Towards Sample Efficient Reinforcement Learning*

Yang Yu
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
yuy@nju.edu.cn

Abstract

Reinforcement learning is a major tool to realize intelligent agents that can be autonomously adaptive to the environment. With deep models, reinforcement learning has shown great potential in complex tasks such as playing games from pixels. However, current reinforcement learning techniques are still suffer from requiring a huge amount of interaction data, which could result in unbearable cost in real-world applications. In this article, we share our understanding of the problem, and discuss possible ways to alleviate the sample cost of reinforcement learning, from the aspects of exploration, optimization, environment modeling, experience transfer, and abstraction. We also discuss some challenges in real-world applications, with the hope of inspiring future researches.

1 Introduction

Decision making is a basic activity in our everyday life. It is also an essential feature of intelligent agents. Particularly, the decision making for a long-term goal requires the intelligence of long-term vision and less greedy behaviors; the decision making in an unknown environment requires the intelligence of adapting the environment. Reinforcement learning [Sutton and Barto, 1998] studies the decision making for long-term goals in unknown environments, thus is at the center of artificial intelligence.

By reinforcement learning, an agent interacts with the environment, explores the unknown area, and learns a policy from the exploration data. In a common setting, the exploration data contains environment state transitions associated with the exploration actions and reward signals. From the data, the quality of the policy can be evaluated by the reward. Reinforcement learning algorithms update the policy model from the evaluations, with the aim of maximizing the reward in total. This *exploration–learning* framework is shared among almost all reinforcement learning algorithms. From the perspective of policy modeling, these algorithms can be categorized as value-function estimation algorithms and policy search algorithms. The former ones estimate a value-function to approximate the long-term reward from the current state and action. The policy is then derived from the value-function straightforwardly. The latter ones directly learn the policy model. Recent algorithms focus more on learning policy models with the help of value-functions, known as actor-critic approaches, inheriting the merits of the both.

The decades of development of reinforcement learning have achieved significant successes, such as in the AlphaGo system defeating top human players [Silver *et al.*, 2016] and playing Atari games exceeding human performance [Mnih *et al.*, 2015]. However, noticing that these successes are mostly in the digital world, there are still large barriers to applying reinforcement learning in real-world applications. A noticeable limitation of current reinforcement learning techniques is the low sample efficiency, which causes a huge amount of interactions with the environment. Such amount of interactions in real-world often means an unbearable cost. Even in complex environments in the digital world, such as playing the full StarCraft game, the low sample efficiency blocks the learning of a good policy.

To the best of our understanding, there are multiple reasons that could limit the sample efficiency. In this article, rather than making a survey, we share our understanding and discuss several aspects that could alleviate the limitations: considering the exploration-learning procedure, how to efficiently explore the environment and how to better optimize the policy; considering the environment, how to learn the environment model; considering multiple environments, how to reuse and transfer experiences across environments; and more essentially, how to abstract states and actions. We will discuss each aspect in a following section. We will also discuss some challenges in real-world applications that may have been less noticed and desire more attentions.

2 Exploration

In an unknown environment, the agent needs to visit states that have not been visited in order to collect better trajectory data. The agent cannot follow its current policy tightly, which has been learned from the previous data and may only lead to follow the previous paths. Exploration strategies are usually employed to encourage veering off the previous paths. Basic exploration methods such as ϵ-greedy and Gibbs sampling inject some randomness in the output actions, i.e., action space noise, so that the probability of executing every action, and

*This work is supported by Jiangsu SF (BK20170013), and Collaborative Innovation Center of Novel Software Technology and Industrialization.

thus visiting every state, is non-zero. A limitation of action space noise is that the resulting policy (i.e., the latent policy corresponds to the randomized output) may be far away from the current policy in the parameter space, or even out of the parameter space, which makes difficulties to the policy update.

Parameter space noise. Exploration by randomization in the parameter space, i.e., parameter space noise, could be more friendly to policy update. Plappert *et al.* [2018] showed that parameter space noise can be more efficient than action space noise. Fortunato *et al.* [2018] proposed another approach that could be rewarded as an intermediate approach between action space noise and parameter space noise. The NoisyNet adds a randomized neural network to the policy network, particularly, near the output layers, to induce randomized actions.

