Deep Reinforcement Learning for Dynamic Recommendation with Model-agnostic Counterfactual Policy Synthesis

Siyu Wang  
siyu.wang5@student.unsw.edu.au  
University of New South Wales  
Sydney, NSW, Australia

Lina Yao  
University of New South Wales  
Sydney, NSW, Australia  
lina.yao@unsw.edu.au

Xiaocong Chen  
University of New South Wales  
Sydney, NSW, Australia  
xiaocong.chen@unsw.edu.au

Julian McAuley  
UCSD  
San Diego, CA, USA  
jmcauley@eng.ucsd.edu

ABSTRACT

Recent advances in recommender systems have proved the potential of Reinforcement Learning (RL) to handle the dynamic evolution processes between users and recommender systems. However, learning to train an optimal RL agent is generally impractical with commonly sparse user feedback data in the context of recommender systems. To circumvent the lack of interaction of current RL-based recommender systems, we propose to learn a general Model-agnostic Counterfactual Synthesis Policy for counterfactual user interaction data augmentation. The counterfactual synthesis policy aims to synthesise counterfactual states while preserving significant information in the original state relevant to the user’s interests, building upon two different training approaches we designed: learning with expert demonstrations and joint training. As a result, the synthesis of each counterfactual data is based on the current recommendation agent interaction with the environment to adapt to users’ dynamic interests. We integrate the proposed policy Deep Deterministic Policy Gradient (DDPG), Soft Actor Critic (SAC) and Twin Delayed DDPG in an adaptive pipeline with a recommendation agent that can generate counterfactual data to improve the performance of recommendation. The empirical results on both online simulation and offline datasets demonstrate the effectiveness and generalisation of our counterfactual synthesis policy and verify that it improves the performance of RL recommendation agents.

KEYWORDS

Recommender Systems, Reinforcement Learning, Counterfactual Inference

1 INTRODUCTION

Traditional recommendations are typically developed using content-based filtering [14, 24] or collaborative filtering approaches [1, 20], which predict the user’s future interest based on past preferences. However, because users’ preferences change over time, modelling based on previous interests may not provide an accurate prediction. In order to grasp shifting user interests, dynamic recommendation was developed as a practical technique to improve recommendation systems by supporting interactive processes [3, 4, 6, 34]. Interactive recommendation is a process in which the system takes an optimal action at each step to maximise the user’s feedback reward. Since Reinforcement Learning (RL) can learn from users’ interactive feedback, it has been regarded as having the ability to address the dynamic evolution process of user feedback loops and as an effective approach for modelling dynamic recommendation system [5, 7, 33]. One of the main obstacles for the RL-based recommender system is that it will struggle to precisely grasp users’ preferences and generate suitable recommendations only with limited interaction data. In RL-based recommender systems, the agent will take action based on the current state of the environment, interact with it, and receive the reward. The system will mistakenly presume that the user is uninterested in the item and return zero reward if there is no record of this circumstance, which will prevent the recommender system from accurately reflecting the users’ actual preferences. Therefore, the sparse interaction between users and items may limit the system’s capabilities and harm user satisfaction.

There has been recent interest in attempting to incorporate causality into recommender systems to address the data sparsity [29, 30]. From the standpoint of causality, building counterfactuals provide an answer to the counterfactual question: “What would the interaction process be if we intervened on some parts of the observational data?” Counterfactuals derived from intervention on observational data can be viewed as supplementing the circumstances not covered by the observational data. Thus, modelling from both observational and counterfactual data distributions serves as a powerful data augmentation technique to assist the recommendation engine in understanding users’ true preferences.

Recent works have made several initial attempts to alleviate the sparsity problem through data augmentation. Zhang et al. [31] propose to measure the similarity between the representation of each item and the target item and replace the top half of items with the lowest similarity scores to obtain the positive counterfactual user sequence. Wang et al. [28] design a sampler model that makes minor changes to the user’s historical items on the embedding space to implement a counterfactual data augment framework. However, these approaches are all based on the embedding space. Although the state representation in RL has similarities to the embedding, they are conceptually different. In contrast to the embedding, the state representation is dynamic and affected by the agent’s actions. State representation in an RL-based recommender system consists of user features, feedback and demographic information, and other information, whereas the standard embedding method only contains
users’ recent actions. As a result, RL-based dynamic recommendations in response to users’ shifting interests are incompatible with existing counterfactual generating frameworks.

To address the above issues, we propose to train a counterfactual synthesis policy to generate counterfactuals. We identify the concepts of essential and trivial components in the state representation based on the different degrees of influence on the user’s interest. Our goal is to enable agents to identify optimal actions that only find and change the trivial components in a state. We treat the modified state as an intervention on the original state to evaluate the causal effect of states on rewards. Since the rewards in a recommender system reflect the users’ interests, we can indirectly measure the causal effect of states on users’ interests through the effect on rewards. Thus, we identify the modified components in a state as trivial components if the modification has a weak causal effect on the reward. The weak causal effect can be indicated if the intervened reward’s distribution is similar to the initial distribution, suggesting a low impact on users’ interest. As a result, the counterfactual synthesis policy is learned by minimising the difference between observational and intervention reward distributions to fulfill the goal of replacing a state’s trivial components.

