A SURVEY ON VALUE-BASED DEEP REINFORCEMENT LEARNING

Feng Xiaolong
xlfeng886@163.com

ABSTRACT

Reinforcement learning(RL) is developed to address the problem of how to make a sequential decision and the target is to maximize the total reward. It is very successful in many traditional fields for decades. But deep learning(DL) become popular in these years, and it has extended the boundary of many tradition research filed. Combine with RL and DL is the fundamental idea for researchers recently. The first breakout if the DQN algorithm, which is proposed by (Mnih et al., 2013). It reopened a new branch of the RL field, as we called it as Deep Reinforcement Learning(DRL). After the DQN algorithm, there are many new algorithms have been proposed, and the number of related papers has grown exponentially in recent years. In this survey, we focus on the value-based deep reinforcement learning method, which is one of the bases of the DRL algorithm sets. Many papers are derived from the original DRL algorithm (Mnih et al., 2013), and the state-of-art algorithm has been halted after the Rainbow algorithm (Matteo et al., 2017). In this paper, we discussed many other algorithms, including the Rainbow algorithm. We hope to help the novel methods of the value-based algorithm, which can reach the new state-of-art.

1 INTRODUCTION

As we consider the essential question in computer science, how can we build an efficient artificial intelligence(AI)? We should imagine that the new creature will face the uncertain of the world. The structure of the world is so complicated, and it should know how to make the sequence decision in the real world. As (Sutton and Barto, 2018) discussed the mechanism of how humans affected by the dopamine, the AI agent should also be derived by the reward and towards the maximum of the total reward during the learning progress. In recent years, DRL is the most popular method to approach the efficient AI system. DRL utilizes the DL method to deal with the unstructured of the world. In the past decades, researchers designed many features for the specific problem, which cost too many resources for a little improvement. But with the rapid development of DL technology and its widespread application in various fields, state-of-art in many areas has been continuously refreshed. The reinforcement learning field is also one of the fields that deeply affected by DL technology. As we know, RL is inspired by the behavioral psychology (Sutton and Barto, 2018). RL technology can handle the sequential decision-making problem. So the combination of reinforcement learning and deep learning leads to a newborn deep reinforcement learning technology. The DRL technology fills us with visions of the future, and it has extended the boundaries of our cognition, allowing revolutionary changes in many applications.

We could find some achievements are due to the DRL technology (LeCun et al., 2015), (Schmidhuber, 2015), (Goodfellow et al., 2016). For example, (Mnih et al., 2015) utilized the DRL agent to learn the raw pixels of the Atari game and achieve human-level performance. (Silver et al., 2016) can defeat human champions in the go game. (Silver et al., 2018) use DRL to play chess and Shogi by the self-play method. (Vinyals et al., 2019) applied DRL in complicated strategy games and achieve a fantastic result. DRL also performs very well in real-world applications, such as robotics (Gandhi et al., 2017) (Pinto et al., 2017), self-driving cars (Pan et al., 2017). Even we can see some very successful application cases in the finance field (Deng et al., 2016).

1.1 DRL ALGORITHM TAXONOMY

Here we want to have an overview of the DRL algorithm taxonomy. As we can see, DRL is a fast-developing field, and it is not easy to draw the taxonomy accurately. We can have some traditional perspectives for this, which ignore the most advanced area of DRL (meta-learning, transfer learning, MAML exploration, and some other fields).

The root branching points in a DRL algorithm is to consider the question of whether we can access to the environment model. If the state transitions and rewards can be predicted from the simulated model, we can consider it as the model-based DRL algorithm. If we have no idea about the transitions and rewards, we should consider it as the model-free DRL algorithm.

In model-free DRL algorithms, we could have policy optimization technology(eg. Policy Gradient (Sutton et al., 2000), A2C (Mnih et al., 2016), A3C (Mnih et al., 2016), PPO (Schulman et al., 2017), TRPO (Schulman et al., 2015)), Q-learning technology(eg. DQN (Mnih et al., 2013), C51 (Bellemare et al., 2017), QR-DQN (Dabney et al., 2018), HER (Andrychowicz et al., 2017)), and hybird alogrithms((Lillicrap et al., 2015), TD3 (Fujimoto et al., 2018), SAC (Haarnoja et al., 2018))

In a model-based DRL algorithm, it should consider whether its model is given or the model is unknown. Alpha Zero (Silver et al., 2018) is the application of the given model. If the model is unknown, we could have some algorithms such as I2A, MBMF, MBVE.

