The Advance of Reinforcement Learning and Deep Reinforcement Learning

Le Lyu[1,†], Yang Shen[*,2,†], Sicheng Zhang[3,†]
[1] Guangdong Shunde Desheng School (international), Guangdong, China, 528300
[2] Computer Science, Emory University, 201 Dowman Drive, Atlanta, GA, 30322
[3] College of Liberal Arts and Sciences, University of Florida, Gainesville, United States, 32611
[*]Corresponding author: alex.shen@emory.edu
[†]These authors contributed equally.

2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA) | 978-1-6654-1606-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/EEBDA53927.2022.9744760

***Abstract*—Reinforcement learning is one of the leading research fields of artificial intelligence. Unlike other machine learning methods, reinforcement learning is learning from the environment to action mappings. Thus, the chosen action could maximize the accumulated reward value from the environment and develop an optimal strategy via trial-and-error. In recent years, the achievements of deep reinforcement learning represented by AlphaGo have attracted wide attention from researchers. This paper first introduces the development of reinforcement learning, including classic reinforcement learning methods and deep reinforcement learning methods. Then, this paper discusses the advanced reinforcement learning work at present, including distributed deep reinforcement learning algorithms, deep reinforcement learning methods based on fuzzy theory, Large-Scale Study of Curiosity-Driven Learning, and so on. Finally, this essay discusses the challenges faced by reinforcement learning.**

Keywords—Reinforcement Learning, Deep Learning, Q-Learning, Distributed System, Neural Network

I. INTRODUCTION

Reinforcement learning means learning "what can be done to maximize the numerical benefit signal." The reinforcement learning model is not telling what actions should be taken but must determine which produces the richest benefits. Actions often affect not only immediate benefits but also the following situation, thus affecting subsequent benefits.

The study of reinforcement learning can be traced back to 1957. Bellman proposed the dynamic programming method to solve the optimal control problem [1]. This work used reinforcement learning to solve Markov Decision Process (MDP), which led to MDP becoming the most common form of defining reinforcement learning problems. Then in 1968, the monte carlo method was applied to reinforcement learning to predict action-value for the first time [2]. In 2006, Coulom raised the Monte Carlo Tree [3], which was implemented and defeated the European champion of go ten years later. In 1988, Sutton proposed the Temporal Difference (TD) algorithm, which combines the idea from Monte Carlo and dynamic programming [4]. TD can directly learn from the experience of the agent without a system model. It can also use the estimated value function to iterate, similar to dynamic programming [5].

Unlike the above works, Watkins proposed Q-learning as the earliest online reinforcement learning algorithm [6, 7]. Q-learning defines the performance function and iteratively learns the Q function by substituting the online data into the update formula. Other than the TD algorithm, Q-learning uses the reward of state-action pair and Q*(s, a) as the estimation function, so each behavior has to be examined in each learning iteration to ensure the learning process's convergence. This is the most effective model-independent reinforcement learning algorithm. It only needs to select the action by greedy strategy, without relying on the optimal strategy of the model. Similarly, SARSA learning proposed by Rummery and Niranjanis is also a temporal difference like Q-learning, which updates the value function through a time difference error [8]. Previous works where the selection of their actions is always based on the current value function to obtain a consistently deterministic strategy. However, their optimal strategy may be random. A slight change in the value function often led to the initially selected action not being selected. This change affects the convergence of the algorithm. Therefore, Horiuchi combined the Q-learning with the profit-sharing plan from the Genetic Algorithm (GA). Horiuchi proposed an exploitation-oriented Q-learning algorithm with a higher learning rate than Q-learning. However, the exploitation-oriented Q-learning algorithm cannot satisfy the convergence requirements due to the reinforcement of useless rules, coupled with its poor performance for the dynamic environment when adopting a significant state step back or alternative ruleset [9].