The proposed counterfactual synthesis policy is model-agnostic, allowing it to be implemented in any RL algorithm and cooperate with the recommender policy to synthesise counterfactual interactive data during interaction. This approach effectively handles the data sparsity problem by modelling both counterfactual and observational interaction data. Our main contributions are summarised as follows:

- We propose to generate counterfactuals based on users’ dynamic interests in RL-based dynamic recommendations and model both the observation and counterfactual distribution to address data scarcity.
- We design a novel model-agnostic counterfactual synthesis policy and provide two learning methods. Our policy can be employed in any RL algorithm to cooperate with the recommender policy and generate counterfactuals during the interaction process.
- We theoretically analyze the identification of causal effects in the recommender system and introduce an effective reward to guide our agents to detect and replace the trivial components in a state.
- We conduct experiments on online simulation and offline datasets to demonstrate that the counterfactual synthesis policy can be applied in several RL algorithms and significantly improve the performance of the recommender policy.

## 2 PRELIMINARIES

### 2.1 RL-based Dynamic Recommendation

The interactive process between users and a recommendation system achieved by RL can be formally described as training an agent that interacts with an environment, which follows a Markov Decision Process (MDP) [25].

Specifically, the agent interacts with the environment in each step of a discrete-time sequence $t = 0, 1, 2, ..., n$. In each interaction at time $t_i$, the agent initially receives the state representation $S_t \in S$ from the environment and chooses an action $A_t \in A(S_t)$ based on the state $S_t$. The interaction with the environment then yields a reward $R_{t+1} \in R$ back to the agent, and the environment will enter into the next state $S_{t+1}$. Formally, the process described above can be formulated by a tuple $(S, A(S_t), R, P, \gamma)$, where:

- $S$: the set of all states, including the initial and terminal state. State representations contain some information of the environment for the agent to make decisions.
- $A(S_t)$: the set of available actions in the state $S_t$.
- $R: S \times A \rightarrow R$ is the set of possible rewards related to user feedback.
- $P: S \times A \times S \rightarrow R$ are the state-transition probabilities.
- $\gamma$: discount factor that satisfies $0 \leq \gamma \leq 1$.

In RL, a policy ($\pi : S \rightarrow A$) is a mapping from perceived states to actions that the agent would take when in those environment states. Each interaction between the agent and the environment will yield an immediate reward $R_t$. The learning agent aims to select the action to maximise discounted return, which is the sum of discounted rewards it receives over the episode:

$$G_t := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots = \sum_{n=1}^{\infty} \gamma^n R_{t+n},$$

where $T$ is the final time step of an episode.

### 2.2 Structural Causal Models

The causal model describes a system by random variables that can be divided into two separate sets: endogenous variables and exogenous variables. The value of an endogenous variable is determined by the state of the other variables in the system. And in contrast, the value of exogenous variables is determined by factors outside the causal system, meaning that it is independent of other variables in the system [10].

A standard approach to modelling the causal relationships of interacting variables is using structural equations. For a complex system, Structural Causal Modeling (SCM) is a practical way to describe the causal relationships between variables with a set of structural equations. In an SCM, causal relationships are generated by functions that compute variables from other variables. Formally, we refer to these functions as assignments and define SCM as follows [19]:

**Definition 1. (Structural Causal Models)** A Structural Causal Model (SCM) $M := (S, P_X)$ is associated with a directed acyclic graph (DAG) $G$, which consists of a collection $S$ of $n$ structural assignments:

$$X_i := f_i(P A_i, U_i), \quad i = 1, ..., n,$$

where $V = \{X_1, ..., X_n\}$ is a set of endogenous variables, and $P A_i \subseteq \{X_1, ..., X_n\} \setminus \{X_i\}$ represent parents of $X_i$, which are also called direct causes of $X_i$. And $U = \{U_1, ..., U_n\}$ are noise variables, determined by unobserved factors. We assume that noise variables are jointly independent. Correspondingly, $P_X$ is the joint distribution over the noise variables. Each structural equation $f_i$ is a causal mechanism that determines the value of $X_i$ based on the values of $P A_i$ and the noise term $U_i$.

Since the definition of SCM requires its underlying graph be acyclic, each SCM $M$ models a unique distribution over the endogenous variables $V = \{X_1, ..., X_n\}$. Intuitively, SCMs with different causal structures have different distributions. Through intervention...
on the causal structures, we can obtain intervention distributions that are usually different from the observational one. We refer the definition of intervention as follows [19]:

**Definition 2. (Intervention).** Given an SCM \( \mathcal{M} := (S, P_N) \) over \( X \), an intervention \( I \) is defined as replacing one or several structural assignments of the SCM \( \mathcal{M} \). Assume we replace the assignment for \( X_k \) by the following expression:

\[
X_k := \tilde{f}_k(P_{A_k \theta}, \tilde{N}_k).
\]

Then we say that variable \( X_k \) have been intervened on, resulting in a new SCM \( \mathcal{M} \). The corresponding distribution is changed from the observational distribution \( p_X^M \) to the intervention distribution \( p_X^\mathcal{M} \), expressed as:

\[
p_X^\mathcal{M} := p_X^{M I \theta(X_k := \tilde{f}_k(P_{A_k \theta}, \tilde{N}_k))},
\]

where the operator \( \theta(X_k := \tilde{f}_k(P_{A_k \theta}, \tilde{N}_k)) \) denotes the intervention that we use to replace the assignment \( X_k \). Note that the new noise variables \( \tilde{N}_k \) are also required to be jointly independent with the original noise variables \( N \) in the SCM \( \mathcal{M} \).

