Weighted Double Deep Multiagent Reinforcement Learning in Stochastic Cooperative Environments

Yan Zheng[1], Zhaopeng Meng[1,2], Jianye Hao[1(✉)], and Zongzhang Zhang[3]

[1] Tianjin University, Tianjin, China
jianye.hao@tju.edu.cn
[2] Tianjin University of Traditional Chinese Medicine, Tianjin, China
[3] Soochow University, Suzhou, China

Abstract. Recently, multiagent deep reinforcement learning (DRL) has received increasingly wide attention. Existing multiagent DRL algorithms are inefficient when faced with the non-stationarity due to agents update their policies simultaneously in stochastic cooperative environments. This paper extends the recently proposed weighted double estimator to the multiagent domain and propose a multiagent DRL framework, named weighted double deep Q-network (WDDQN). By utilizing the weighted double estimator and the deep neural network, WDDQN can not only reduce the bias effectively but also be extended to scenarios with raw visual inputs. To achieve efficient cooperation in the multiagent domain, we introduce the lenient reward network and the scheduled replay strategy. Experiments show that WDDQN outperforms the existing DRL and multiagent DRL algorithms, i.e., double DQN and lenient Q-learning, in terms of the average reward and the convergence rate in stochastic cooperative environments.

Keywords: Multiagent learning · Deep learning · Weighted Q-learning

1 Introduction

The goal of reinforcement learning (RL) is to learn an optimal behavior within an unknown dynamic environment, usually modeled as a Markov decision process (MDP), through trial and error [11]. By far, deep RL (DRL) has achieved great successes in mastering various complex problems [6], which can be credited to the *experience replay* and *target network* [6,10].

In multiagent domains, approaches like [5,13] have been proposed by extending Q-learning to address the coordination problems in cooperative multiagent systems (MAS). However, they are only verified using relative simple problems. Recently, employing DRL in MAS draws wide attention [3,4,7]. These algorithms, however, still suffer from two intrinsic difficulties: stochasticity due to the noisy reward signals; and non-stationarity due to the dynamicity of coexisting agents. The stochasticity introduces additional biases in estimation,

X. Geng and B.-H. Kang (Eds.): PRICAI 2018, LNAI 11013, pp. 421–429, 2018.
https://doi.org/10.1007/978-3-319-97310-4_48

while the non-stationarity harms the effectiveness of experience replay, which is crucial for stabilizing deep Q-networks. These two characteristics result in the lack of theoretical convergence guarantees and amplify the difficulty of finding the optimal Nash equilibriums, especially in cooperative multiagent problems.

This work focuses on learning algorithms of independent learners (ILs) in cooperative MAS. In such setting, agents are unable to observe other agents' actions and rewards [2], share a common reward function and learn to maximize the common expected discounted reward (a.k.a. return). To handle the stochastic and non-stationary challenges in MAS, we propose the weighted double deep Q-network (WDDQN) with two auxiliary mechanisms, the *lenient reward network* and the *scheduled replay strategy*, to help ILs in finding the optimal policy, maximizing the common return.

Our contributions are three-fold. First, we extend weighted double Q-learning (WDQ) [14], a state-of-the-art traditional RL method, to the multiagent DRL settings. Second, we introduce lenient reward network inspired by the lenient Q-learning [7,8]. Third, we propose a scheduled replay strategy to stabilize and speed up the learning process in complex multiagent problems with raw visual inputs. Empirical results demonstrate that on a fully cooperative multiagent problem, WDDQN with new mechanisms contribute to increasing the algorithm's convergence, decreasing the instability and helping ILs to find an optimal policy simultaneously.

2 Preliminaries

Weighted Double Q-Learning (WDQ). [14] uses a dynamic heuristic value β to balance between the overestimation of the single estimator and the underestimation of the double estimator during the iterative Q-value update process:

$$Q(s,a)^{U,WDQ} = \beta Q^U(s,a^*) + (1-\beta)Q^V(s,a^*), \tag{1}$$

where a linear combination of two estimators Q^U and Q^V is used for updating Q-value. When a^* is chosen by $Q^U, i.e., a^* \in \arg\max_a Q^U(s,a)$, $Q^U(s,a^*)$ will be positively biased and $Q^V(s,a^*)$ will be negatively biased, and vice versa. $\beta \in [0,1]$ balances between the positive and negative biases.

