Multimedia Tools and Applications (2023) 82:20079–20100
https://doi.org/10.1007/s11042-022-14231-x

Proximal policy optimization based hybrid recommender systems for large scale recommendations

Vaibhav Padhye[1] · Kailasam Lakshmanan[1] · Amrita Chaturvedi[1]

Received: 19 March 2022 / Revised: 20 June 2022 / Accepted: 4 November 2022 /
Published online: 15 December 2022

Abstract
Recommender systems have become increasingly popular due to the significant rise in digital information over the internet in recent users. They help provide personalized recommendations to the user by selecting a few items out of a large set of items. However, with the growing size of item space and users, scalability remains a key issue for recommender systems. However, most existing policy gradient approaches in recommendations suffer from high variance leading to an increase in instability during the learning process. Policy Gradient Algorithms such as PPO are proven to be effective in large action spaces (a large number of items) as they learn the optimal policy directly from the samples. We use the PPO algorithm to train our Reinforcement Learning agent modeling the collaborative filtering process as a Markov Decision Process. PPO utilizes the actor-critic framework and thus mitigates the high variance in Policy Gradient Algorithms. Further, we address the cold start issue in Collaborative filtering with autoencoder-based content filtering. Proximal Policy Optimization (PPO) methods are today considered among the most effective reinforcement learning methods, achieving state-of-the-art performance and even outperforming Deep Q learning methods. In this paper, we propose a switching hybrid recommender system using the two different recommender system techniques. A switching hybrid system can switch between recommendation techniques depending on some criterion and can tackle its constituent recommender system's shortfall using the other counterpart in a particular situation. We show that our method outperforms various baseline methods on the popular Movielens datasets for different evaluation metrics. On Movielens 1m, our method outperforms the baseline by 9.19% in terms of R@10 and 3.86% and 6.58% in terms of P@10 and P@20, respectively. For the Movielens 100k dataset, our method improves on the baseline methods by 4.10% in terms of P@10 and 3.90% and 2.40% in terms of R@10 and R@20.

Keywords Proximal policy optimization · Hybrid recommender system · Reinforcement learning · Policy gradient algorithms

✉ Vaibhav Padhye
vaibhavpadhye10@gmail.com

[1] I.I.T BHU, Varanasi BHU, Varanasi, India

1 Introduction

Recommender systems suggest items to users based on available information such as previous usage patterns, user usage patterns, and features of the things themselves. Recommendation systems make it easier for the user to access the product according to his preference. People look to find their most preferred options from an extensive collection of items in the internet age. Generally, recommender systems provide the results in a specific domain: movies, books, e-commerce items, etc., to a user from a large database corpus. We classify Recommender systems into collaborative filtering and content-based systems. Hybrid systems are another type that combines the properties of two or more recommender systems in one. For example, it could be a combination of collaborative filtering and content-based, knowledge-based and collaborative filtering or two types of collaborative filtering systems to improve recommendation performance by usually dealing with the cold-start problem. Further, Collaborative filtering techniques are generally Memory-based or Model-based. Memory-based approaches typically include neighborhood-based methods and use rating data to find the similarity between the users or items [6, 28]. Model-based approaches learn a model from a collection of ratings and use it to make a prediction; examples include Singular Value Decomposition (SVD) based models [25]. However, with the growing size of items and users, recommender systems also face the issue of scalability [32]. With the ever-increasing space of users and items, finding a personalized search for the users is an important characteristic of recommender systems. Other than that, collaborative filtering systems also face the Cold Start issue. It is mainly due to insufficient ratings associated with items or users to make optimal recommendations. The cold start problem is further divided into complete cold start, i.e., when there are no ratings associated with the item or users, and incomplete cold start, when very few interaction data is available for the item or user. Collabarative filtering techniques have been extensively applied in recommender system, however they face challenges such as sparsity of interactive rating matrix, integeral nature of data, scalability, etc. [2]. Content based filtering on the other hand requires domain knowledge and need existing user interests profile to make recommendations and thus suffer with diversity in recommendations [32, 49]. Deep Learning based approaches have been used efficiently in recommender system [20, 39, 43, 48], but face challenges such as scalability and cold start [49].

Recently, Reinforcement Learning based recommender systems have been proposed and proven to be effective in modeling the recommendation process [14, 19, 31, 50, 51, 55]. Multi-Armed Bandits based approaches have been applied to recommender systems [16] which learns the user's preferences are learned with continuous interaction. However, MAB approaches assume user preferences remain static and do not change over a period of time during the recommendation process and thus fail to model the dynamic user preferences [55]. In [31], the authors proposed an MDP-based recommender system and used Q-learning for the learning process. However, with a very large number of items Q-learning becomes intractable as it limits its efficiency when action space becomes very large. RL based approach are generally classified as :*Model-based* and *Model-free*. Later, model-free approaches were applied to recommender systems and are classified as-value based [9, 51, 52, 55] and policy-based approaches [12, 19]. Value based approaches involves computation of Q values of all the actions for a state and then choose the maximum of all the Q-values as the best action for that state. Thus, evaluating Q-values for all the actions for each state, this approach can be inefficient if action space is very large, i.e millions of items as in recommender systems [11]. Thus, policy based approaches are preferred the most if the action space is large. Model-free approaches use either Monte Carlo or TD

Learning approach for calculating estimates. Monte-carlo methods suffer from high variance in large scale tasks whereas TD methods provide better efficiency through bootstrapping techniques but they do suffer from from a problem known as *Deadly triad* [38], that arises due to combination of function approximation, bootstrapping and offline training and leads to instability and inefficiency. Monte-Carlo based methods can lead to an very large action space and an unbounded importance weight of training samples leading to instability and convergence [54]. A Reinforcement Learning based recommender system has also been applied to address the cold-start issue in recommender system [55], where the authors used Q-leaning based approach for training the agent. However Q-learning suffers with curse of dimensionality as computing an value function for each action in a state becomes intractable as the state-action space becomes large, i.e., with very large number of items as in recommender systems. Deep Q Network(DQN) makes it possible to learn a high-dimensional state space by using a neural network for functional approximation and to stabilize learning. DQN based methods were proposed which combine the powerful approximation capabilities of neural network with Reinforcement learning. Deep Q Network based recommender system was proposed in [9, 50, 52]. However, DQN cannot be used when the action space is continuous because it is necessary to obtain actions that maximize the current action value in each state. Thus, DQN based methods are more suitable to small discrete action spaces whereas recommender systems generally comprise of normally contains large and high-dimensional action spaces [11]. Methods such as DQN, DDPG use maximization over the set of all actions at each step for the selection of action also becomes intractable with large action space size [3].

In [24], authors proposed Policy Gradient for Contextual Recommendation (PGCR), utilising the contextual information and used REINFORCE algorithm to train the agent. A policy gradient method based on REINFORCE was proposed in [10] for explainaible recommmendation. Another natural policy gradient based recommender system was proposed in [10] for top k recommender problem applying off policy learning, and performed experiments over youtube datasets.

Policy gradient algorithms learn the optimal policy directly by learning the policy parameters instead through neural networks. Policy-based approaches are useful for dealing with large action spaces, including continuous spaces with an infinite number of actions [11]. The agent learns the policy directly and chooses an action from a probability distribution of the action space in policy-based techniques. Further, policy-based methods can learn stochastic policy, unlike value-based methods, and thus handle the explore/exploitation conflict automatically. However, the traditional Policy Gradient approach have high variance due to gradient estimation and a large state space and action space will cause sample inefficiency issue [35, 47]. Proximal Policy Optimization, is an Actor-Critic method that combines the advantages of both DQN and Policy Gradient based approach and has achieved state-of-the-art performance in Reinforcement Learning, outperforming even Deep Q learning methods. PPO ensures low variance during learning by making sure updated policy isn't too much different from the old policy with a clipped objective surrogate function. The reduction of variance helps to increase the stability of the learning process. Further, the PPO agent limits policy gradient step and reduces the variance of the estimation using an advantage function so it does not move too much away from the original policy, causing overly large updates that often makes the policy unstable. We address the issues above using the state-of-the-art Proximal Policy Optimization, which uses on poilcy learning and thus avoid deadly triad. We also use Probabilistic Matrix factorization approach to obtain our state from the user and item matrix. This approach further helps with the sparsity issue as PMF approach performs very well on large, sparse and imbalanced matrices [22]. In this paper, we propose a

Reinforcement learning-based based approach to address these issues modeling the recommendation process as a Markov Decision Process. We utilise the policy gradient approach for recommendation in large action space. Further, we combine the MDP based collabarative filtering with autoencoder-based content filtering to address the cold start problem.

