PyRecGym: A Reinforcement Learning Gym for Recommender Systems

Anonymous Author(s)

ABSTRACT

Recommender systems (RS) share many features and objectives with reinforcement learning (RL) systems. The former aim to maximise user satisfaction by recommending the right items to the right users at the right time, the latter maximise future rewards by selecting state-changing actions in some environment. The concept of an RL *gym* has become increasingly important when it comes to supporting the development of RL models. A gym provides a simulation environment in which to test and develop RL agents, providing a state model, actions, rewards/penalties etc. In this paper we describe and demonstrate the *PyRecGym* gym, which is specifically designed for the needs of recommender systems research, by supporting standard test datasets (MovieLens, Yelp etc.), common input types (text, numeric etc.), and thereby offering researchers a reproducible research environment to accelerate experimentation and development of RL in RS.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Modeling methodologies*; Model verification and validation.

KEYWORDS

Reinforcement learning, Recommender systems, Reinforcement learning gym

ACM Reference Format:

Anonymous Author(s). 2019. PyRecGym: A Reinforcement Learning Gym for Recommender Systems. In *Proceedings of ACM Conference on Recommender Systems (ACM RecSys'19).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Online Recommender Systems (RS) share many features with the reinforcement learning (RL) systems. Both aim to maximize some notion of reward (for example, end-user satisfaction, click-through rates, revenue etc.) by observing an environment (i.e., user, item, context features), selecting actions to take (i.e. recommending products, requesting feedback, generating an explanation etc.), and by updating the internal states (i.e. the recommendation model). Recently researchers have explored the use of RL ideas in a recommendation context to good effect [7, 13].

ACM RecSys'19, September 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

As the RL field has matured the concept of an RL *gym* has become important, to provide researchers and developers with access to a reproducible environment for experimenting with, evaluating, and developing RL agents and techniques. A gym defines the current state of the RL agents, offers available actions, and decides the rewards/penalties for each action. Usually gyms are developed with specific RL tasks in mind, so that they can be tailored for the particular needs of such tasks. For instance, the well-known OpenAI [3] gym focuses on the use of RL in gaming domains. Recently, as interest in RL has grown among RS researchers, the advantages of a gym to support RS research, have become apparent. However, a number of challenges remain. For a start, it is non-trivial to develop, train, and test an RL-based recommendation model in a gym that supports standard test datasets (Yelp, MovieLens etc), the variety of input types routinely used by RS (text, numeric ratings, preferences etc.), and the generally accepted best-practice evaluation metrics used by the RS community.

Typically researchers set up a bespoke RL environment using one or two datasets relevant to the task at hand ([7, 13]). This makes it difficult for others to reproduce or benchmark against their own work. One recent project, RecoGym [8], is a first step in this direction, offering an RL gym but is tailored to recommending products in an online advertising setting. While it is a step in the right direction, its dependency on synthetic data is a limitation that is difficult to overcome in more realistic RS settings. Moreover, it is not possible to benchmark RL-based models with other recommendation models.

The main contribution of this short paper is to introduce, and evaluate PyRecGym, an RL gym that is specifically designed for RS research. It offers a standardised setup process, the ability to work with existing RS datasets and input types, and comes fully loaded with a number of baseline agents, reward functions, and performance metrics. Our aim is to make it easier for RS researchers to experiment with RL approaches. In the next section, we review related work at the intersection of RL and RS before going on to describe the design of PyRecGym and its baseline functionality. We compare the PyRecGym with alternative RL-based RS gym(s) and in the Section 5, we finish by summarizing our contributions. PyRecGym is open-source and will be available on github [1].

