Multiagent Soft Q-Learning

Ermo Wei and **Drew Wicke** and **David Freelan** and **Sean Luke**
Department of Computer Science, George Mason University, Fairfax, VA USA
ewei@cs.gmu.edu, dwicke@gmu.edu, dfreelan@gmu.edu, sean@cs.gmu.edu

Abstract

Policy gradient methods are often applied to reinforcement learning in continuous multiagent games. These methods perform local search in the joint-action space, and as we show, they are susceptable to a game-theoretic pathology known as *relative overgeneralization*. To resolve this issue, we propose Multiagent Soft Q-learning, which can be seen as the analogue of applying Q-learning to continuous controls. We compare our method to MADDPG, a state-of-the-art approach, and show that our method achieves better coordination in multiagent cooperative tasks, converging to better local optima in the joint action space.

Introduction

Multiagent reinforcement learning (or MARL) is a type of Reinforcement Learning (RL) involving two or more agents. The mechanism is similar to traditional reinforcement learning: the environment is some current *state* (which the agent can only sense through its observation), the agents each perform some *action* while in that state, the agents each receive some *reward*, the state transitions to some new state, and than the process repeats. However in MARL, both transitions from state to state and the rewards allotted are functions of the *joint action* of the agents while in that state. Each agent ultimately tries to learn a *policy* that maps its observation to the optimal action in that state: but these are individual actions, not joint actions, as ultimately an agent cannot dictate the other agents' actions.

Multiagent Learning has been investigated comprehensively in discrete action domains. Many methods have been proposed for *equilibrium learning* (Littman 1994; 2001; Hu and Wellman 2003; Greenwald, Hall, and Serrano 2003), where the agents are trying to learn policies that satisfy some equilibrium concept from Game Theory. Almost all the equilibrium learning methods that have been proposed are based on off-policy Q-learning. This is not surprising, as multiagent equilibrium learning is naturally off-policy, that is, the agents are trying to learn an equilibrium policy while exploring the environment by following another policy. However, this situation does not apply to continuous games, that is, games with continuous actions. When RL must be applied to continuous control, policy gradient methods are often taken into consideration. However, in the past, it does not combine with off-policy samples as easily as the tabular Q-Learning. For this reason, RL has not been able to achieve as good performance in continuous games as it has in discrete domains.

In this paper, we consider cooperative games, where the agents all have the same reward function. Cooperative MARL problems can be categorized based on how much information each agent knows. If we have a central controller to control the learning process of each agent, then we have centralized training with decentralized execution (Oliehoek, Spaan, and Vlassis 2008). If the agents are learning concurrently, and each agent is told what the other agent or agents did, then the problem is known as a *joint action learner* problem. If the agents are learning concurrently but are not told what the others did, then we have an *independent learner* problem.

When the information is limited for learners in cooperative games, as is the case with independent learners, a pathology called *relative overgeneralization* can arise (Wei and Luke 2016). Relative overgeneralization occurs when a suboptimal Nash Equilibrium in the joint space of actions is preferred over an optimal Nash Equilibrium because each agent's action in the suboptimal equilibrium is a better choice when matched with arbitrary actions from the collaborating agents. For instance, consider a continuous game in Figure 1. The axes i and j are the various actions that agents A_i and A_j may perform (we assume the agents are performing deterministic actions), and the axis rewards (i, j) is the joint reward received by the agents from a given joint action $\langle i, j\rangle$. Joint action M has a higher reward than joint action N. However, the average of all possible rewards for action i_M, of agent A_i is lower than the average of all possible rewards for action i_N. Thus, the agents tend to converge to N.

In this paper, we first analytically show how relative overgeneralization prevents policy gradient methods from achieving better coordination in cooperative continuous games. This is even true in centralized training if we are not using the information wisely. Then we tackle the relative overgeneralization problem in these games by introducing Multiagent Soft Q-Learning, a novel method based on Soft Q-Learning and deep energy-based policies (Haarnoja et al. 2017). Our method is similar to MADDPG (Lowe et al.

