Reinforcement learning usage in the development of recommender systems

Jelica Cincović, Dražen Drašković
University of Belgrade, School of Electrical Engineering, Belgrade, Serbia
jelica.cincovic@etf.rs, drazen.draskovic@etf.rs

***Abstract* — Recommender systems are used for predicting user preferences mostly for commercial purposes. As the need for recommender systems has increased significantly, due to the rapid lifestyle we all lead, machine learning has also found its wide application in the development and improvement of these systems. Reinforcement learning emerged as one of the newer machine learning techniques, which can make recommender systems more flexible using user interactions. This paper focuses on analyzing and comparing existing implementations of recommender systems based on reinforcement learning, and drawing final conclusions when combining these two techniques, as well as defining possible further upgrades.**

I. INTRODUCTION

Machine learning (ML) is a type of artificial intelligence (AI) that enables computers to learn from data and predict new output values. This process is performed with the help of machine learning algorithms, that are trained to extract essential data and make the required predictions. Digital assistants, robots, self-driving cars are just some of the things that rely on machine learning.

Machine learning methods can be divided into four categories:

- Supervised machine learning – uses labeled data as input to produce output values, which can be compared with correct predictions to improve the algorithm,
- Unsupervised machine learning – uses unlabeled data as input, while algorithm finds hidden structures from data,
- Semi-supervised machine learning – uses both labeled and unlabeled data to achieve the best results,
- Reinforcement machine learning – uses reward and error cues, based on which the environment should be directed towards an appropriate outcome.

This paper is focused on reinforcement learning, that uses interaction with the environment for the sake of progress. After each step, the environment receives a response in the form of a reward or punishment and based on it decides its next action aimed at maximizing the reward. Reinforcement learning is mostly used in robotics, games [1][2], and resource management. Another possible application of reinforcement learning is in the development of the recommender system, which is further discussed.

Recommender (recommendation) systems are used to predict user preferences. They are very important asset to companies, that can adjust their strategies and attract more users with recommenders help. The most used approaches when implementing recommender systems are collaborative filtering, content-based filtering, and hybrid systems, based on a combination of both.

Improvements in recommenders can be achieved with integration of reinforcement learning algorithms. Therefore, this paper provides an overview of current state-of-the-art solutions in this area, as well as their comparison, which is presented in the following chapters.

II. REINFORCEMENT LEARNING

Reinforcement learning setting consists of the agent and the environment. The agent is the learning algorithm itself, while the environment is the object over which the algorithm is executed. The environment provides the algorithm with state information, and the algorithm decides which action will be performed. The action performed in the environment will produce a new state of the environment, as well as the reward that the algorithm receives for the selected action. Based on the state and reward, the algorithm selects a new action. The strategy that decides what action to take based on the current state is also called policy. The described behavior is illustrated in Fig. 1.

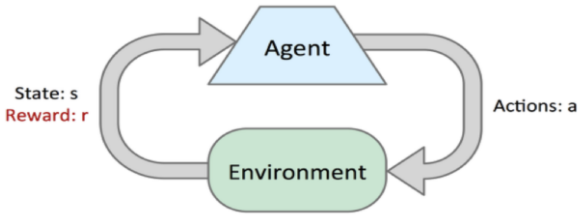

Figure 1. The agent-environment interaction

The main division of reinforcement learning algorithms is according to whether they are model-based or model-free.

The concept of model-based algorithms is to learn the whole simulation of the environment, and to know before time, which action needs to be taken in each of the possible states, in order to progress towards the solution, the fastest. This approach is not suitable for problems with many states and actions.

On the other hand, model-free algorithms dynamically, according to the principle of trial and error, learn the most optimal policy, and do not need the whole model.

The second parameter by which we can divide the algorithms is by the way the policy is updated, and the algorithms can use on-policy or off-policy learning.

On-policy learning is based on updating the policy, based on the experience gained from applying the current policy, while off-policy learning allows the use of multiple policies, and the search for an updated policy is based on experience gained from previous policies.

This chapter gives an overview of the most popular reinforcement learning algorithms, some of which will also be mentioned in later discussion: Q-learning, Deep Q-Network and Actor-Critics.

A. *Q-learning*

Q-learning [3] belongs to the group of off-policy model-free reinforcement learning algorithms. This algorithm tends to find a policy that maximizes the overall reward.

The basic steps of Q-learning are:

- Creating a q-table – initializing a matrix of state and actions, to zeros. This table will serve later to help the algorithm choose the next action,
- Making updates - when an agent (algorithm) chooses its action, it has two options. The first option is *exploiting*, when it selects the action from the q-table that has the highest value for the given state. Another option is *exploring*, when choosing random action for that state. It is important to have a good balance of exploiting and exploring for algorithm to find the best policy. After the action is completed q-table is updated. The basic q-learning update rule is:

$$Q[state, action] = Q[state, action] + lr * (reward + gamma * np.max(Q[new\_state, :]) - Q[state, action])$$

with, lr being learning rate and gamma being a discount factor. This formula shows that with the help of lr we influence how much we accept new values, while with gamma we influence the balance between the current and future reward.