We summarize our contributions as follows:

1. We propose a Reinforcement learning-based hybrid recommender system for large action space, i.e., for a large set of items, utilizing the state-of-the-art PPO algorithm to train our agent.
2. We use the Proximal Policy Optimization approach to train our agent and modeled recommender system as a Markov Decision Process. The PPO based algorithm mitigates the high variance and leads to increase in stability as described above and further in detail in Section 2.4.
3. In our approach, we addressed the cold start issue using a switching Hybrid recommender setting. We divided users into cold users and hot users depending on the number of ratings provided by them.
4. We demonstrate the effectiveness of the proposed framework on the two popular datasets : Movielens 1m and Movielens-100k, for different evaluation metrics.

The rest of this paper is organized as follows. In Section 2, we describe the preliminaries of the Hybrid Recommender system and Reinforcement Learning. We formally describe the Markov Decision Process and the Policy Gradient algorithm. In the next section, Section 3, we provide the literature survey and the related work. In Section 4, we formally define our problem and provide the approach of our solution. We describe the MDP Based Collaborative filtering in detail, outlining the MDP model with reward structure. Further, we also describe Autoencoder-based content filtering to address the cold start issue. Section 5 carries out the experiments on the popular Movielens datasets. In Section 5, we describe our algorithms in detail. Section 6 describes the experimental settings and evaluation metrics to carry out the experiments on the two datasets. In Section 7, we present the results. Finally, Section 8, concludes this paper and provides future research scope for this work.

2 Background

2.1 Hybrid recommender systems

Hybrid recommender systems [8] are generally composed of two or more recommender systems. It can address issues in recommender systems such as data sparsity, cold start, scalability, etc. Collaborative filtering is quite effective in dealing with personalized recommendations, but they struggle with the cold start issue. Hybrid recommender systems try to address the cold start issue by combining multiple recommender systems, generally collaborative filtering, along with some other types such as content-based, demographic-based, knowledge-based, etc. There are different types of hybridization techniques such as switching, weighted, cascaded, mixed, etc. Switching hybrid systems use specific recommender systems depending on some criteria and can switch to another type accordingly. Weighted systems combine the scores of various types of recommenders involved. Cascade recommender type performs staged recommendation where one recommender system generates a candidate list of items that are further refined by the subsequent recommender.

2.2 Reinforcement learning

In reinforcement learning, the agent learns the policy that can maximize long-term cumulative reward through interacting with the environment [34]. MDP consists of a tuple of five elements $(\mathbf{S}, \mathbf{A}, \mathbf{P}, \mathbf{R}, \gamma)$, where S is the set of states, A is the set of action, $T(s_{t+1}|s_t, a) : S \times A \times S \rightarrow \mathbb{R}$ is the transition probability of reaching state s_{t+1} after executing action a on state s_t. , $R(s, a) : S \times A \rightarrow \mathbb{R}$ is the immediate reward after executing action a from state s_{t+1} , and γ is the discount factor.

Solving an MDP typically refers to finding a policy $\pi : S \rightarrow A$ such that from any given state s, executing action $\pi(s)$ and then acting optimally (following the optimal policy π^*). The Reinforcement learning framework is shown in Fig. 1, depicting an RL agent receiving the state from the environment and generates an action. The environment further provides feedback on the action taken by the agent.

In Reinforcement Learning, the agent's goal is to maximize the expected cumulative reward. So, the agent needs to find an optimal policy that resembles the optimal strategy for the agent to act in an environment. A policy is generally a function that outputs an action to take, given the current state of the environment outputs an action .and can either be stochastic or deterministic.

For example, consider the Cartpole game in which a pole is attached by a joint to a cart, which moves along a frictionless track. The state consists of cart position, cart velocity, pole angle, and pole velocity at the tip. The system is controlled by applying a force of +1 or -1 to the cart (moving left or right). The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The game is divided into terms of episodes, each containing a specific number of steps. The episode ends when the pole is more than 15 degrees from vertical or the cart moves more than 2.4 units from the center. The goal of the game is to have the cumulative reward as high as possible.

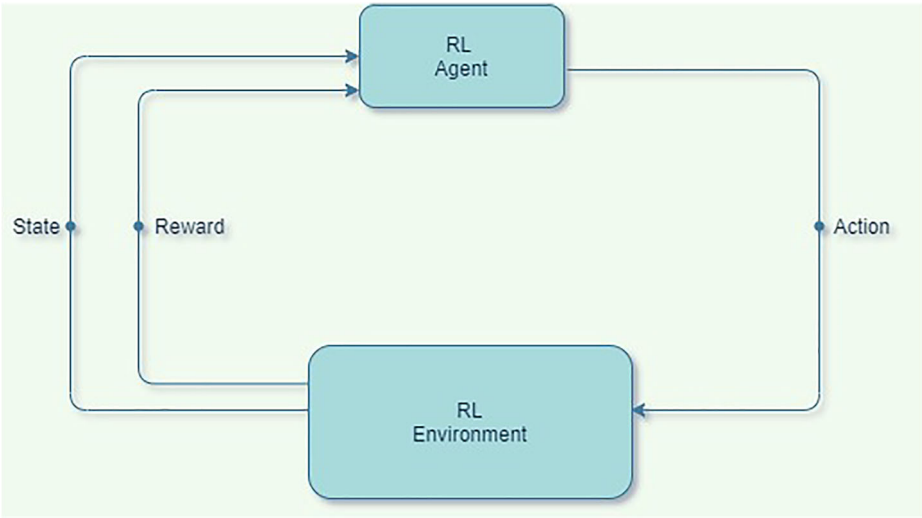

Fig. 1 Reinforcement learning framework

2.3 Policy gradient algorithms

Policy gradient methods [34] is a type of reinforcement learning technique that directly optimize policy with respect to the expected return instead of evaluating optimal value functions associated with the value-based methods. Let π_θ denote a policy with parameters θ, and $J(\pi_\theta)$ denote the expected finite-horizon undiscounted return of the policy. The goal in the Policy Gradient method is to find the optimal choice of θ that maximizes the objective $J(\pi_\theta)$. We have $J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$, where τ is a trajectory and $R(\tau)$ denotes the reward over the trajectory.

The gradient of $J(\pi_\theta)$ is

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta} \right], \tag{1}$$

where A^{π_θ} is the advantage function for the current policy, $A_t = Q(s_t, a_t) - V(s_t)$. The gradient of policy performance, $\nabla_\theta J(\pi_\theta)$, is called the policy gradient, and algorithms that optimize the policy this way are called policy gradient algorithms. Policy gradient implementations typically compute advantage function estimates based on the infinite-horizon discounted return, denoting how much better or worse taking action a in the state is compared to acting according to the policy. The policy gradient algorithm works by updating policy parameters via stochastic gradient ascent on policy performance,

$$\theta_{k+1} = \theta_k + \alpha \nabla_\theta J(\pi_{\theta_k}).$$

2.4 Proximal Policy Optimization

Proximal Policy Optimization (PPO) [29], is a recently proposed and effective policy gradient algorithm. TRPO uses a second-order derivative matrix which increases the complexity for large-scale problems. Mostly policy gradient involves a second-order derivative which makes it inefficient for large-scale problems as the computational complexity is too high for real tasks.PPO instead uses a first-order optimizer like the Gradient Descent method, which makes it more applicable for large-scale tasks and considerably efficient for computation and can be used in both discrete and continuous environments.PPO tries to limit the difference from one policy to the next so as to regulate new policy not to be too different from the current one. PPO uses the Minorize-Maximization MM algorithm by iteratively maximizing a lower bound function M approximating the expected reward η locally (See (2)). PPO starts with an initial policy guess and finds a lower bound M for η at this policy. It optimizes M and uses the optimal policy for M as the next guess. It approximates a new lower bound again and repeats the iterations until the policy converges. There are two primary variants of PPO: PPO-Penalty and PPO-Clip. PPO-Penalty uses KL-divergence to measure how much policy changes in each iteration. KL-divergence measures the difference between two data distributions, p, and q.

