Reinforcement learning for long-term reward optimization in recommender systems

Anton Dorozhko
Departament of Mechanics and Mathematics
Novosibirsk State University
Novosibirsk, Russia
a.dorozhko@g.nsu.ru

Evgeniy Pavlovskiy
Departament of Mechanics and Mathematics
Novosibirsk State University
Novosibirsk, Russia
e.pavlovskiy@g.nsu.ru

***Abstract*—Recommender systems help users to orient in the vast space of goods, services, and events. A user interacts with recommender engine in a sequence of exchanges of recommendations and user feedback. The idea that previous interaction influence the later ones and the importance of the sequence of interactions can be modeled using Markov decision processes and solved by reinforcement learning. Several recent articles applying reinforcement learning to recommender systems has proved the viability of this direction. But it is still difficult to compare different approaches. We propose an environment with a unified interface that will permit to compare different modelization of recommender process and different algorithms on the same underlying sequential data. We also performed the extensive parameter study for deep deterministic policy gradient methods on the well-known MovieLens dataset.**

***Index Terms*—recommender systems, reinforcement learning, long-term value, deep reinforcement learning (DRL), DDPG**

I. Introduction

Usually, the recommendation process is viewed as a static, in other words, we train a model on some log history dataset and then serve it for some period of time without modifications. In order to update the model to account for users and items evolution in time, we need to retrain it. In that case, we do not model the influence of the interaction between user and recommender system (RS) and optimize only for a short-term reward (Compare: will a user like this product? vs will our RS provide agreeable user experience and increase overall usage of the service by a user?).

By casting the problem of recommendations into Reinforcement Learning (RL) framework we hope to achieve several important benefits. Firstly, we do not discard change in users preference w.r.t. time. Secondly, we can optimize not only the immediate reward/feedback of the users to the item but also the longterm contributions that our recommendation policy can achieve following one sequence of recommendations or another.

The paper is organized as follows. In Section II we provide a background for Markov decision processes and reinforcement learning methods to solve them. Related work is described in Section III. Proposed modelization and algorithm are presented in Section IV. Section V describes the testing procedure and the experimental results. Conclusion and final remarks are given in Section VI.

II. Background

We are interested in the optimization of the interaction between an environment (users with their preferences for recommended items) and an agent (recommender system).

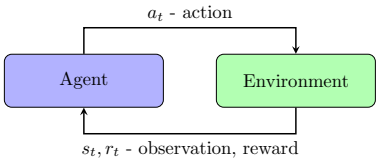

Fig. 1. Agent-environment interaction

A. Markov decision process

The underlying mathematical formalization of the environment is Markov decision process (MDP). MDP is a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$.

- State space $\mathcal{S}$ - representation of the observation.
- Action space $\mathcal{A}$ - what kind of actions can an agent choose
- Transition function $\mathcal{P}$ - represents the transition probability between current state s_t and the next state s_{t+1} given an action a_t
- Reward function $\mathcal{R}$ - outputs a scalar value on each transition
- Discount factor γ - defines the length of the sequence of actions for with we consider the reward.

B. Reinforcement learning

The goal of agent is to find optimal policy $\pi : \mathcal{S} \to \mathcal{A}$ that will maximaze cumulative discounted return:

$$G = \sum_{t}^{T} \gamma^t r_t \tag{1}$$

There are two approaches to find a good policy.

- Q-learning - estimate action-value function $Q^\pi(s, a) = \mathbb{E}[G|s_0 = s, a_0 = a]$ and take a policy $\pi(s) = arg\max_a Q(s, a)$
- Policy optimization (policy search) - take a parametrized function as policy $\pi_\theta(s) = f(s; \theta)$ and maximaze total expected reward by these parameters.

In many applications, we do not know the model of the environment (the transition and the reward functions). In that case, we can either estimate $\mathcal{P}$ and $\mathcal{T}$ and use a model-based approach, or we can use model free algorithms.

III. Related work

Various techniques, such as content-based collaborative filtering [1], matrix factorization methods [2], [3], logistic regression, factorization machines [4] models recommendation as static task or use time only as a feature.

Contextual bandits [5] provide good performance, but the contexts are supposed to be i.i.d and we want to model the influence of recommendations on user state.

