A Literature Review on Deep Reinforcement Learning

Shangyi Li 2819****

SOUTHEAST UNIVERSITY - MONASH UNIVERSITY JOINT

GRADUATE SCHOOL

12 May 2017

Abstract

Deep reinforcement learning (DRL) is a new research hotspot in the artificial intelligence community. This paper focuses on three main categories of deep reinforcement learning methods. Firstly, we summarize value-based methods which use deep neural networks to approximate the value function. Secondly, value-based DRL methods are introduced, which directly approximate the polices with deep neural networks. Thirdly, we introduce multiagent DRL methods which have the ability of cooperation and communication between multiple agents. Finally, we end up with discussing some cutting-edge DRL methods and DRL's bright future.

Key Words: Deep Reinforcement learning, Reinforcement Learning

1. Introduction

Reinforcement learning (RL) usually solves sequential decision making problems. An RL agent interacts with an environment over time. However, learning to control agents directly from high-dimensional sensory inputs like vision and speech is one of the long-standing challenges of RL (Minh et al., 2013). DRL integrates the advantages of the perception of deep learning and decision making of reinforcement learning, and gains the output control policy directly on raw inputs by the end-to-end learning process.

DRL has recently achieved notable successes in a variety of domains, like Atari games (Minh et al., 2013; 2015), the game of go (Silver et al., 2015), robot control (Lillicrap et al., 2016), and computer vision (Oh et al., 2015). DRL now is considered to be an important way to AGI (Artificial General Intelligence).

2. Scope and method

This article mainly focuses on breakthroughs in the field of DRL. In this review, we describe three main categories of DRL methods. Firstly, we introduce value-based DRL methods, which use deep neural networks to approximate the value function. Deep Q-network, which is the groundbreaking work in the field of DRL, and its improved variants are introduced in this session. Secondly, we summaries policy-based DRL methods, which directly approximate the policies with deep neural networks. We highlight two outstanding algorithms, and among which, Asynchronous Advantage Actor-Critic is now the most successful DRL algorithm. Thirdly, multiagent DRL methods which make multiple agents cooperate or compete in complex tasks are introduced.

All the papers are found on Google scholar, with typing in the key word "deep reinforcement learning". We choose the papers which have made great contribution to the field of DRL, with averagely high numbers of citations.

3. Review

3.1 Value-based DRL

3.1.1 Deep Q-network

Minh et al. (2013) proposed Deep Q-network (DQN) model, which combines the convolution neural network with the Q-learning algorithm of traditional RL. This model is used to deal with vision-based control tasks and ignited the field of DRL. In environments with discrete action choices, DQN has achieved super-human performance on a large collection of diverse Atari 2600 video games by using only raw sensory input (screen images) and the reward signal.

In deep Q-learning, a neural network is optimized to predict the average discounted future return of each possible action given a small number of pixel images. The action with the highest predicted return is chosen by the agent. Specifically, the input of a DQN model is the last four pre-processed images from the current moment. The input is subjected to a non-linear transformation of three convolution layers and two fully connected layers, resulting in the Q value of each action at the output layer.

In order to train a neural network to approximate the optimal Q value, DQN minimizes a sequence of loss function $L_i(\theta_i)$ that changes at each iteration.

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim \mathrm{U}(D)}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]$$

in which r is the immediate reward taking action a, γ is the discount factor determining the agent's horizon, θ_i are the parameters of the Q-network at iteration i and θ_i^- are the network parameters used to compute the target at iteration i. The target network parameters θ_i^- are only updated with the Q-network parameters θ_i every C steps and are held fixed between individual updates.

The use of a replay memory to decorrelate samples is a critical element of DQN, as is the use of a target network. Both mechanisms help to alleviate the instability of using a non-linear network as the approximation function.

3.1.2 Double Q-network

In Q-learning and DQN, the max operator uses the same values to both select and evaluate an action. This can therefore lead to overoptimistic value estimates (Hasselt et al., 2015). Hasselt et al. (2015) proposed Deep Double Q-Network (DDQN), based on double Q-learning (Hasselt, 2010). In double Q-learning, there are two sets of parameters: θ and θ^{-}. θ is used to select the action which has the maximum Q value, while θ^{-} is used to evaluate the Q value of the optimal action.

