MARL-PPS: Multi-agent Reinforcement Learning with Periodic Parameter Sharing

Extended Abstract

Safa Cicek
UCLA, Los Angeles, United States
safacicek@ucla.edu

Alireza Nakhaei
Honda Research Institute, Mountain View, United States
anakhaei@honda-ri.com

Stefano Soatto
UCLA, Los Angeles, United States
soatto@ucla.edu

Kikuo Fujimura
Honda Research Institute, Mountain View, United States
kfujimura@honda-ri.com

ABSTRACT

We present a multi-agent reinforcement learning algorithm that is a simple, yet effective modification of a known algorithm. External agents are modeled as a time-varying environment, whose policy parameters are updated periodically at a slower rate than the planner to make learning stable and more efficient. Replay buffer, which is used to store the experiences, is also reset with the same large period to draw samples from a fixed environment. This enables us to address challenging cooperative control problems in highway navigation. The resulting Multi-agent Reinforcement Learning with Periodic Parameter Sharing (MARL-PPS) algorithm outperforms the baselines in multi-agent highway scenarios we tested.

ACM Reference Format:
Safa Cicek, Alireza Nakhaei, Stefano Soatto, and Kikuo Fujimura. 2019. MARL-PPS: Multi-agent Reinforcement Learning with Periodic Parameter Sharing. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

Our goal is to learn a long-term policy for highway driving where vehicles travel in the same direction at high speed and are expected to cooperate. In Fig. 1-left, the green vehicle can avoid the obstacle in black without changing speed only with the cooperation of the red vehicle. To enable cooperation, the red vehicle must maintain a predictive model of the green one (See Fig. 1-right). If during training, the policy of other agents changes rapidly relative to the update of the ego-vehicle's plan, one cannot learn an accurate prediction of future states, and therefore an optimal policy. In this work, we focus on this "moving environment" problem.

We aim to design planning algorithms to address these cases. A natural tool to this end, in the absence of expert data, is reinforcement learning (RL). In particular, Deep Q-Networks (DQN) [9], introduced for single-agent cases, have been extended to multi-agent reinforcement learning (MARL) in high-dimensional observation spaces [4, 13].

To train and evaluate MARL-PPS, we developed a highway simulator where each vehicle has limited ego-centric observations (See Fig. 2). Observations are given in the form of an occupancy grid, a representation suitable for ingestion by a convolutional neural network (CNN). Vehicles can observe 100m ahead and behind in the horizontal axis. In the vertical axis, agents observe 1.5 lanes from its center. All the vehicles are modeled with the second order Dubin motion model [6]. Agents do not have any knowledge of the global map such as the number of lanes, current lane, distance to the closest lane or road width. The sides of the highway, static obstacles and vehicles are all represented as occupied areas in the observation, making the occupancy grid a binary map. The ego-vehicle state is also fed to a fully connected layer for preprocessing. Outputs of the fully connected layer and the CNN are concatenated and fed to a Long short-term memory (LSTM) [3]. [1] Its output is processed with another fully connected layer to map into space whose dimension is the same as the number of actions. This map gives the Q-value predictions for each action. As a final step, the epsilon-greedy block chooses the action (e.g. acceleration and steering velocity) to execute.

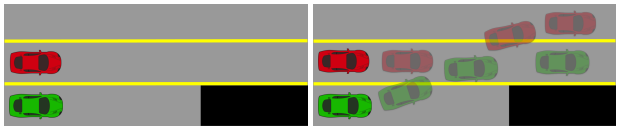

Figure 1: Left: initial positions of two vehicles. The black rectangle is a static obstacle to be avoided. Right: the red vehicle must maintain a predictive model of the green one for cooperative behaviour.

2 MARL-PPS

Given the samples from the simulator, minimizing this DQN objective can be considered as supervised learning (SL). But, unlike SL, the target is moving within training for the given observation o_t either because of (i) moving target network parameter or (ii) moving next observation o_{t+1}. The first problem is tackled in [9] by updating target parameters with large period T_t. The instability of o_{t+1} given o_t is because of the changing policies of other agents. To handle this problem, we suggest to apply a similar idea: We update the policies of other agents (non-ego agents) with the policy of the

[1] Since each agent has limited observation range, the sequential decision-making process can be formulated with POMDP. Hence, we are using LSTM as a function approximator as in [2].

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.

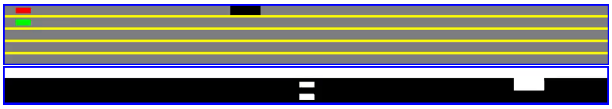

Figure 2: Screenshots are from the highway simulator RL agents are trained on. The top panel is the scene with two vehicles (red and green rectangles) and a static obstacle to be avoided in the top lane (black rectangle). The bottom panel shows the observation of the red vehicle.

ego-agent periodically with a fixed number of iterations $T_n \gg 1$. The period T_n should be chosen large enough to keep the training objective stable until the ego-agent can optimize for it.[2] Within T_n iterations, the target network of the ego-agent should also be updated many times for both the ego Q-network and the ego target Q-network to converge. One can consider the T_n iterations in which non-ego agent policies kept fixed, as one stage of learning. So, within that stage, the target network parameters are updated many times to optimize for that particular environment i.e. the period at which the ego-target network is updated (T_t) is chosen to be smaller than T_n.

Experience replay introduced by [9] was essential to solve the moving target problem in fitted Q-learning approaches. However, in the multi-agent setting, the use of the replay buffer becomes problematic due to the changes in the policies of other agents. When the replay buffer has samples from different policies of other agents, it gets harder for the ego-agent to fit a particular environment. Hence, we choose to reset the replay buffer whenever we update other agent policies to make sure that samples are drawn for the same policy of other agents. So, the replay buffer is reset at every T_n time steps.