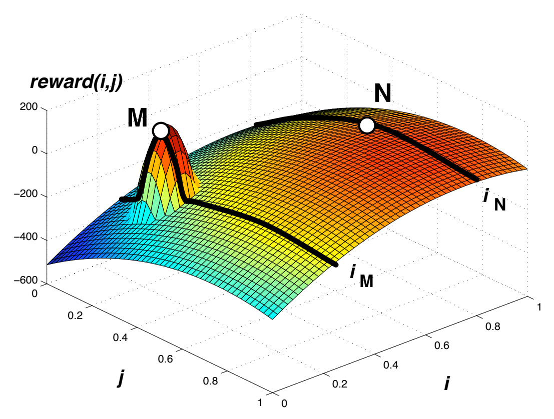

Figure 1: The relative overgeneralization pathology in continuous games.

2017), a recently proposed centralized learning algorithm. Thus, it belongs to the centralized training with decentralized execution paradigm. In this setting, since the training is centralized and we use the information wisely, it avoids the co-adaptation problem in Multiagent RL and greatly reduces the sample complexity, as the environment for agents is stationary.

Background

In this section, we first give an introduction to Markov Decision Processes (MDP) and various generalizations. We then introduce policy gradient methods.

Markov Decision Processes and Stochastic Games

A Markov Decision Process (or MDP) can be used to model the interaction an agent has with its environment. An MDP is a tuple $\{S, A, T, R, \gamma, H\}$ where S is the set of states; A is the set of actions available to the agent; T is the transition function $T(s, a, s') = P(s'|s, a)$ defining the probability of transitioning to state $s' \in S$ when in state $s \in S$ and taking action $a \in A$; R is the reward function $R : S \times A \mapsto \mathbb{R}$; $0 < \gamma < 1$ is a discount factor; and H is the horizon time of the MDP, that is, the number of steps the MDP runs.[1] An agent selects its actions based on the policy $\pi_\theta(a|s)$, which is a distribution over all possible actions a in state s parameterized by $\theta \in \mathbb{R}^n$.

The concept of an MDP can be extended to partially observable (POMDP) settings, where agents do not directly sense the state s. Rather, they receive some observation o sampled from a distribution conditioned on s.

MDPs can also be generalized to a cooperative multiagent settings, called a *Cooperative Stochastic Game* or CSG. This is a game with n agents (or players), defined by the tuple $\{S, \mathcal{A}, R, T, \gamma, H\}$, where S is the state space, $\mathcal{A} = A^1 \times ... \times A^n$ is the joint action space of n agents, $R : S \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function for each agent i, and $T(s, \vec{a}, s') = P(s'|s, \vec{a})$ is the transition function, where $\vec{a} = \langle a^1 \cdots a^n \rangle \in \mathcal{A}$ is the joint action of all agents. Thus the reward the agents receive and the state to which they transition depends on the current state and agents' joint action. Each agent i determines its action using a policy π^i. We will also use $-i$ to denote all agents except for agent i. A POCSG can be thought as taking CSG into a partially observable setting.

In the multiagent setting, a rational agent will play its *best response* to the other agents' strategy. If all agents are following a policy that implements this strategy, they will arrive at a Nash equilibrium defined as a solution where $\forall i \;\; R_i(s, \pi_1^*, \dots \pi_i^* \dots \pi_n^*) \geq R_i(s, \pi_1^*, \dots, \pi_{i-1}^*, \pi_i, \pi_{i+1}^*, \dots, \pi_n^*)$ for all of the strategies π_i available to agent i. π_i^* denotes the best response policy of agent i.

