed. Thus, we present two distributions of cosine similarities between pairs of items using PMF and EDRR embeddings[6] on ML(1M) dataset in Figure 5. Specifically, Figure 5 (a) plots such distribution based on all the item pairs in the dataset, and Figure 5 (b) plots the distribution based on the item pairs on the training set with strong association relationship. The association relationship is defined by a sequential association mining rule. Specifically, we find $X \rightarrow Y$ from $(i_{t-L}, ..., i_{t-2}, i_{t-1}) \rightarrow i_t$ with support count $sup(XY) > 5$ and the support confidence $\frac{\text{sup}(XY)}{\text{sup}(X)} > 40\%$ with $L = 2$.



(a) (b)

Figure 5: Embedding similarity distribution analysis on ML (1M) dataset.

From Figure 5 (a), we find the distances between random pairs of item embeddings in EDRR are more widely distributed than

[5] In the training process, the original rewards for the recommended items that the users never rate before are 0.

[6] Here we take the embeddings of DQN-att and DQN-att (EDRR) as example to compare the embedding similarity distribution, where former one takes the PF strategy, and the later one is learned from scratch.

Table 2: Overall Ranking performance on the three dataset.

	model	ML (100k)			ML (1M)			Jester		
		precision@20	NDCG@20	MAP	precision@20	NDCG@20	MAP	precision@20	NDCG@20	MAP
Conventional	Popularity	0.5685	0.8720	0.6017	0.5094	0.8727	0.5783	0.5608	0.8415	0.6248
	PMF	0.5845	0.8849	0.6446	0.5213	0.8734	0.6180	0.5876	0.8428	0.6477
	SVD++	0.5876	0.8866	0.6461	0.5183	0.8785	0.6191	0.5908	0.8515	0.6416
	DeepFM	0.6362	0.8941	0.7097	0.5750	0.8841	0.7096	0.6023	0.8729	0.6635
Policy-based RL	DDPG-R	0.6052	0.8870	0.6713	0.5364	0.8776	0.6342	0.5947	0.8550	0.6429
	DRR	0.6564	0.8982	0.7425	0.6227	0.8912	0.7238	0.6075	0.8834	0.6850
	DRR-att	0.6784	0.9025	0.7768	0.6315	0.8946	0.7516	0.6124	0.8889	0.6937
	DRR-att (EDRR)	**0.6941***	**0.9054****	**0.7907***	**0.6485***	**0.8996****	**0.7703***	**0.6196****	**0.8907****	**0.6984****
Value-based RL	DQN-R	0.6076	0.8815	0.6704	0.5377	0.8759	0.6365	0.5952	0.8548	0.6425
	DEERS	0.6481	0.8933	0.7226	0.5963	0.8878	0.7174	0.6025	0.8744	0.6752
	DQN-att	0.6753	0.9001	0.7670	0.6252	0.8935	0.7505	0.6104	0.8859	0.6897
	DQN-att (EDRR)	**0.6925***	**0.9062****	**0.7914***	**0.6488***	**0.8985****	**0.7717***	**0.6168****	**0.8898****	**0.6967****

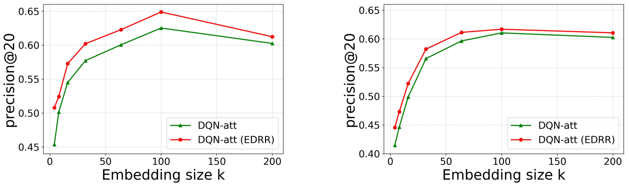
Table 3: The average rewards on the three datasets.

	Model	ML (100k)	ML (1M)	Jester
MAB	LinUCB	0.4266	0.4996	0.2391
	HLinUCB	0.3214	0.5428	0.2488
Policy-based RL	DDPG-R	0.5783	0.5937	0.2805
	DRR	0.7105	0.6746	0.3315
	DRR-att	0.7466	0.6824	0.3437
	DRR-att (EDRR)	**0.7574***	**0.6933***	**0.3485****
Value-based RL	DQN-R	0.5806	0.5944	0.2791
	DEERS	0.7035	0.6635	0.3274
	DQN-att	0.7418	0.6835	0.3428
	DQN-att (EDRR)	**0.7552***	**0.6924***	**0.3471****

that in PMF. This indicates the learned embeddings in EDRR are more discriminative. From Figure 5 (b), we find the mean value of distribution produced by EDRR tends to shift to positive value (recall that the cosine similarity being positive means positively correlated), while the mean value of PMF is around zero. The fact suggests that EDRR successfully captures the positive correlations between items in association set, while the pre-training strategy fails to do so. Overall, EDRR can model the sequential correlations between items.