Another work line closely related to reinforcement learning is deep learning. Deep learning originated from artificial neural networks. In 1986, Geoffrey Hinton proposed the Back Propagation (BP) algorithm for optimizing multilayer neural networks [10]. LeCun applied the BP algorithm into a multilayer neural network in 1989, while the embryonic form of the neural network was formed until 1998 when LeCun proposed the model of LeNet-5 [11]. After that, recurrent neural networks (RNN) were developed in 1990. This is a powerful dynamic system that considers the relevance between data and the application of previous information into the computation of output of the current layer [12]. RNN is usually used to deal with the time-series database and for predictive text and speech recognition. In 2006, a deep belief network identified features, classified, and generation of data [13,14]. It is formed by restricted Boltzmann machines and extracting abstract features by learning the probability density distribution of data. In 2012, Convolutional Neural Network (CNN) attracted the attention of many researchers that the AlexNet was constructed by Hinton [15]. At the same time, this was the first time a GPU was applied to speed up the computation of the model. CNN's architecture is conducive to image recognition. It can extract more abstract features from original data with the

increasing number of network layers. The Visual Geometry Group (VGG) network is a typical deep convolutional network applied in large-scale image recognition tasks [16]. Besides, stacked auto-encoder, composed of several auto-encoders (AE) and obtains parameter weights of each layer auto-encoder by unsupervised training method, improves the performance of the deep neural network and therefore is applied into the pre-training of the area of image and speech.

The decision ability of reinforcement learning and perception ability of deep learning provide ideas for solving the perceptual decision problem of complex systems. By applying a neural network, the dimension of complex high-dimensional data is reduced and transformed into low-dimensional feature space for reinforcement learning processing. Combining two learning methods improves the ability of machine learning and presents a new research direction, deep reinforcement learning. In 2013, Mnih proposed the deep Q network (DQN), a deep reinforcement learning based on convolutional neural networks [17]. DQN combined Q learning and convolutional neural network ideas, which realized the end-to-end learning control and made a breakthrough in video games. At the same time, the efficiency of data use is increased, and the correlation between data is reduced while DQN repeatedly samples historical data. After that, DQN has been further developed. For example, the DQN with prioritized experience replay proposed by Schaul considers the importance of historical data and prioritized experience processing, thus improving the learning effect and speeding up the learning process [18]. Also, Guo has realized the real-time processing of Atari games by combining DQN and Monte Carol Tree Search (MCTS) algorithm [19]. Moreover, Double-DQN proposed by Van in 2015 effectively avoids the problem that the inherent estimation error of Q learning overestimates the value of the action with a large-scale of data [20].

In recent years, many remarkable achievements have been made in deep reinforcement learning. Table I shows the number of papers on deep reinforcement learning in recent years on the Web of Science.

TABLE I NUMBER OF PUBLISHED PAPERS ON DEEP REINFORCEMENT LEARNING IN RECENT YEARS

Years	2000~2010	2011~2015	2016	2017	2018	2019	2020
Numbers	75	118	191	446	1439	3096	4390

However, deep reinforcement learning also faces some bottlenecks such as generalization and robustness. This paper first reviews the development of reinforcement learning in the first section. Then, in the second section of this paper, four advanced research fields of reinforcement learning are introduced in detail, including the fuzzy Q-Learning and game theory based on reinforcement learning, distributed deep reinforcement learning algorithms, and so on. In addition, the challenges faced by reinforcement learning are further discussed in the third section. The fourth section summarizes this paper.

II. METHOD

This section introduces in detail four kinds of work of reinforcement learning, including (1) Fuzzy Q-Learning and Game Theory on traffic control. (2) Distributed Deep Q-Learning. (3) Tuning Computer Gaming Agents using Q-Learning. (4) Large-Scale Study of Curiosity-Driven Learning.

A. Fuzzy Q-Learning and Game Theory on Traffic Control

This paper proposed a hybrid fuzzy Q-learning and game theory method for traffic light control [21]. Meanwhile, reinforcement learning techniques to "let" each traffic light decide how much time they should have for each green light to significantly reduce traffic jams. Any agent would learn from experience and neighbor action through a reward system. Each agent would provide the input fuzzily and generate a decision by a fuzzy inference system.

The agents' goal is to make the optimum decision, and they do so via trial and error. The agents start with random values, and after each action, they would receive a tuple consisting of the current state, action, reward, and the next state, which the action leads to from the current state. Moreover, due to the dynamic situations at real-world traffic intersections, states and actions are stated continuously. Therefore, a Q table might not be possible, thus introducing fuzzy theory.

