On Deep Reinceforcement Learning Algorithms

Wuwei Lin
Shanghai Jiao Tong University

Abstract

Reinforcement learning has been boosted by deep neural networks. In this paper, we survey several deep learning based algorithms on reinforcement learning (DQN, Dueling Networks, A3C). We analyze and make comparations among these algorithms.

1. Introduction

Reinforcement learning involves interactions with the environment in a closed loop in discrete time steps by an agent. In each step, an agent observes reward and takes a action, which influences the environment in the future. This is generally modelled as a Markov Decision Process (MDP), where transitions of the state of the environment concerns the last step only. Classical reinforcement learning mainly focuses on learning the best policy to maximize reward collected in the future. Algorithms in reinforcement learning including dynamic programming, Monte Carlo and Temporal Difference have been widely studied. These algorithms attempt to estimate the value of each state or action. As values are estimated separately, in a complex system where both the action space and the state space are fairly large, these algorithms become computationally impractical due to its complexity. Function approximators have been applied to this domain.

Great advances have been made in deep neural networks (DNN) in recent years and it is proven pragmatical to apply DNN in reinforcement learning. However, different from vanilla deep learning based on a large amount of handlabelled data, reinforcement learning involves signals from temporal sequences, where data tends to be noisy sparse, delayed. In addition, as the environment is influenced by the agent continuously, it becomes a non-stationary problems. Model learned from previous results may not be applicable to estimations of the future as data distributions vary. Training such neural networks become rather challenging due to poor quality and quantity of training data. Different network architectures and techniques including action repetition, experience replay have been proposed.

In this paper, first we review the classical reinforcement learning, and then we compare some state-of-the-art deep reinforcement learning algorithms, namely Deep Q-Networks, Duel Networks and Asychronous Advantage Actor Critic. We also discuss furthur imporvements on these algorithms.

2. Reinforcement Learning

In reinforcement learning, an agent interacts with its environment in discrete time steps. In each step t, it chooses an action a_t following the policy π and observes reward r_t. Then the environment state transite to s_{t+1} from s_t by the dynamic $P^a_{ss'}$. The goal of the agent is to optimize its policy to maximize the rewards collected in the future

$$E[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...] \tag{1}$$

where γ is the discount factor. Policy π can be defined as distribution of actions under a state $\pi(\cdot|s)$. We define the value functoin of each state as $V(s) = E[G_t|S_t = s]$, and the action-value function of each action taken in each state as

$$Q(s, a) = E[G_t|S_t = s, A_t = a] \tag{2}$$

Reinceforcement learning can be approached by *policy gradient* or *value iteration*. Policy gradient methods are simple and direct, which represents policy as a parameterized function of features of states. It learns the action distribution on the state $\pi(s)$, The policy is parameterize by features of states. Policy gradient is model free, which does not require explicit MDP models, namely the reward function and the transition on actions $P^a_{ss'}$. It has good performance when handling high-dimentional input as the learning the distrubitions over actions is straightforward and be easier to approximate than the action-value function. However, policy gradient may have higher variance and computation complexity.

In this paper, we mainly focus on value iteration methods. Value iteration methods are based on *Bellman Equation*

$$V_\pi(s) = E[R_{t+1} + \gamma V(S_{t+1})|S_t = s] \tag{3}$$

$$Q_\pi(s, a) = E[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})| S_t = s, A_t = a] \tag{4}$$

Bellman equations can be solved by dynamic programming, Monte Carlo methods or TD methods iteratively. Value functions are updated according to Bellman equation, which will converge to the optimal value function as $t \to \infty$ theorically. Following value function, it is straightforward to make decesion in each step. In order to trade off between exploration and exploitation, the action can be picked by greedy algorithms such as ϵ-greedy algorithm.

2.1. Q Learning

Q learning is a model-free and off-policy method based on learning on action value functions. It learns from previous experience following

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t)) \tag{5}$$

The learning target is $R_{t+1} + \gamma Q(S_{t+1}, A')$, where A' is chosen using the policy being learned such that $Q(S_t, A_t)$ will converge to the optimal state action function. Therefore, (5) can be written as

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \underset{a'}{max} Q(S', a') - Q(S_t, A_t)) \tag{6}$$

There are some issues to be addressed. First, Q value cannot be updated efficiently as in each step reward of only state-action pair is observed. Second, estimating value of each state-action pair saperately incurs unaffordable computation complexity when the domain becomes large. Third, the Markov property may not hold in many cases when dealing with real-world problems. Actions may impose influences on the environment in many steps in the future.

2.2. Actor Critic Methods

Actor-critic methods combine value-based methods and polocy-based methods. It learns policies and state value functions or state action value functions simultaneously. Unlike value-based methods where policies are generated greedy according to learned value functions, policies in actor-critic methods are learned directly. Actions are evaluated by the learned value function. Parameterized policies are then updated with respect to gradients of the value function.