Q-learning is a very popular algorithm that is quite easy to implement and which, in addition to the current reward, tries to include the future reward in its formula.

B. *Deep Q-Network (DQN)*

Due to its popularity, the Q-learning algorithm has given rise to many other algorithms based on it. One such algorithm is Deep Q-learning.

The main problem with the Q-learning algorithm is the impracticality in cases of many states and actions. Tables can take up a lot of memory and are inefficient. In such situations, Deep Q-learning can be switched to, and Q values can be calculated using neural networks with Θ parameters.

Deep Q-Network algorithm [4] was created in 2015, by DeepMind, with the idea to solve Atari games. In DQN algorithm, the next action is determined based on the output of the neural network (the action with the highest Q-value is selected), and not based on a q-table, and the neural network allows to work with an unlimited number of states. The difference between Q-learning and Deep Q-learning is shown in Fig. 2. In addition, DQN introduced the concept of *experience replay*.

Experience replay allows the algorithm to remember previous actions by being able to replay them. From time to time, a set of actions in the buffer is remembered, and that buffer is given to the neural network to take into consideration.

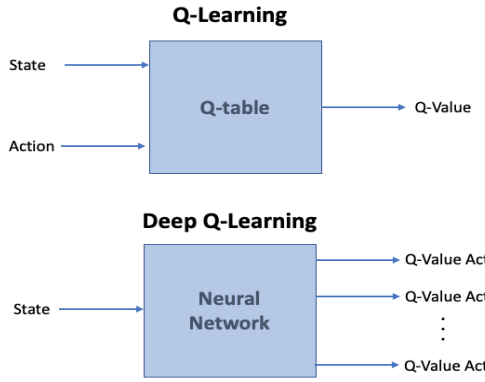

Figure 2. The difference between Q-learning and Deep Q-learning

C. *Actor-Critics*

So far, algorithms have been seen that use either the off-policy or on-policy method, but there have been attempts to combine these two ideas, and thus the Actor-Critic algorithm has emerged [5]. This algorithm draws the best from these two methods and overcomes their shortcomings.

Actor is based on the on-policy method, and for a given state, as an output it gives the best action to be performed. On the other hand, Critic is based on off-policy method. Critic is tasked with evaluating Actor's action. In time, both parts will improve in their work. In the Fig. 3 an architecture od Actor-Critic algorithm is given.

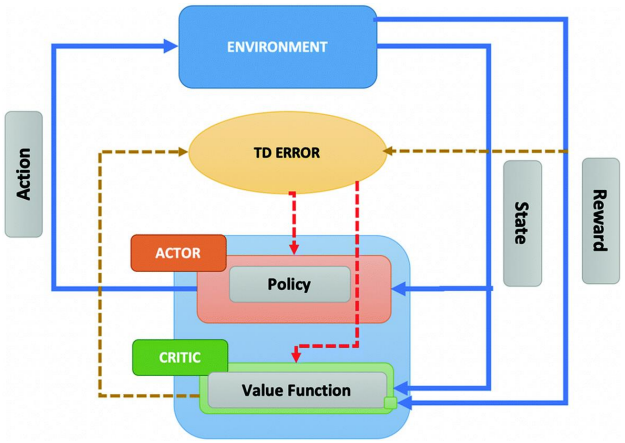

Figure 3. Actor-Critic architecture

The first extension of this algorithm is Advantage Actor-Critic (A2C), which is based on Critic learning not only Q values but Advantage values. The Advantage function indicates not only how good the action is, but also how much better the action is, compared to others. Another possible extension of this algorithm is the Asynchronous Advantage Actor-Critic (A3C). The main difference from the Advantage Actor-Critic is the asynchronization feature, which allows for more independent networks that can cover more space in less time. Those independent networks are trained in parallel, and they update the main (global) network, from time to time. A3C architecture is shown in Fig. 4.

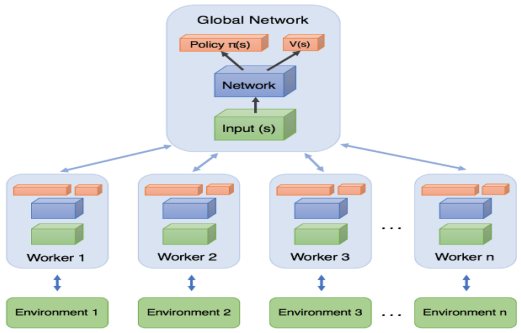

Figure 4. Asynchronous Advantage Actor-Critic architecture

III. Analysis of Reinforcement Learning Algorithms Used in Recommender Systems

During the analysis of the area of integration of reinforcement learning with recommendation systems, two groups of possible solutions were quite expectedly highlighted. Some implementations have opted for the model-free method and others for the model-based.