Interaction-*unaware* reward for the motion planning task penalizes collision with either of the static obstacles, highway edges or other agents. It also encourages driving at the desirable speed with desirable direction. Finally, small jerk is encouraged. To make agents cooperative, we include the reward of other agents into the reward that ego-agent is maximizing for. This interaction-*aware* reward for the ego-agent at time t can be written as $r_{1,t} = \tilde{r}_{1,t} + \lambda_{\text{coop}} \sum_{j=2}^{J} \tilde{r}_{j,t}$ where λ_{coop} is the cooperativeness measure of the agent and $\tilde{r}_{j,t}$ for $j > 1$ are the interaction-*unaware* reward functions of other agents. Only the rewards of agents which are within 25m ahead or behind of the ego-vehicle are added to the interaction-*aware* reward to reduce the effect of credit assignment problem [8, 12].

3 RESULTS

We compare our algorithm MARL-PPS to two baselines that we will call "independent-DQN" [13] and "synchronic-DQN" [1]. In independent-DQN, each agent updates its DQN policy with its own observations concurrently. In synchronic-DQN, one ego-agent updates its policy with its own observations and shares its parameters at every time step with others.

[2]In practice, ego-agent is also not updated at every time step to make sure that buffer has enough new samples. But, the period for this (learning frequency) is very small (e.g. 4).

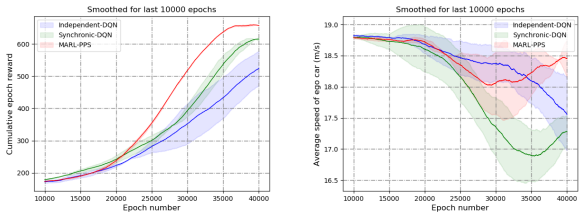

Figure 3: Training curves for baselines and the proposed algorithm. The left plot shows the mean epoch rewards for different methods. The right plot is for the average speed of the agents.

The experimental setting is illustrated in Fig. 2. The expected behaviour is the green vehicle to steer into the middle lane, so they can both move at the optimal speed. An episode ends for an agent when it collides or exceeds the maximum allowed number of iterations.

Training plots are given in Fig. 3. All the results are averaged over 3 different runs for each algorithm and associated means are plotted along with deviations. Fig. 3-left is the average over cumulative reward within an episode vs episode number. As it can be seen, MARL-PPS converges to a better solution benefiting from the stability of the training. Note that agents could learn to slow down and stop to avoid collisions. To make sure that this does not happen, we are rewarding moving in the desired speed and direction. Fig. 3-right shows the average speed of agents verifying that they do not learn to stop.

4 DISCUSSION

Decentralized learning is a common approach to solve MARL problems [5, 7, 10, 11]. A naive way is to let all agents store independent policies updated concurrently with their individual observations. Then, the number of value networks (and therefore memory usage) is linear in the number of agents. Furthermore, the number of updates and therefore training time will scale with the number of agents making this approach impractical when J is very large. Parameter sharing (PS) can tackle the problems of growing memory and computation time. In PS, observations and actions are individual, but parameters are shared during training thus one ego-agent learns from its own observations and shares its parameters with others. This way, better sample efficiency and memory usage can be achieved. But, just like concurrent learning, in PS also environment dynamics change as policies of other agents are updated, making stable training harder.

We proposed a novel MARL algorithm that is communication-cost and memory efficient outperforming the baselines in the highway planning. With MARL-PPS, only two networks are used at training time (one for ego-agent and the other for all the other agents) for any $J > 1$ making it applicable to large J. Furthermore, parameters are shared only at every $T_n \gg 1$ steps making communication cost T_n times less than the synchronic-parameter updating.

Acknowledgment Research supported by ONR N00014-17-1-2072 and ARO W911NF-17-1-0304.

REFERENCES

[1] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*. Springer, 66–83.

[2] Matthew Hausknecht and Peter Stone. 2015. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527* (2015).

[3] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[4] Zhang-Wei Hong, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, and Chun-Yi Lee. 2018. A Deep Policy Inference Q-Network for Multi-Agent Systems. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1388–1396.

[5] Max Jaderberg, Wojciech M Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio Garcia Castaneda, Charles Beattie, Neil C Rabinowitz, Ari S Morcos, Avraham Ruderman, et al. 2018. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *arXiv preprint arXiv:1807.01281* (2018).

[6] Steven M LaValle. 2006. *Planning algorithms*. Cambridge university press.

[7] Miao Liu, Kavinayan Sivakumar, Shayegan Omidshafiei, Christopher Amato, and Jonathan P How. 2017. Learning for multi-robot cooperation in partially observable stochastic environments with macro-actions. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1853–1860.

[8] Laëtitia Matignon, Guillaume Laurent, and Nadine Le Fort-Piat. 2007. Hysteretic Q-Learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams.. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'07*. 64–69.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[10] Shayegan Omidshafiei, Ali-Akbar Agha-Mohammadi, Christopher Amato, Shih-Yuan Liu, Jonathan P How, and John Vian. 2017. Decentralized control of multi-robot partially observable Markov decision processes using belief space macro-actions. *The International Journal of Robotics Research* 36, 2 (2017), 231–258.

[11] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P How, and John Vian. 2017. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. *arXiv preprint arXiv:1703.06182* (2017).

[12] Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. 2018. Lenient multi-agent deep reinforcement learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 443–451.

[13] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. 2017. Multiagent cooperation and competition with deep reinforcement learning. *PloS one* 12, 4 (2017), e0172395.