The first line of works in applying RL to recommender systems is to study the performance on generated or available datasets. [6] proposed a variant of synthesized environment that modeled user and item embeddings as a vector with normally distributed components with varied level of correlation. They combined the flow of 'organic' views made by users independently of RS and 'bandit' views - the feedback on the items proposed by RS in one interface. [7] used DDPG with additional user-item representation module and evaluated on MovieLens, Yahoo music, Jester datasets. [8] tried to optimize diversity of recommendations proposed by RL algorithm but evaluated on a subset of MovieLens dataset.

Another approach is to use proprietary data and model specific sequential behavior according to the domain knowledge about the recommendation process. In the articles from JD.com [9] they used RL to jointly generate a set of recommendation and a specific order to display them. The agent has Actor-Critic architecture and model a state by feeding the embeddings of previously clicked items into the RNN. Then policy network outputs a vector that can be interpreted as a concatenated preferences for each position in the 2D display. Products are chosen as argmax of the scalar product of item embedding and preferences vector. Another article [10] optimize in one whole-chain approach different scenarios of user behavior: entrance page, item detail pages. They used a multi-agent reinforcement learning (MARL) approach to jointly optimize where each agent was responsible for one particular scenario. Alibaba Group research team [11] proposed a model-based formalism as search session Markov decision process with page histories, conversion and abandon states, where they estimate transition the transition function using function approximation. [12] used RL to model news recommendations. DQN they use continuous state feature representation and continuous action feature representation. As a reward, they used not only click/noclick but also how frequent user returns to the application. [13] used RL agent to traverse some tree structure over items to provide recommendations.

Facebook has developed recently a framework for RL called Horizon [14] and make recommendations of push messages.

A recent article on the difficulty of baseline evaluation [15] brought to light the problem of the quality of benchmarks and baseline finetuning. They were able to outperform the recent algorithms on the MovieLens 10M and Netflix prize datasets using older methods but with better hyperparameters and some modifications. Their research confirms the difficulty of the proper evaluation of recommender systems.

IV. Proposed method

In this article, we will consider simple MDP formalism and an adaptation of the DDPG algorithm. We will propose one possible way to make an OpenAI gym environment for recommender process.

A. MDP

Following [16] it is possible to model a state space as a list of N last interacted items and an action space as a list of K items. In our experiments we will consider $K = 1$. Each item is represented by an embedding of product in some space or a feature vector because working with indices of items and users don't allow a policy to capture useful patterns efficiently.

More formally,

- **State space** $\mathcal{S}$: A state $s_t = \{s_t^1, \cdots, s_t^N\} \in \mathcal{S}$ is defined as the browsing history of a user, i.e., previous N items that a user liked before time t.
- **Action space** $\mathcal{A}$: An action $a_t = \{a_t^1, \cdots, a_t^K\} \in \mathcal{A}$
- **Reward** $\mathcal{R}$. The agent receives immediate reward $r(s_t, a_t)$ according to the user's feedback on each action in the list and a chosen aggregation rule, i.e. simple sum of rewards or weighted by position.
- **Transition probability** $\mathcal{P}$: Transition probability $p(s_{t+1}|s_t, a_t)$ defines the probability of state transition from s_t to s_{t+1} when agent takes action a_t. If user skips all the recommended items, then the next state $s_{t+1} = s_t$; while if the user clicks/orders part of items, then the next state s_{t+1} updates.
- **Discount factor** γ: $\gamma \in [0, 1]$ defines the discount factor when we measure the present value of future reward.

B. DDPG Agent

Deep Deterministic Policy Gradient is an off-policy algorithm [17] that scales well with huge action spaces. There are two parts: actor that model deterministic policy $a(s)$ and critic that approximate action-value function $Q(s, a)$.

Actor network models state-specific scoring function $f_{\theta^\pi} : s_t \rightarrow w_t$ Then we recommend K items without repetition with maximal

$$score_i = w_t^k i_i^T \tag{2}$$

Critic takes state S_t and action a_t and computes $Q(S_t, a_t)$.

Optimal policy should satisfy:

$$a^*(s) = arg\max_a Q^*(s, a) \tag{3}$$

Using that fact we can avoid computing max over action-values of each item as in DQN, instead we will just use $Q(s, a(s))$.

