Sequential Recommendation with Batch Reinforcement Learning

Xinhao Huang
Shenzhen, China
3475635952@qq.com

***Abstract*—”Reinforcement Learning based Recommender System Challenge : Item Combination Generation” is a new task in the field of recommender systems. It can be naturally formalized as a multi-step decision-making problem in which the recommender system recommends an appropriate product for users in each step. Existing recommender systems are designed to maximize immediate user satisfaction in a greedy manner. However, the item-wise greedy recommendation strategy is imperfect fitting to the real recommendation system. The sequential nature of the recommendation problem can be formulated as a Markov decision process (MDP) and reinforcement learning (RL) methods can be employed to solve it. We emphasize that with online reinforcement learning, the construction of simulator is complex. We have to construct the user model accurately, in other words, we need to accurately predict users’ feedback for the given product. In this work, we utilize batch reinforcement learning, a variant of reinforcement learning that requires the agent to learn from a fixed batch of data without exploration. Through batch reinforcement learning, We can skip the construction process of the simulator, reduce the risk of error, and focus on solving the sequence decision problem.**

I. Introduction

Modeling the users’ dynamic preferences based on the their historical behavior is challenging but essential for recommendation systems. To effectively capture such interaction for recommendations, we need to solve a key problem: how to update recommending strategy according to users’ historical preferences. Most existing recommendation systems regard the recommendation process as a static process and make recommendations according to a fixed greedy strategy. However, these methods may not be able to capture the dynamic characteristics of the user’s preferences, nor can they efficiently and continuously update the recommendation strategy based on the user’s real-time feedback. Therefore, in this work, we regard the recommendation process as a sequential interaction between the user and the recommendation agent, and use reinforcement learning to automatically learn the optimal recommendation strategy.

The wide application of deep reinforcement learning makes more and more researchers devote themselves to it. Reinforcement learning is suitable for solving Markov Decision Process(MDP), and the mode of the recommendation system and the user interaction can be a good way to perform Markov modeling. Therefore, in recent years, more and more attention has been focused on combining the recommendation system and reinforcement learning, looking forward to realizing personalized recommendations to users.

Existing works regard the recommendation task as a process of interaction between the recommendation system (Agent) and the recommended object (User), which is the environment usually defined by reinforcement learning, that is, the agent observes the user history in an ideal state. The preference record makes a recommendation and displays it to the user. The user will accept the recommendation or reject the recommendation. If the recommendation is accepted, a positive return will be given to the Agent. If the recommendation is rejected, a negative return will be added. At the same time, modify the user’s historical preference record, and so on. To utilizing online reinforcement learning, we have to construct the simulator and user model. However, the construction of simulators is complicated and detailed. We must accurately construct the user model based on the recommended products and the hidden relationships between them. Moreover, the construction of the user model requires us to complete the task of track 1 well, that is, we need to accurately predict users’ feedback for the given product.

Batch reinforcement learning is a variant of reinforcement learning that requires the agent to learn from a fixed batch of data without exploration. Batch reinforcement learning is about deriving the best policy possible given the data. This gives us the hope of out-performing the demonstration data. However, due to extrapolation error, an error in off-policy value learning which is introduced by the mismatch between the dataset and true state-action visitation of the current policy, standard off-policy deep reinforcement learning algorithms, such as DQN and DDPG, cannot learn without data correlated to the distribution under the current policy, which makes them invalid for this fixed batch setting. We adapt the Batch-Constrained Q-learning(BCQ) algorithm to the discrete-action setting, which restricts the action space in order to force the agent towards behaving close to on-policy with respect to a subset of the given data.

II. Method

A. Feature Engineering

The goal of feature engineering is simply to make data better suited to the problem. As show in Fig. 1 figure In this work, we considered the following characteristics: user portrait, user click historical behavior, and product feature. For user portrait, we apply mean-standard deviation normalization to normalize the original data. The user click historical behavior is another essential feature, which contains users’ historical preferences.

A simply but effective approach is to use the provided user click history product id. We set the length of user click history to fixed 249, truncated more than 249, and padded 0 for less. Product feature is composed of item vector, price, and location. We also average the click history product features as part of the input vector. Finally, the dimension of the feature vector input to the neural network is 273.

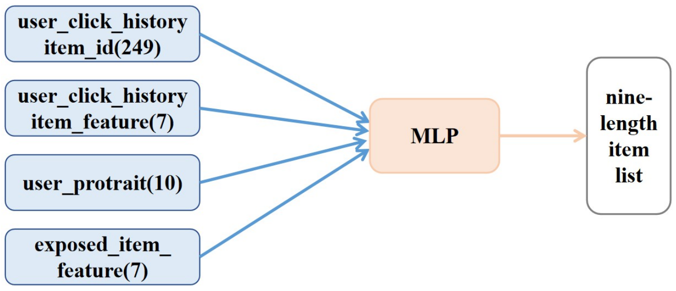
user_click_history item_id(249)
user_click_history item_feature(7)
user_protrait(10)
exposed_item_ feature(7)
MLP
nine-length item list

Fig. 1. The framework of Our Method.