2 RELATED WORK

Much research has been carried out to apply RL techniques to make better recommendation models that are appropriate for online recommendations [7, 11, 13]. Zhao et al. [11] provides an overview of deep RL approaches in search, recommendation and online advertising, highlighting how many recommendation methods employ RL settings to solve various challenges. Li et al. [7] attempts to solve the *explore-exploit* dilemma, while Wu et al. [10] and Chen et

[1] http://github.com/nnn

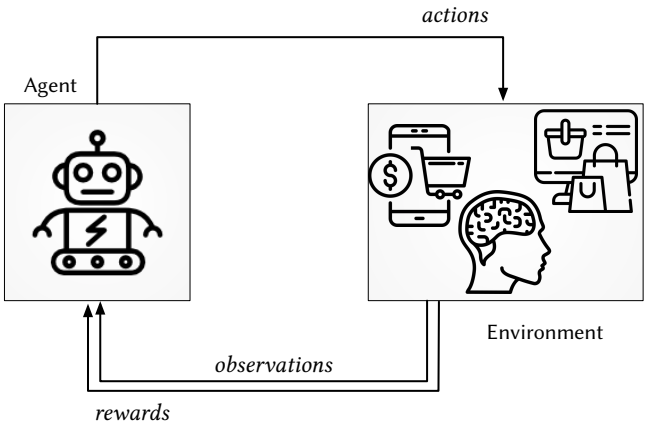

Figure 1: Reinforcement learning

al. [4] consider temporal dynamics while applying RL to RS problems. Zheng et al. [13] and Zhao et al. [12] aim to increase long term user engagement and Wang et al. [9] and Choi et al. [5] make page-wise recommendations by employing RL techniques. Many of these works set up their custom RL environment upon one or two chosen datasets. However, it is not easy for other researchers to replicate their environment and get access to their particular dataset in order to compare their recommendation model.

The simulation of the environment with which the RL agents interact, requires the set up of a "gym", which facilitates the execution of an RL algorithm, allowing the researcher to fully customise it to her preferences, defining the current state of the RL agents, showing available actions, and deciding the rewards/penalties for each action. A well-known RL gym library is developed by OpenAI [3], but is mainly developed for gaming, not for RS. A recent work, RecoGym [8], creates an RL gym for online advertisements using synthetic data, which can be far from the user behaviours captured in real RS datasets. Also, it is not possible to use such data for benchmarking RL-based models vs. other recommendation models.

Our proposed gym, PyRecGym, can take in any RS dataset and set up an appropriate RL gym. Also, it supports different kinds of input data types (e.g., textual, numeric) and offers a fully customizable environment, allowing the gym user to define the states, rewards and outputs according to the needs of the algorithm. Using PyRecGym, an RL-based recommendation model can be developed, trained, tested and compared against other types of recommendation models. Baseline RL algorithms such as multi-armed bandits are already part of PyRecGym, as can be seen section 3.4, as well as different kinds of performance metrics (e.g., accuracy, click through rate)

3 MODEL OVERVIEW

3.1 Preliminaries

The classical formalisation of RL contains *agent, environment, action, state, observation* and *reward*. As shown in Figure 1, the agent executes an action and the environment returns the reward and observation as a result of that action. The definition of the RL-concepts are as follows [1]:

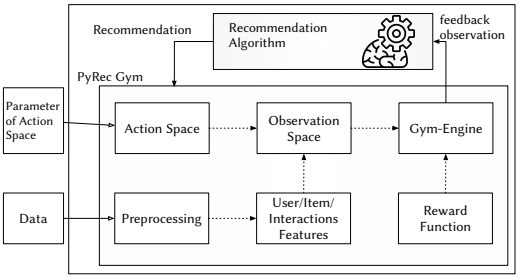

Figure 2: Design of the PyRecGym

Agent: The agent operates the actions based on the observations. For example, in the Mario game, Mario himself is the agent.

Environment: The environment is the world where the agent performs actions and where states and rewards are computed. For example, in the Mario game, the game itself (e.g., the enemies, the blocks and the clock) forms the environment.

Action: The actions are the set of all possible moves the agent can make at any given point in time. For example, in the game, Mario can jump, duck, move forward or backwards.

State: The state represents the 'concrete and current' situation of the agent in the environment. For example, in the Mario game, the location of Mario, the current score, and the location of the enemies on the screen form the state.

Observation: The observation indicates what the agent observes from the environment. The terms state and observation are highly related, where the state can be partially or fully observable.

Reward: The reward is the feedback which measures the success or the failure of the actions. For example, in the Mario game, the points are the reward.

Table 1 gives examples of how RL concepts are used for the Mario game [2] and how they are used in our proposed PyRecGym. There is a state-space which describes the setting and a finite set of possible actions that an agent can take in order to move through the state space. Taking an action results in a state transition, after which the agent receives some observations and a reward. The overall goal is to learn a policy, which is a way to make a decision of what action to take at any particular state in order to maximise the expected reward over some time horizon.