$$D_{KL}(P \,||\, Q) = \mathbb{E}_x \log \frac{P(x)}{Q(x)}, M = L(\theta) - C.\overline{KL} \tag{2}$$

Here M is a lower bound funcion as described above, approximating expected reward. $L(\theta)$ equals $\mathbb{E}_t[\,\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t$, denoting the expected advantage function for the current policy which is estimated by the new policy and then reordered using the probability ratio between

the current and the old policy. The second term denotes the KL Divergence, which measures the difference between the two policies. The main objective in PPO-Penalty can be summarized as :

$$\max_{\theta} \text{s.t.} \quad \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \hat{A}_t \right] = \mathbb{E}_t[\, r_t(\theta) \hat{A}_t] - \beta \mathbb{E}_t[\, \text{KL}[\pi_{\theta_{old}}(.|s_t), \pi_\theta(.|s_t)]] \tag{3}$$

, here β controls the weight of the penalty. It penalizes the objective if the new policy is different from the old policy and thus changes between iterations to ensure the KL Divergence constraint is satisfied. $\mathbb{E}_t[\, \text{KL}[\pi_{\theta_{old}}(.|s_t), \pi_\theta(.|s_t)]]$ is the KL-divergence between the old and the new policy. If it is higher than a target value, we reduce β and vice versa. PPO-Clip doesn't have a KL-divergence term in the objective and doesn't have a constraint at all. Instead relies on specialized clipping in the objective function to remove incentives for the new policy to get far from the old policy. It uses a clipping function to make sure the new policy does not deviate much from the older policy. With clipped objective, we compute a ratio between the new policy and the old policy, $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. This ratio measures how difference between two policies. If the new policy is far away from the old policy , then we clip the estimated advantage function. The objective function in PPO Clip can be summarized as:

$$L_\theta^{CLIP} = \mathbb{E}[\, \min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\,] \tag{4}$$

Thus, PPO maintains the low variance as the new policy obtained from the old policy does not differ much, which is ensured through the clipped objective surrogate function. The advantage function above reduces the variance of the estimation, so it does not move too much away from the original policy, avoiding large up- dates which can lead to unstable policy. Therefore, if the probability ratio between the new policy and the old policy falls outside the range $(1-\epsilon)$ and $(1+\epsilon)$, the advantage function will be clipped. For eg, if ϵ is .25, then the $r_t(\theta)$ could vary between .75 to 1.25. And finally, it takes the minimum of the two, i.e., clipped and unclipped objective, and so thus, the algorithm doesn't become too greedy and prohibits making too many updates at once.

3 Related work

Recommender systems today have been extensively applied in diverse fields such as e-commerce items [17, 20, 37, 39, 44], healthcare [4, 5], news [16, 53], travel [33], Internet of Things (IoT) [18], etc. Collaborative filtering systems [28] are the most popular recommender system technique. In the CF algorithm, the core idea is that the past preference behaviors of users have a significant influence on their future behaviors, and their previous behaviors are consistent with future behaviors. Generally speaking, we estimate the similarity between users according to the user's historical behavior. And then, according to the evaluation of high similarity neighbors to the target users, the target users are predicted to be interested in the item. CF techniques are generally memory-based or model-based. Memory-based CF the recommendation uses the user's historical rating data to find the similarity between the items or users, i.e., item-based or user-based types. They generally use neighborhood-based methods [28] to calculate the similarity between two users or items. Model-based techniques [21] use machine learning techniques such as clustering

methods, SVD, PCA, Bayesian models, Matrix factorization, etc., to calculate the user's rating on required items. These methods generally assume that latent vectors determine user preferences in a low dimensional space that reflects users' interests. Probabilistic matrix factorization [22] model scales linearly and perform well on large and sparse data sets, while [23] presented Bayesian probabilistic matrix factorization (BPMF) model, extending PMF to the Bayesian model, which increased the accuracy.

Knowledge-based systems [1] recommend items based on specific domain knowledge about how certain item features meet the user's needs and preferences and how the item is useful for the user. Demographic recommender systems [41] categorize users or items based on their personal attributes and make a recommendation based on demographic categorizations. Hybrid systems [8] typically combine two or more approaches to provide better recommendations; usually, content-based and collaborative filtering approaches or variations of collaborative filtering approaches [15]. The works [41, 42] employ a hybrid recommender system utilizing demographic and feature information of users and items. In [41], the authors used demographic information about users and items to provide more accurate predictions for user-based and item-based CF. Deep learning-based recommender systems have also been applied recently and use different neural network techniques such as Autoencoder, RBM, CNNRNN deep belief networks, etc., to perform recommendations. Autoencoders have been used for CF problems [30]. Zheng et al. [46] proposed the Collaborative De-noising AutoEncoder (CDAE), which utilizes a Denoising AutoEncoder [40] to perform collaborative filtering. They learn hidden structures that can reconstruct a user's ratings given previous ratings as inputs. A deep hybrid recommender system was proposed in [20] using neural networks to obtain user and item features from side information to perform collaborative filtering. In [39], deep belief networks are used in music recommendations while [27] proposed a restricted Boltzmann machine-based recommender. Collaborative Deep Learning (CDL) [43] is a hierarchical Bayesian model which integrates stacked denoising autoencoder (SDAE) into probabilistic matrix factorization.

Reinforcement learning has been used to recommend web pages, travel information, books, news, etc. Policy-based approaches [12, 14, 50] are preferred over value-based approaches in large state-action space as they directly evaluate the policy instead of calculating the value over all the actions. Deep RL-based algorithms [50, 53] are also applied recently in Recommender Systems, which combine deep learning techniques with reinforcement learning. A Deep Deterministic Policy Gradient (DDPG) based algorithm was proposed in [19] to tackle the large action space issue prevalent in RL-based recommender systems.

WebWatcher [36], exploits Q-Learning to assist users to their desired pages in World Wide Web. It represented pages as states, hyperlinks as actions, and computed rewards based on the similarity of the pages and user profiles. A travel recommender system based on the Q learning approach was proposed in [33] which used a linear function to rank the different trips based on various attributes like trip price, location duration, etc. An MDP-based recommender system [31] was proposed, which made the recommendation as a sequential optimization problem.

An interactive recommender system was proposed in [50], and a similar page-wise recommender using deep reinforcement learning utilizing real-time feedback from the users to display the set of recommended items on a page with a proper display. Zhang et al. [51] proposed another deep reinforcement learning based recommender system by capturing both positive and negative feedback. Multi-armed bandits based solutions have also been applied to recommender systems. LinUCB [16] modeled news article recommendations as a contextual bandit problem where the algorithm recommends articles to users based on the

contextual information related to the users and articles. It represented news feature vectors, and the click-through rates of news were taken as the payoffs.

4 A reinforcement learning based hybrid recommender system

In RL, an agent should decide the best action to select based on his current state. When this step is repeated, the problem is known as a Markov Decision Process. The recommender system can be modeled as an MDP. An agent interacts with the environment and, in return, receives feedback. As a result, the agent moves to the next state. We used probabilistic matrix factorization to obtain the latent vector representing the state.

The recommender provides an item to the user, then receives a rating on the item given by him/her. After considering the observed rating, the recommender updates its knowledge about the user and provides a new item at the next time step. Suppose such a recommender–user interactive process will last for T time steps. The goal of the recommender is to provide user u with the most interesting items that maximize the reward received over T steps. In our approach, we addressed the cold start issue using a switching Hybrid recommender setting. We divided users into cold users and hot users depending on the number of ratings provided by them. Further, for users who have given very few ratings, i.e., cold set users, we applied autoencoder-based content filtering, whereas, for the other set of hot users, we applied MDP-based collaborative filtering. For users with no ratings at all, we used demographic information to recommend items to the user(Section 4.2). Below we describe the two techniques.

4.1 MDP based collaborative filtering

Here we modeled the collaborative filtering system as a Markov Decision Process.

- **State Space S**: a state $s_t^1, ...s_t^n \in S$ is represented as a latent feature vector of the item with meta information. The state is obtained from the Matrix factorization method. Matrix factorization is a collaborative filtering technique used in recommender systems that work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. We used Probabilistic Matrix factorization as described below to obtain our MDP states.