Lenient Q-Learning. [9] updates the policies of multiple agents towards an optimal joint policy simultaneously by letting each agent adopt an optimistic dispose at the initial exploration phase. This contributes to discovery the optimal joint policy and has been empirically verified in [7,8,13]. To be specific, during training, lenient agents keep track of the temperature $T_t(s,a)$ for each state-action pair (s,a) at time t, which is initially set to a defined maximum temperature value and used for measuring the leniency $l(s,a)$ as follows:

$$l(s_t;a_t) = 1 - e^{-K*T_t(s_t,a_t)}, \tag{2}$$

where K is a constant determining how the temperature affects the decay in leniency. As suggested by [13], $T_t(s_t, a_t)$ is decayed using a discount factor $\kappa \in [0, 1]$ and $T_{t+1}(s_t, a_t) = \kappa T_t(s_t, a_t)$. Given the TD error $\delta = Y_t^Q - Q_t(s_t, a_t; \theta_t)$, the iterative update formula of lenient Q-learning is defined as follows:

$$Q(s_t, a_t) = \begin{cases} Q(s_t, a_t) + \alpha\delta & \text{if } \delta > 0 \text{ or } x > l(s_t, a_t), \\ Q(s_t, a_t) & \text{otherwise.} \end{cases} \tag{3}$$

The random variable $x \sim U(0, 1)$ is used to ensure that a negative update δ is performed with a probability $1 - l(s_t, a_t)$. We absorb this interesting notion of forgiveness into our lenient reward network to boost the convergence in cooperative Markov games which will be explained later.

3 Weighted Double Deep Q-Networks

In the section, we introduce a new multiagent DRL algorithm, weighted double deep Q-networks (WDDQN), with two auxiliary mechanisms, i.e., the lenient reward approximation and the scheduled replay strategy, to achieve efficient coordination in stochastic multiagent environments, where reward could be extremely stochastic due to the environments' inherent characteristics and the continuous change of the coexisting agents' behaviors. For the stochastic environments, WDDQN uses the combination of the weighted double estimator and a reward approximator to reduce the estimation error. As for the non-stationary coexisting agents, we incorporate the notion of leniency [7,8] into the reward approximator to provide an optimistic estimation of the expected reward under each state-action pair $r(s, a)$. Besides, directly applying prioritized experience replay [10] in multiagent DRL leads to poor performance, as stored transitions can become outdated because agents update their policies simultaneously. To address this, we propose a scheduled replay strategy to enhance the benefit of prioritization by adjusting the priority for transition sample dynamically. In the remainder of this section, we will describe these facets in details.

WDDQN is adapted from WDQ by leveraging the neural network as the Q-value approximator to handle problems with high-dimensional state spaces. The network architecture is depicted in Fig. 1. To reduce the estimation bias, the combination of two estimators, represented as Deep Q-networks Q^U and Q^V with the same architecture, is adopted to select action $a = \max_{a'} \frac{Q^U(s,a') + Q^V(s,a')}{2}$. Besides, the target $Q^{\texttt{Target}}(s, a)$ used for Q-value updating in back-propagation is replaced with a weighted combination like Eq. 1, balancing between the overestimation and underestimation. Besides, a lenient reward approximator and an efficient scheduled replay strategy are incorporated in WDDQN to achieve bias reduction and efficient coordination in multiagent stochastic environments.

Lenient Reward Network (LRN) is a neural network estimator to explicitly approximate the reward function $R(s, a)$, being conducive to reduce noise in

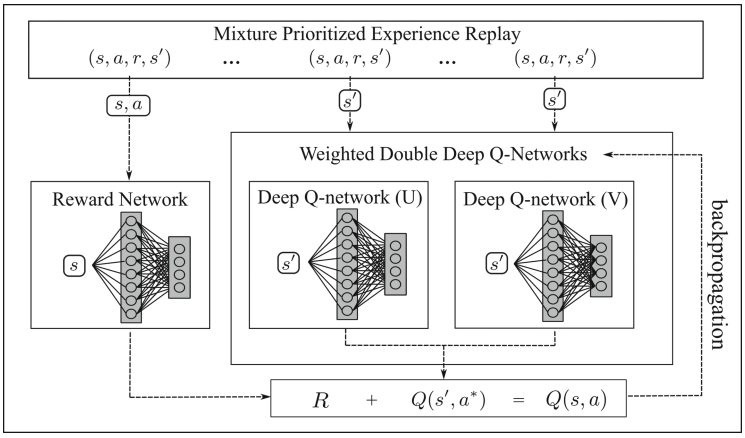


Fig. 1. Network architecture of WDDQN

stochastic rewards. LRN can reduce bias in immediate reward r yielded from stochastic environments by averaging all rewards for distinct (s, a) pair and be trained using the transitions stored in the experience replay during the online interaction. Instead of using the reward r in transition (s, a, r, s') from experience memory, WDDQN uses the estimated reward by the reward network as shown in Fig. 1.