3.2 System Design

RS datasets usually contain three types of data: (i) the user profile, which contains detailed information about users (e.g., gender, age); (ii) the item profile, which contains information about items (e.g., title, description); and (iii) the interaction data, which records users' explicit (e.g., ratings, likes) or implicit (e.g., click, read) interactions with items, along with the context of the interaction (e.g., time, location).In PyRecGym we enable the extraction of all types of information from the given dataset and we use them as input for simulating the RL environment based on user behaviours captured inside.

The mapping of the RL-concepts to an RL-based RS is exemplified in Table 1. In PyRecGym, the recommendation algorithm/model is the agent, the user profiles and context of the recommendation

Table 1: Examples for the classical formalization of RL

RL-concepts	Mario Game	PyRecGym
Agent	Mario	Recommendation algorithm/model
Environment	The enemies, the blocks and the clock	Combination of User/item profiles and contextual information
Action	Jump, duck and move forward	Recommend the candidate item/not
State	The location of Mario and the current score	User/item profiles and contextual information
Observation	The location of Mario and the current score	User/item profiles and contextual information
Reward	Score	Feedback (e.g., like/dislike)

requests are the states of the RL environment, the decision of recommending/not the candidate item or the rating assigned to the candidate item is the action, and users' responses are the rewards.

Following the same setup as the OpenAI Gym, PyRecGym gym does not make any assumptions about the agent and provides a standardized environment. Figure 2 presents the general structure of PyRecGym and its main components:

- *Preprocessing*: receives as an input the dataset on which the RL algorithm will run and preprocesses the data in the proper format to be used for feature extraction
- *Feature extraction*: receives the pre-processed data and extracts the user/item/interaction features required by the RL algorithm.
- *Action Space*: receives the parameters of the action space and sets the model's actions , i.e. binary classification, regression.
- *Observation space*: sets the available states of the model as a combination of the user, item and interaction features together with any available contextual information.
- *Reward function*: this is a user-set function for the rewards or penalties that the engine will give to the agent for its actions.
- *Gym Engine*: this is the main component that plays the role of the environment that interacts with the RL algorithm (agent). The engine gets the current state, actions and observations, receives the result as the recommendation and sends the feedback to the algorithm according to the reward function. The engine also performs the evaluation of the RL algorithm, using metrics such as accuracy or click-through-rate.

The PyRecGym overall has three main functions:

Initialization: initializes the feature vectors for all users from the input data, sets up the action space and the observation space and initializes with the initial state of the system for all users.

Reset: randomly selects a user for the next episode, subsets the user related states and returns the initial state to the RL agent.

Step: reacts to the RL agent's actions by calculating rewards and returns the next state. As is the case for the OpenAI Gym, this function returns four objects [3]: (i) *Observation*: Provides the observation of the environment for the current state. (ii) *Reward*: Provides the reward information calculated by the reward function as a result of the executed action. For example, if the reward function calculates the CTR, each like/usage of the user returns a positive reward. (iii) *Done*: Indicates if the sequence to be executed is finished/not. In PyRecGym, the Done value is set to true when all available observations (states) of the current user have been processed. (iv)*Info*: Provides the logging information showing the last observation and total number of steps executed until that point.