6.4 Parameter Study

embedding size k. In this subsection, we analyse the effect of embedding size k on the proposed EDRR framework. Specifically, we tune the embedding size as $k = 4, 8, 16, 32, 64, 100, 200$, and report the results in Figure 6. From the results, we observe that the performance of DQN-att and DQN-att (EDRR) is increased at first and then decreased when we enlarge the embedding size k. Both models achieves the best performance in terms of precision@20 on ML-1M and Jester datesets when $k = 100$.

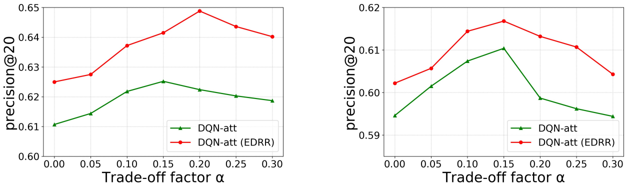


(a) ML-1M

(b) Jester

Figure 6: Parameter study on embedding size k in ML (1M) and Jester datasets.

Trade-off Factor α. In this subsection, we analyze the effect of the hyper-parameter α of the reward shaping and present the result in Figure 7. We observe that the performance is at first increased and then decreased when we enlarge α on both datasets. Specifically, on ML-1M, the DQN-att (EDRR) and DQN-att achieve the best performance in terms of precision@20 when $\alpha = 0.2$ and $\alpha = 0.15$ respectively, while on Jester, both models achieve the best performance when $\alpha = 0.15$. We can conclude in two-fold: (1) Reward shaping indeed contributes to the performance promotion as we utilize a ranking based metric NDCG to supervise the recommendation; (2) Too large α will restrict the exploration of reinforcement learning and reduce the performance.



(a) ML-1M

(b) Jester

Figure 7: Parameter study on trade-off factor α in ML (1M) and Jester datasets.

7 CONCLUSION

In this paper, we proposed an end-to-end deep RL based recommendation framework EDRR. Specifically, a SL component is introduced to guide the update directions to embedding component, and three incorporating ways are designed and compared. Extensive experimental results show that the proposed EDRR achieves the end-to-end training goal on both the policy-based and value-based RL recommendation models, and produces better performance than state-of-the-art interactive recommendation methods.

ACKNOWLEDGMENTS

This research was supported in part by the National Key R&D Program of China, 2018YFB2101100, 2018YFB2101101 and NSFC under Grant Nos. 61972111, 61602132, U1836107 and 61572158.

REFERENCES

[1] Rocío Cañamares and Pablo Castells. 2018. Should I Follow the Crowd?: A Probabilistic Analysis of the Effectiveness of Popularity in Recommender Systems. In *In SIGIR*. ACM, New York, NY, USA, 415–424.

[2] Olivier Chapelle and Lihong Li. 2011. An Empirical Evaluation of Thompson Sampling. In *NeurIPS, Granada, Spain*. 2249–2257.

[3] Haokun Chen, Xinyi Dai, Han Cai, Weinan Zhang, Xuejian Wang, Ruiming Tang, Yuzhou Zhang, and Yong Yu. 2018. Large-scale Interactive Recommendation with Tree-structured Policy Gradient. *arXiv preprint arXiv:1811.05869* (2018).

[4] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H Chi. 2019. Top-K Off-Policy Correction for a REINFORCE Recommender System. In *In WSDM*. ACM, 456–464.

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. *CoRR* abs/1606.07792 (2016).

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[7] Gabriel Dulac-Arnold, Richard Evans, Peter Sunehag, and Ben Coppin. 2015. Reinforcement Learning in Large Discrete Action Spaces. *CoRR* abs/1512.07679 (2015).

[8] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *In IJCAI*. 1725–1731.

[9] Kaiming He, Ross Girshick, and Piotr Dollár. 2018. Rethinking imagenet pre-training. *arXiv preprint arXiv:1811.08883* (2018).

[10] Junqi Jin, Chengru Song, Han Li, Kun Gai, Jun Wang, and Weinan Zhang. 2018. Real-Time Bidding with Multi-Agent Reinforcement Learning in Display Advertising. *CoRR* abs/1802.09756 (2018).

[11] Yu-Chin Juan, Yong Zhuang, Wei-Sheng Chin, and Chih-Jen Lin. 2016. Field-aware Factorization Machines for CTR Prediction. In *In RecSys*. 43–50.

[12] Gregory Kahn, Adam Villaflor, Bosen Ding, Pieter Abbeel, and Sergey Levine. 2018. Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation. In *In ICRA 2018*. IEEE, 1–8.

[13] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *In SIGKDD*. ACM, 426–434.

[14] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* 42, 8 (2009), 30–37.