Consider each traffic light as a learning agent. There are two critical sources of information: the number of vehicles on the North-South and West-East routes based on its sensors and cameras (number of vehicles is fuzzified). The green time from its neighbor agent (the total time of green and red is fixed). Each agent uses these key points to schedule its own green time.

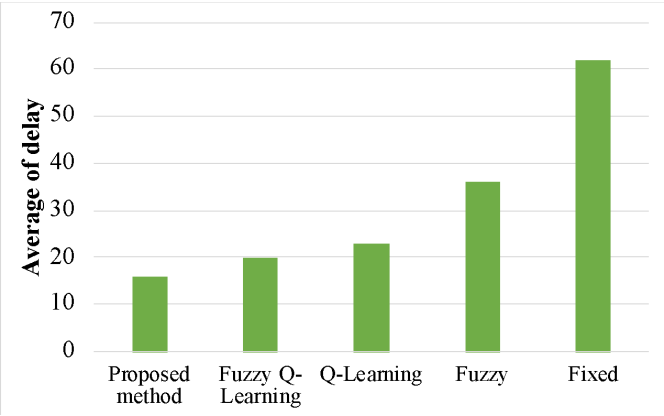

Figure 1. Average of delay for the proposed method, fixed time, fuzzy, Q-learning, fuzzy Q-learning [21]

Using fuzzy Q-learning, each agent tries to learn and decide its own green time. Each agent would receive a reward from its neighbor. The reward is combined with weighted functions of neighbor agents from up the learning algorithm. Compared with other methods and the fixed-time method, the simulation results show that a hybrid fuzzy Q-learning and game theory method dramatically reduces the average delay in traffic at intersections, as shown in Figure 1.

B. Distributed Deep Q-Learning

The distributed deep learning model is aimed at using multiple machines to process huge input data frames simultaneously [22]. To accomplish this, the agent is introduced to an observe-action-reward system to learn from experience, as shown in Figure 2.

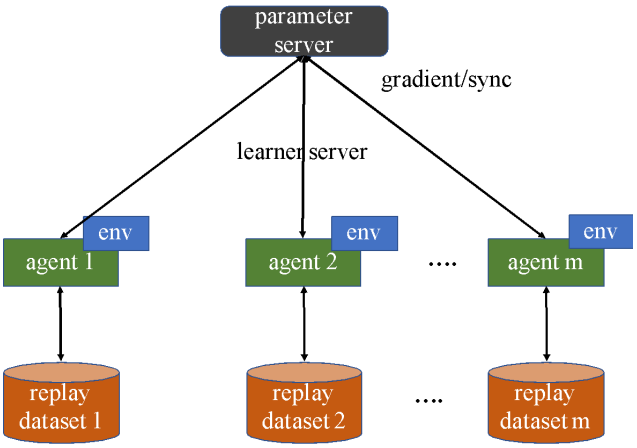

Figure 2. A variant of Downpour SGD adapted to deep Q-learning [22]

The algorithm tries to reduce the dimension of input data. It adopts experience replay to control the flow of reinforcement learning. reinforcement learning can be unstable and diverging, caused by correlations within observations, the impact on the policy from Q, and the correlations between Q and target values. With experience replay, the behavior distribution is averaged over many of its states, thus avoiding divergence or oscillations.

As for model parallelism, if the process was done on one machine, a strict upper bound on the model's size is made. Therefore, Caffe deep learning framework is used to compute gradients on each machine. The work done by a single node is parallelized across CPU or GPU cores. Suppose the model solely uses the GPU for all computation. In that case, the trade-off is between the acceleration of computation and the size of the model. The memory limit can often take place on low-end GPUs. On the other hand, CPU could hold a massive amount of data as well as the model.

The server needs to communicate with all workers and update each worker with the latest model. Each worker only communicates with the server about the gradient, but not with each other. Moreover, the number of workers is something worth considering. If increasing the number of workers, some problems might occur. The server can only perform a finite amount of updates per second. If the number of workers reaches a certain amount, no more improvements will occur as add workers. Moreover, as more frequently update the gradient, a gradient received by the server was likely from an outdated model.