Policy Gradient Methods

In single agent continuous control tasks, it is common to apply a *policy gradient* method to determine an optimal policy. We describe that process here. To start, we define the expected return $J(\theta)$ of a policy π_θ as

$$J(\theta) = E_{P_\theta(\tau)}\big[R(\tau)\big] = \sum_\tau P_\theta(\tau)R(\tau), \qquad (1)$$

where $P_\theta(\tau)$ is the probability distribution over all possible state-action trajectories $\tau = \langle s_0, a_0, s_1, a_1, \dots, s_H, a_H \rangle$ induced by following policy π_θ, and $R(\tau) = \sum_{t=0}^{H} \gamma^t R(s_t, a_t)$ is the discounted accumulated reward along trajectory τ. We want to compute the gradient $\nabla_\theta J(\theta)$, so that we can follow the gradient to a local optimum in the space of policy parameters. To do this we use the likelihood-ratio trick (Williams 1992), where we write the gradient as

$$\begin{aligned}\nabla_\theta J(\theta) =& \sum_\tau \nabla_\theta P_\theta(\tau)R(\tau) = \sum_\tau P_\theta(\tau)\frac{\nabla_\theta P_\theta(\tau)}{P_\theta(\tau)}R(\tau) \\ =& E_{P_\theta(\tau)}\big[\nabla_\theta \ln P_\theta(\tau)R(\tau)\big]\end{aligned} \qquad (2)$$

and estimate it by performing m sample trajectories $\langle \tau^{(1)}, \dots, \tau^{(m)} \rangle$, calculating the corresponding terms, and then taking the average, that is, $\nabla_\theta J(\theta) \approx \frac{1}{m}\sum_{j=1}^{m} \nabla_\theta \ln P_\theta(\tau^{(j)})R(\tau^{(j)})$. This policy gradient method can also use off-policy samples by introducing importance sampling, where we scale each term in the empirical expectation by $\frac{P_\theta(\tau)}{Q(\tau)}$, where Q is another distribution from which our off-policy samples come. The intuition behind Equation 2 is that the reward term $R(\tau)$ scales the gradient proportionally to the reward along the trajectories.

One problem with using this likelihood-ratio estimator in practice is that it suffers from a large variance, and thus requires a great many samples to give an accurate estimation. There are various methods proposed to deal with this. A first approach is to replace the Monte Carlo estimation

[1]Any infinite horizon MDP with discounted rewards can be ϵ-approximated by a finite horizon MDP using a horizon $H_\epsilon = \frac{\log_\gamma(\epsilon(1-\gamma))}{\max_{s,a}|R(s,a)|}$ (Jie and Abbeel 2010)

of the reward along trajectories $R(\tau)$ with a value function. This leads to the *Stochastic* (SPG) and *Deterministic Policy Gradient* (DPG) Theorems (Sutton et al. 1999; Silver et al. 2014), shown below respectively:

$$\begin{aligned}\nabla_\theta J(\theta) &= \int_S \rho^{\pi_\theta}(s) \int_A \nabla_\theta \pi_\theta(a|s) Q^{\pi_\theta}(s,a)\, da\, ds \\ &= E_{s\sim\rho^{\pi_\theta}, a\sim\pi_\theta}\left[\nabla_\theta \ln \pi_\theta(a|s) Q^{\pi_\theta}(s,a)\right] \\ \nabla_\theta J(\theta) &= \int_S \rho^{\pi_\theta}(s) \nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}\, ds \\ &= E_{s\sim\rho^{\pi_\theta}}\left[\nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}\right],\end{aligned}$$

where $\rho^{\pi_\theta}(s') = \int_S \sum_{t=1}^{\infty} \gamma^{t-1} P(s) P(s \to s', t, \pi_\theta)\, ds$ is the discounted distribution over states induced by policy π_θ and starting from some state $s \in S$. Specifically, $P(s \to s', t, \pi)$ is the probability of going t steps under policy π from state s and ending up in state s'. The theorems introduced a class of algorithms (Peters and Schaal 2008; Degris, White, and Sutton 2012) under the name *actor-critic* methods, where the *actor* is the policy π and the *critic* is the Q-function.

The actor-critic algorithms have also been used in an off-policy setting through importance sampling (Degris, White, and Sutton 2012). Recently, another method called a *replay buffer* (Mnih et al. 2013) has drawn people's attention for being able to do off-policy learning with actor-critic algorithms (Lillicrap et al. 2015). In this method, we store all the samples in a buffer and at every step of learning we sample a mini-batch from this buffer to estimate the gradient of either Q-Function or policy.