Another approach to modifying the SCM is to maintain the causal relationship between the variables, but change the distribution of the noisy variables in the causal structure. In a recommendation system, for example, the counterfactual would deal with the question: “What does the user want to buy if he/she has previously clicked on a different item?”. This counterfactual query is based on observed data (the same list of items) but considers a different intervened on actions. Figure 1b shows that both the essential and trivial components may be changed when transitioning from state \( S_t \) to \( S_{t+1} \). Considering users’ dynamic preferences, we assume that not all historical interaction data has a substantial causal effect on inferring users’ preferences. That is, only the most recent \( m \) interactions are more relevant to users’ interests. Since the rewards in a recommender system reflect users’ interests, we measure the causal effect on users’ interests through rewards. Formally, we identify the essential components of the state \( S_t \) containing essential information of the user’s interest. Changes to the essential components will result in significant changes to the rewards as well as to the items recommended. In contrast, the trivial components are the part of the state that are less important for representing the user’s interest. Thus changing the trivial components have a weak causal effect on the reward in the SCMs. Given the identification of two components, the essence of our work is to find and replace the trivial components in a state. Figure 1c shows that we only change the trivial components in the state \( S_t \) to generate counterfactual state \( S_c \) by intervening on the action by \( A_t \), denoted as \( \theta(A_t) := A_c \).

\[ S_{t+1} := P(S_t, A_t), A_t := \pi_t(S_t), R_{t+1} := f_R(S_t, A_t). \]

Our work decomposes the state \( S_t \in S \) into two disjoint components, essential components \( S_t^{ess} \) and trivial components \( S_t^{tri} \), to identify their different levels of influence on learning users interest representations. We denote the decomposition of state as \( S_t = S_t^{ess} \oplus S_t^{tri} \). As shown in Figure 1b, both the essential and trivial components may be changed when transitioning from state \( S_t \) to \( S_{t+1} \). Considering users’ dynamic preferences, we assume that not all historical interaction data has a substantial causal effect on inferring users’ preferences. That is, only the most recent \( m \) interactions are more relevant to users’ interests. Since the rewards in a recommender system reflect users’ interests, we measure the causal effect on users’ interests through rewards. Formally, we identify the essential components of the state \( S_t \) containing essential information of the user’s interest. Changes to the essential components will result in significant changes to the rewards as well as to the items recommended. In contrast, the trivial components are the part of the state that are less important for representing the user’s interest. Thus changing the trivial components have a weak causal effect on the reward in the SCMs. Given the identification of two components, the essence of our work is to find and replace the trivial components in a state. Figure 1c shows that we only change the trivial components in the state \( S_t \) to generate counterfactual state \( S_c \) by intervening on the action by \( A_t \), denoted as \( \theta(A_t) := A_c \).

### 3.2 Counterfactual Synthesis Policy

The reinforcement learning method handles problems by considering the interaction of a goal-directed agent with an uncertain environment. It specifies that the agent maximises the total reward it receives over time and adjusts its policy based on its experience. The policy also provides guidance for the agent to choose the optimal action in a given state to maximise overall reward. Our goal is to allow the agent to change its policy such that the policy can guide the agent to discover the action to only replace the trivial components in a state.
Formally, the agent chooses an action $A_t$ as the intervention on the action $A_t$, which can be formulated with do calculus as $do(A_t := A_c)$. According to the Peal’s rules of do calculus [18], the probability distribution for state $S_{t+1}$ induced after intervention can be calculated by:

$$
p_{M^{do(A_t := A_c)}}(S_{t+1}) = \sum_{S_t} P(S_{t+1}|do(A_t), S_t) P(S_t|do(A_t))
$$

$$
= \sum_{S_t} P(S_{t+1}|A_c, S_t) P(S_t) \quad (7)
$$

$$
= E_{S_t} P(S_{t+1}|A_c, S_t).
$$

which is a special case of the back-door adjustment formula. Hence, we assume that the state variable $S_t$ in the SCM defined in Section 3.1 satisfies the back-door criterion such that it is sufficient for identifying $p_{M^{do(A_t := A_c)}}(S_{t+1})$.

**Assumption 1. (Back-Door).** For the SCM $M$ defined in Section 3.1 with the corresponding DAG $G$ shown in Figure 1a, the variable $S_t$ satisfies the back-door criterion with regard to the pair of variables $(A_t, S_{t+1})$ because it meets the following criteria:

- There is no descendant of $A_t$ in $S_t$.
- All paths containing an arrow into $A_t$ between $A_t$ and $S_{t+1}$ are blocked by $S_t$.