In this article, we only survey the algorithms which are value-based.

2 PROBLEM FORMULATION

Reinforcement Learning (Sutton and Barto, 2018) is to learn how to make sequential decision when the AI agent interact with the environments E. We will formalize the E as a Markov Decision Processes (MDPs)[xxx]. it could be described as the tuple (S, A, P, R). That means in each time step, the AI agent interact with the observed state $s_t \in S$, and chooses the action $a_t \in A$. The action will lead to the reward $r_t \sim R$ and next state $s_{t+1} \sim P$.

The purpose of the learning progress is to maximize the total reward when the AI agent interacts with the environments E. In the infinite condition, we also consider the discount factor γ per time-step. The total discounted return at time step t should be $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where T is the time-step and we ignore the step before time step t due to the causality law.

The optimal action-value function $Q^*(s, a)$ is the maximum expected return on the condition below s_t and a_t. We can define it as $Q^*(s, a) = \max_\pi \mathbb{E}[R_t|s_t = s, a_t = a, \pi]$, where π is the policy. The $Q^*(s, a)$ function will follow the Bellman equation[xxx] and update the next Q value iteratively as $Q_{i+1}(s, a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s', a')|s, a\right]$. This is proved to converge to optimal Q^* in tabular case (Sutton and Barto, 2018). In practice, we usually use a function approximator to estimate the Q value. The linear function approximator and neural network are both the reasonable choice.

In the reinforcement learning community, this is typically a linear function approximator, but sometimes a non-linear function approximator is used instead, such as a neural network. When we use the neural network, we parameterize the Q-network according to the weight θ and train it by minimizing the mean square error(MSE) loss function. The parameter θ will change iteratively as below

$$argmin_\theta \, L_i\left(\theta_i\right) = E_{s,a\sim P(\cdot)}\Big(E_{s'\sim E}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})|s, a] - Q\left(s, a; \theta_i\right)\Big)^2 \quad (1)$$

In the optimization progress, previous parameters θ_{i-1} will be fixed. The optimization method could be one of the most popular methods, such as ADAM, RMSprop, Momentum SGD, etc.

3 RELATED WORK

For decades, scholars have made many outstanding contributions to the field of RL, and traditional RL algorithms are now very mature. We could find many papers that review the traditional RL, such as (Gosavi, 2009),(Kaelbling et al., 1996),(Sutton and Barto, 2018), (Szepesvári, 2010).

Benefit from the rapid development of DL technology, it is continuously refreshing the state-of-the-art in the RL field. There is also some survey cover the developments in DRL field, such as (Arulkumaran et al., 2017). There is also some paper survey the DRL application in NLP filed, such as (Ranzato et al., 2015) and (Bahdanau et al., 2016). For robot applications, we can refer to [survey on robot].

4 VALUE-BASED RL

The value-based RL algorithms will define the policy π, by constructing a function of the state-value. Q-learning algorithm (Watkins and Dayan, 1992) is the bias and most popular value-based RL algorithm. We can find some traditional ways to improve the Q-learning by parameterized function approximator (Gordon, 1996).

Before we dive into the Q-learning, we should consider the Sarsa algorithm at first. We should consider Sarsa as an on-policy algorithm and consider Q-learning as an off-policy algorithm.

4.1 SARSA ALGORITHM

Unlike Q-learning algorithm to learn state value, the Sarsa algorithm is to learn the action-value function, $q_\pi(s, a)$. The reward will generated by $q_p i(s_t, a_t)$ and lead to new state-action pair (s_{t+1}, a_{t+1}), so that we get the chain according to the state, action and reward. The detail of Sarsa algorithm can refer to (Sutton and Barto, 2018)

4.2 Q-LEARNING ALGORITHM:

It will lookup the Q value table for the simplest version of Q-learning. It applies the Bellman equation to update the Q-value function (Bellman and Kalaba, 1962). The ideal is straightforward and works not very well in the high-dimensional state-action space. The parameterized value function $Q(s, a; \theta)$ must be used to fit the value. The detail of the basic Q-learning algorithm can refer to (Sutton and Barto, 2018).

4.3 FITTED Q-LEARNING

As we could get a lot of experience, in the form of (s, a, r, s'). In fitted Q-learning (Gordon, 1996), it use an function approximator to update the $Q(s, a; \theta_k)$ in each iteration, where θ_k with respect to the parameter of the function in k_{th}iteration. So whe could obtain the target value as the equation below:

$$y_k = r + \gamma \max_{a' \in A} Q(s', a'; \theta_k)$$

4.4 NEURAL FITTED Q-LEARNING (NFQ)

The neural fitted Q-learning (NFQ) (Riedmiller, 2005), The neural Q-network is fed by the state information and output the probability of each action. The network is parameterized with a neural network $Q(s, a; \theta_k)$ and apply SGD to minimize the MSE.