Related Work

The idea of learning to cooperate through policy gradient methods has been around for a long time, but mainly for discrete action domains (Banerjee and Peng 2003). Peshkin et al. have applied the REINFORCE policy gradient to both CSG and POCSG tasks. However, as we will show later, a naive use of this gradient estimator is dangerous in the multiagent case. Nair et al. proposed *Joint Equilibrium-Based Search for Policies* (JESP), applied to POCSGs. The main idea here is to perform policy search in one agent while fixing the policies of other agents. Although this method is guaranteed to converge to a local Nash Equilibrium, it is essentially a round-robin single agent algorithm.

Recently, with the boom of Deep Reinforcement Learning (DRL), deep MARL algorithms have been proposed to tackle large scale problems. One of the main streams is the centralized training with decentralized execution. Foerster et al. proposed a method to learn communication protocols between the agents. They use inter-agent backpropagation and parameter sharing. Foerster et al. studied how to stablize the training of multiagent deep reinforcment learning using importance sampling. Two actor-critic algorithms have been proposed in (Foerster et al. 2017a; Lowe et al. 2017). They argue that by using a central critic we can ease the training of multiple agents, and that by keeping a separate policy, the agent can execute with only its local information, which makes it possible to learn in POCSGs. Among these two algorithms, MADDPG (Lowe et al. 2017) is most relevant to us. It uses the learning rule from DDPG (Lillicrap et al. 2015) to learn a central off-policy critic based on Q-learning, and uses the following gradient estimator to learn the policies for each agent i:

$$\nabla_{\theta^i} J(\theta^i) = E_{s,a^{-i}\sim D}\left[\nabla_{\theta^i} \pi^i_{\theta_i}(a_i|o_i) \nabla_{a_i} Q(s,\vec{a})|_{a_i=\pi^i_{\theta_i}(o_i)}\right],$$

where θ^i is the agent i's policy parameters, D is the replay buffer, and o_i is the local observation of agent i. During the centralized training process, the critic has access to the true state $s = [o_1, \ldots, o_n]$. But at execution time, each agent only has access to o_i.

Multiagent Actor-Critic Algorithms

As we described earlier, if we have limited information for our agent, we can suffer from the relative overgeneralization problem. In this section, we demonstrate how this affects the actor critic algorithm. We will first derive the policy gradient estimator for the cooperative multiagent case and then discuss several problems that occurs if we naively use this estimator, which will shed light on the reasons why Multiagent Soft Q-Learning may be useful.

Proposition 1. *For any episodic cooperative stochastic game with n agents, we have the following multiagent stochastic policy gradient theorem:*

$$\begin{aligned}\nabla_{\theta^i} J(\vec{\theta}) = \int_s \rho^{\pi^1,\cdots,\pi^n}(s) \int_{A^i} \nabla_\theta \pi(a^i|s) \\ \int_{A^{-i}} \pi^{-i}(a^{-i}|s) Q^{\pi^1,\cdots,\pi^n}(s,\vec{a})\, da^{-i}\, da^i\, ds\end{aligned}$$

The proof of this proposition is provided in Proof A at the end of this paper. From Proposition 1, we can see that the policy gradient for agent i at each state is scaled by $Q^{\pi^i}(s,a^i) = \int_{A_{-i}} \pi^{-i}(a^{-i}|s) Q^{\pi^1\cdots\pi^n}(s,\vec{a}) da^{-i}$, which are the joint-action Q-values averaged by the other agents' policies. Their are several problems with this estimator. First, for any agent i, the joint-action Q-function is an on-policy Q-function. That is, it is learned under policy π^i, and π^{-i} which is not the best response policy of other agents. Thus, the joint-action Q-function may not scale the gradient in right magnitude. Second, if we are an independent learner and play as A_i in game shown in Figure 1, we only have access to $Q^{\pi^i}(s,a^i)$, since this value is averaged by other's policies, the value of the action $\langle i_N \rangle$ would be higher than the value of action $\langle i_M \rangle$ even under the optimal Q-function, thus mistakenly scaling the gradient to towards $\langle i_N \rangle$.