Curiosity-driven exploration. All the above exploration strategies are generally applicable, however, are all blind searches. The agent may repeatly try a bad path many times, since it does not know if the path has been explored before. This might be a major reason that the current general reinforcement learning algorithms require a lot of samples — find a good path by luck. Curiosity-driven exploration [Singh *et al.*, 2004] can be much more efficient than random exploration. The agent records the counts of the visits of every states and actions. According to the counts, an intrinsic reward is added to the environment reward to encourage visiting states that are less visited. This kind of approaches have been addressed a decade ago, where the state space and action space are small and discrete. For high-dimensional state space, an obstacle of implementing the curiosity-driven exploration is that it is hard to if a state has been visited before. Recently, Pathak *et al.* [2017] proposed the Intrinsic Curiosity Module (ICM) to overcome this obstacle. It employs the state prediction error as a measure to determine if a state has been visited. Meanwhile, it employs the self-supervision to learn a low-dimensional representation of the states. Intrinsic reward, however, is a delayed feedback to drive the agent. Mechanisms that directly encourage the exploration might be desired.

3 Optimization

After the exploration step that collects interaction data from the environment, the learning step updates the model of policy or value function from the data. Nowadays, neural networks might be the most popular choice as the models. However, find an appropriate neural network model is not that straightforward. Consider the policy search methods, the direct objective is to maximize the expected long term reward. This objective can be represented as the reward integrated over the current state and action distribution, where the distribution is determined by the policy. Unlike in the supervised learning scenario where the samples are fixed, optimizing this objective first requires to generate samples by the policy. Once the policy is updated, the distribution will be changed, and new samples have to be generated by the updated policy. Therefore, the optimization faces a non-static context, and achieving the best objective value on the current samples is not the goal, and exploration is necessary to find better samples.

Optimization from samples. The mainstream model updating approaches often rely on the gradient of the objective or surrogate objectives, such as in the TNPG [Duan *et al.*, 2016] and the TRPO [Schulman *et al.*, 2015] methods. However, these methods consider only the policy update from the samples, but do not touch the exploration. Meanwhile, there is another kind of optimization methods, optimization from samples, also known as derivative-free optimization, which can have their own advantages.

Derivative-free optimization algorithms share a common structure. Initialization from some random samples in the search space, they learn an area of potential better samples from these observations previous samples. They then generate new samples from that area, and repeat this sampling-and-learning iteration. Representative algorithms include evolutionary algorithms [Beyer and Schwefel, 2002], Bayesian optimization [Snoek *et al.*, 2012], etc.

Applying derivative-free optimization for reinforcement learning, a straightforward way is to define the search space as the policy parameters and the objective function as the expected long-term reward. A derivative-free optimization then tries to sample different policy parameters, and learn where to sample in the next iteration. We may have noticed that derivative-free optimization methods involve the exploration in the search process. Therefore, they can take part of the duty of exploration for reinforcement learning, and have been shown to have better performance on some tasks, both a decade ago [Whiteson, 2012] and very recently [Pet, 2018]. However, derivative-free optimization algorithms also share limitations, such as slow convergence, hard to scale up, noisy sensitive, and no theoretical guarantee. Recent progress in this direction include theoretical-grounded derivative-free optimization methods [Hu *et al.*, 2017], scaling-up methods for high-dimensional search spaces [Qian *et al.*, 2016], and noise handling methods [Wang *et al.*, 2018].

Hybrid optimization. Another direction to overcome the limitations of derivative-free optimization methods is to combine them with gradient-based methods. Jaderberg *et al.* [2017] borrowed the population idea of derivative-free methods to maintain a sample set of models, while the model optimization is still by a gradient-based method. Stochastic gradient Langevin dynamics (e.g., [Raginsky *et al.*, 2017]) recently attracted many attentions. It can be viewed as a hybrid method too, since the Langevin dynamics is equivalent with the a kind of random sampling. However, there are only quite a few studies about the hybrid of the two types of optimization methods. Hybrid optimization is be an interesting and promising direction, as it could overcome both the greedy nature of gradient-based methods and the slow convergence issue of derivative-free methods.

4 Environment Modeling

While model-free reinforcement learning algorithms have taken a large body of the research, model-based algorithms can be much more efficient, as long as the environment model can be efficiently constructed. An environment model includes a transition function, telling how the state will change after taken an action, and a reward function, telling how a transition would be rewarded. It is easy to see that the environment model learning is a supervised learning task. From some sam-

pled paths, a transition data set can be extracted in the way that the state together with the action at time t are composed as the input and the state at time $t+1$ as the output. Once the environment model has been built, planning in the model is free of real-world samples, thus would be an ideal approach to improve the sample efficiency. Unfortunately, such supervised learning approach works well only in discrete or small state space, but rarely works for large/high-dimensional cases.