In addition to stochasticity, in a cooperative MAS, the coexisting agents introduce additional bias to r as well. The mis-coordination may lower the reward r for (s, a^*) although the agent has adopted the optimal action. To address this, LRN draws on the lenient concept in [9], making the agent keep optimistic during the initial exploration phase. LRN is updated periodically as follows:

$$R_{t+1}(s_t, a_t) = \begin{cases} R_t(s_t, a_t) + \alpha\delta & \text{if } \delta > 0 \text{ or } x < l(s_t, a_t), \\ R_t(s_t, a_t) & \text{otherwise.} \end{cases} \tag{4}$$

where $R_t(s_t, a_t)$ is the reward approximation of state s and action a at time t, and $\delta = \bar{r}_t^{(s,a)} - R(s_t, a_t)$ is the TD error between the $R_t(s_t, a_t)$ and the target reward $\bar{r}_t^{(s,a)} = 1/n \sum_{i=1\ldots n} r_i^{(s,a)}$ obtained by averaging all immediate reward $r_i^{(s,a)}$ of (s, a) pairs in experience memory. Note that $l(s_t, a_t)$ inherits from Eq. 3, and gradually decayed each time a state-action (s, a) pair is visited. Consequently, the LRN contributes to reduce bias by reward approximation and can help agents to find optimal joint policies in cooperative Markov games.

Scheduled Replay Strategy (SRS) is a new strategy adapted from the prioritized experience replay (PER), selecting vital samples to replay in stochastic multiagent environments. In vanilla PER, the probability of a sample being chosen for training is proportional to its TD error. However, in stochastic multiagent environments, due to the noisy reward and the continuous behavior changes of

coexisting agents, the vanilla PER may deteriorate the algorithm's convergence and perform poorly. Given a transition (s, a, r, s, d) with an extremely biased reward r, PER will treat it as an important sample for its large TD error and will frequently select it for update the network, though it is incorrect due to the big noise in r. To address this, we replace r with an estimation $R^N(s, a)$ using LRN to correct TD error, by which the PER can distinguish true important samples.

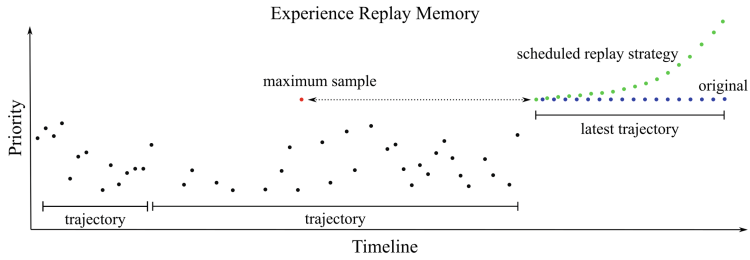


Fig. 2. Comparison between the prioritized experience replay and the scheduled replay strategy: each dot represents a sample (s, a, r, s), and a trajectory consists of an ordered sequence of samples. The x-axis represents the order that each sample comes into the relay memory and the y-axis is the priority of each sample. (Color figure online)

Another potential problem is that PER gives all samples in the new trajectory the same priority, thus resulting in the indistinguishability of importance for all new samples. To be specific, in Fig. 2, the sample with the maximum priority is colored by red dot. PER gives all samples (blue dots) in the latest trajectory with an identical priority[1]. However, in cooperative multiagent environments, the trajectories that agents succeed in cooperation are relatively rare, and in these trajectories, the samples closer to the terminal state is even more valuable than the ones far from the terminal state. Besides, the $Q(s, a) = r + Q(s', a^*)$ far from the terminal state can further deteriorate if bootstrap of action value $Q(s', a^*)$ is already highly inaccurate, since inaccurate estimation will propagate throughout the whole contiguous samples.