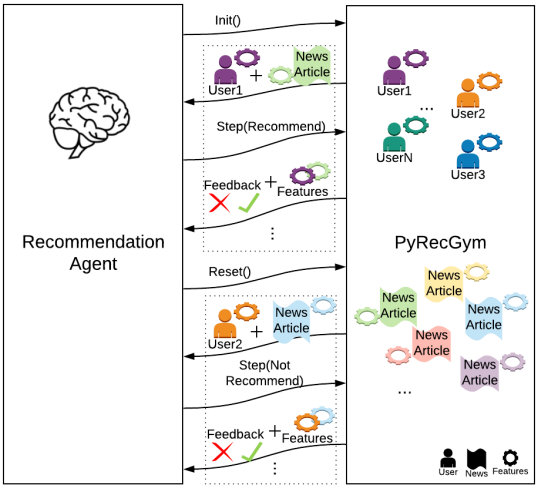

Figure 3: Recommendation agent

An important feature of PyRecGym, is that it can ingest existing RS datasets and use them to play out sequences of interactions between users and the RL system. The dataset is divided into initialisation data and interaction data. The initialisation data is used to set up the environment, by initialising a set of observations in the form of user, item and user-item feature vectors. Each feature vector encapsulates the information contained in the dataset about users and items, such as the set of past user-item interactions. The user profile and the states of the system are represented by a joined feature vector of the user/item features. A generalised data pipeline is provided to convert any dataset into the feature vector representation, and this pipeline may be customised, depending on the type of input data available and the RL algorithm to be applied. The interaction data are used as the RL action. The gym executes a reward and observation loop, to compute user responses to actions and the consequent observations and rewards. For instance, the action of recommending an item to a user results in a positive click, if the user-item pair exists within the interaction data.

3.3 Baselines included in PyRecGym

When testing new algorithms it's always useful to have some baselines to compare against. In this respect, PyRecGym already includes functionality for running many baseline RL algorithms such as:

Random recommendation: Recommends items randomly, without utilizing the RL environment or learning from the interactions.

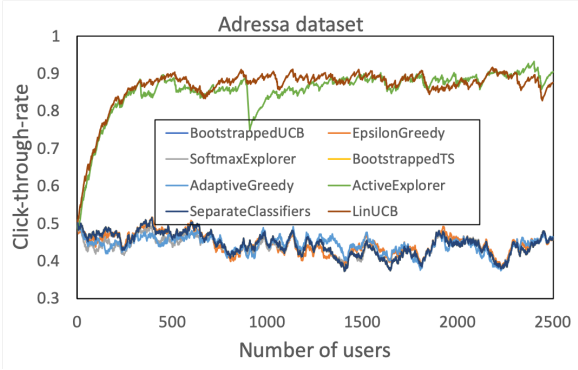

Figure 4: Training results on Adressa dataset

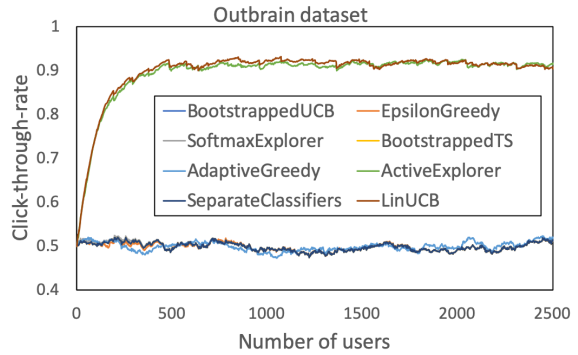

Figure 5: Training results on Outbrain dataset

Multi-arm bandit based recommendation: Inspired by Cortes et al. [6] PyRecGym includes as baselines contextual multi-armed bandits using the contextualbandits python package[2]

The process for including a new algorithm into the system is designed to be very simple for the end user. To add a new RL algorithm, the example baseline algorithm class should be modified with the following changes: (i) the reward function should be updated according to the needs of the algorithm, and (ii) the RL algorithm to be tested should be modified to interact with the gym using the three functions (init, reset and step).

4 EVALUATION

Comparison against state of the art. Testing RL-based RS algorithms requires an environment to be setup to interact with the RL model. The few past attempts for introducing a gym for RS have proven to be difficult to generalise to new recommender system settings. For example, Table 2 compares PyRecGym with its closest peer in the RS space, RecoGym[8].

Both the RecoGym and the PyRecGym target RS but RecoGym focuses specifically on computational advertising, while PyRecGym is designed to be more generic so that it can accommodate a greater variety of common recommender systems tasks (e.g. recommending movies, news using implicit/explicit ratings). Moreover, while RecoGym utilizes simulated data, PyRecGym uses real-world datasets (e.g., Outbrain, MovieLense). As a result, PyRecGym makes it easier to compare the RL-based RS algorithms with the state-of-the-art

[2]https://github.com/david-cortes/contextualbandits

Table 2: RecoGym vs. PyRecGym

	RecoGym	PyRecGym
Customized for RS	✓	✓
Generalized recommendation task	X	✓
Simulated dataset	✓	X
External real/benchmark dataset	X	✓
Support different types of input data and features	X	✓

RS algorithms, which may or may not be based on RL. PyRecGym is able to support many input data types (i.e., text, integer/float, category, location, timestamp data), as well as various kinds of features (i.e., user, item, user-item, context features). The RecoGym supports click-through rate (CTR) as the reward function, but the users cannot define other reward functions according to their needs.However, PyRecGym provides this kind of flexibility to its users.