4.1.1 Probabilistic matrix factorization

Probabilistic matrix factorization proposed in [22] follows probabilistic approximation on the Matrix factorization and factors the base interaction matrix into two lower rank matrices. PMF performs effectively on sparse datasets as, generally, the real-world datasets scale well with the input. In our problem, PMF factorizes the rating feedback interaction matrix into user and movie matrices. Let the feedback rating matrix be X. In MF, X is factorized into the product of two low-rank matrices U, V. Both U, V are d–dimensional matrix where d corresponding to feature dimension with vectors u_i and v_j describing the behavior of the i-th user and j-th item respectively.

The aim of the PMF is to obtain the matrix U and V, which are the parameters of the model. PMF uses Bayesian learning to estimate parameters to find the posterior distribution of the model parameters by applying the Bayes rule. The main idea in the Bayesian learning process is to iteratively use the knowledge about the data distribution to adjust the model

parameters fitting to the data. The posterior distribution parameters are plugged into the prior distribution in the next iteration of the learning process, and the process is continued till the posterior attains the convergence.

The PMF algorithm uses interaction matrix X as the dataset with U and V as its parameters. Using the Bayes inference, PMF obtains the posterior distribution of model parameters as,

$$p(U, V|X, \sigma^2) = p(X|U, V, \sigma^2)p(U, V|\sigma_U^2, \sigma_V^2) \tag{5}$$

We obtain the user and item latent factors from the rows and columns associated in matrix U and V, respectively. The prior for feature vectors is determined by the user's ratings in PMF, with assumption that users who provide similar ratings to the items have identical prior distributions over the feature vectors.

$$X_{ij} = U_i^T V_j = \sum_k U_{ik}^T V_{jk}$$

In PMF, the ratings X are modeled as draws from a Gaussian distribution.

$$X \sim \mathcal{N}(\mu, \sigma^2),$$

with $\mu = U_i^T V_j$ and σ denoting variance .

- Let U_i and V_j denote the user and movie feature vectors, the probability distribution of the corresponding rating can be obtained as:

$$p(X_{ij}|U_i, V_j, \sigma^2) = \mathcal{N}(X_{ij}|U_i^T, V_j, \sigma^2)^{I_{i,j}},$$

where I_{ij} is an indicator function having value 1 if rating at row i, column j is present, and zero otherwise.

PMF places zero-mean Gaussian priors on the user and movie feature vectors and obtains the prior distributions for U and V as,

$$p(U|\sigma_U^2) = \prod_{i=a}^{b} \mathcal{N}(U_i|0, \sigma_U^2 I), \;\; p(V|\sigma_V^2) = \prod_{i=a}^{b} \mathcal{N}(V_i|0, \sigma_V^2 I) \tag{6}$$

It can be observed that maximizing the posterior above is same as maximizing log posterior and can be obtained as minimising the sum of squared errors over features with regularisation.

$$\frac{1}{2} \sum_{j=1}^{\infty} \sum_{k=1}^{\infty} I_{ij}(X_{ij} - U_i, V_j)^2 + \frac{\lambda_U}{2} \|U_i\|_{Fro}^2 + \frac{\lambda_V}{2} \|V_j\|_{Fro}^2 \tag{7}$$

where , $\lambda_U = \frac{\sigma^2}{\sigma_U^2}$, $\lambda_V = \frac{\sigma^2}{\sigma_V^2}$, and $\|.\|$ represents Frobenius norm .

By minimising the preceding equation and performing gradient descent until convergence, we obtain the revised update equations for U_i and V_j.

$$U_i^* \leftarrow U_i - \lambda_U \frac{\partial L(U, V)}{\partial U_i} \;\; , \;\; V_j^* \leftarrow V_j - \lambda_V \frac{\partial L(U, V)}{\partial V_j} \tag{8}$$

with , $\lambda_U \frac{\partial L(U,V)}{\partial U_i} = -\sum_j I_{ij}(X_{ij} - U_i, V_j)V_j + \frac{\sigma}{\sigma_u} U_i$, $\lambda_V \frac{\partial L(U,V)}{\partial U_i} = -\sum_i I_{ij}(X_{ij} - U_i, V_j)U_i + \frac{\sigma}{\sigma_u} V_j$.

- Action space A: An action $a_t^1, ...a_t^n \in A$, is to recommend items to a user at time t based on the current state s_t. We recommend a single item at a time. The selection of

the particular action depends on the current policy Action determines the next item to recommend. We use an actor policy network that predicts the rating.
A policy $\pi : S \rightarrow A$, is a function that indicates for each state $s \in S$, the action a taken by the Agent in that state. In a standard actor–critic approach, the policy is explicitly defined by a parameterized actor function $\pi_{(\theta)} : S \rightarrow$ A. We implement our policy using the actor–critic approach. We use the actor policy network to generate actions and utilize the critic to refine the actor's choices. We use a policy network to predict the ratings of a particular item, and if the predicted rating is 3 or above, then we recommend the item most similar to the item corresponding to the current state. We used the nearest neighbor technique employing cosine similarity for finding similar items. Otherwise, we recommend a random item, thus increasing exploration. As a result, the Agent moves to the next state (item). Both actor and critic methods are implemented with multi-layer perceptron networks. We used the PPO algorithm here to train our policy.

- Transition between the states is deterministic as our agent moves to a specific state. The selection of the particular action depends on the Agent's current policy and determined through the actor-critic framework.
- Reward R: After the recommender agent takes action a_t at the state s_t, i.e., recommending an item to a user, the user provides his feedback. Our reward $r(s_t, a_t)$, is in the range [-1 , 0 ,+1] , depending on the user feedback. We provide a -1 reward in the case user has given a rating below or equal to 2. We provide reward 0 in case the user has not rated the particular item or rated 3. Similarly, reward +1 is given in the case user has given a rating of 4 or 5.
- Discount factor $\gamma : \gamma \in [0, 1]$ defines the discount factor when we measure the present value of the future reward. In particular, when $\gamma = 0$, RA only considers the immediate reward. In other words, when $\gamma = 1$, all future rewards can be counted fully into that of the current action.

4.2 Autoencoder based content recommender system

Autoencoders are neural networks used in unsupervised learning methods, including generative modeling, dimensionality reduction, etc. They encode the data into a reduced lower-dimensional form and reconstruct the data back from the reduced representation.

Content-based filtering use product features to generate recommendations such as tags, text description, reviews, and other meta information. Content-based recommender systems are more robust against the cold start problem than collaborative systems as they can utilize product information.

Embeddings in natural language processing are a kind of word representation that allows words with similar meanings to have a similar representation by mapping them to a latent vector space and are often obtained via neural networks. Word embeddings are better than one hot encoding because they can retain the similarity information between different words, which is lost due to the orthogonal nature of the one-hot encodings.

We use neural network-based embeddings for the cold users, which are learned from textual information describing the items and demographic data associated with users. We utilized cosine similarity to find the similarity between the movies. It ranges between -1 to 1 and is determined by the dot product between two vectors divided by their magnitudes. Embedding vectors receive a high cosine similarity score if they point in the same direction and vice versa. Item embeddings thus can be obtained from the content-related information associated with the data.

Movielens dataset contains movie tag information generated by the users. The tags are generally a single word or a phrase. The corpus of documents is created composed of movies and the tags related to them. Each document, in particular, is the movie, along with all the tags for the particular movie. We created the Term Frequency-Inverse Document Frequency (TF-IDF) representation from the tag documents in the corpus.