[15] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *In WWW*. ACM, 661–670.

[16] Xutao Li, Gao Cong, Xiaoli Li, Tuan-Anh Nguyen Pham, and Shonali Krishnaswamy. 2015. Rank-GeoFM: A Ranking based Geographical Factorization Method for Point of Interest Recommendation. In *In SIGIR*. ACM, 433–442.

[17] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971 (2015).

[18] Bin Liu, Ruiming Tang, Yingzhi Chen, Jinkai Yu, Huifeng Guo, and Yuzhou Zhang. 2019. Feature Generation by Convolutional Neural Network for Click-Through Rate Prediction. In *The World Wide Web Conference, San Francisco, CA, USA, May 13-17*. ACM, 1119–1129.

[19] Feng Liu, Ruiming Tang, Xutao Li, Weinan Zhang, Yunming Ye, Haokun Chen, Huifeng Guo, and Yuzhou Zhang. 2018. Deep Reinforcement Learning based Recommendation with Explicit User-Item Interactions Modeling. *arXiv preprint arXiv:1810.12027* (2018).

[20] Odalric-Ambrym Maillard, Daniil Ryabko, and Rémi Munos. 2011. Selecting the state-representation in reinforcement learning. In *NeurIPS*. 2627–2635.

[21] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *SIGKDD*. ACM, 1222–1230.

[22] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NeurIPS*. 3111–3119.

[23] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *NeurIPS*. 1257–1264.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[25] Raymond J. Mooney and Loriene Roy. 2000. Content-based book recommending using learning for text categorization. In *ACM DL*. 195–204.

[26] Yanru Qu, Han Cai, Kan Ren, Weinan Zhang, Yong Yu, Ying Wen, and Jun Wang. 2016. Product-Based Neural Networks for User Response Prediction. In *ICDM 2016, December 12-15, 2016, Barcelona, Spain*. 1149–1154.

[27] Yanru Qu, Bohui Fang, Weinan Zhang, Ruiming Tang, Minzhe Niu, Huifeng Guo, Yong Yu, and Xiuqiang He. 2019. Product-based Neural Networks for User Response Prediction over Multi-field Categorical Data. *ACM Trans. Inf. Syst.* 37, 1 (2019), 5:1–5:35.

[28] Steffen Rendle. 2010. Factorization Machines. In *In ICDM, Sydney, Australia, 14-17 December 2010*. 995–1000.

[29] Graeme D Ruxton. 2006. The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test. *Behavioral Ecology* 17, 4 (2006), 688–690.

[30] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[31] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. 2014. Deterministic Policy Gradient Algorithms. In *ICML 2014, Beijing, China, 21-26 June 2014*. 387–395.

[32] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[33] Huazheng Wang, Qingyun Wu, and Hongning Wang. 2016. Learning Hidden Features for Contextual Bandits. In *In CIKM*. ACM, 1633–1642.

[34] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *In SIGIR*. ACM, 501–508.

[35] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *In SIGKDD*. ACM, 974–983.

[36] Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li. 2016. Online Context-Aware Recommendation with Time Varying Multi-Armed Bandit. In *In SIGKDD*. 2025–2034.

[37] Weinan Zhang, Ulrich Paquet, and Katja Hofmann. 2016. Collective Noise Contrastive Estimation for Policy Transfe Learning. In *AAAI*. AAAI Press, 1408–1414.

[38] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. 2018. Deep reinforcement learning for page-wise recommendations. In *RecSys*. ACM, 95–103.

[39] Xiangyu Zhao, Long Xia, Yihong Zhao, Dawei Yin, and jiliang Tang. 2019. Model-Based Reinforcement Learning for Whole-Chain Recommendations. *arXiv preprint arXiv:1902.03987* (2019).

[40] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Recommendations with Negative Feedback via Pairwise Deep Reinforcement Learning. In *In SIGKDD 2018, London, UK, August 19-23, 2018*. 1040–1048. https://doi.org/10.1145/3219819.3219886

[41] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Dawei Yin, Yihong Zhao, and Jiliang Tang. 2017. Deep reinforcement learning for list-wise recommendations. *arXiv preprint arXiv:1801.00209* (2017).

[42] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. 2013. Interactive collaborative filtering. In *In CIKM, San Francisco, CA, USA, October 27 - November 1, 2013*. 1411–1420.

[43] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. DRN: A Deep Reinforcement Learning Framework for News Recommendation. In *In WWW 2018, Lyon, France, April 23-27, 2018*. 167–176.

[44] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *In SIGKDD*. ACM, 1059–1068.

[45] Barret Zoph and Quoc V. Le. 2016. Neural Architecture Search with Reinforcement Learning. *CoRR* abs/1611.01578 (2016).