In actor-critic methods, actions can be picked directly like policy-based methods, which is critical for domains such as continuous control. Moreover, as it learn explicit policies, it can tackle sceanios where Markov properties do not hold. As actor-critic methods learn both policies and value functions, extra computation costs may be introduced.

3. Deep Reinforcement Learning

Deep Q Networks (DQN), Duel Networks and Asychoronous Advantage Actor Critic (A3C) deep reinforcement learning algorithms based on are Q learning or actor-critic methods. We will delve into these algorithms in this section.

3.1. Deep Q-Networks

The action value function can be approximated by deep neural networks. Deep Q-Networks proposed by Mnih *et al*. [2, 3] is the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. DQN involves a deep networks with parameters θ for prediction that is being learned and a fixed target networks whose parameters θ^- are updated with θ periodically. The estimation error is given as $Q(s, a; \theta) - Q(s, a; \theta^-)$

Deferred update of the target network stablize the learning process as the learning target is less sensitive to immediate rewards.

Experience replay is applied in DQN. In each step, the agent save the state transitions into the experience buffer and learn from data sampled from its experience. As data for learning is randomly ordred, data correlation is alleviated. Besides, it allows the saved state to be use repeatedly, accelerating the training process.

DQN suffers from overestimation due to its builtin property [6]. DQN uses a max operator over Q value of actions given by target network in its learning target, and uses the same parameters θ^- to evaluate actions. Hasselt *et al*. [7] proposes DDQN as an imporvement of DQN to address the issue of overestimation. Instead of θ, DDQN uses θ^- to evaluate actions. This simple technique is progmatical to alleviate overestimation without introducing other issues.

3.2. Dueling Deep Q Learning

Wang *et al*. [8] proposes a novel network architecture that estimates state value $V(s)$ and action advantage $A(s, a)$ in two streams saperately and then combines the two streams to compute the Q value.

Estimating value of each state-value pair is unnecessary as some of them do not have actual meanings, resulting in inefficiency of the learning process. In dueling networks, Q value can be learned efficiently as it is devided into two parts. Value of a state is updated for each training sample, while advantage value of only one state action pair is updated, compared with Q-learning or DQN where Q value of only one state action is updated. Frequent updates of the state value function in dueling networks accerate the vanilla Q-learning by allocating more resources to the state value function.

The network architecture of dueling networks is similar

with DQN and other deep networks, with the last layer replaced with a dueling module that divides the output into two streams. Therefore, dueling networks can be a additional module of deep models like DQN or DDQN that further enhance their performance.

The output of the dueling networks is the state action function. Therefore, it is model-free and off-policy, which can cooperate with other algorithms such as SARSA. Besides, other pragmatic mechanisms including experience replay can be easily applied to the dueling networks.

3.3. Asychronous Advantage Actor Critic

Asynchronous methods *et al*. [1, 4] are a new paradiam of deep reinforcement learning. In asynchronous framworks, multiple agents sharing the same network and interacting with multiple instances of the environment run in parallel, allowing agents at the same state to explore in independantly.

Different from experience replay where batches are sampled from previous experience, asychronous methods are capable to learn online as samples are generated by multiple agents at the same time. As indenpendant agents can choose different policies, they are able to explore different states and actions widely under a shared network. Thus, signals received by different agents are less relevant compared with DQN where signals are generated sequently from the interaction with the environment by a sole agent.

Among asychronous frameworks proposed in [1], asychronous advantage actor critic (A3C) has the best overall performance. It follows the actor-critic paradiam and adds a entropy regularization term to its loss to prevent it converging to suboptimal results. A3C is an effective algorithm as parallalism can solve the problem of data inefficiency as experience replay but makes better use of exploration. Besides, unlike other deep learning frameworks that require huge computation resources, A3C that runs parallelly in multiple thread is rather efficient.

4. Discussion

DQN, DDQN, the dueling networks and A3C are all model-free and off-polocy algorithms. A3C can learn online from parallel agents, while others utilize experience replay to reduce data cooralation and data inefficiency. A3C can run effectively in CPUs as it fully utilize CPU resources by running in multiple threads, while others require more computation resources such that high performance GPUs are indispensible. As A3C explore differnet states and actions in vast space, it is likely have less variance than other algorithms.

References

[1] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *CoRR*, 2016.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing Atari with Deep Reinforcement Learning. *CoRR*, 2013.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[4] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, S. Legg, V. Mnih, K. Kavukcuoglu, and D. Silver. Massively Parallel Methods for Deep Reinforcement Learning. *arXiv.org*, (arXiv:1511.06581), July 2015.

[5] Schaul, Tom, Quan, John, Antonoglou, Ioannis, and Silver, David. Prioritized Experience Replay. *CoRR*, 2015.

[6] H. van Hasselt. Double Q-learning. *NIPS*, 2010.

[7] H. van Hasselt, A. Guez, and D. Silver. Deep Reinforcement Learning with Double Q-Learning. *AAAI*, 2016.

[8] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas. Dueling Network Architectures for Deep Reinforcement Learning. *ICML*, 2016.