TF-IDF is a technique that measures how relevant a word is to a document in a corpus of documents. It is obtained by the product of two metrics: how many times a word appears in a document (TF) and the inverse document frequency of the word across a set of documents(IDF). The higher the TF*IDF score, the rarer the term and thus more relevant and vice versa. However, TF-IDF representations can be high dimensional, so to compress the data high-dimension TF-IDF vector into low-dimension embeddings, we use autoencoders to learn the representation of higher dimensional data into a corresponding lower form by reducing the noise. We further reduced the dimensions to 50 and 100 in the ML 100K and Movielens 1m dataset, respectively. We find the similar TF-IDF vectors, i.e., movies, corresponding to the movies rated highest by the current user using the cosine similarity metric to calculate the similarity between the movies. It calculates the cosine angle between the two vectors in a high dimensional space and varies between -1 to 1.

$$sim(A, B) = cos(\theta) = \frac{A.B}{||A||||B||}$$

5 MDP based PPO Algorithm

Collaborative filtering systems generally do not perform well with cold users due to less user interaction history associated with the items. We, therefore, perform hybrid recommendation by using the autoencoder-based content recommender systems for the set of cold set users and Reinforcement learning-based collaborative filtering for the other user set. In our solution, we divide the user sets into two. We employed content-based filtering for the users in cold set and employ an MDP-based recommender for the rest of the users. The flowchart for the complete process is depicted in Fig. 2. If the user is classified as a hot user based on his rating profile in the interaction matrix, we perform MDP-based collaborative filtering, as depicted in the Figure. First, it obtains the state from PMF using the user and item matrix. The state is then fed to the RL policy, i.e., the PPO agent, which outputs an action. The action corresponds to recommending an item. After, the agent receives a reward from the environment and proceeds to the next state. If a user has not provided any rating, i.e., complete cold start, we used user embedding to find similar users and recommend the items corresponding to them. We used the demographic information included in the dataset to obtain user embeddings. The demographic data consists of user gender, age, occupation, zip code, etc. If the user has provided few ratings, i.e., incomplete cold start problem, we use the technique mentioned above using autoencoders to give recommendations to the user. The two algorithms for the MDP-based collaborative filtering are depicted in Algorithms 1 and 2. Algorithm 1 shows the process of generating MDP transitions. First, it uses PMF to obtain the state and then select the action according to the policy. Then it obtains the rating from the policy network and recommends the item. Finally, it receives the reward and proceeds to the next state. Algorithm 2 shows training the PPO agent. First, we obtain the set of partial trajectories from Algorithm 1. We then utilize the advantage function to obtain the advantage estimate. Further, we update the policy by maximizing the PPO clip objective(see equation 4) and then compute the policy objective using gradient descent.

s_0 initial state , obtained with pmf , described above
for $t = 1, 2, \ldots, N$ **do**
  Obtain the next state S_i for item I_i according to PMF
  select action according to policy
  obtain rating $r = \pi(s_i)$ from the policy network
  recommend item I_j as described in 5.1
  Get feedback from the user for item I_j as U_{rat}
  if $U_{rat} \geq 4$ **then**
    $reward = +1$
  else if $U_{rat} = 3$ **then**
    $reward = 0$
  else
    $reward = -1$
  end if
end for

Algorithm 1 Generating Transitions.

Initial policy parameters θ_0
s_0 initial state , obtained with pmf , described above
for $k = 1, 2, \ldots$ **do**
  collect set of partial trajectories D_k on policy $\pi_{(k)} = \pi_{(\theta_k)}$ as in Algorithm 1
  Estimate adavantage A using A(S,A) (Section 3.3)
  update policy by maximizing ppo clip objective using (4)
  Compute policy update

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}}\, L_{\theta_k}^{CLIP}(\theta)$$

  using Gradient Descent where ,

$$L_{\theta_k}^{CLIP} = \mathbb{E}_{\tau \sim \pi_k}[\sum_{t=0}^{r}[\min(r_t(\theta)A_t^{\pi_k}, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t^{\pi_k}]]$$

end for

Algorithm 2 MDP Based PPO Algorithm.

6 Experiments

6.1 Experimental settings

We used the Movielens datasets ML 1m and ML 100k datasets for our experiments. ML 1m contains contains 1,000,209 anonymous ratings of 3,952 movies made by 6,040 MovieLens users, in which ratings are on the scale 1-5, while ML 100k contains 100000 ratings of 943 users and 1682 movies on the same rating scale.

We used time 80% of the dataset for training purposes, whereas the remaining 20% is used for testing the recommendations. We used grid search for tuning the hyperparameters.

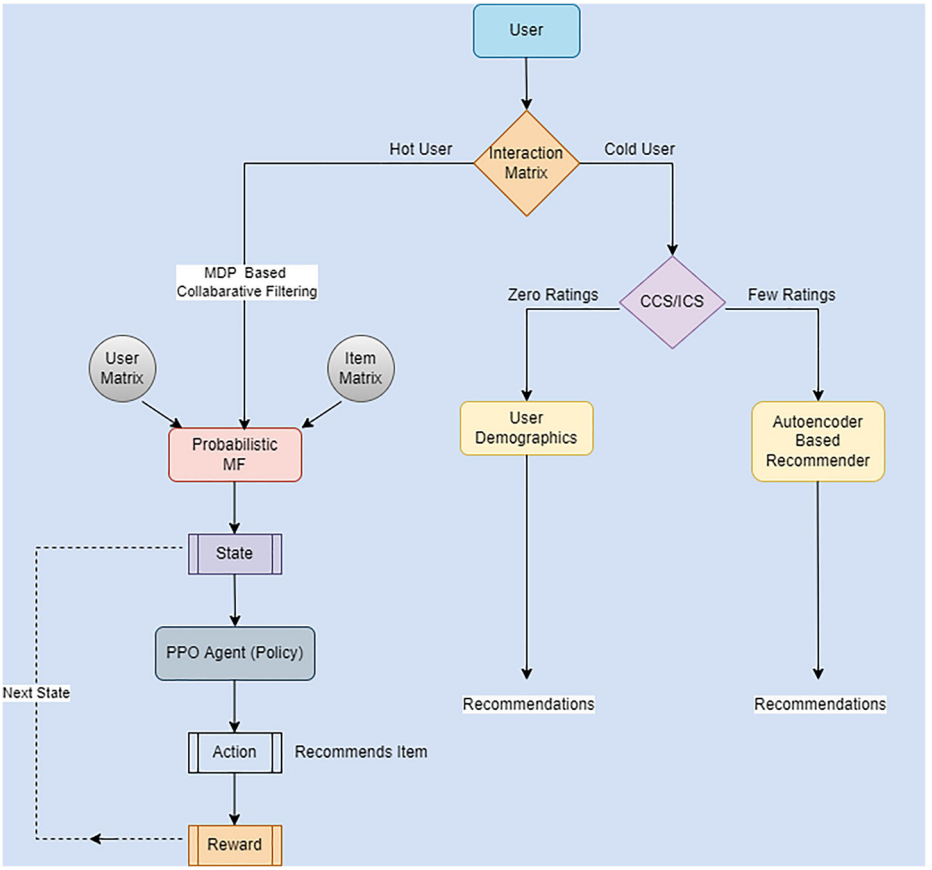

Fig. 2 Flowchart depicting the modeling of recommendation process

We used batch size of 128, and the learning rate for actor-critic networks as 0.001. We ran the PPO based agent for over 1000 episodes. We varied the percent of the cold user, i.e., the least n percentage of users who have given a minimum number of ratings to the movies. The different values for clipping ratio, cold user threshold, and discount factor for the hyper-parameters are also shown in Table 1. We also used different embedding sizes obtained from the PMF for finding the optimal ones on the two datasets. We used the Open AI Gym Environment to create a Markov Decision Process agent [7]. Further, we used the PPO stable-baselines framework for training the agent [26]. We ran the tests 5 times for the agent and reported the average results in the tables and figures below.

Table 1 Hyperparameters

Cold user percent	3%	5%	7%	10%
Discount Rate	0.85	0.90	0.95	0.99
Clipping Ratio	0.05	0.10	0.15	0.20

6.2 Evaluation baselines and criterion

We included the following popular metrics for evaluating the recommender system.

Precision: For a recommender system, it measures the fraction of recommended items that are relevant to the user.
Precision@k = (# of recommended items @k that are relevant) / (# of recommended items @k)
Recall : recall represents the fraction of relevant documents that have been selected.
Recall@k = (# of recommended items @k that are relevant) / (total # of relevant items).
F1 score combines precision and recall and is obtained by computing the harmonic mean of the precision and recall.

We used the following baselines for comaprisons:

Random: A method that randomly recommends the items to a user.
PopRank: A popularity based methood which recommends the most popular items.
Content Based Filtering : A content-based method that recommends items that are similar to those that the individual user has liked in the past.
LinUCB: It's multi armed bandit approach which recommends items to the user based on the contextual information about the user and items [16].
SVD: It is a popular algorithm utilizing Singular Value Decomposition for the process of recommendation [25].
Deep-MF: A state-of-the-art neural network architecture based Matrix Factorization Method for recommendation [48].
Neural-MF: A state-of-the-art method using deep neural network structure for Collaborative Filtering [13].