These two traits explain why samples close to the terminal state should be frequently used for network training. To this end, we propose the scheduled replay strategy, using a precomputed rising schedule $[w_0, w_1, ..., w_n]$ with size n to assign different priorities according to the sample's position i in the trajectory. The values for $w_i = e^{\rho_c * u^i}$ are computed using an exponent ρ^c which grows with a rising rate $u > 1$ for each i, $0 \leqslant i < n$. The priority p_i assigned to sample with index i is obtained by multiplying the current maximum priority $p_{\max}$ in experience memory (priority of the red point in Fig. 2) by w_i:

$$p_i = p_{\max} \times w_i$$

[1] See OpenAI source code for details: https://github.com/openai/baselines.

The SRS assigns a higher priority to samples near the terminal state (the green dot in Fig. 2) to ensure they are more likely to be sampled for network training. In this way, the estimation bias of the $Q(s, a)$ near the terminal state is expected to decrease rapidly. This can significantly speed up the convergence and improve the training performance, as to be verified in the following section.

4 Experiments

Pacman Game is an $n \times n$ gridworld problem (Fig. 3), where the agent starts at the s_0 (top left cell) and moves towards the goal (pink dot) using four actions: {north, south, east, west}. Every action leads the agent to move one cell in the corresponding direction, and the goal appears randomly in any position. A stochastic reward of -30 or 40 is received with equal probability for any action entering into the goal. Moving north or west will get a reward of -10 or $+6$, and south or east get -8 or $+6$ at a non-goal state. The environment is extremely noisy due to the uncertainty in the reward function.

Fig. 3. Pacman game (Color figure online)

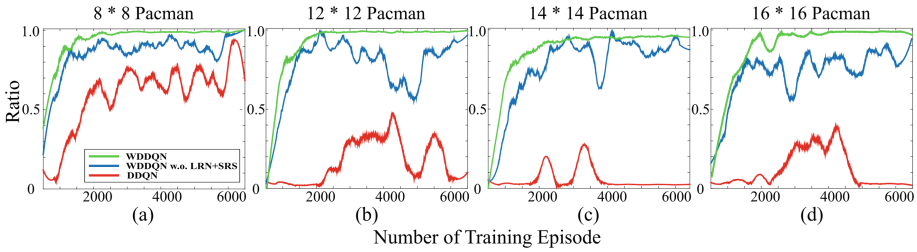


Fig. 4. Comparisons of double DQN (DDQN), WDDQN with/without LRN and SRS, denoted by WDDQN and WDDQN w.o. LRN+SRS, on Pacman with 4 different sizes. The X-axis is the number of training episodes and the Y-axis is a ratio of the number of minimum steps to the goal to the number of steps that the agent actually used during training.

As shown in Fig. 4, under extremely stochastic environments, DDQN takes a long time to optimize the policy, while WDDQN and WDDQN w.o. LRN+SRS learn fast due to the weighted double estimator. DDQN and WDDQN w.o. LRN+SRS oscillate too frequently to converge, while WDDQN performs steadily and smoothly due to LRN. Another finding is that the training speed of WDDQN is faster than the others, which is attributed to the SRS. In general, WDQ works not as well as in relatively simple RL problems and both DDQN and WDDQN w.o. LRN+SRS may not converge even after a very long training time. By contrast, WDDQN learns efficiently and steadily due to the LRN and SRS.

Predator Game is a more complex cooperative problem adapted from [1], where two agents (robots) try to enter into the goal state (G or S) at the same time to achieve coordination. S is a suboptimal goal with +10 reward while G is a global optimal with +80 reward. A thick wall (in gray) in the middle separates the area into two zones. Different from settings in Pacman, a reward of 0 is received whenever entering into a non-goal state and a reward of −1 is received as punishment for miscoordination. We investigate whether WDDQN and related algorithms can find cooperative policies moving towards the S (suboptimal) or G (optimal), especially the optimal policy (Fig. 5).

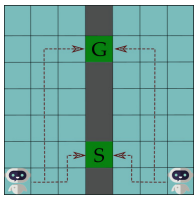
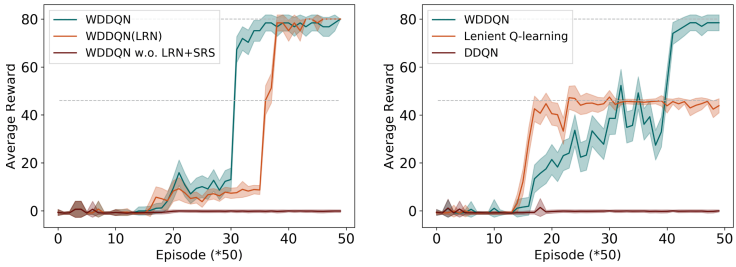


Fig. 5. Predator game



(a) deterministic rewards. (b) stochastic rewards.