Testing Example. Figure 3 presents the sequence of execution of the phases of PyRecGym, namely the init(), where the initialization of the components takes place, the sequential steps for each one of the users of the dataset and the reset() phase inbetween the users. After calling the init() function, PyRecGym contains the information of the input features (i.e., user, item, interactions and context) and the actions (i.e., recommend an article). During training, the RL algorithm (the agent) interacts with the environment by suggesting a news article to the target user and by getting feedback about the clicks of that user (i.e. reward) and the updates of the states.

As proof of concept, we use two example news datasets, namely Adressa and Outbrain as the input datasets. The datasets contain information on the users, the items (news articles) and their interactions (i.e., which articles the users have clicked/read). We used the CTR as the reward function, i.e., the overall aim of the algorithm is to maximize the number of clicks of the users in the long term. Figure 4 and 5 show the click-through-rate (CTR) performance of the training of the baseline algorithms on these two datasets.

5 CONCLUSION

RL is an exciting new direction for RS research and it has already been embraced by a number of early-adopters in recent work [3, 8]. In order to further accelerate this type of work, in this short paper, we follow in the footsteps of other communities (e.g. computer gaming) by providing a bespoke toolkit or gym, called PyRecGym, to support the development of recommender systems agents using RL techniques. The PyRecGym accommodates standard benchmark datasets (e.g., MovieLense, Yelp), works with different input data types (e.g., textual, numeric), and comes pre-loaded with a number of baseline agents with which to get started.

REFERENCES

[1] 2019. A Beginner's Guide to Deep Reinforcement Learning. https://skymind.ai/wiki/deep-reinforcement-learning
[2] 2019. Reinforcement Learning (Part 1) The Mario Bros Example. https://cai.tools.sap/blog/the-future-with-reinforcement-learning-part-1/
[3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. *CoRR* abs/1606.01540 (2016). arXiv:1606.01540
[4] 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 (KDD '18)*. ACM, 1187–1196.

[5] Sungwoon Choi, Heonseok Ha, Uiwon Hwang, Chanju Kim, Jung-Woo Ha, and Sungroh Yoon. 2018. Reinforcement Learning based Recommender System using Biclustering Technique. *CoRR* abs/1801.05532 (2018). arXiv:1801.05532

[6] David Cortes. 2018. Adapting multi-armed bandits policies to contextual bandits scenarios. *CoRR* abs/1811.04383 (2018). arXiv:1811.04383

[7] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*. ACM, 661–670.

[8] David Rohde, Stephen Bonner, Travis Dunlop, Flavian Vasile, and Alexandros Karatzoglou. 2018. RecoGym: A Reinforcement Learning Environment for the problem of Product Recommendation in Online Advertising. *CoRR* abs/1808.00720 (2018). arXiv:1808.00720

[9] Yingfei Wang, Hua Ouyang, Chu Wang, Jianhui Chen, Tsvetan Asamov, and Yi Chang. 2017. Efficient Ordered Combinatorial Semi-Bandits for Whole-Page Recommendation.

[10] Qingyun Wu, Naveen Iyer, and Hongning Wang. 2018. Learning Contextual Bandits in a Non-stationary Environment. *SIGIR âĂŹ18 The 41st International ACM SIGIR Conference on Research Development in Information Retrieval* (Jul 2018).

[11] Xiangyu Zhao, Long Xia, Jiliang Tang, and Dawei Yin. 2018. Reinforcement Learning for Online Information Seeking. *CoRR* abs/1812.07127 (2018). arXiv:1812.07127

[12] Xiangyu Zhao, Long Xia, Yihong Zhao, Dawei Yin, and Jiliang Tang. 2019. Model-Based Reinforcement Learning for Whole-Chain Recommendations. arXiv:cs.IR/1902.03987

[13] 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 on World Wide Web*. International World Wide Web Conferences Steering Committee, 167–176.