For SVD, we used a learning rate of 10^{-4}, regularization rate of 0.02 and a factor-size of 50 for ML-100K and 100 for ML-1M, same as that of our algorithm. In LinUCB, the only hyperparameter is α, which determines the trade-off between exploration and exploitation. We varied α for different values [0.1, 0.5, 1, 2] for selecting the optimum value. For content based filtering, we use tf-idf and cosine similarity for finding K Most similar movies from the dataset. For K, we varied 20, 50 and 100. For Neural-MF(NMF), we randomly initialized model parameters with a Gaussian distribution, with mean = 0 and standard deviation = 0.01, and optimizing the model with mini batch Adam. We used a batch size of 512 and varied the learning rate as [0.0001, 0.0005, 0.001, 0.005] for testing. Further, for neural network, we employed 3 hidden layers for MLP. For Deep CF, we set the depth of hidden layers to 3, the batch size to 256 and the varied the learning rate as for NMF. We conducted the experiments on Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz processor with 264 GB RAM. The machine has x86_64 architecture with 40 CPUs and multiple level cache. We ran the experiment on Python 3.6 framework on Ubuntu 16.04 LTS.

7 Results

This section compares our approach to the baseline methods mentioned earlier for the evaluation metrics described above. Our approach is specified as MDP, and we tuned the hyperparameters with The results of comparing our proposed approach with the baselines are presented in Tables 2 and 3 in terms of Precision@k Recall@k and F1@K on the two

Table 2 Results for different metrics on Movielens 100k

Method	P@10	R@10	F1@10	P@20	R@20	F1@20
Random	0.1641	0.0782	0.1057	0.1124	0.1271	0.1190
PopRank	0.2163	0.1002	0.1368	0.1572	0.1714	0.1639
CB	0.2458	0.1072	0.1491	0.1788	0.1702	0.1743
LinUCB	0.3210	0.1354	0.1921	0.2860	0.2035	0.2380
SVD	0.3432	0.1613	0.2218	0.2824	0.2453	0.2715
DeepMF	0.3224	0.1428	0.1979	0.3077	0.2283	0.2621
NMF	0.3441	0.1601	0.2185	0.3142	0.2438	0.2745
MDP	0.3573	0.1676	0.2282	0.3130	0.2512	0.2787

datasets. Our approach, MDP, outperformed the different baseline methods for most metrics. For the Movielens 100k dataset, our model outperformed the different baselines on all metrics except for P@20 and R@20 for the Movielens 1m dataset. For both of these metrics, Neural MF gave the best performance.

Among baselines, Neural MF performs the best, followed by the Deep CF method and SVD, whereas PopRank performs significantly better on Movielens 1m dataset compared to Movielens 100k. LinUCB outperforms SVD on the ML-1M dataset for the recall@20 metric and on the P@20 on the Movielens-100k Dataset. Further, On Movielens 1m, our method outperforms the baseline by 3.86% and 6.58% in terms of P@10 and P@20, respectively, and 9.19% in terms of R@10. For the Movielens 100k dataset, our method improves on the baseline methods by 4.10% in terms of P@10 and 3.90% and 2.40% in terms of R@10 and R@20. In Fig. 3, we show the reward distribution of our algorithm over the two datasets, Movielens 100k and Movielens 1m. We can observe that algorithm converges better on the Movielens 1m as compared to the Movielens 100k dataset. For the Movielens 1m dataset, we can observe the stability in reward distribution after 700 episodes, which again we can speculate due to the Policy gradient method achieving stability in convergence due to more sample data associated with the dataset as compared to Movielens 100k. Further, as the number of episode increase, the agent achieves more stable learning on both the datasets, implying gradual convergence of gradient descent. Thus we can observe stability of our algorithm, *MDP* from the figure. The effect of varying the parameters such as discount factor and clipping ratio on the Ml-100k dataset is further shown in Figs. 4 and 5, respectively. In Fig. 6, we showed the effect of different embedding sizes obtained from the

Table 3 Results for different metrics on Movielens 1m

Method	P@10	R@10	F1@10	P@20	R@20	F1@20
Random	0.1380	0.0723	0.0946	0.0950	0.1134	0.1033
PopRank	0.2841	0.1121	0.1607	0.2108	0.1823	0.1955
CB	0.2712	0.1128	0.1593	0.2047	0.1774	0.1900
LinUCB	0.2988	0.1167	0.1678	0.2213	0.2385	0.2295
SVD	0.3167	0.1318	0.1853	0.2762	0.2308	0.2514
DeepMF	0.3043	0.1286	0.1806	0.2551	0.2391	0.2468
NMF	0.3180	0.1348	0.1893	0.2714	0.2612	0.2662
MDP	0.3314	0.1472	0.2039	0.2944	0.2576	0.2748

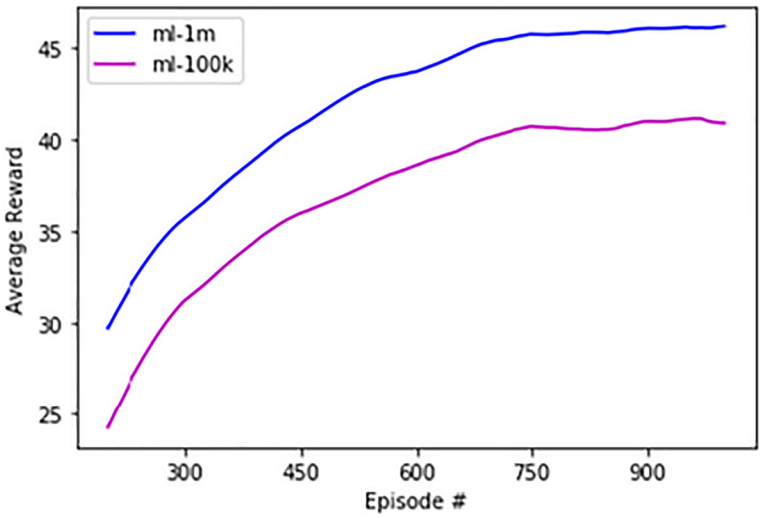

Fig. 3 Reward Distribution of MDP over 2 datasets

PMF on the two datasets. In Fig. 7, we further display the change in precision metric for the two datasets with respect to the threshold of cold user percentage. We can observe that our algorithm performs better on the Movielens 1m dataset as compared to Movielens 1ook. We can speculate as the Movielens 1m data has much more interactive rating data available as compared to Movielens-100k, our model is able to learn a better model as the policy gradient algorithm has relatively more samples and can learn a better model through gradient descent. Another observation we can make from the result is the two deep learning method also perform relatively better on ML 1M dataset as compared to ML 100K, as deep neural network is able to learn a more complex model on ML 1M due to more rating data. From Fig. 6, we can observe the effect of varying embedding sizes on the precision metric. For the Movielens 100k dataset, we get the highest precision at an embedding size of 50 and for Movielens 1m, we get the best result at an embedding size of 100. For the Movielens 100k dataset, the drop becomes steeper once we increase the size after 50, whereas for Movielens 1m dataset, the changes in precision are more stable for different embedding sizes.