Critic minimizes mean-squared Bellman error (the difference between bootstrapped action-value and current action-value)

$$\mathcal{L} = \mathbb{E}\left[(Q(s, a) - (r + \gamma Q_{target}(s', a_{target}(s'))^2\right] \tag{4}$$

where target subscript defines the function with the same structure but different weights, targets are updated softly or after some number of iterations M.

Actor's weights are also updated with gradient descent to maximaze:

$$\max_{\theta} \mathbb{E}[Q(s, a_\theta(s))] \tag{5}$$

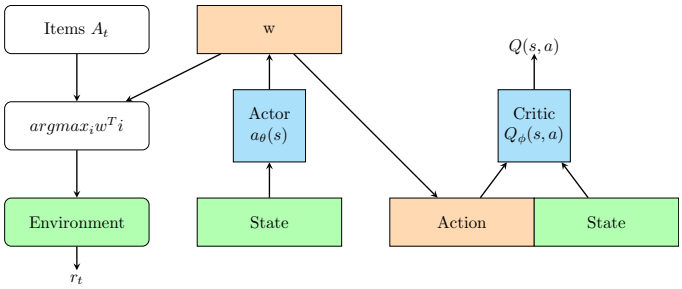

Fig. 2. DDPG

Algorithm 1 Deep Deterministic Policy Gradient

1: Initialize replay memory $\mathcal{D}$ to capacity N
2: Initialize action-value function Q with random weights
3: Set target parameters $\theta_{target} \leftarrow \theta$, $\phi_{target} \leftarrow \phi$
4: Initialise sequence s_1, $A_1 = a_1, ..., a_{N_1}$
5: **for** $t = 1, T$ **do**
6: Observe state s
7: Actor returns scoring function $a_t = a_\theta(s_t)$, select items as $argmax_i a_t^T i$
8: Observe reward for chosen items r_t, state s_{t+1} and possible items A_{t+1}
9: Store transition $(s_t, a_t, r_t, s_{t+1}, d)$ in $\mathcal{D}$
10: Sample random minibatch B of transitions from $\mathcal{D}$:

$$(s_j, a_j, r_j, s_{j+1}, d_j)$$

11: Set target
$y_j = r_j + \gamma(1 - d_j) Q_{\phi_{target}}(s_{j+1}, a_{target}(s_{j+1}))$
12: Update Q-function by one step of gradient DESCENT:

$$\nabla_\phi \frac{1}{|B|} \sum_j (y_j - Q_\phi(s_j, a_j))^2$$

13: Update policy by one step of gradient ASCENT:

$$\nabla_\theta \frac{1}{|B|} \sum_j Q_\phi(s_j, a_\theta(s_j))$$

14: update target networks

$$\theta_{target} = \rho\theta_{target} + (1 - \rho)\theta$$

$$\phi_{target} = \rho\phi_{target} + (1 - \rho)\phi$$

15: **end for**

C. Environment

As was mentioned by a recent study of the evaluation of the baselines [15] for MovieLens and Netflix Prize datasets it is a really difficult task. Better evaluations can be obtained through collaborative efforts that can be organized as a competition or clearly defined benchmarks. Another good reason for creating OpenAI Gym environment is the ability to run already implemented RL algorithms with small changes implemented in environment wrapper.

To construct such environment it is better to use OpenAI gym [18] library that proposes a convenient simple interface. Then it can be easily shared, thus facilitating reproducibility. Two main methods reset and step are introduced in Fig. 3. The episode is an interaction with one user, it starts by reset method. The episode is considered finished when the step method return $done = True$. If for a particular dataset we have ground truth recommended actions and we want to learn from them we can provide them in $info$ variable.