MADDPG solves the previous two issues by using the following methods. First, it uses the replay buffer (Lillicrap et al. 2015) to learn an off-policy optimal Q-function very much like what we learn for Q-Learning (Silver et al. 2014). This is not doable with traditional importance sampling based off-policy learning. Second, it's using the centralized training method which gives it direct access to the joint-action Q-function, but not the policies.

However, MADDPG fails to use the optimal action for gradient scaling, making it still vulnerable to the relative overgeneralization problem. To see that, consider its gradient estimator,

$$\nabla_{\theta^i} J(\theta^i) = E_{s,a^{-i}\sim D}\left[\nabla_{\theta^i}\pi^i_{\theta_i}(a_i|o_i)\nabla_{a_i}Q^*(s,\vec{a})|_{a_i=\pi^i_{\theta_i}(o_i)}\right]$$

We see that for agent i, it tries to ascend the policy gradient based on $Q^*(s,\vec{a})$, where a^{-i} is from the replay buffer D rather than the optimal policy, which is another way of averaging the Q-values based on others' policies. As we showed in Figure 1, this average-based estimation can lead to relative overgeneralization.

Multiagent Soft Q-Learning

In this paper, we propose MARL method for cooperative continuous games. We show that on the one hand, our method is an actor-critic method, which thus can benefit from the centralized training method, with one central critic and multiple distributed policies. And on the other hand, our method resembles Q-Learning, and thus, it efficiently avoids the relative overgeneralization problem. We first introduce Soft Q-Learning and then describe how we use it for multiagent training.

Soft Q-Learning

Although Q-Learning has been widely used to deal with control tasks, it has many drawbacks. One of the problems is that at the early stage of learning, the max operator can bring bias into the Q-value estimation (Fox, Pakman, and Tishby 2016). To remedy this, Maximum Entropy Reinforcment Learning (MERL) was introduced, in which tries to find the following policy:

$$\pi^*_{\text{MaxEnt}} = \text{argmax}_\pi \sum_t E_{(s_t,a_t)\sim\rho^\pi}[r(s_t,a_t)+\alpha\mathcal{H}(\pi(\cdot|s)]$$

where $\mathcal{H}(\pi(\cdot|s))$ is the entropy of the policy. The parameter α controls the relative importance of the reward and entropy: when it goes to 0, we recover ordinary RL. From this objective, a learning method similar to Q-Learning can be derived, called Soft Q-Learning (Haarnoja et al. 2017). Its learning algorithm is

$$Q_{\text{soft}}(s_t,a_t) \leftarrow r_t + \gamma E_{s_{t+1}}[V_{\text{soft}}(s_{t+1})] \quad \forall s_t, a_t,$$
$$V_{\text{soft}}(s_t) \leftarrow \alpha \log \int_A \exp(\frac{1}{\alpha}Q_{\text{soft}}(s_t,a'))da'.$$

Haarnoja et al. have shown that by using this update rule, Q_{soft} and V_{soft} can converge to Q^*_{soft} and V^*_{soft} respectively, and by driving $\alpha \to 0$, Q-learning with a hard max operator can be recovered. For this reason, Haarnoja et al. named this Soft Q-learning.

Once we have the learned Q-function above, we can get the optimal max entropy policy as

$$\pi^*_{\text{MaxEnt}}(a_t|s_t) = \exp(\frac{1}{\alpha}Q^*_{\text{soft}}(s_t,a_t) - V^*_{\text{soft}}(s_t)) \propto Q^*_{\text{soft}}(s_t,a_t).$$

A nice property of this policy is that it spreads widely over the entire action space in continuous control tasks. Thus, if we have such a policy, and if there are multiple modes in the action space, we can find them much more effectively than with more deterministic policies (e.g. Gaussian policy) which are typically used in actor-critic algorithms. However, since the form of this policy is so general, sampling from it is very hard. Soft Q-Learning solves this issue by using Stein Variational Gradient Descent (SVGD) (Liu and Wang 2016) to approximate the optimal policy through minimizing the KL-divergence:

$$D_{\text{KL}} = \left(\pi_\theta(\cdot|s_t)\big|\big|\exp\left(\frac{1}{\alpha}Q^*_{\text{soft}}(s_t,a_t) - V^*_{\text{soft}}(s_t)\right)\right), \quad (3)$$