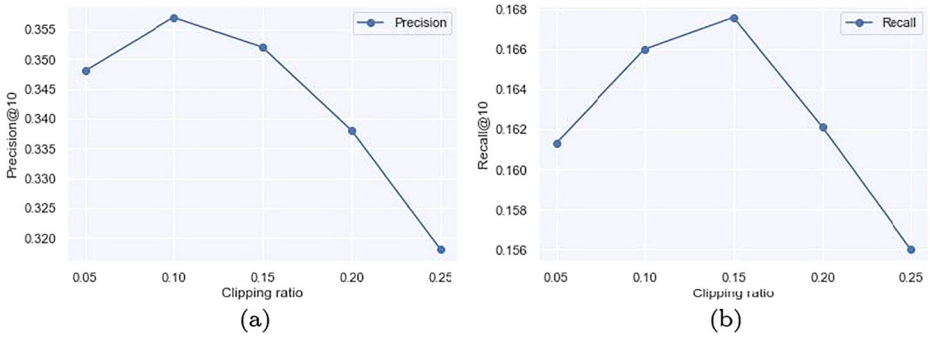

Fig. 4 Parameter Sensitivity by varying Clipping ratio

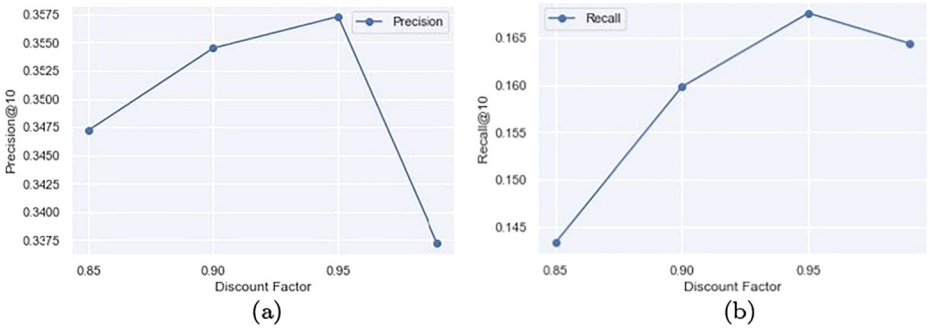

Fig. 5 Result of Varying Discount factor

We can see that for the Movielens 100k dataset, the optimum threshold is 5%, whereas, for the Movielens 1m dataset, the optimal threshold is 7% as depicted in Fig. 7. Specifically, precision decreases significantly for both datasets as the percent of cold users starts increasing. We achieve the highest precision at a clipping ratio of 0.9 and discount factor of 95%, depicted in Figs. 4 and 5, respectively.

For checking the statistical significance of the results, we used Wilcoxon-Signed test [45]. It is a non-parametric test and does not assume the normal distribution over the input like the student t-test. The null hypothesis is the two distributions are not statistically different. The test was performed using the results of the MDP algorithm against each of the baselines at a 5% significance level. We can observe that Tables 4 and 5 list the p-values obtained by the test, where p-values less than 0.05 indicate the rejection of the null hypothesis, implying there is a significant difference at a level of 5%. The p-values in the tables

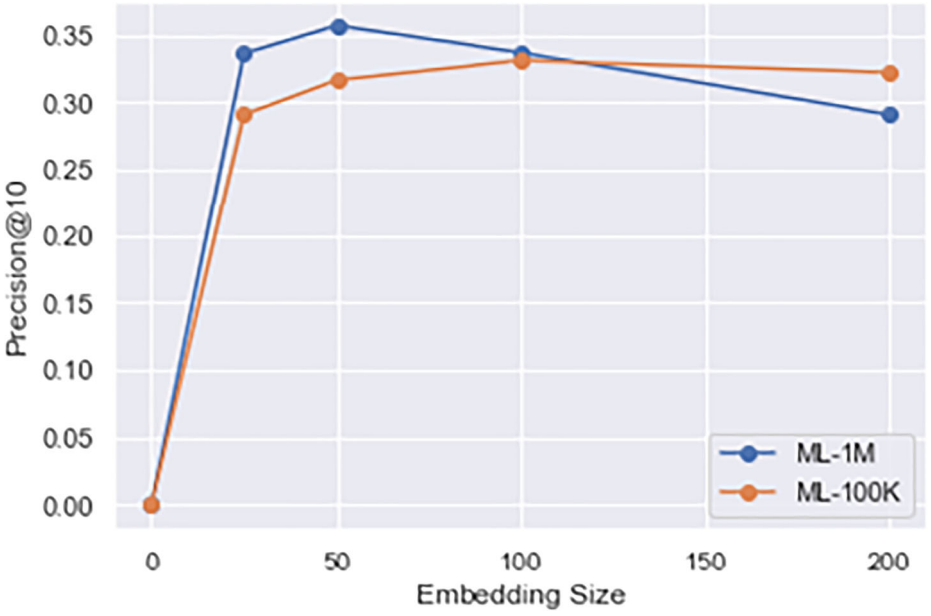

Fig. 6 Effect of different embedding sizes on Precision

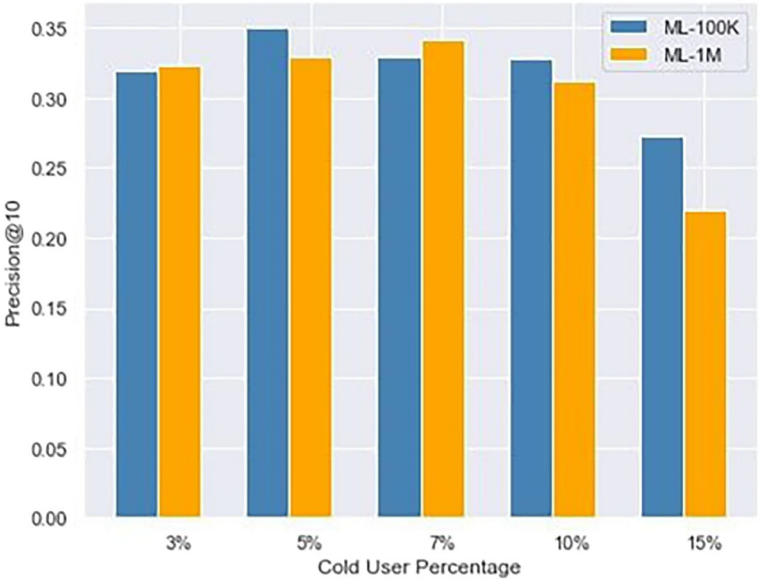

Fig. 7 Cold User Threshold

Table 4 Wilcoxon Signed P value against different baselines on Movielens 100k

MDP vs	P@10	R@10
Random	0.0008	0.0078
PopRank	0.0585	0.0100
LinUCB	0.0158	0.0089
CB	0.0093	0.0411
SVD	0.0002	0.0178
Deep MF	0.0067	0.0010
Neural CF	0.0312	0.0076

Table 5 Wilcoxon Signed P value against different baselines on Movielens 1M

MDP vs	P@10	R@10
Random	0.0014	0.0066
PopRank	0.0158	0.0074
LinUCB	0.0365	0.0289
CB	0.0027	0.0088
SVD	0.0721	0.0218
Deep MF	0.0011	0.0006
Neural CF	0.0043	0.0198

confirm that the improvement achieved by MDP based algorithm is statistically significant over the majority of the baseline methods(apart from P@10 metric on PopRank on ML-100K and SVD on ML-1M, respectively).

8 Conclusion and Future Work

In this paper, we used the Hybrid recommender system based on Reinforcement learning over the Movielens datasets. We combined the features of two popular recommender systems techniques to address the cold start issue. We modeled the recommender system in the Reinforcement Learning framework. We incorporated the Collaborative filtering technique in it by modeling the states of the MDP using Matrix factorization and using the neighborhood technique inside the MDP process for calculating the next item. We utilized a Content-based filtering approach further for the cold user set, thereby further improving the recommendation process. We used the Policy gradient approach for training our algorithm as they are more suited for large state spaces compared to the value-based approach. We performed the experiments on two Movielens datasets and outperformed different state-of-the-art baseline methods.

For future work, we are interested in extending our work to more domains in recommender systems such as e-commerce items and deploying multi-agent reinforcement learning based recommender system frameworks to model the recommender system process. Further research directions could be the techniques that can lead to including more diversity in the recommendation process. We can modify the action in our MDP architecture from the proposed solution to have more than one item recommended at a time, utilizing the correlation between the items. Further, with a multi-agent approach, we can consider multiple scenarios of the recommendation process, each modeled by a particular agent. We also think our method can be extended to a Deep Reinforcement Learning framework using different architecture for Actor-Critic networks.

Data Availability All the data required for this research work, i.e., Movielens 1M and Movielens 100k dataset, is available publicly on the Movielens Website. The data can be accessed from https://grouplens.org/datasets/movielens/.

Declarations

Conflict of Interests All authors declare that they have no conflicts of interest and this research received no funding.