With Assumption 1, the causal effect of $A_t$ on $S_{t+1}$ is identifiable and is given by the Equation (7). We consider the state $S_{t+1}$ after intervention as the counterfactual state $S_c$ of the state $S_t$, where only the trivial components have been affected, that is:

$$
S_c = S_t^{ess} \otimes S_t^{tri} \; \text{and} \; S_c \sim E_{S_t} P(S_{t+1}|A_c, S_t) \quad (8)
$$

Since the agent always learns to maximize its reward, we propose to set up the rewards to evaluate whether only the trivial components have been replaced after intervention on the action. Given the SCMs defined in Section 3.1, we regard the counterfactual state $S_c$ as an intervention on the state $S_t$ and evaluate the counterfactual state by calculating the causal effect on the reward. Formally, we perform intervention on the state $S_t$ via $do(S_t := S_c)$, and the do calculus provides us:

$$
p_{M^{do(S_t := S_c)}}(R_{t+1}) = \sum_{A_t} P(R_{t+1}|do(S_t), A_t) P(A_t|do(S_t))
$$

$$
= \sum_{A_t} P(R_{t+1}|S_c, A_t) P(A_t|S_c), \quad (9)
$$

from which we can identify the causal effect of $do(S_t := S_c)$ on the reward $R_{t+1}$. If the intervened probability distribution of reward is similar to the original distribution, substituting $S_t$ with $S_c$ has a minor causal effect on reward, indicating that it also has a minor influence on learning the user’s interest.

To this end, the reward will take the measurement of the distance between two probability distributions into account. We adopt the KL-Divergence to measure how the intervened reward probability distribution $P_{M^{do(S_t := S_c)}}(R_{t+1})$ is different from the original distribution $P(R_{t+1})$:
where \( \epsilon \) and the reward will be written as:

\[
 D_{KL}(p_{M\text{do}}(S_t=S_{t+1})\|p_{M}(R_{t+1})) = \sum_{r\in R} p_{M\text{do}}(S_t=S_{t+1})(R_{t+1}) \log \left( \frac{p_{M\text{do}}(S_t=S_{t+1})(R_{t+1})}{p_{M}(R_{t+1})} \right)
\]

and the reward will be written as:

\[
 R_{t+1} = 1 - \frac{1}{D_{KL}(p_{M\text{do}}(S_t=S_{t+1})\|p_{M}(R_{t+1}))} + \epsilon
\]  

where \( \epsilon \) is a small constant to prevent the denominator from being zero. In order to maximise the total reward, the agent will select the optimal action to make the intervened probability distribution similar to the original one. Thus, the agent will adjust its policy to achieve only replacing the trivial components in a state.

### 3.3 Model-agnostic Learning and Integration

Given the above-described reward function, it is possible to develop a policy that meets our goal of identifying a counterfactual state for the current state in which only the trivial components have been replaced. The key to establishing the above-described reward function is obtaining the reward probability distribution of the observational data and the intervened reward probability distribution. To this end, we utilize an additional policy to help extract both the observational and intervened reward probability distribution. Our proposed counterfactual synthesis policy is a model-agnostic approach due to its key point on the reward function. Thus, it can be easily implemented in current RL-based algorithms to achieve the training process and perform data augmentation. We design two strategies for implementing the architecture as mentioned above: the first one is learning the counterfactual synthesis policy assisted by the expert demonstrations. The second one is joint training of the counterfactual synthesis policy and recommendation policy, in which the recommendation policy also serves as the distribution construction policy.

#### 3.3.1 Learning with Expert Demonstrations

We introduce a pre-trained policy as the expert policy that uses external knowledge to obtain the observational and intervened reward distribution. The pre-trained policy can be learned using any RL-based algorithm, and so does our policy. The idea is motivated by the policy distillation [21], in which the student policy is learned by minimizing the divergence between each teacher policy and itself over the dataset of the teacher policy. Consider an expert recommendation policy \( \pi_e \) that has learned the knowledge about user interest. It can guide the agent to select an optimal action to get positive feedback from users. By interacting with the environment, the agent following the expert policy \( \pi_e \) will construct the reward distribution over the observational data that reflects user interests. When retaining the same actions, the expert policy \( \pi_e \) can also assemble the reward distribution under the intervention of \( S_t \) with the Equation (9).

The overall schema of learning with expert demonstrations architecture is illustrated in the left part of Figure 2. We use one actor-critic framework to learn the counterfactual synthesis policy \( \pi_c \) and load the expert policy \( \pi_e \) in an additional agent. The process starts from the expert agent interacting with the environment. Depending on the current state \( S_t \) at time \( t \), the agent following the expert policy \( \pi_e \) takes the action \( A_t \) and interacts with the environment to obtain the reward \( R_{t+1} \). Meanwhile, the counterfactual...
synthetic agent will apply intervention on the action under the same environment whose current state is $S_t$ to generate a counterfactual state. By replacing the action $A_t$ with the action $A_t'$ taken by our training agent, the environment would enter into the counterfactual state $S'_t$. For the expert policy, the generated counterfactual state $S'_t$ can be regarded as an intervention on the state $S_t$. Under this intervention, the expert policy can construct the intervened reward distribution and evaluate the causal effect of this intervention on the user interest. Specifically, we estimate the reward by putting the agent with expert policy $\pi_e$ in the environment, of which the current state is counterfactual $S'_t$, to receive the reward $R_e$.