C. Tuning Computer Gaming Agents using Q-Learning

Due to high demand for interesting, challenging, and sophisticated video games, the selection of Q-learning, a kind of reinforcement learning technique, to evolve the intelligent rots becomes the key to solving a series of problems, such as a waste of time, robot's fixed logic, easily identified robots and so on, appeared in the earliest game research and development [23]. Therefore, this paper examines how Q-learning evolves the computer game bots and how the robots learn from abstract models to achieve self-evolvement. In the experimentation, a simplified simulation of the FPS game is designed to test the efficiency of Q-learning in evolving the FPS game bots to learn to fight, plant the bomb, and learn for development when there is only limited guidance available for setting reinforcement learning parameters. Also, the inspiration for this simulation comes from the Frames Per Second (FPS) game Counter-strike. The Q-values learned from the experiment as shown in Table II.

TABLE II Q-TABLE FOR PLAN 0 AND 4 [23]

State	Attack	Ignore	State	Attack	Ignore
0 A n	14.72	19.33	4 A n	-0.10	0.00
0 A p	2.10	1.83	4 A p	0.00	0.00
0 B n	26.89	35.14	4 B n	-0.10	0.00
0 B p	41.38	32.92	4 B p	-0.10	0.00
0 C n	33.67	33.65	4 C n	-0.10	0.00
0 C p	41.02	33.10	4 C p	-0.10	0.00
0 D n	1.25	1.19	4 D n	0.00	0.00
0 D p	-0.09	1.89	4 D p	0.00	0.00
0 E n	18.54	18.47	4 E n	-0.10	0.00
0 E p	34.33	23.51	4 E p	-0.10	0.00
0 F n	19.58	19.57	4 F n	-0.10	0.00
0 F p	19.99	19.98	4 F p	-0.10	0.00
0 G n	1.47	1.58	4 G n	0.00	0.00
0 G p	2.26	1.82	4 G p	0.00	0.00
0 H n	0.00	0.00	4 H n	0.00	0.00
0 H p	0.00	0.00	4 H p	0.00	0.00

This work developed a scaled-down abstraction of Counter-strike in Java and simulated the bots in this environment. They only implemented the Q-learning algorithm for the green agents. However, they kept the blue agents following a hard-coded plan. The agents with a Q-learning algorithm select the best action based on its current state argmax a Q (s, a). Every action the agent takes affects the current state and generates a reward or a punishment that promotes the agents to learn. In addition, the agent uses the enemy present of size two as the state. In total, there are 96 states, and the agent will choose to perform in one of the Attacks (0) and Ignore (1). After making decisions, the agents update the Q-table. As a result, an agent's reward is known in a future state, and the exploration rate is similar to the epsilon greedy. After determining the right combination of different parameters in the Q-learning algorithm, the experiment begins with training the bots with a small number of abstract states of the superset of the more detailed states in an actual game. Then in the experiment, the fitness of agents is measured, and the reward function is modified.

Overall, the evolved bots can learn and perform better and more based on their rewards compared with the hard-coded ones. Also, training the bots with a small number of states can help them compete better for a large number of states. In the experiment, the bots with a Q-learning algorithm can learn to attack or ignore the enemy according to different factors, such as location, plan, enemy presence, and so on, and select the actions based on the utility values. Also, the agents can learn from the methods proposed by humans and select the plan from past experiences by considering the utilities for using that plan. Combining all these learning techniques, the learning speed of agents can be improved a lot to increase efficiency. Lastly, this paper mentions that what they need to test in the future is applying this approach inaccurate game simulation and the competition with human players.