where policy $\pi_\theta(\cdot|s)$ is our approximate policy. Since $-\frac{1}{\alpha}Q^*_{\text{soft}}(s_t,a_t)$ can be viewed as an energy function, and the authors are using a deep neural network to approximate the Q-function, they call this a *deep energy-based* policy. It has been demonstrated that using the Soft Q-Learning with deep energy based policies can learn multimodal objectives. In Soft Q-Learning we need to learn both the Q-function and the energy-based policy $\pi(\cdot|s)$. Thus, Soft Q-Learning can be thought as an actor-critic algorithm. Now consider the multiagent case. To make it clear, we first recall how we can achieve coordination in a discrete domain. In discrete domains, when we have the $Q^*(s,a)$ function, we simply apply the *argmax* operator to it and then let each agent do its own part of the optimal action. This is possible since we can do global search in the joint-action space for a given state. Now, with Soft Q-Learning and a deep energy-based model, we can mimic what we did in the discrete case. In this situation, we start with a high α to do global search in the joint-action space, then quickly anneal the α to lock on some optimal action, like the argmax operator. It has been shown that by annealing the α, we can get a deterministic policy from deep energy-based policies (Haarnoja et al. 2017).

Algorithm 1: Multiagent Soft Q-Learning

```
input: A central critic Q, N policies for all N agents, $\alpha$,
       and the epoch start to annealing $t$
for episode = 0 to M do
    Update central critic Q using the method from Soft
      Q-Learning.
    for agent = 1 to N do
        Update the joint policy for agent $i$ using
          equation (3)
    end
    if episode $\geq t$ then
        anneal $\alpha$
end
```

As we described before, Soft Q-Learning is also an actor-critic method. Thus, we can borrow the idea of learning a centralized joint action critic with Soft Q-Learning from MADDPG. Then for each of the agents, instead of learning a mapping for its own observation to its own action, we learn a mapping from its own observation to the joint-action. When the agent interacts with the environment, it just performs its own part of the joint action. We start the learning with high α value and let it explore the joint action space, we then quickly anneal the α to let each agent find a better local optima in joint-action space. Our algorithm is given at Algorithm 1.

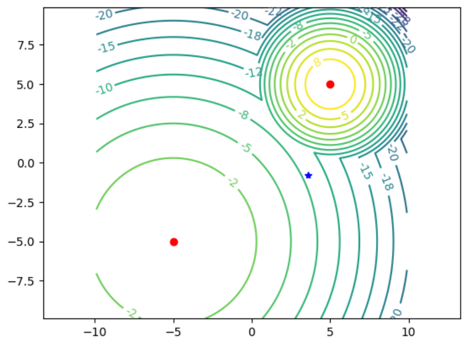

Figure 2: The Max of Two Quadratic game. The dots mark the two local optima in the joint action space while the star marks the joint action of the two agents. The contour shows the reward level.

Experiments

To show that our Multiagent Soft Q-Learning method can achieve better coordination, we consider the Max of Two Quadractics game from previous literature (Panait, Luke, and Wiegand 2006). This is a simple single state continuous game for 2 agents, one action dimension per agent. Each agent has a bounded action space. The reward for a joint action is given by following equation

$$f_1 = h_1 \times \big[-(\frac{a_1 - x_1}{s_1})^2 - (\frac{a_2 - y_1}{s_1})^2 \big]$$
$$f_2 = h_2 \times \big[-(\frac{a_1 - x_2}{s_2})^2 - (\frac{a_2 - y_2}{s_2})^2) \big] + c$$
$$r(a_1, a_2) = \max(f_1, f_2)$$

where a_1, a_2 are the actions from agent 1 and agent 2 respectively. In the equation above, $h_1 = 0.8, h_2 = 1, s_1 = 3, s_2 = 1, x_1 = 5, x_2 = 5, y_1 = -5, y_2 = -5, c = 10$ are the coefficients to determine the reward surface (see Figure 2). Although the formulation of the game is rather simple, it poses a great difficulty to gradient-based algorithms as, over almost all the joint-action space, the gradient points to the sub-optimal solution located at (-5, -5).