Two sets of parameters separate the action selection and policy evaluation, reducing the risk of overestimating the Q value. Thus, DDQN uses the parameter θ of the current Q-network to select the optimal action, and use the parameter θ^{-} of the target Q-network to evaluate the optimal action. This can be achieved with a minor change to the DQN algorithm, replacing y_t^{DQN} with

$$y_t^{DDQN} = r_{t+1} + \gamma Q\left(s_{t+1}, \max_{a} Q(s_{t+1}, a_t; \theta_t^{-}\right)$$

Experiments show that DDQN can estimate more accurate Q value, and can obtain more stable and effective policies in some Atari 2600 game.

3.1.3 prioritize experience replay

In DQN, experience transitions are uniformly sampled from the replay memory, regardless of the significance of experiences. Also because of the limited capacity of the replay memory, some samples are abandoned before they have been fully utilized. Schaul et al. (2016) proposed to prioritize experience replay, built on top of DDQN. This method replaces the uniform sampling in DQN with the priority-based sampling method, improves the sampling probability of some valuable samples, thus accelerating the learning process of the optimal policy.

This sample method uses TD errors to measure the importance of experience transitions. The larger the error is, the higher the probability that the corresponding transition is sampled. In the sampling process, the method uses both stochastic prioritization and importance-sampling weights. Among them, stochastic prioritization can not only make full use of transitions with larger TD error, but also ensure the diversity of samples. The use of importance-sampling weights slows down the speed of parameter updates and ensures the stability of learning process. Experiments show that DDQN with prioritize experience replay can improve the training speed and get better performance on Atari 2600 games.

3.1.4 DRQN

Real-world tasks are often feature incomplete and have noisy state information. In traditional RL, partial observability of state information has always been a problem to be solved. DRL has yielded proficient controllers, like DQN, for complex tasks. However, these controllers have limited memory and rely on being able to perceive the complete game screen at each decision point.

Hausknecht et al. (2016) proposed DRQN, using a recurrent neural network to remember the historical state information on the time axis. DRQN replaces the first full-link layer component of the DQN with 256 long short-term memory units (LSTM). In this case, the input of the model is only an image at the current time, which reduces the computational resources consumed by extracting image feature. Experiments show that, in the case of partial state observability, DRQN shows better performance than DQN.

3.2 Policy-based DRL

Policy gradient is a common policy optimization method, that updates policy parameters by continually calculating the gradient of the expectation of total reward, and eventually converges to an optimal policy. Therefore, when solving a DRL problem, we can use a deep neural network with parameter θ to approximate a policy, and optimize the policy by policy gradient method. It is worth noting that when solving a DRL problem, policy gradient algorithms are the first choice. The reason is that it can directly optimize the expected total reward, searching for an optimal policy in the policy space in an end-to-end manner. Thus, compared with DQN and its improved models, policy-based DRL methods have a wider range of application and the better performance in policy optimization.

3.2.1 Actor-critic DRL method

Traditional policy-based methods optimize the policy which is approximated by a neural network, by directly optimizing parameters of the network with policy gradient methods. This kind of methods requires sampling mini-batches of trajectories to update policy gradient at each iteration. However, in many complex scenes, it is difficult to obtain a large amount of training data online. For instance, in robot control tasks, collecting and making use of a large number of training data online will take a very expensive price, while the continuous characteristic of action space makes it impossible to sample all the trajectories. The above problems lead to convergence to local optimality while using policy gradient methods. Actor-critic (AC) method in RL can effectively solve the problems.

Lillicrap et al. (2015) modified deterministic policy gradient methods with the idea of DQN, and proposed deep deterministic policy gradient (DDPG) algorithm, based on AC method. DDPG is able to solve DRL problems with continuous action space. DDPG uses two networks with parameters θ^{μ} and θ^{Q} separately to approximate deterministic policy $a = \pi(s|\theta^{\mu})$ and value function $Q(s, a|\theta^{Q})$. The policy network is used to update the policy, corresponding to the actor in AC methods, while the value network is used to evaluate the current policy, corresponding to the critic, by approximating the value function.

In DDPG, the objective function is defined as discounted total reward.

$$J(\theta^{\mu}) = \mathrm{E}_{\theta^{\mu}}[r_1 + \gamma r_2 + \gamma^2 r_3 + \cdots]$$

Policy gradient algorithm is adapted to optimize the objective function end to end. DDPG uses replay memory mechanism to sample transitions in the memory D, and then send gradient information about Q value function of actions from to critic network the actor network. The actor network then updates parameters in the direction of increasing the Q value.

Experiments show that DDPG is stable in a series tasks of continuous action space and faster in convergence to the optimal policy than DQN. Compared to value-based DRL methods, policy-based methods with AC framework are more efficient and faster.