With the observational and intervention reward distribution, the counterfactual synthesis policy is learned by minimizing the KL-Divergence between two distribution as formulated in Equation (10):

$$\min_{\theta} \sum_r \left( \sum_t \sum_{s'} P(R_{t+1} | S'_t, A_t) P(A_t) \left( \frac{\sum_a P(R_{t+1} | S'_t, A_t) P(A_t) (A_t)}{P(R_{t+1})} \right) \right).$$

(12)

Once learned, the counterfactual synthesis policy can be used in any RL-based dynamic recommendation to collaborate with the recommendation policy for counterfactual synthesis. The procedure follows the right part in Figure 2. Formally, based on the current state $S_k$, the trained policy $\pi_e$ is used to find and replace the trivial components in $S_k$ to synthesize counterfactual state $S_{c,k}$. The recommendation policy $\pi_N$ receives the counterfactual state $S_{c,k}$ but perform the action $A_k$ based on the state $S_k$ to get the reward $R_{c,k+1}$. Then the counterfactual transition $(S_{c,k}, A_k, S_{c,k+1}, R_{c,k+1})$ can be put into the replay buffer. The generated counterfactual can provide additional information about the user’s interests for the recommendation policy, as it is based on changing states. This can also be regarded as exploration for recommendation policy while retain the current level of exploitation.

3.3.2 Joint Training. The goal of joint training is to combine the training processes for the recommendation policy $\pi_N$ and the counterfactual synthesis policy $\pi_C$. To do this, the training procedure is divided into three stages. The conventional recommendation policy training process, which can be based on any RL algorithm, is the first stage. We begin by training the recommendation policy until we store a policy with an average episode reward greater than a certain threshold. The stored policy can be used to build the observation and intervention reward distribution. The second stage, training the counterfactual synthesis policy, can then begin. This stage adheres to the approach outlined in Section 3.3.1 as well as the learning purpose mentioned in Equation (12). The training for policy $\pi_C$ will end when the average episode reward of policy $\pi_C$ reaches a particular threshold. Then the third stage is to apply the learned policy $\pi_C$ to the first stage to construct the counterfactual together with the policy $\pi_N$. In this approach, the recommendation policy aids policy $\pi_C$ training, while the trained policy $\pi_C$ provides counterfactuals to supplement the transition in the recommendation policy’s reply buffer, assisting the recommendation policy in comprehending the users’ interests. We raise the threshold each time we finish training the policy $\pi_C$. If the recommended policy meets the new threshold, we will use it to restart the second stage to improve the policy $\pi_C$. The above-described three stages can use any RL algorithm as an underlying framework. We take the DDPG as an example and present the process in Algorithm 1, in which the training objective can be indicated as minimising the loss function:

$$L(\theta_p, D) = \mathbb{E}_{(s,a,s',r) \sim D} \left[ \left( r + y \left( \mu_p(s', \phi'_{\theta_p}(s')) - \mu_p(s, a) \right) \right)^2 \right].$$

(13)

where $D$ is a set of mini-batch of transitions $(s, a, s', r)$ for $s \in S$, $a \in A(s)$, $r \in \mathbb{R}$, and $s' \in S^*$ ($S^*$ is $S$ plus a terminal state). $\theta_p$ and $\theta'_{\phi_p}$ represent parameters for the critic and actor network, respectively. And $\mu_p(s', \phi'_{\theta_p}(s'))$ represent the target critic network.

4 EXPERIMENTS

4.1 Experimental Setup

4.1.1 Data. We conduct experiments and evaluate our model in both online and offline manners. A public simulation platform, VirtualTaobao [23], is used for online evaluation. The benchmark datasets MovieLens-100k and MovieLens-1M are used for offline evaluation.

VirtualTaobao mimics a real-world online retail environment for recommender systems. It is trained using hundreds of millions of genuine Taobao data points, one of China’s largest online retail sites. The VirtualTaobao simulator provides a “live” environment by generating customers and generating interactions, in which the agent may be tested with virtual customers and the recommendation system. It uses the GAN-for-Simulating Distribution (GAN-SD) technique with an extra distribution constraint to produce varied clients with static and dynamic features. The dynamic attributes represent changing interests throughout an interactive process. It also employs the Multi-agent Adversarial Imitation Learning (MAIL) technique to concurrently learn the customers’ policies and the platform policy to provide the most realistic interactions possible.

MovieLens [11]. MovieLens-100k and MovieLens-1M are stable benchmark datasets based on user ratings on watching movies on the MovieLens website collected during different periods. Ratings are given on a 5-star scale, with each user having at least 20 ratings. Each user is assigned five features, whereas each movie has 23 features with 19 different genres.

4.1.2 Baselines and evaluation metrics. The proposed Counterfactual Synthesis Policy is model-agnostic that can be employed in various popular RL algorithms. Although some priors works also approach counterfactual reasoning by generating counterfactual sequences, their works are designed for supervised learning, which is different from RL. Existing RL-based recommendation methods do not have unified state representations, which cannot produce a fair comparison. Therefore, we mainly focus on the following RL algorithms as baselines:

- Deep Deterministic Policy Gradient (DDPG) [12]. DDPG is an off-policy method for environments with continuous action spaces. DDPG employs a target policy network to compute an action that approximates maximisation to deal with continuous action spaces.
- Soft Actor Critic (SAC) [9]. SAC is an off-policy maximum entropy Deep Reinforcement Learning approach that optimises a stochastic policy. It employs the clipped double-Q method and entropy regularisation that trains the policy to maximise a trade-off between expected return and entropy.
• Twin Delayed DDPG (TD3) [8]: TD3 is an algorithm that improves on baseline DDPG performance by incorporating three key tricks: learning two Q-functions instead of one, updating the policy less frequently, and adding noise to the target action.