But there is some issue inside it, and the target will be changed as the weights change and generate errors in the state-action space. We can not guarantee the convergence by using this algorithm (Baird, 1995) (Tsitsiklis and Van Roy, 1997) (Gordon, 1996) (Riedmiller, 2005). The other problem is the overestimate issue (Van Hasselt et al., 2016).

5 VALUE-BASED DRL

Here we specifically survey the deep Q-network (DQN) algorithm (Mnih et al., 2015). The DQN algorithm is the first algorithm that can achieve a superhuman level in the Arcade Learning Envi-

ronment (*ALE*) environment (Bellemare et al., 2013). We will also survey various improvements based on the DQN algorithm.

5.1 DEEP Q-NETWORKS

There is some common sense as NFQ, and the DQN algorithm is so attractive due to the excellent performance in the variety of ATARI games. It directly learns the raw pixels from the video and keeps the same structure and parameters in all games. This is unimaginable in previous algorithm implementations, which face the curve of dimension problem in high dimension space. To address the unstable and divergence problem, (Mnih et al., 2015) have proposed two methods, experience replay, and target Q-network.

(a) Experience replay mechanism: DQN initializes a replay buffer $\mathbf{D}$ with transitions (s_t, a_t, r_t, s_{t+1}) and randomly choose samples from the buffer to train the deep neural network. The simple idea is very efficient in implementation.

(b) Fixed target Q-network: To address the shift of the Q-value problem, the target Q-network will periodically synchronize parameters with the main Q-networks. The performance of DQN seems to be more stable for this technology.

5.2 DOUBLE DQN

To address the over-estimations issue, (Van Hasselt et al., 2016) propose a solution to divide the max operation into action selection and action evaluation. Two networks evaluate and choose action values according to the loss function as below:

$$Loss = \Big[r_t + \gamma Q_{evaluate}\Big(s_{t+1}, \arg\max_{a_{t+1}} Q_{estimate}\big(s_{t+1}, a_{t+1}; \theta_{estimate}\big); \theta_{evaluate}\Big) - Q_{evaluate}(s_t, a_t; \theta_{estimate})\Big]^2$$

The estimate network is parameterized by $\theta_{evaluate}$, and apply $\epsilon - greedy$ policy. The evaluate will be the policy of the estimate network and parameterized by $\theta_{evaluate}$. The evaluate network will periodically be synchronized with an estimate network.

5.3 DUELING DQN

We can decompose the Q-value function into value function and advantage function. So the dueling DQN used a specialized dueling network architecture to implement this simple idea. (Wang et al., 2015) proposed to change the single output network to a double output network. They share the same encoder. The formulation can be considered as below:

$$Q(s, a; \theta_1, \theta_2) = V(s; \theta_1) + \Big(A(s, a; \theta_1) - \frac{\sum_{a'} A(s, a'; \theta_1)}{|A|}\Big),$$

where θ_1 is the parameter of advantage function structure, and θ_1 is the parameter of the value function structure.

5.4 PRIORITIZED REPLAY

In the DQN algorithm, it proposes to use replay buffer technology and the algorithm uniformly sample from the buffer to train the neural network. (Schaul et al., 2015) have a one more step ideal for this technology, the proposed prioritized replay buffer. They set the priority of the experience in the buffer according to the defined rules, and then sample the experience by the priority. The high priority experience is worse to learn. Priority rules are usually set using TD error, but the choice of this method could be defined according to the particular problem. So it is not easy when we could not know what experience is essential and what is not.

5.5 ASYNCHRONOUS MULTI-STEP DQN

The previous DQN algorithm needs to consider using off-policy algorithms to improve efficiency, and it uses the replay buffer, which occupied much memory and computation resources. (Mnih

et al., 2016) proposed an idea to use many agents to train the deep neural network. It applies the asynchronous gradient descent method to train the multiple agents. Each subprocess maintains its own environment, and all the accumulated gradient will be applied to the center. The n-step methods [sutton] will be used to update the reward function, and it will be used to trade off the bias and variance problem.