Combining model-based and model-free. A trend has emerged that, since the environment model is hard to be accurately learned, the agent does not fully rely on the learned environment model to derive the policy, rather it extracts guiding information from an inaccurate model. In [Tamar *et al.*, 2016], the proposed Value Iteration Network employs a planning structured network to implement the value iteration, while the environment transition is learned together with the value iteration. The planning result from the value iteration is then used as an augmented features for the policy input, instead of directly learn a policy by the value iteration. Weber *et al.* [2018] proposed the Imagination-Augmented Agents involving an imagination modular, which learns the environment model. The roll-out paths in the modular are encoded as augmented features for the policy input. In [Pong *et al.*, 2018], instead of learning the environment transitions, it learns a Q-function that predicts the distance to the goal. This function serves as an immediate reward to guide the learning. We can see that these approach involve environment model learning can lead to significant improvement from model-free methods, and thus are quite promising. Meanwhile, modeling stochastic environments is still hard, and these methods currently work in restricted environments and are not yet general enough.

Manually constructed environment. In many applications, environment models, in other words, simulators, have been constructed manually, such as the simulators for aircraft and robot design. These simulators can be employed directly for training reinforcement learning with cheap cost. However, there are more sophisticated applications that are hard to construct simulators for by hand. For example, an online retail system involve buyers and sellers, which are hard to be simulated by coarse behavior models. Shi *et al.* [2018], based on the recent technique of generative adversarial networks, proposed a multi-agent imitation learning approach to reconstruct human policy from experience data. They showed that the approach can learn a simulator called Virtual Taobao that tightly mimic the customer behaviors in the Taobao environment, which is the one of the world's largest online retail platform. This work inspires that adversarial learning might have the power to simulate the physical world truthfully, when the experience data is sufficient. This approach separates the environment model learning with policy learning, while a combined one might be more general and powerful.

5 Experience Transfer

Human does not accomplish every task from scratch, but instead continually learn and accumulate experiences from many tasks. The accumulated experience from previous tasks can accelerate the learning in future tasks. In the similar way, an agent can also learn more efficiently in a task if experiences are available. This is in the subfield of transfer reinforcement learning, which also has been consistently studied for decades. Many methods have been proposed from various aspects, such as transfer of samples [Lazaric *et al.*, 2008], transfer of representation [Ferrante *et al.*, 2008], and transfer of skills/options [Sutton *et al.*, 1999] that is related to the abstraction.

Recent progress includes learning jumpstart models. Finn *et al.* [2017] proposed MAML to learn a kind of average model but is ready to be updated to different tasks. Therefore the model can be quickly updated to the specific tasks. Meanwhile, learning a jumpstart model has to assume a narrow distribution of tasks. Another way to adapt to the task is to sense the environment. Peng *et al.* [2018] proposed to employ an LSTM network to automatically infer the environment parameter from the interactions. Yu *et al.* [2018] proposed to obtain environment parameters by probing the environment through executing some roughly trained policies. Zhang *et al.* [2018] proposed to learn a set of calibration actions to probe the environment parameters. They showed that 5 probing samples could be sufficient to find a good policy in a new environment in their experiments. By sensing the environment, the policy learning task is reduced to the environment identification task, which could require only a few samples. But all these methods work only in restricted cases. General transfer reinforcement learning approaches are still missing.

6 Abstraction

We believe abstraction is in the core of the sample efficiency issue. Generally, an abstraction of the state space could lift to a higher-level state space with lower dimensions. Once this is possible, the exploration as well as the environment modeling in the abstract-level space can be much more efficient. However, abstraction has being a long-standing problem in artificial intelligence, which is far from being well solved.

Hierarchical reinforcement learning. One particular direction of abstraction in reinforcement learning is hierarchical reinforcement learning, which has been developed for decades. Early work includes learning with options [Sutton *et al.*, 1999], where an option is an abstraction of actions and is defined by the entrance condition, exit condition, and the option sub-policy; Hierarchical of Abstract Machines (HAMs) [Parr and Russell, 1997], where predefined automata are sub-policies; and MaxQ framework [Dayan and Hinton, 1992] which learns with decomposed subgoals. While there were studies on learning the hierarchical structure automatically, general applicable approaches were missing and hierarchical approaches still relied heavily on hierarchies given as priori knowledge.