In terms of evaluation measures, click-through rate is the primary indicator used by VirtualTaobao. For dataset evaluation, three widely used numerical criteria are utilised: Precision, Recall, and Accuracy.

4.2 Overall Comparison

**Online Experiment.** The overall comparison results conducted on the VirtualTaobao platform are depicted in Figure 3. In a nutshell, we find that both our approaches of learning through expert demonstrations and joint training achieve significant improvements over the chosen baselines. Among the baselines, DDPG achieves the best performance. It obtains a stable policy around the 70,000 episodes but suffers more considerable variance than others. The learning pace of SAC is slow. At the end of 100,000 episodes, SAC does not finish learning and does not reach a plateau. One probable explanation is that SAC utilizes a stochastic policy that introduces extra noise to the agent. TD3 initially receives a better policy but does not maintain it. The downward trend and the fluctuation may be ascribed to the delayed update parameter mechanism.

Applying a policy learned through expert demonstrations to DDPG shows a similar rising tendency as DDPG, shown in Figure 3a. However, with our policy, DDPG rises dramatically with each ascent. This may be due to the fact that the counterfactual transactions generated by our policy assist the recommendation policy in better learning the dynamic interests of users. The counterfactual synthesis policy trained by expert demonstrations assists DDPG in rising more steadily than others and achieving the highest CTR. The joint training method helps DDPG rapidly grow and discovers a good policy around 60,000 episodes. However, we can also observe a fluctuation in the last section. One of the possible reasons for the different performances of the two methods is that the counterfactual synthesis policy is frozen under the expert demonstrations setting while continuously optimising under the joint training scenario.

As shown in Figure 3b, the performance of SAC has a significant improvement when using our policy learned with either approach. The joint training procedure even assists SAC in reaching a plateau of roughly 90,000 episodes. Although the expert demonstration method drops dramatically near the 80,000 episodes, it rapidly returns and continues to increase. Because the TD3 employs a special strategy to force agent to conduct exploration at the beginning of training, Figure 3c shows that all three lines start with high CTRs. We can also observe that the expert demonstrations approach initially achieves a very high CTR. Moreover, TD3 utilising our strategy learned from the two approaches displays a similar tendency. They all descend to a relatively stable level after a defined number of steps at the start, which can be attributed to the fact that TD3 changes the policy less frequently and introduces noise into the target action for target policy smoothing.

**Offline Experiment.** The comparison results of employing our policy with baselines are listed in Table 1. On both datasets, the statistical results show that using our policy learned with any approach clearly outperforms the compared underlying models, DDPG, SAC, and TD3. In a nutshell, The results demonstrate the efficacy of our model-agnostic counterfactual synthesis policy in enhancing any RL-based recommendation model. It is worth mentioning that expert demonstration learning and joint training approaches generate comparable progress across all three baselines. We attribute the improved prediction result to the fact that our policy assists the recommendation policy in understanding users’ interests. The improved performance of RL-based recommendation systems in both online and offline experiments demonstrates the generality of our policy.

4.3 Hyper-parameter Study

We examine the number of hidden sizes to investigate how they affect the performance of counterfactual synthesis policies. Because the DDPG using our policy learned through expert demonstrations achieves the best results, we will utilise it as the baseline for the trial. We set the hidden size number in the range [64, 128, 256], and present the comparison result in Figure 4. The graph demonstrates that the best performance is obtained when the hidden size is set to 128. Either a too small or too large number would harm the performance of our counterfactual synthesis policy. Although the standard deviation of the result with 64 hidden sizes is small, it has a slower convergence speed and worse performance, which may be attributed to too few neurons in the hidden layers to collect information adequately. A too large number of hidden sizes leads to a high standard deviation. One possible explanation is that using too many neurons in the hidden layers causes overfitting, making it difficult to converge.

5 RELATED WORK

**RL-based interactive recommendation.** Since RL trains an agent that is able to learn from interaction trajectories, it has shown great effectiveness in modelling interactive recommendation processes. Mahmood and Ricci [17] modify the Markov Decision Process (MDP) to achieve an adaptive interaction recommender system using Reinforcement Learning (RL) techniques. Zheng et al. [33] apply a DQN structure and Dueling Bandit Gradient Descent method in reinforcement learning framework to address the dynamic nature of news recommendation. To handle the drawback of DQN in large and dynamic action space scenarios, Zhao et al. [32] build the list-wide recommendation model with the capability of dealing with large and dynamic item space upon the Actor-Critic framework. Chen et al. [4] utilize the knowledge graphs on the actor-critic network to improve the stability and quality of the critic network for interactive recommendation.