5.6 DISTRIBUTIONAL DQN

Normally,bellman equation is used to calculate the expected reward value. But (Bellemare et al., 2017) introduces a different solution that uses the distributional perspective to update the Q-value function. Their motivation is to address the issue caused by the uncertainty of the system, which has multimodal distribution in action space.

In there work, they let $Z(s, a)$ be the return obtained by the current policy. Z is not a scalar but a distribution. Then we derived the distributional version of Bellman equation as follows: $Z(s, a) = r + \gamma Z(s', a')$.

This advanced proposal has been well used in practicecite (Bellemare et al., 2017),(Dabney et al., 2018), (Rowland et al., 2018). This method could lead to risk-aware behavior (Morimura et al., 2010) and leads to more robust learning in practice. The reason could be that the distributional perspective maybe provides more training signals than the previous method.

5.7 NOISY DQN

(Fortunato et al., 2017) proposal the Noisy Net. The noisy network could have the same structure as before, but its bias and weights could be perturbed iteratively. The motivations are to solve the limitations of exploring using ϵ-greedy policies. It usually will combine the deterministic and Gaussian noisy stream. In the implementation status, the randomized action-value function replaces the ϵ-greedy policy. Then, the MLP layers will be parameterized by the noisy network.

5.8 RAINBOW DQN

(Matteo et al., 2017) propose a solution that integrates all advantages of the seven methods, include DQN, double DQN, prioritized DDQN, dueling DDQN, multi-step, distributed DQN, and noisy DQN. Due to the ensemble of the seven methods, it called the Rainbow DQN. The combinations lead to a mighty learning algorithm, which is more stable, robust, and efficiency than any single method.

REFERENCES

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, O. P., and Zaremba, W. (2017). Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. (2017). A brief survey of deep reinforcement learning. *arXiv preprint arXiv:1708.05866*.

Bahdanau, D., Brakel, P., Xu, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. (2016). An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*.

Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier.

Bellemare, M. G., Dabney, W., and Munos, R. (2017). A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

Bellman, R. and Kalaba, R. (1962). Dynamic programming applied to control processes governed by general functional equations. *Proceedings of the National Academy of Sciences of the United States of America*, 48(10):1735.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2018). Distributional reinforcement learning with quantile regression. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. (2016). Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. (2017). Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*.

Fujimoto, S., van Hoof, H., and Meger, D. (2018). Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*.

Gandhi, D., Pinto, L., and Gupta, A. (2017). Learning to fly by crashing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3948–3955. IEEE.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Gordon, G. J. (1996). Stable fitted reinforcement learning. In *Advances in neural information processing systems*, pages 1052–1058.

Gordon, G. J. (1999). Approximate solutions to markov decision processes.

Gosavi, A. (2009). Reinforcement learning: A tutorial survey and recent advances. *INFORMS Journal on Computing*, 21(2):178–192.

Gu, S., Holly, E., Lillicrap, T., and Levine, S. (2017). Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3389–3396. IEEE.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

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*.

Matteo, H., Joseph, M., Hado, H., Tom, S., Georg, O., Will, D., Dan, H., Bilal, P., Mohammad, A., and David, S. (2017). Rainbow: Combining improvements in deep reinforcement learning. *arXiv: 1710.02298 v1 [cs. AI]*.

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*, pages 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., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Morimura, T., Sugiyama, M., Kashima, H., Hachiya, H., and Tanaka, T. (2010). Nonparametric return distribution approximation for reinforcement learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 799–806.

Pan, X., You, Y., Wang, Z., and Lu, C. (2017). Virtual to real reinforcement learning for autonomous driving. *arXiv preprint arXiv:1704.03952*.

Pinto, L., Andrychowicz, M., Welinder, P., Zaremba, W., and Abbeel, P. (2017). Asymmetric actor critic for image-based robot learning. *arXiv preprint arXiv:1710.06542*.

Puterman, M. L. (2014). *Markov Decision Processes.: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Riedmiller, M. (2005). Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer.

Rowland, M., Bellemare, M. G., Dabney, W., Munos, R., and Teh, Y. W. (2018). An analysis of categorical distributional reinforcement learning. *arXiv preprint arXiv:1802.08163*.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103.

Tsitsiklis, J. N. and Van Roy, B. (1997). Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, pages 1075–1081.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.

Vinyals, O., Babuschkin, I., Chung, J., Mathieu, M., Jaderberg, M., Czarnecki, W. M., Dudzik, A., Huang, A., Georgiev, P., Powell, R., et al. (2019). Alphastar: Mastering the real-time strategy game starcraft ii. *DeepMind Blog*.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*.

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.