In the model-free methods, the most common implementations are those that use Deep Q-learning and Actor-Critic algorithms. The two solutions based on these methods are described below.

The first analyzed solution [6] is based on the DQN framework, for online news recommendation. There are 4 categories of features, which are entrances to the multilayer Deep Q-Network, that predicts the reward. The feature categories are:

- News features – category, topic, click counts etc.,
- User features – features of the news that user clicked on,
- User news features – the frequency of occurrence of a single news entity in the user's history,
- Context features - the state of the system at the time of requesting a particular news item.

User and Context features are represented as state features, and User news and News features as action features. When given to the network they produce value and advantage function, that together produce Q-function. This architecture is shown in Fig. 5.

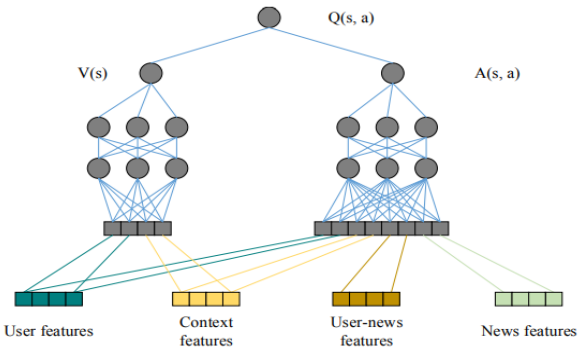

Figure 5. DQN network architecture

One way to compare this implementation with other state-of-the-art solutions, are the CTR (click through rate) parameter and normalized discounted cumulative gain (nDCG). The presented solution, according to these parameters, gave the best result, compared to other techniques when recommending news, such as Logistic Regression, Factorization Machines, or Wide and Deep methods.

The second analyzed solution [7] is based on Actor-Critic algorithm, used for list wise recommendations. The Actor is given the current state as input, and with the help of deep neural network it produces a list of weights. The function which generates the action that should be performed, on state of these weights and possible items, computes the scores for these items, and select the one with the highest score. The Critic assesses whether the action given by the Actor, satisfy for a particular state, producing Q value, with the help of DQN architecture. The Actor-Critic algorithm architecture implemented in this solution is shown in Fig. 6.

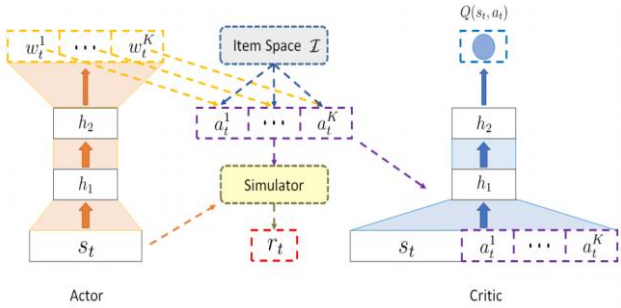

Figure 6. Actor-Critic algorithm architecture

The comparison of this solution with other possible implementations, such as Collaborative Filtering, Factorization Machines or Deep Neural Network, was done also by nDCG, and it gave much better results (similar to previous solution). When comparing Actor-Critic algorithm and DQN, Actor-Critic training time is much shorter than DQN.

The next solutions are focused on the model-based method, which seeks to find a complete model of user interaction with the environment.

The third and fourth analyzed solutions [8][9] are based on creating a model with the help of Generative Adversarial Networks (GANs). GANs [10] can generate new instances of data, without that data originating from the user, and they are created using neural networks. GANs consist of two neural networks. One neural network is a generator (G) and the other a discriminator (D). As the generator produces new data, the discriminator tries to evaluate them, and find out which ones are real and which ones are not, as shown in Fig. 7. These two networks compete, hence the name adversarial.