**Counterfactual for Recommendation.** Causality and counterfactual reasoning have received increased attention in a variety of fields. Many researchers leverage counterfactual reasoning in recommender systems to release bias issues. Schnabel et al. [22] propose a paradigm for dealing with selection bias in recommendation
Figure 3: Overall comparison result between the baselines, baselines with counterfactual synthesis policy learned with expert demonstrations and joint training of baselines and our policy: (a) DDPG as the baseline; (b) SAC as the baseline; (c) TD3 as the baseline.

Table 1: Performance comparisons of baselines, baselines with counterfactual synthesis policy learned with expert demonstrations and joint training of baselines and our policy on the MovieLens-1M dataset and the Amazon Fashion dataset. "-E" indicates that the counterfactual synthesis policy is learned with expert demonstration and "-J" indicates a joint training approach. The best results are highlighted in bold and the second best ones use the symbol *.

|                | Recall     | Precision | Accuracy   | Recall     | Precision | Accuracy   |
|----------------|------------|-----------|------------|------------|-----------|------------|
| MovieLens-100k |            |           |            | MovieLens-1M|           |            |
| DDPG           | 0.6979±0.2251 | 0.5019±0.0071 | 0.6338±0.1487 | 0.7207±0.0632 | 0.4225±0.006 | 0.7198±0.0716 |
| DDPG-E         | 0.8361±0.0698* | 0.5105±0.0048* | 0.7285±0.0462* | 0.7994±0.0189* | 0.4324±0.0031* | 0.8035±0.0171 |
| Improvement    | 19.81%     | 1.70%     | 14.94%     | Improvement | 10.92%    | 2.36%      | 11.63%     |
| DDPG-J         | 0.7755±0.002* | 0.5111±0.0009 | 0.7468±0.0178 | 0.8033±0.0266 | 0.4329±0.0042 | 0.7846±0.0285* |
| Improvement    | 11.13%     | 1.83%     | 17.83%     | Improvement | 11.46%    | 2.47%      | 9.00%      |
| SAC            | 0.6899±0.15 | 0.499±0.0079 | 0.6266±0.1002 | 0.7149±0.0438 | 0.4103±0.0086 | 0.7016±0.0557 |
| SAC-E          | 0.7849±0.0861 | 0.5102±0.0025* | 0.6951±0.0559 | 0.7974±0.0616* | 0.4257±0.0084* | 0.7751±0.0215 |
| Improvement    | 13.76%     | 2.248%    | 10.93%     | Improvement | 11.53%    | 3.767%     | 10.476%    |
| SAC-J          | 0.7795±0.0041* | 0.5118±0.0017 | 0.6912±0.0604* | 0.8055±0.0493 | 0.4222±0.0088* | 0.7663±0.0323* |
| Improvement    | 12.992%    | 2.573%    | 10.302%    | Improvement | 12.679%   | 2.899%     | 9.210%     |
| TD3            | 0.6822±0.1352 | 0.5001±0.0043 | 0.623±0.092 | 0.7034±0.0546 | 0.4201±0.0055 | 0.7121±0.0585 |
| TD3-E          | 0.7706±0.0785 | 0.5068±0.0045* | 0.6827±0.0521* | 0.7885±0.0047 | 0.4315±0.0029* | 0.8077±0.0107 |
| Improvement    | 12.954%    | 1.349%    | 9.584%     | Improvement | 12.097%   | 2.713%     | 13.426%    |
| TD3-J          | 0.7541±0.0253* | 0.5074±0.0051 | 0.698±0.0443 | 0.7711±0.0087* | 0.4338±0.0083 | 0.7805±0.0139* |
| Improvement    | 10.353%    | 1.472%    | 12.049%    | Improvement | 9.617%    | 3.261%     | 9.602%     |

Figure 4: Performance comparison on the hidden size of 64, 128 and 256.
help to learn a robust user representation for sequential recommendation. Another line of work focuses on data augmentation of training samples for data-scarce problems. Wang et al. [28] propose to generate counterfactual sequences by finding replacement items in the embedding space for sequential recommendation models. Lu et al. [15] focus on the learning of a causal mechanism by using a GAN-like adversarial framework for counterfactual data augmentation.

6 CONCLUSION

In this paper, a novel counterfactual synthesis policy has been proposed based on the casual view of MDP. The linking reward with the divergence between the observational and intervening policy performs well on different RL frameworks and achieves a users’ preferences. Results show that the counterfactual synthesis policy is simple to implement in various RL frameworks and works with the recommender agent. During the interaction between the recommender agent and the environment, our agent provides counterfactual data that takes into account the dynamic preferences. Results show that the counterfactual synthesis policy performs well on different RL frameworks and achieves a considerable improvement for all compared baselines.

REFERENCES

[1] Justin Basilico and Thomas Hofmann. 2004. Unifying collaborative and content-based filtering. In Proceedings of the twenty-first international conference on Machine learning. 9.

[2] Stephen Bonner and Flaviani Vasile. 2018. Causal embeddings for recommendation. In Proceedings of the 12th ACM conference on recommender systems. 104–112.

[3] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. 2018. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 1187–1196.

[4] Xiaocong Chen, Chaoran Huang, Lina Yao, Xiandi Wang, Wenjie Zhang, et al. 2020. Knowledge-guided deep reinforcement learning for interactive recommendation. In 2020 International Joint Conference on Neural Networks (IJCNN). IEEE, 1–8.