Recent studies may reduce the requirement of hand-crafted hierarchies. SNN4HRL [Florensa *et al.*, 2017] utilized the information theory to train sub-policies automatically, which still requires domain knowledge to design the intrinsic reward. After that, the high-level policy then learns how to utilize the sub-policies to accomplish tasks. The Option-Critic [Bacon *et al.*, 2017] employed options but with no predefinitions. With the policy gradient method, it trains both the high-level policy of selecting options, and the options. FeUdal network [Vezhnevets *et al.*, 2017] employed the manager (high-level policy) and worker (low-level policy) structure. The high-level

policy sends signals to guide the behavior of the low-level policy, such that there are no explicitly defined sub-goals or sub-policies.

Hierarchical reinforcement learning approaches can benefit from the hierarchies in the way that the high-level policy is trained with a shorter horizon, and thus are more efficient. However, how the training horizon should be shorten has not been clear yet. Most of the approaches force that the high-level decision making takes place only every fixed steps. In this way, the high-level policy may not change the sub-policy on time. Moreover, hierarchical reinforcement learning might not be best fitted for solving only one task. In a multi-task and transfer learning scenario, sub-policies may be more defined as policy segments that can be reused across tasks, as partly in [Sahni *et al.*, 2017]. It is also noticeable that there are few studies about multiple levels hierarchies.

Neural network with symbols. Inside the policy model, it is noticed that, although the commonly employed neural networks are capable of abstraction from the raw input to some concept levels such as recognition the objects in a picture, they are not capable of operating in the abstraction level. But once they can, the policy model might be powerful to learn an abstraction of the state/action internally. It is worth noticing that a set of recent studies are pushing toward extending such ability of neural networks. Graves *et al.* [2016] enabled the memory ability of neural networks by simulating a memory cells. Hu *et al.* [2016] and Evans and Grefenstette [2018] embedded logic components as differentiable parts in neural networks. Unfortunately, due to the inability of doing recursions in the current neural network models, a reasoning path has to be expanded in advance, which makes the model too large to be stored. Dai *et al.* [2018] integrated a complete Prolog system within neural networks, so that the networks can perform first-order logical deductions using the efficient discrete search tree. They demonstrated that the new model that learns from algebra equation pictures can understand the algebra rules correctly, thus has a strong generalization performance on longer equations. We can imagine that, when a neural network model can do general abstractions with first-order logical reasoning ability, it could be possible to model the environment efficiently and could transfer its high-level reasoning across tasks.

7 Application Challenges

Other than the aspects discussed above, real-world applications may pose more challenges to reinforcement learning. We would like to take recommendation systems as the example, which have become a large source of benefit for many companies. The first challenge could be non-technical: to recognize that the application problem is a reinforcement learning problem. It is not until very recently, such as in [Hu *et al.*, 2018], recommendation systems are accepted as a reinforcement learning problem. This is a crucial first step for recognizing the potential value of reinforcement learning.

Dynamic environment. In real recommendation systems applications, the customers, which are the main part of the environment for the recommendation agent, keep changing all the time. Thus the environment is dynamic instead of static assumed in traditional reinforcement learning algorithms. Chen *et al.* [2018] identified two factors of the impact that the dynamic environment can lead to, high variance and fake reward, and used two tricks to alleviate the impact. However, how to fundamentally improve the learning algorithm so that it can infer the real outcome of its actions in a dynamic environment is still an open problem.

Very large action space. A large-scale recommendation system usually have a large number of items, say tens of thousands, to recommend. The recommendation of each item is usually formulated as an action. However, by classical algorithms, each discrete action has to be well explored, which requires many samples. Dulac-Arnold *et al.* [2016] proposed to learn an embedded action space, in which each point can be mapped to a discrete action. Meanwhile, the items commonly have their description features and categories. These information can organize items in a hierarchical tree with a similarity function, which could be useful for the learning, and might connect with hierarchical reinforcement learning.