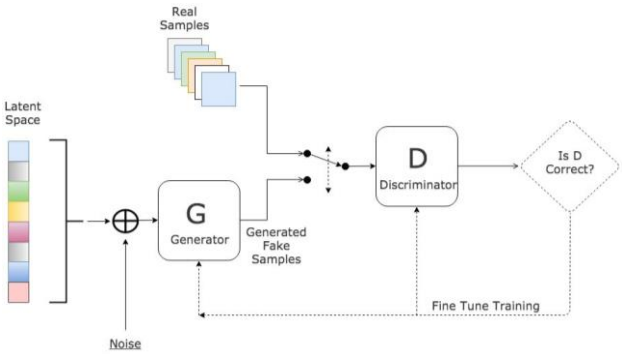

Figure 7. Generative Adversarial network architecture

TABLE I.
OVERVIEW OF THE COMPARED SOLUTIONS

Num.	Year	Institution	Recommendation type	Algorithm				Results
				Model-free	Model-based	On-policy	Off-policy	
1.	2018.	Pennsylvania State University and University Park	News	+			+	nDCG parameter gives much higher values compared to standard recommender techniques (Collaborative Filtering, Factorization Machines)
2.	2019.	Michigan State University	Diverse item space (tested on e-commerce dataset)	+			+	
3.	2019.	Stony Brook University, Tsinghua University and University of Virginia	Diverse item space (tested on CIKM Cup dataset)		+	+		precision@1 parameter gives much higher values compared to other recommender techniques (Wide and Deep, LSTM)
4.	2019.	Georgia Institute of Technology	Diverse item space (tested on ant financial news, movies, listening records)		+		+	

After the created model, which is a simulation of user actions, the most optimal policy can be established with the help of various algorithms. One solution used policy gradient methods REINFORCE, and other used cascading Q-networks.

These solutions are compared by different metrics, but the common is the precision @ k metric. Observing p @ 1 metric with Wide and Deep model, LSTM model, or GAN models but with different methods of learning policy, shows that observed solutions measure significant improvements.

Table I represents an overview of the compared solutions. All other considered solutions were based on similar implementations and were not further discussed.

IV. DISCUSSION

Recommendation systems have already taken off, and with reinforcement learning we can only improve them. This was also shown in the analyzed recommenders that used reinforcement learning, which were better in terms of CTR, nDCG and prediction accuracy compared to standard methods.

As there is no doubt that reinforcement learning algorithms are the direction in which to think when it comes to recommenders, it is important to look at and compare the two main approaches, model-based and model-free algorithms.

Model-based algorithms require large amounts of data representing user interaction with the environment. This can be quite problematic when viewed from the side of spent memory, but if this is not the case then the accuracy of such models is a great advantage. Of course, care must be taken to reduce the bias.

On the other hand, model-free algorithms are more often used, due to their ability to improve during the user's interaction with the environment. This is exactly one of the shortcomings, too frequent communication with the environment, but that is why such a solution is very suitable for large sets of states and actions.

V. CONCLUSION

Recommender systems will be more and more represented in all spheres of our lives. Companies depend on such systems, which bring big profits. That is why a lot of effort has been put into these systems, and reinforcement learning has brought visible improvement. Based on this paper, developers have an insight into current solutions in this area, that can serve them as a starting point for their further training.

Possible improvements can include: software for comparison of different reinforcement learning algorithms in recommenders (software would be tested over the same data sets, and would be compared by CTR, nDCG, total reward, and speed), and recommender literature system for students, based on they could more easily get the necessary knowledge.

Overall, the field of reinforcement learning in recommender systems will develop more and more and new solutions in this field can be expected.

REFERENCES

[1] Draskovic, D., Brzakovic, M., and Nikolic, B., “A comparison of machine leaming methods using a two-player board game,” IEEE EUROCON 2019 -18th International Conference on Smart Technologies, Novi Sad, Serbia, 2019, pp. 1-5.

[2] Xu, M., Shi, H., and Wang, Y., “Play games using Reinforcement Learning and Artificial Neural Networks with Experience Replay”, 17th IEEE/ACIS International Conference on Computer and Information Science, June 2018.

[3] Violante A., “Simple Reinforcement Learning: Q-learning”, Accessed: 06.02.2021., https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56

[4] Karagiannakos S., “Q Learning and Deep Q Networks”, Accessed: 07.02.2021., https://towardsdatascience.com/q-learning-and-deep-q-networks-436380e8396a

[5] Karagiannakos S., “The idea behind Actor-Critics and how A2C and A3C improve them”, Accessed: 06.02.2021., https://theaisummer.com/Actor_critics/

[6] Zheng, G., Zhang, F., Zheng, Z., Xiang, Y., Yuan, N. J., Xie, X., Li, Z., “DRN: A Deep Reinforcement Learning Framework for News Recommendation” In P.-A. Champin, F. L. Gandon, M. Lalmas and P. G. Ipeirotis (eds.), ACM, 2018.

[7] Zhao, X., Zhang, L., Xia, L., Ding, Z., Yin, D., Tang, J, “Deep Reinforcement Learning for List-wise Recommendations”, 2017.

[8] Bai, X., Guan, J., Wang, H., “A Model-Based Reinforcement Learning with Adversarial Training for Online Recommendation”, 2019.

[9] Chen, X., Li, S., Li, H., Jiang, S., Qi, Y., Song, L., “Neural Model-Based Reinforcement Learning for Recommendation”, 2018.

[10] “Generative Adversarial Networks”, Accessed: 08.02.2021., https://developers.google.com/machine-learning/gan