Fig. 6. (Left) Comparisons of WDDQN and its variants using the Predator game with deterministic rewards; and (right) comparisons of WDDQN and other algorithms using the Predator game with stochastic rewards. Note that, each point in the x-axis consists of 50 episodes, the y-axis is the corresponding averaged reward, and the shadow area ranges from the lowest to the highest rewards.

Evaluation on WDDQN. Comparison of WDDQN, WDDQN(LRN), which uses only LRN, and WDDQN w.o. LRN+SRS in terms of the average reward is conducted and depicted in Fig. 6(a). As the learning convergence is no longer guaranteed in multiagent domains, WDDQN w.o. LRN+SRS's thus fails in finding the cooperative policy by directly combining WDQ with the neural network. By contrast, WDDQN(LRN), due to the LRN, achieves coordination more quickly and efficiently finds the optimal policy. Moreover, by leveraging the SRS, WDDQN learns the optimal policy much faster than the two others.

Evaluation Against Other Algorithms. Comparisons of WDDQN, DDQN [12] and lenient Q-learning [7] is conducted under the same settings except that the agent receives a reward of +10 or +100 with the possibility of 60% or 40% at goal S and a deterministic reward of +80 at goal G. S is still suboptimal as its average reward is 46. This stochasticity may mislead the agent to converge

to the suboptimal goal where a higher reward may appear accidentally. Results in terms of the average reward are depicted in Fig. 6(b), where two dashed lines indicate optimal (80) and suboptimal (46) solutions. Both WDDQN and lenient Q-learning outperform DDQN in terms of the convergence speed and the average reward in all experiments, which confirms the infeasibility of directly applying DRL algorithms in multiagent problems. Note that, WDDQN, due to the LRN and SRS, is more stable, performs better and is more likely to find the optimal solution than lenient Q-learning in such a stochastic environment.

5 Conclusion

We propose WDDQN with the lenient reward network and the scheduled replay strategy to boost the training efficiency, stability and convergence under stochastic multiagent environments with raw image inputs, stochastic rewards, and large state spaces. Empirically, WDDQN achieves promising performance in terms of the average reward and convergence rate on two stochastic environments.

Acknowledgments. The work is supported by the National Natural Science Foundation of China under Grant No.: 61702362, Special Program of Artificial Intelligence of Tianjin Municipal Science and Technology Commission (No.: 569 17ZXRGGX00150) and Science and Technology Program of Tianjin, China (Grant Nos. 15PTCYSY00030 and 16ZXHLGX00170).

References

1. Benda, M., Jagannathan, V., Dodhiawala, R.: On optimal cooperation of knowledge sources - an empirical investigation. Technical report BCS-G2010-28, Boeing Advanced Technology Center, Boeing Computing Services (1986)
2. Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. In: AAAI Conference on Artificial Intelligence (AAAI), pp. 746–752 (1998)
3. Gupta, J.K., Egorov, M., Kochenderfer, M.: Cooperative multi-agent control using deep reinforcement learning. In: Sukthankar, G., Rodriguez-Aguilar, J.A. (eds.) AAMAS 2017. LNCS (LNAI), vol. 10642, pp. 66–83. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-71682-4_5
4. Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Perolat, J., Silver, D., Graepel, T., et al.: A unified game-theoretic approach to multiagent reinforcement learning. In: Advances in Neural Information Processing Systems (NIPS), pp. 4193–4206 (2017)
5. Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. Knowl. Eng. Rev. **27**(1), 1–31 (2012)
6. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)

7. Palmer, G., Tuyls, K., Bloembergen, D., Savani, R.: Lenient multi-agent deep reinforcement learning. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (2018, to appear)
8. Panait, L., Sullivan, K., Luke, S.: Lenient learners in cooperative multiagent systems. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (2006)
9. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-58484-6_269
10. Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. In: International Conference on Learning Representations (ICLR) (2016)
11. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press, Cambridge (1998)
12. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double Q-learning. In: AAAI Conference on Artificial Intelligence (AAAI), pp. 2094–2100 (2016)
13. Wei, E., Luke, S.: Lenient learning in independent-learner stochastic cooperative games. J. Mach. Learn. Res. **17**(84), 1–42 (2016)
14. Zhang, Z., Pan, Z., Kochenderfer, M.J.: Weighted double Q-learning. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 3455–3461 (2017)