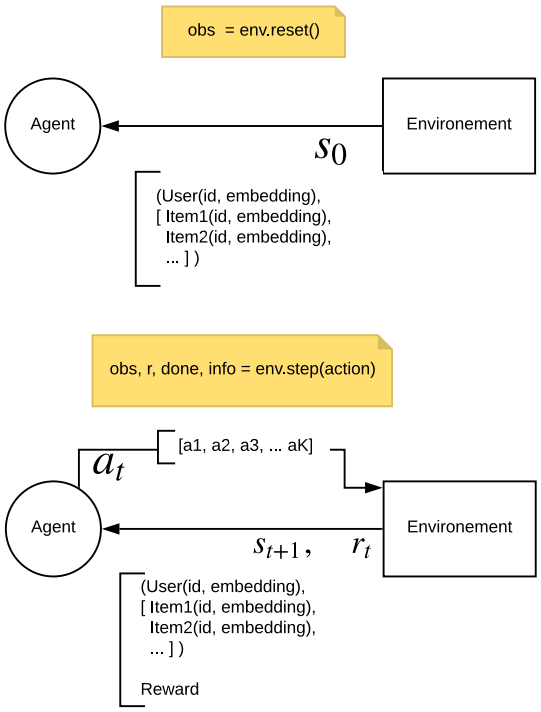

Fig. 3. Interaction loop with environment

To obtain different types of states and actions we can use wrappers with the same interface that will preprocess the initial states and actions. It is also useful to add another method that will return the action taken by the user if we want to learn not only from a signal of item ratings but also the sequences of real user decisions.

D. Baselines

- **Random Agent** - sample randomly K items from A_t
- **SVD** - SVD++ matrix factorization [19]
- **Popularity Agent** - recommend items with highest mean rank so far, or highest number of positive ratings

- **LinUCB** - linear contextual bandits [5]

V. Experimental results

A. MovieLens

One straightforward approach to evaluation is an offline evaluation of MovieLens [20].

TABLE I
DATASET STATISTICS

	ml-100k	ml-1m
users	943	6040
items	1682	3952
ratings	100000	1000209

To perform offline evaluation we will ask the algorithm to rerank known items from user sessions. The movies have ratings from 1 to 5. We will consider movies with ratings 4 and 5 as relevant to compute our metrics. Rewards are normalized to $[-1, 1]$.

We can split interactions into sessions by time or just by a number of interactions. DDPG algorithm will perform poorly if we only use categorical genres as representations of items. To provide a kind of structured embeddings we computed SVD with size 40 on train ratings.

Algorithm 2 Offline Evaluation Algorithm

```
n state window size
R reward function
Observe set 𝓘 from log
for t = 1, T do
    Observe current state s_t = {i_1, ..., i_n}
    Execute action w_t = a_θ(s_t)
    Choose i_t with highest score (2)
    Get reward r_t = R(s_t, a_t) from log
    Update state s_{t+1} = P(s_t, a_t, r_t)
    Remove i_t from 𝓘
end for
```

We compute the following metrics: precision@k, NDCG@k.

$$\text{precision} = \frac{\text{\# of receommended in first k in } \mathcal{I}}{\text{\# of recommended}} \tag{6}$$

$$\text{DCG@k} = \sum_{i=1}^{k} \frac{r_i}{log(1+i)} \tag{7}$$

$$\text{nDCG@k} = \frac{DCG@k}{IDCG@k} \tag{8}$$

where $IDCG@k$ is Ideal Discounted Cumulative gain, where recommended items sorted by their relevance. At each time step we recommend $K = 1$ items, metrics @k are computed for all recommendations during one session. sXX gives the size of the session in items.

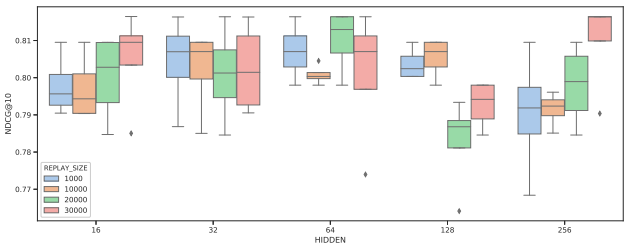

Fig. 4. Parameter study of DDPG algorithm on MovieLens100k

To understand the behavior of DDPG algorithm with various parameters with a varied number of hidden nodes in actor and critic network, experience replay buffer size and batch size.

The best set of parameters was identified by NDCG@10 metric (see Fig. 4): actor and critic are MLPs with 64 hidden nodes and tanh activations. They were optimized by Adam optimizer with learning rate 10^{-3} and batch size 64. Replay buffer size is 20000. Soft target update rate $\rho = 0.01$.