B. Batch Reinforcement Learning : Discrete BCQ

In batch reinforcement learning, we additionally assume the data set is fixed, and no further interactions with the environment will occur. Batch deep reinforcement learning algorithms have been shown to be susceptible to extrapolation error, induced by generalization from the neural network function approximator. The idea of BCQ is to run normal Q-learning, but in the maximization step (which is normally $max_{a'}Q(s', a')$), instead of considering the max over all possible actions, we want to only consider actions a' such that (s', a') actually appeared in the batch of data. Or, in more realistic cases, eliminate actions which are unlikely to be selected by the behavior policy π_b (the policy that generated the static data). Discrete BCQ is a variant of the BCQ algorithm which operates on a discrete action space. We summarize discrete BCQ in figure 2.

Algorithm 1 BCQ

```
1: **Input:** Batch $\mathcal{B}$, number of iterations $T$, target_update_rate, mini-batch size $N$, threshold $\tau$.
2: Initialize Q-network $Q_\theta$, generative model $G_\omega$ and target network $Q_{\theta'}$ with $\theta' \leftarrow \theta$.
3: **for** $t = 1$ **to** $T$ **do**
4:     Sample mini-batch $M$ of $N$ transitions $(s, a, r, s')$ from $\mathcal{B}$.
5:     $a' = \text{argmax}_{a'|G_\omega(a'|s')/\max \hat{a}\ G_\omega(\hat{a}|s')>\tau} Q_\theta(s', a')$
6:     $\theta \leftarrow \text{argmin}_\theta \sum_{(s,a,r,s')\in M} l_\kappa \left(r + \gamma Q_{\theta'}(s', a') - Q_\theta(s, a)\right)$
7:     $\omega \leftarrow \text{argmin}_\omega - \sum_{(s,a)\in M} \log G_\omega(a|s)$
8:     If $t$ mod target_update_rate $= 0$: $\theta' \leftarrow \theta$
9: **end for**
```

Fig. 2. BCQ Algorithm.

The model structure is very simple, shown in table I. We tried to use LSTM, but the performance was not surpassed fully connected networks.

C. Action Mask

Action mask is a environment-knowledge-based pruning method, which is developed to guide explorations during the reinforcement process. To improve the training efficiency, action mask is used to incorporate the correlations between action elements at the final output layers of the policy based on prior knowledge of experiences, which helps prune the exploration of RL. Specifically, our action mask helps eliminate unreasonable aspect: as show in Fig. 3 the recommendation engine will respond with 3 item lists (3 items per list), and the next item list is locked until the items of the current list are sold out.

TABLE I
AVG. STEPS

Layer	Input	Output
Linear&Relu	273	512
Linear&Relu	512	256
Linear	256	9

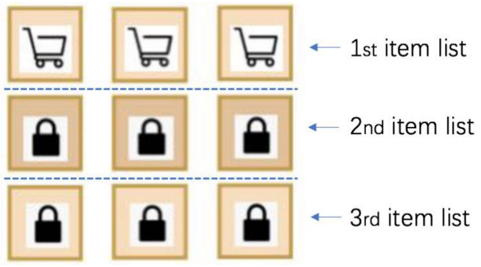

Fig. 3. An illustration of the recommended product.

Through the action mask, the agent will select the product that matches the location at each step, and the recommended product will not be recommended repeatedly.

III. Future Work

Another important problem is how to capture the hidden relationship between products. Most of the existing methods use recurrent neural networks to encode the left-to-right interactions of historical product that users have clicked into hidden representations to make recommendations. But these methods restrict the power of hidden representation in users' behavior sequences. Bidirectional Encoder Representations from Transformer(BERT) achieve great success in natural language processing tasks like text understanding. We wish to employs BERT to extract the hidden representation of users' behavior sequences. However, it is not straightforward to train the BERT model for sequential recommendation. To solve this problem, we need to pre-train the BERT model on related tasks. Specifically, we can randomly mask some items in the input sequence, and then predict the id of these masked items based on the context around them.

We know quite well that feature engineering is extremely simple. Due to a lack of time and computational resources, as well as the enormous amount of data provided by the challenge, we are unable to do more detailed experiments. If we can continue to explore, we will utilize BERT to capture the hidden relationships between user click on historical products in the hopes of improving performance even further.

IV. Result

Kaggle account : Ralph

Email : 3475635952@qq.com

Score : 1210506602

Ranks: 8

Source code : https://github.com/Ppaddington/BigData2021-Reinforcement-Learning-based-Recommender-System-Challenge