3.2.2 A3C

To alleviate the instability caused by the combination of traditional policy gradient methods and neural networks, various deep policy gradient methods, like DDPG, all adapt experience replay mechanism to eliminate the correlation between training data. However, this mechanism has two shortcomings. 1) Each real-time interaction between the agent and the environment requires a lot of memory and computing power. 2) Experience replay mechanism requires agent learn off-policy, while off-policy methods can only update based on data generated by old policy.

Mnih et al (2016) proposed a light-weighted DRL architecture, based on the idea of asynchronous reinforcement learning, overcoming the above problems. This architecture can use asynchronous policy gradient method to optimize parameters of network controllers, and what's more, can be combined with various RL algorithms. Among which, asynchronous advantage actor-critic (A3C) perform best in a large variety of tasks with continuous actions spaces, reach the state-of-the-art level.

To be specific, A3C algorithm takes advantage of multi-threading function of CPU, running multiple agents asynchronously in parallel. Thus, at any time, parallel agents will go through many different states, avoiding the correlation between state transition samples in training process. Therefore, this kind of asynchronous methods can perfectly replace the experience replay mechanism.

A3C reduces hardware requirements in the training process. Deep policy gradient algorithms relies deeply on strong computing power of GPU, while A3C only need a standard multi-core CPU in practice. The average performance of A3C algorithm in Atari games has been significantly improved with far less training time. Therefore, A3C can learn to navigate random 3D mazes only using a visual input. In conclusion, A3C algorithm is the most versatile and successful DRL algorithm today. Of course, combining A3C with some recent deep policy gradient methods may further improve its performance.

3.3 Multiagent DRL

The decision-making capability of a single agent system is far from enough when faced with complex decision-making problems in some real scenes. For example, in multi-player Atari games, different players interact with the environment and competitive and collaborative behaviors may emerge in the system. Therefore, in some specific situations, DRL models have to be extend to a multiagent system that various agents coordinate, communicate and compete.

3.3.1 Multiagent cooperation and competition

In multiagent RL algorithms, in most cases, training mechanisms are assigned separately to each agent. For example, applying independent Q-learning algorithm to train each agent. This kind of distributed learning framework reduces the difficulty of learning and complexity of computation. For DRL problems with large state space, replacing Q-learning with DQN algorithm to train each agent separately, can easily construct a simple multiagent DRL system. Tampuu et al. (2015) made use of the above idea, adjusting the rewarding scheme dynamically according to different targets, and proposed a DRL model that agents have the ability of cooperation and competition.

4. Conclusion

Deep reinforcement learning is a new research hotspot in the artificial intelligence community. In this review, we introduce three main categories in the field of DRL, which are value-based, policy-based and multiagent DRL methods. Except for these methods, there are still some cutting-edge research directions we haven’t covered, including search and supervision based DRL methods which combines DRL models with Monte Carlo tree search algorithm, such as AlphaGo, and hierarchical DRL methods which decompose the ultimate goal into a series of sub-goals which has significant performance when reward signals are sparse and long-delayed. DRL has made substantial breakthroughs in a variety of tasks requiring both rich perception of high-dimensional raw inputs and policy control since it was proposed. In Atari games and the game of go, performance of DRL approaches significantly exceed human level, which implies the bright future of DRL. Although DRL algorithms do not have the ability to think, reason and learn like human so far, we believe that we will reach artificial general intelligence in the near future, with the development of DRL theories and methods.

References

Hasselt, H.V., 2010. Double Q-learning. In *Advances in Neural Information Processing Systems*, pp. 2613-2621.

Hausknecht, M. and Stone, P., 2015. Deep recurrent q-learning for partially observable mdps. arXiv preprint arXiv:1507.06527.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K., 2016. Asynchronous methods for deep reinforcement learning.

In International Conference on Machine Learning, pp. 1928-1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. Nature, 518(7540), pp. 529-533.

Oh, J., Guo, X., Lee, H., Lewis, R.L. and Singh, S., 2015. Action-conditional video prediction using deep networks in atari games. In Advances in Neural Information Processing Systems, pp. 2863-2871.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. and Dieleman, S., 2016. Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), pp. 484-489.

Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J. and Vicente, R., 2015. Multiagent Cooperation and Competition with Deep Reinforcement Learning. arXiv preprint arXiv:1511.08779.

Van Hasselt, H., Guez, A. and Silver, D., 2016. Deep Reinforcement Learning with Double Q-Learning. In AAAI, pp. 2094-2100.