TABLE II
METRICS FOR DIFFERENT SESSION SIZE FOR MOVIELENS100K

	ml-100k s20		ml-100k s30	
	NDCG@10	***Precision@10***	***NDCG@10***	***Precision@10***
Random	0.78	0.56	0.76	0.56
Popularity	**0.86**	**0.68**	**0.86**	**0.69**
SVD	0.80	0.60	0.77	0.57
LinUCB	0.85	0.65	0.84	0.67
DDPG	0.83	0.63	0.81	0.63

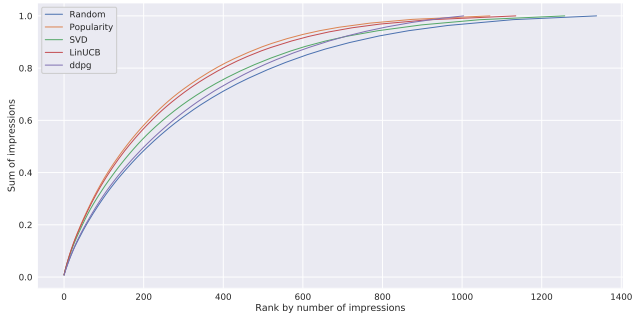

Fig. 5. Distribution of impressions over items: lower curve means that algorithm served more different items

As we can see results of RL algorithms are quite comparable with LinUCB and Popularity but provides slightly better variety as their CDF curve are lower Fig. 5, 6.

VI. Conclusion

In this project, we have studied the applications of reinforcement learning algorithms to the task of recommendations.

TABLE III
METRICS FOR DIFFERENT SESSION SIZE FOR MOVIELENS 1M

	ml-1m s20		ml-1m s30	
	NDCG@10	***Precision@10***	***NDCG@10***	***Precision@10***
Random	0.76	0.65	0.78	0.66
Popularity	0.85	0.78	**0.88**	**0.81**
SVD	0.76	0.66	0.77	0.66
LinUCB	0.85	0.78	0.87	0.8
DDPG	**0.87**	**0.81**	0.83	0.74

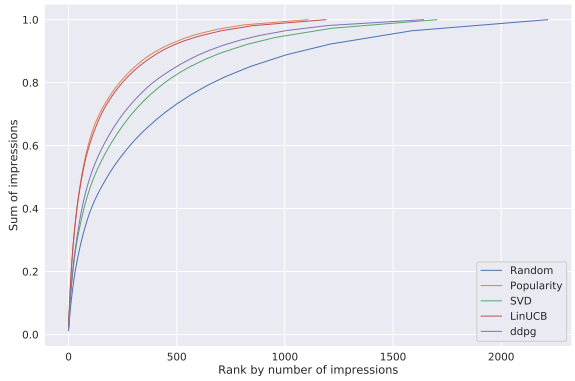

Fig. 6. Distribution of impressions over items for MovieLens 1M: lower curve means that algorithm served more different items

The main contributions of this work are 2 folds:

1) Proposed an interface for recommendation environment
2) Experiments with DDPG agents and comparison against baselines

That research direction is very promising and vast. Main advantages of the application of RL to recommendation process are

- we consider recommendation process as dynamic and optimize for long-term rewards
- we model the influence of the recommendations on user state
- MDP formalism is flexible and different scenarios can be modeled easily in this framework

But it is still not mature approach for building recommender systems and extensive ablation studies and comparison between different MDP models could be fruitful directions of future research.

REFERENCES

[1] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, pp. 76–80, 2003. [Online]. Available: doi.ieeecomputersociety.org/10.1109/MIC.2003.1167344

[2] H.-J. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems." *IJCAI*, pp. 3203–3209, 2017.

[3] S. Bonner and F. Vasile, "Causal embeddings for recommendation," in *Proceedings of the 12th ACM Conference on Recommender Systems*, ser. RecSys '18. New York, NY, USA: ACM, 2018, pp. 104–112. [Online]. Available: http://doi.acm.org/10.1145/3240323.3240360

[4] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica, "Ad click prediction: A view from the trenches," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13, A. Macintyre, L. Pacholski, and J. Paris, Eds. ACM, 2013, pp. 1222–1230.