D. Large-Scale Study of Curiosity-Driven Learning

Curiosity-driven learning, an intrinsic reward function using prediction error as a reward signal, is used to test the effect of using different feature spaces and show the limitations of the prediction-based rewards [24]. Due to the high challenging work related to applying extrinsic reward function in reinforcement learning, the denser and more well-shaped intrinsic reward function is introduced to solve this

problem and guide the agent to find the next reward itself. In the experiment, a large scale of curiosity-driven explorations is investigated through various environments. Also, the feature spaces and the limitations of such methods are evaluated to conclude. In this paper, a new term, feature space, is introduced: A suitable feature space with compact, sufficient, and stable characteristics can promote the forward dynamics model, making predictions efficiently. In order to achieve this goal, this paper compares four methods: Pixels, Random Features (RF), Variational Autoencoders (VAE), and Inverse Dynamics Features (IDF). Curiosity-driven learning without extrinsic rewards is begun to test what happens, the effect of the different feature learning variants, and the agents' behavior in different learning environments or games by comparing four feature learning methods. Figure 3 shows that the 54 environments investigated in this work.

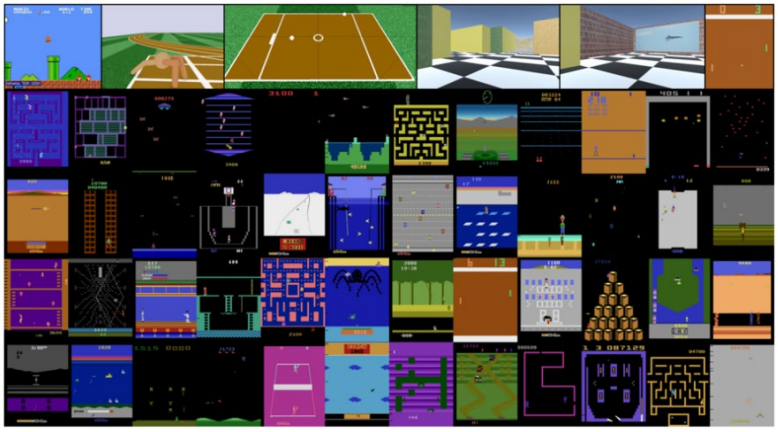

Figure 3. A snapshot of the 54 environments

Experimental results show that curious agents can effectively learn and explore valuable skills, so the next goal is to "utilize abundant 'unlabeled' environments without reward functions by showing generalization to novel environments," again, by comparing different feature learning methods in the game Super Mario Bros. In the third experiment, which is the opposite of the first one, the researchers evaluate the effectiveness of curiosity with a sparse external reward in helping the agents work on a task and compare the result with the previous works. In addition, this work evaluates the limitation of prediction error-based curiosity, like handling stochastic dynamics and the problem of random transitions in the environment.

III. Prospects and Challenges

Deep reinforcement learning needs loads of training steps to be at a comparable level with human performance. Deep reinforcement learning can theoretically solve any real-world problem. However, the unknown model takes tons of samples to train, adding more drag to efficiency.

One of the critical features of reinforcement learning is the reward system, but setting up a proper reward is trickier than intuition. First, the reward function needs to be set (in most cases), and setting a universal reward function for the complete training can be challenging. It needs to be precise and concise at the same time. There are two types of reward: shaped reward and sparse reward. The scant reward is only given at the goal state, meaning that the agent can only know if it receives a reward after the whole process. Take maze as an example. The goal is to find the shortest route, but sparse reward only tells the model if the model has taken the correct route at the finish line, which could be time-consuming for the agent.

In contrast, the shaped reward constantly informs the agent about each decision it makes throughout. The problem of shaped reward is that if not designed carefully, it could bias the results. For example, the agent is playing a racing game. Reaching the finishing line is rewarding, and the less time it takes, the bigger the reward. However, drifting along the way also adds points. Therefore, drifting also comes with positive rewards. Moreover, suppose the reward function is not designed correctly. In that case, the agent could keep drifting the car but ignoring that the priority is to finish quickly.

In addition, deep reinforcement learning can build up a model that fits that specific environment crazily well but is not any good for other environments. The deep RL models are environment-specific and usually not transferable, which makes researchers sometimes think it is not worth the while and go more alternative methods.

IV. Conclusion

This paper reviews the development of reinforcement learning and deep reinforcement learning. It then introduces four frontier studies of reinforcement learning, including (2) Distributed Deep Q-Learning. (3) Tuning Computer Gaming Agents using Q-Learning. (4) Large-Scale Study of Curiosity-Driven Learning. This paper further discusses the current challenges of reinforcement learning. It has been argued that reinforcement learning training and transfer are challenges that need to be continuously paid attention to and solved.