References

1. Akerkar B, Sajja P (2010) Knowledge-based systems. MIT press Cambridge, 978-0763776473
2. Aljunid MF, Manjaiah DH (2020) An efficient deep learning approach for collaborative filtering recommender system procedia computer science. Third International Conference on Computing and Network Communications (CoCoNet'19) 171:829–836
3. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) A brief survey of deep reinforcement learning. IEEE Signal Proc Mag 2:26–38
4. Bhatti UA, Huang M, Wang H, Zhang Y, Mehmood A, Wu D (2018) Recommendation system for immunization coverage and monitoring. Human Vaccines & Immunotherapeutics, 165–171
5. Bhatti UA, Huang M, Wu D, Zhang Y, Mehmood A, Han H (2019) Recommendation system using feature extraction and pattern recognition in clinical care systems. Enterprise Information Systems, 329–351

6. Breese JS, Heckerman D, Kadie C (2013) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, pp. 43–52, UAI'98, Madison, Wisconsin
7. Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) Openai Gym. arXiv:1606.01540
8. Burke R (2002) Hybrid recommender systems: survey and experiments. User Modeling and User-adapted Interaction 12:331–370
9. Chen H (2021) A DQN-based Recommender System for Item-list Recommendation. In: IEEE International Conference on Big Data (Big Data), pp 5699–5702
10. Chen M, Beutel A, Covington P, Jain S, Belletti F, Chi Ed (2019) Top-K Off-Policy Correction for a REINFORCE Recommender System, Association for Computing Machinery, New York, NY, USA, 456–464
11. Chen X, Yao L, McAuley J, Zhou G, Wang X (2021) A survey of deep reinforcement learning in recommender systems: a systematic review and future directions. arXiv:2109.03540
12. Dulac-Arnold G, Evans R, Hasselt HV, Sunehag P, Lillicrap T, Hunt J, Mann T, Weber T, Degris T, Coppin B (2015) Reinforcement learning in large discrete action spaces. Corr, abs/1512.07679
13. He X, Li L, Zhang H, Nie L, Hu X, Chua TS (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web, pp 173–182
14. Hu Y, Da Q, Zeng A, Yu Y, Xu Y (2018) Reinforcement learning to rank in e-commerce search engine Formalization, analysis, and application. Corr, abs/1803.00710
15. Koren Y (2008) Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Association for computing machinery, New York, USA, pp. 426–434
16. Li L, Chu W, Langford J, Chapire RE (2010) A contextual-bandit approach to personalized news article recommendation. J Mach Learn Res, 661–670
17. Li W, Zhou X, Shimizu S, Xin M, Jiang J, Gao H, Jin Q (2019) Personalization recommendation algorithm based on trust correlation degree and matrix factorization. IEEE Access 7:45451–45459
18. Lin W, Zhang X, Qi L, Li W, Li S, Sheng VS, Nepal S (2021) Location-Aware Service recommendations with Privacy-Preservation in the internet of things. IEEE Trans Comput Social Syst 8:227–235
19. Liu F, Tang R, Li X, Zhang W, Ye Y, Chen H, Guo H, Zhang Y (2018) Deep reinforcement learning based recommenda- tion with explicit user-item interactions modeling, arXiv:1810.12027
20. Liu Y, Wang S, Khan MS, He J (2018) A novel deep hybrid recommender system based on auto-encoder with neural collaborative filtering. Big Data Mining and Analytics, 211–221
21. Marlin B (2003) Modeling User Rating Profiles for Collaborative Filtering. MIT Press. Cambridge, MA, USA
22. Mnih A, Salakhutdinov R (2008) Probabilistic matrix factorization. NIPS, 1257–1264
23. Mnih A, Salakhutdinov R (2008) Bayesian probabilistic matrix factorization using markov chain monte carlo. ICML, pp. 880–887
24. Pan F, Cai Q, Tang P, Zhuang F, He Q (2019) Policy Gradients for Contextual Recommendations. In: Association for Computing Machinery, New York, NY, USA, 1421–1431
25. Polat H, Du W (2005) SVD-based collaborative filtering with privacy. Association for Computing Machinery, 791–795, New York, NY, USA
26. Raffin A, Hill A, Gleave A, Kanervisto A, Ernestus M, Dormann N (2021) Stable-Baselines3 reliable reinforcement learning implementations. J Mach Learn Res 22:1–8
27. Salakhutdinov R, Mnih A, Hinton G (2007) Restricted Boltzmann machines for collaborative filtering. NIPS, 791–798
28. Sarwar B, Karypis G, Konstan J, Riedl J (2017) Item-based collaborative filtering recommendation algorithms. arXiv:1707.06347
29. Schulman J, Wolski P, Dhariwal P, Radford A, Klimo O (2017) Proximal policy optimization algorithms. arXiv:1707.06347
30. Sedhain S, Menon AK, Sanner S, Xie L (2015) Autorec: Autoencoders meet collaborative filtering. In: Proceedings of the 24th International Conference on World Wide Web, WWW '15, Companion, New York, NY. USA, 111–112
31. Shani G, Heckerman D, Brafman I (2005) An mdp-based recommender system. J Mach Learn Res 15324435:1265–1295
32. Singh M (2020) Scalability and sparsity issues in recommender datasets: a survey. Knowl Inf Syst, 1–43
33. Srivihok A, Sukonmanee P (2005) E-Commerce intelligent agent personalization travel support agent using q learning. In: Association for Computing Machinery, New York, NY USA
34. Sutton R (1998) Reinforcement learning: an introduction. vol. 1, no. 1., MIT press Cambridge
35. Sutton R, Singh S, McAllester D (2000) Comparing policy-gradient algorithms. IEEE Transactions on Systems Man, and Cybernetics

36. Taghipour N, Kardan A (2008) A hybrid web recommender system based on q-learning. In: Proceedings of the ACM Symposium on Applied Computing (SAC), Fortaleza, Ceara, Brazil, pp. 1164–1168, March, 16–20, 2008
37. Tao Y, Wang C, Yao L, Li W, Yu Y (2021) Item trend learning for sequential recommendation system using gated graph neural network. Neural Comput & Applic, 1–16
38. Van Hasselt H, Doron Y, Strub F, Hessel M, Sonnerat N, Modayil J (2018) Deep reinforcement learning and the deadly triad, arXiv:1812.02648
39. Van den Oord A, Dieleman S, Schrauwen B (2013) Deep content-based music recommendation. NIPS, 2643–2651
40. Vincent P, Larochelle H, Bengio Y, Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 1096–1103
41. Vozalis M, Margaritis K (2004) Collaborative filtering enhanced by Demographic Correleation
42. Vozalis M, Margaritis K (2006) On the enhancement of collaborative filtering by demographic data. Web Intelligence and Agent Systems 2:117–138
43. Wang H, Wang N, Yeung DY (2015) Collaborative deep learning for recommender systems. KDD, 1235–1244
44. Wei K, Huang J, Fu S (2007) A survey of e-commerce recommender systems. In: 2007 International conference on service systems and service management. pp 1–5
45. Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics Bulletin, 80–83
46. Wu Y, DuBois C, Zheng AX, Ester M (2016) Collaborative denoising auto-encoders for top-N recommender systems. In: Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, pp 153–162
47. Wu C, Rajeswaran A, Duan Y, Kumar V, Bayen AM, Kakade S, Mordatch I, Abbeel P (2018) Variance reduction for policy gradient with action-dependent factorized baselines, arXiv:1803.07246
48. Xue H, Dai X, Zhang J, Huang S, Chen J (2017) Deep matrix factorization models for recommender systems, IJCAI 3203–3209, 17, Melbourne, Australia
49. Zhang S, Yao L, Sun A, Tay Y (2019) Deep Learning Based Recommender System: A Survey and New Perspectives. ACM Comput. Surv., 52
50. Zhao X, Xia L, Zhang L, Ding Z, Yin D, Tang J (2018) Deep reinforcement learning for page-wise recommendations, abs/1801.00209
51. Zhao X, Zhang L, Ding Z, Xia L, Tang J, Yin D (2018) Recommendations with negative feedback via pairwise deep reinforcement learning. Corr2, abs/1802.06501
52. Zheng G, Zhang F, Zheng Z, Xiang Y, Nicholas J, Xie X, Li ZDRN (2018) A deep reinforcement learning framework for news recommendation. International World Wide Web Conferences Steering Committee 2:167–176
53. Zheng G, Zhang F, Zheng Z, Xiang Y, Yuan NJ, Xie X, Li ZDRN (2018) DRN: a deep reinforcement learning framework for news recommendation. WWW 2018, Lyon, France, April, 23-27, 167–176
54. Zou L, Xia L, Du P, Zhang Z, Bai T, Liu W, Nie JY, Yin D (2020) Pseudo Dyna-Q: a reinforcement learning framework for interactive recommendation. In: Association for computing machinery, New York, NY, USA, pp. 816–824
55. Zou L, Xia L, Gu Y, Zhao X, Liu W, Huang J, Yin D (2020) Neural interactive collaborative filtering. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp 749–758