[5] Xiaocong Chen, Lina Yao, Julian McAuley, Guanglin Zhou, and Xianzhi Wang. 2021. A survey of deep reinforcement learning in recommender systems: A systematic review and future directions. arXiv preprint arXiv:2109.03540 (2021).

[6] Xiaohong Cheng, Lina Yao, Xianzhi Wang, and Julian McAuley. 2021. Locality-Sensitive Experience Replay for Online Recommendation. arXiv preprint arXiv:2102.08520 (2021).

[7] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2015. Deep reinforcement learning in large discrete action spaces. arXiv preprint arXiv:1512.07679 (2015).

[8] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In International conference on machine learning. PMLR, 1587–1596.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In International conference on machine learning. PMLR, 1861–1870.

[10] Joseph Y Halpern and Judea Pearl. 2020. Causes and explanations: A structural-model approach. Part II: Explanations. The British journal for the philosophy of science (2020).

[11] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. ACM transactions on interactive intelligent systems (tiis) 5, 4 (2015), 1–19.

[12] Timothy P Lillicrap, Jonathan J Hunt, Alexander Fritzke, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015).

[13] Dugang Liu, Pengxiang Cheng, Zhenhua Dong, Xiuqiang He, Weike Pan, and Zhong Ming. 2020. A general knowledge distillation framework for counterfactual recommendation via uniform data. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 831–840.

[14] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. Recommender systems hand- book (2011), 73–105.

[15] Chaoshao Lu, Biwei Huang, Ke Wang, Jose Miguel Hernandez-Lobato, Kun Zhang, and Bernhard Schölkopf. 2020. Sample-efficient reinforcement learning via counterfactual-based data augmentation. arXiv preprint arXiv:2012.09092 (2020).

[16] Prashan Madumal, Tim Müller, Lise Sonenberg, and Frank Vetrore. 2020. Explainable reinforcement learning through a causal lens. In Proceedings of the AAAI conference on artificial intelligence, Vol. 34. 2499–2508.

[17] Tariq Mahmood and Francesco Racci. 2007. Learning and adaptivity in interactive recommender systems. In Proceedings of the ninth international conference on Electronic commerce. 75–84.

[18] Judea Pearl. 2009. The Logic of Structure-Based Counterfactuals (2 ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511801611.009

[19] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. 2017. Elements of causal inference: foundations and learning algorithms. The MIT Press.

[20] Andy Ramlatchan, Mengyun Yang, Quan Liu, Min Li, Jianxin Wang, and Yashang Li. 2018. A survey of matrix completion methods for recommendation systems. Big Data Mining and Analytics 1, 4 (2018), 308–323.

[21] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. 2015. Policy distillation. arXiv preprint arXiv:1511.06295 (2015).

[22] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. In international conference on machine learning. PMLR, 1670–1679.

[23] Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. 2019. Virtual-taobao: Virtualizing real-world online retail environment for reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33. 4902–4909.

[24] Jieun Son and Seoung Bum Kim. 2017. Content-based filtering for recommendation systems using multivariate networks. Expert Systems with Applications 89 (2017), 404–412.

[25] Richard S Sutton and Andrew G Barto. 2018. Reinforcement Learning: An introduction. MIT press.

[26] Juntao Tan, Shuyuan Xu, Yingtian Ge, Yunqi Li, Xu Chen, and Yongfeng Zhang. 2021. Counterfactual explainable recommendation. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 1784–1793.

[27] Yixin Wang, Dawen Liang, Laurent Charlin, and David M Blei. 2020. Causal inference for recommender systems. In Fourteenth ACM Conference on Recommender Systems. 426–431.

[28] Zhenlei Wang, Jingsen Zhang, Hongteng Xu, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, and Ji-Rong Wen. 2021. Counterfactual data-augmented sequential recommendation. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 347–356.

[29] Kun Xiong, Wenwen Ye, Xu Chen, Yongfeng Zhang, Wayne Xin Zhao, Binbin Hu, Zhiqiang Zhang, and Jun Zhou. 2021. Counterfactual Review-based Recommendation. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2231–2240.

[30] Mengyue Zhang, Quanyu Bai, Zhenhui Dong, Xu Chen, Xiuqiang He, and Jun Wang. 2021. Top-N Recommendation with Counterfactual User Preference Simulation. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2342–2351.

[31] Shengyu Zhang, Dong Yao, Zhou Zhao, Tat-Seng Chua, and Fei Wu. 2021. Causerec: Counterfactual user sequence synthesis for sequential recommendation. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval. 367–377.

[32] Xiangyu Zhao, Liang Zhang, Long Xia, Zhuoyue Ding, Dawei Yin, and Jihang Tang. 2017. Deep reinforcement learning for list-wise recommendations. arXiv preprint arXiv:1801.02609 (2017).

[33] 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 Proceedings of the 2018 World Wide Web Conference. 367–377.

[34] Lixin Zou, Long Xia, Pan Du, Zhub Zhang, Ting Bai, Weidong Liu, Jian-Yun Nie, and Dawei Yin. 2020. Pseudo Dyn-Q: A reinforcement learning framework for interactive recommendation. In Proceedings of the 13th International Conference on Web Search and Data Mining. 816–824.