8 Conclusion

This article discusses the sample efficiency issue of reinforcement learning, from multiple aspects. Note that instead of doing a survey, we only discuss these aspects with some representative studies. Also, there are other aspects, such as reward function design and imitation learning, that are related to the sample efficiency issue but are not covered. Reinforcement learning is a fast growing area. New ideas and approaches are blooming out. Nevertheless, the state-of-the-art is still far away from the fantastic ultimate goal of reinforcement learning. We can see there are plenty of room for inventing novel approaches, which, hopefully, would lead the artificial intelligence systems to the next level.

Acknowledgements

The author would like to thank Wei-Yang Qu and Wen-Ji Zhou for improving the paper.

References

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, 2017.

Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies–A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

Shi-Yong Chen, Yang Yu, Qing Da, et al. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In *KDD*, 2018.

Wang-Zhou Dai, Qiu-Ling Xu, Yang Yu, , and Zhi-Hua Zhou. Tunneling neural perception and logic reasoning through abductive learning. *arXiv:1802.01173*, 2018.

Peter Dayan and Geoffrey E. Hinton. Feudal reinforcement learning. In *NIPS*, 1992.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, et al. Deep reinforcement learning in large discrete action spaces. *arXiv:1512.07679*, 2016.

Richard Evans and Edward Grefenstette. Learning explanatory rules from noisy data. *Journal of Artificial Intelligence Research*, 61:1–64, 2018.

Eliseo Ferrante, Alessandro Lazaric, and Marcello Restelli. Transfer of task representation in reinforcement learning using policy-based proto-value functions. In *AAMAS*, 2008.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. In *ICLR*, 2017.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, et al. Noisy networks for exploration. In *ICLR*, 2018.

Alex Graves, Greg Wayne, Malcolm Reynolds, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471, 2016.

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. *arXiv:1603.06318*, 2016.

Yi-Qi Hu, Hong Qian, and Yang Yu. Sequential classification-based optimization for direct policy search. In *AAAI*, 2017.

Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *KDD*, 2018.

Max Jaderberg, Valentin Dalibard, Simon Osindero, et al. Population based training of neural networks. *arXiv:1711.09846*, 2017.

Alessandro Lazaric, Marcello Restelli, and Andrea Bonarini. Transfer of samples in batch reinforcement learning. In *ICML*, 2008.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Ronald Parr and Stuart J. Russell. Reinforcement learning with hierarchies of machines. In *NIPS*, 1997.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. *arXiv:1710.06537*, 2018.

Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv:1712.06567*, 2018.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, et al. Parameter space noise for exploration. In *ICLR*, 2018.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control. In *ICLR*, 2018.

Hong Qian, Yi-Qi Hu, and Yang Yu. Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In *IJCAI*, 2016.

Maxim Raginsky, Alexander Rakhlin, and Matus Telgarsky. Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. In *COLT*, 2017.

Himanshu Sahni, Saurabh Kumar, Farhan Tejani, and Charles Isbell. Learning to compose skills. *arXiv:1711.11289*, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *ICML*, 2015.

Jing-Cheng Shi, Yang Yu, Qing Da, Shi-Yong Chen, and An-Xiang Zeng. Virtual-Taobao: Virtualizing real-world online retail environment for reinforcement learning. *arXiv:1805.10000*, 2018.

David Silver, Aja Huang, Chris J. Maddison, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Satinder Singh, Andrew G. Barto, and Nuttapong Chentanez. Intrinsically motivated reinforcement learning. In *NIPS*, 2004.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, 2012.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value iteration networks. In *NIPS*, 2016.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, et al. Feudal networks for hierarchical reinforcement learning. In *ICML*, 2017.

Hong Wang, Hong Qian, and Yang Yu. Noisy derivative-free optimization with value suppression. In *AAAI*, 2018.

Théophane Weber, Sébastien Racanière, David P. Reichert, et al. Imagination-augmented agents for deep reinforcement learning. *arXiv:1707.06203*, 2018.

Shimon Whiteson. Evolutionary computation for reinforcement learning. In Marco Wiering and Martijn van Otterlo, editors, *Reinforcement Learning: State-of-the-Art*, pages 325–355. Springer, Berlin, Heidelberg, 2012.

Yang Yu, Shi-Yong Chen, Qing Da, and Zhi-Hua Zhou. Reusable reinforcement learning via shallow trails, 2018.

Chao Zhang, Yang Yu, and Zhi-Hua Zhou. Learning environmental calibration actions for policy self-evolution. In *IJCAI*, 2018.