[5] L. Li, W. Chu, J. Langford, and R. E. Schapire, "Proceedings of the 19th international conference on world wide web," in *A Contextual-bandit Approach to Personalized News Article Recommendation*, ser. WWW '10. ACM, 2010, pp. 661–670.

[6] D. Rohde, S. Bonner, T. Dunlop, F. Vasile, and A. Karatzoglou, "Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising," in *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, S. Pera, M. D. Ekstrand, X. Amatriain, and J. O'Donovan, Eds. ACM, 2018. [Online]. Available: http://dl.acm.org/citation.cfm?id=3240323

[7] F. Liu, R. Tang, X. Li, Y. Ye, H. Chen, H. Guo, and Y. Zhang, "Deep reinforcement learning based recommendation with explicit user-item interactions modeling," *CoRR*, vol. abs/1810.12027, 2018. [Online]. Available: http://arxiv.org/abs/1810.12027

[8] Y. Liu, Y. Zhang, Q. Wu, C. Miao, L. Cui, B. Zhao, Y. Zhao, and L. Guan, "Diversity-promoting deep reinforcement learning for interactive recommendation," *CoRR*, vol. abs/1903.07826, 2019. [Online]. Available: http://arxiv.org/abs/1903.07826

[9] X. Zhao, L. Xia, L. Zhang, Z. Ding, D. Yin, and J. Tang, "Deep reinforcement learning for page-wise recommendations," in *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*, S. Pera, M. D. Ekstrand, X. Amatriain, and J. O'Donovan, Eds. ACM, 2018, pp. 95–103. [Online]. Available: https://doi.org/10.1145/3240323.3240374

[10] X. Zhao, L. Xia, Y. Zhao, D. Yin, and J. Tang, "Model-based reinforcement learning for whole-chain recommendations," *CoRR*, vol. abs/1902.03987, 2019. [Online]. Available: http://arxiv.org/abs/1902.03987

[11] Y. Hu, Q. Da, A. Zeng, Y. Yu, and Y. Xu, "Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*, Y. Guo and F. Farooq, Eds. ACM, 2018, pp. 368–377. [Online]. Available: https://doi.org/10.1145/3219819

[12] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li, "DRN: A Deep Reinforcement Learning Framework for News Recommendation," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web - WWW '18*. ACM Press, 2018. [Online]. Available: https://doi.org/10.1145/3178876.3185994

[13] H. Chen, X. Dai, H. Cai, W. Zhang, X. Wang, R. Tang, Y. Zhang, and Y. Yu, "Large-scale interactive recommendation with tree-structured policy gradient," *CoRR*, vol. abs/1811.05869, 2018. [Online]. Available: http://arxiv.org/abs/1811.05869

[14] J. Gauci, E. Conti, Y. Liang, K. Virochsiri, Y. He, Z. Kaden, V. Narayanan, and X. Ye, "Horizon: Facebook's open source applied reinforcement learning platform," *CoRR*, vol. abs/1811.00260, 2018. [Online]. Available: http://arxiv.org/abs/1811.00260

[15] S. Rendle, L. Zhang, and Y. Koren, "On the difficulty of evaluating baselines: A study on recommender systems," *CoRR*, vol. abs/1905.01395, 2019. [Online]. Available: http://arxiv.org/abs/1905.01395

[16] X. Zhao, L. Zhang, Z. Ding, D. Yin, Y. Zhao, and J. Tang, "Deep reinforcement learning for list-wise recommendations," *CoRR*, vol. abs/1801.00209, 2018. [Online]. Available: http://arxiv.org/abs/1801.00209

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1509.02971

[18] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman,

J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016. [Online]. Available: http://arxiv.org/abs/1606.01540

[19] N. Hug, "Surprise, a Python library for recommender systems," http://surpriselib.com, 2017.

[20] F. M. Harper and J. A. Konstan, "The MovieLens datasets," *ACM Transactions on Interactive Intelligent Systems*, vol. 5, no. 4, pp. 1–19, Dec. 2015. [Online]. Available: https://doi.org/10.1145/2827872

[21] S. Pera, M. D. Ekstrand, X. Amatriain, and J. O'Donovan, Eds., *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*. ACM, 2018. [Online]. Available: http://dl.acm.org/citation.cfm?id=3240323