References

[1] BELLMAN R. Dynamic programming and Lagrange multipliers [J]. Proceedings of the National Academy of Sciences, 1956, 42(10): 767-769.

[2] MICHIE D, CHAMBERS R A. BOXES: An experiment in adaptive control [J]. Machine Intelligence, 1968, 2(2): 137 – 152.

[3] COULOM R. Efficient selectivity and backup operators in Monte-Carlo tree search [M] //Computers and Games. Berlin Heidelberg: Springer, 2006: 72 – 83.

[4] SUTTON R S, BARTO A G. Reinforcement Learning: An Introduction [M]. Cambridge MA: MIT Press, 1998.

[5] SUTTON R S. Learning to predict by the methods of temporal differences [J]. Machine Learning, 1988, 3(1): 9 – 44.

[6] WATKINS C J C H. Learning from delayed rewards [D]. Cambridge: University of Cambridge, 1989.

[7] Tsitsiklis, John N. Asynchronous stochastic approximation and Q-learning. Machine Learning, 1988, 3:9-44

[8] RUMMERY G A, NIRANJAN M. On-Line Q-Learning Using Connectionist Systems [M]. Cambridge: University of Cambridge, Department of Engineering, 1994.

[9] Horiuchi T, Katai O. Q-PSP learning: An exploitation-oriented Q-learning algorithm and its applications. Transactions of the Society of Instrument and Control Engineers, 1999, 39(5): 645-653.

[10] DE Rumelhart, Hinton G E, Williams R J . Learning Representations by Back Propagating Errors[J]. Nature, 1986, 323(6088):533-536.

[11] LeCun, Y, Bottou, L, Bengio, Y, Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278 - 2324.

[12] Elman J L. Finding Structure in Time[J]. Cognitive Science, 1990, 14(2):179-211.

[13] Hinton, G, E, et al. Reducing the Dimensionality of Data with Neural Networks.[J]. Science, 2006.

[14] Hinton G E, Osindero S , Teh Y W. A Fast Learning Algorithm for Deep Belief Nets[J]. Neural Computation, 2014, 18(7):1527-1554.

[15] Krizhevsky A, Sutskever I, Hinton G E. ImageNet Classification with Deep Convolutional Neural Networks. pp. 1–9.

[16] SIMONYAN K, ZISSERMAN A. Very deep convolutional net-works for large-scale image recognition [EB/OL] //arXiv preprint. 2015. arXiv:1409.1556[cs.CV].

[17] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning [C] //Proceedings of the NIPS Work- shop on Deep Learning. Lake Tahoe: MIT Press, 2013.

[18] SCHAUL T, QUAN J, ANTONOGIOU I, et al. Prioritized experience replay [C] //Proceedings of the International Conference on Learning Representations. San Juan: ACM, IEEE, 2016.

[19] GUO X, SINGH S, LEE H, et al. Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning [C] //Advances in Neural Information Processing Systems. Montreal: MIT Press, 2014: 3338 – 3346.

[20] VAN H H, GUEZ A, SILVER D. Deep reinforcement learning with double Q-learning [C] //Proceedings of the 30th AAAI Conference on Artificial Intelligence. Phoenix: AAAI, 2016: 1813 – 1819.

[21] Daeichian, A., & Haghani, A. (2018). Fuzzy q-learning-based multi-agent system for intelligent traffic control by a game theory approach. Arabian Journal for Science and Engineering, 43(6), 3241-3247.

[22] Ong, H. Y., Chavez, K., & Hong, A. (2015). Distributed deep Q-learning. arXiv preprint arXiv:1508.04186.

[23] Patel, P. G., Carver, N., & Rahimi, S. (2011, September). Tuning computer gaming agents using q-learning. In 2011 Federated Conference on Computer Science and Information Systems (FedCSIS) (pp. 581-588). IEEE.

[24] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., & Efros, A. A. (2018). Large-scale study of curiosity-driven learning. arXiv preprint arXiv:1808.04355.