We trained the MADDPG agent along with our Multagent Soft Q-Learning agent in this domain. As this was a simple domain, we used two-hidden-layer networks with size $\{100, 100\}$, and we trained the agents for 150 epochs for 100 steps per epoch. The training was not started until we had 1000 samples in the replay buffer. Both agents scaled their reward by 0.1. For our Multiagent Soft Q-Learning agent, we started the annealing at epoch 100, and finished the annealing in 15 epochs. We started with $\alpha = 1$, and annealed it to 0.001. For the rest of the parameters, we used the default setting from the original DDPG (Lillicrap et al. 2015) and Soft Q-Learning (Haarnoja et al. 2017) papers. In addition, to mimic the local observation setting where centralized learning was suitable, we gave the two agents in both algorithms different observation signals, where the first agent would always sense the state as $\langle 0 \rangle$, and the second agent would always sense it as $\langle 1 \rangle$. Then the state for the central critic was $\langle 0, 1 \rangle$.

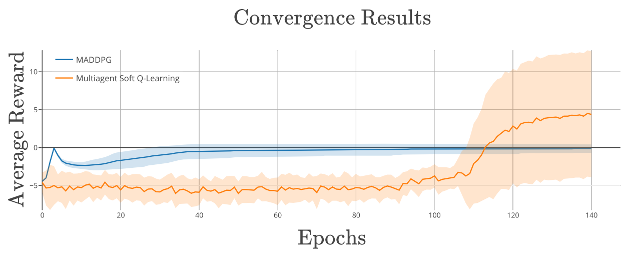

Figure 3: The average reward for both algorithms. Multiagent Soft Q-Learning finds the better local optima quickly after we anneal α.

The result is in Figure 3. However, the plot is an average over all 50 experiment runs, and hence, may hide some critical information. On closer investigation, we found that Multiagent Soft Q-Learning converged to the better equilibrium **72%** of the time, while MADDPG **never** converged to it.

Conclusion and Future Work

In this paper, we investigated how to achieve better controls in continuous games. We showed why the traditional policy gradient methods is not suitable for these tasks, and why the gradient-based method can fail to find better local optima in the joint-action space. We then proposed Multiagent Soft Q-Learning based on the centralized training and decentralized execution paradigm, and showed that, we can achieve much better coordination with higher probability. And since we are using centralized training, the co-adaption problem can be avoided, thus, making our method sample-efficient compared to independent learners. We argue that Multiagent Soft Q-Learning is a competitive RL learner for hard coordination problems.

There are some issues that we haven't been able to investigate thoroughly in this work. First, so far we have only applied our learner in the single state games. To better understand the algorithm, we would like to try our algorithm on sequential continuous games with hard coordination problems. Second, as we show in the experiment, with Soft Q-Learning we are not able to converge the better equilibrium for 100% of the time. In the future, we would like to investigate different annealing methods to improve the convergence rate. Last, we notice that Multiagent Soft Q-Learning models the joint action of all the agents, and thus the dimension of the action can explode with more agents. To solve this issue, we will investigate how to apply Soft Q-Learning in the independent learner case, where the algorithm scales well.

Acknowledgments

The research in this paper was conducted with the support of research infrastructure developed under NSF grant 1727303. We thank Tuomas Haarnoja and Haoran Tang for their helpful comments on implementing Soft Q-Learning.

Proof A

We first denote $\vec{\pi}$ as the joint-policy. This proof requires that $P(s), P(s'|s,\vec{a}), \pi^i(a^i|s), \nabla_{\theta^i}\pi^i(a^i|s)$, and $Q^{\vec{\pi}}(s,\vec{a})$ be continuous in all parameters and variables $s, s', \vec{a}$. This regularity condition implies that $V^{\vec{\pi}}(s)$ and $\nabla_{\theta^i}V^{\vec{\pi}}(s)$ are continuous functions of θ and s. S is also required to be compact, and so for any θ, $||\nabla_{\theta^i}V^{\vec{\pi}}(s)||$ is a bounded function of s. The proof mainly follows along the standard Stochastic Policy Gradient Theorem. We assume agent i follows the policy $\pi^i(a^i|s)$ parameterized by θ^i. We denote π^{-i} as the joint policy of all agents but agent i, and a^{-i} as the joint action of all agents except agent i. For notation simplicity, we denote:

$$\begin{aligned}
&\int_{A^{-i}} \pi^{-i}(a^{-i}|s) f(a^{-i})\, da^{-i} \\
=&\int_{A^1}\pi^1(a^1|s)\cdots\int_{A^{i-1}}\pi^{i-1}(a^{i-1}|s) \\
&\int_{A^{i+1}}\pi^{i+1}(a^{i+1}|s)\cdots\int_{A^n}\pi^n(a^n|s) f(a^{-i}) \\
&da^n\cdots\, da^{i+1}\, da^{i-1}\cdots\, da^1
\end{aligned}$$

Using this new notation the proof follows:

$$\begin{aligned}
&\nabla_{\theta^i}V^{\vec{\pi}}(s) \\
=&\nabla_{\theta^i}\int_{A^1}\pi^1(a^1|s)\cdots\int_{A^n}\pi^n(a^n|s)\, Q^{\vec{\pi}}(s,\vec{a})\, d\vec{a} \\
=&\int_{A^{-i}}\pi^{-i}(a^{-i}|s)\nabla_{\theta^i}\int_{A^i}\pi^i(a^i|s)Q^{\vec{\pi}}(s,\vec{a})\, da^i\, da^{-i} \\
=&\int_{A^{-i}}\pi^{-i}(a^{-i}|s)\int_{A^i}\Big[\nabla_{\theta^i}\pi^i(a^i|s)Q^{\vec{\pi}}(s,\vec{a}) \\
&\quad + \pi^i(a^i|s)\nabla_{\theta^i}Q^{\vec{\pi}}(s,\vec{a})\Big]\, da^i\, da^{-i} \\
=&\int_{A^{-i}}\pi^{-i}(a^{-i}|s)\int_{A^i}\Big[\nabla_{\theta^i}\pi^i(a^i|s)Q^{\vec{\pi}}(s,\vec{a}) \\
&\quad + \pi^i(a^i|s)\int_S \gamma P(s'|s,\vec{a})\nabla_{\theta^i}V^{\vec{\pi}}(s')\, ds'\Big]\, da^i\, da^{-i}
\end{aligned}$$

We used Leibniz integral rule to exchange order of derivative and integration using the regularity condition and expanding $Q^{\vec{\pi}}(s,\vec{a})$ above. We use $P^{\pi}(s'|s,t)$ as short for $P(s\to s',t,\pi)$. Now we iterate the formula,

$$\begin{aligned}
=&\int_{A^{-i}}\pi^{-i}(a^{-i}|s)\int_{A^i}\Big[\nabla_{\theta^i}\pi^i(a^i|s)Q^{\vec{\pi}}(s,\vec{a}) + \pi^i(a^i|s) \\
&\int_S \gamma P(s'|s,\vec{a})\Big[\int_{A^{-i}}\pi^{-i}(a^{-i}|s')\int_{A^i}\Big(\nabla_{\theta^i}\pi^i(a^i|s')Q^{\vec{\pi}}(s',\vec{a}) \\
&+ \pi^i(a^i|s')\int_S \gamma P(s''|s',\vec{a})\nabla_{\theta^i}V^{\vec{\pi}}(s'')\, ds''\Big)\, da^i\, da^{-i}\Big]\, ds'\Big] \\
&da^i\, da^{-i} \\
=&\int_{A^{-i}}\pi^{-i}(a^{-i}|s)\int_{A^i}\nabla_{\theta^i}\pi^i(a^i|s)Q^{\vec{\pi}}(s,\vec{a})da^i da^{-i} \\
&+\int_S \gamma P^{\vec{\pi}}(s'|s,1)\int_{A^{-i}}\pi^{-i}(a^{-i}|s') \\
&\int_{A^i}\nabla_{\theta^i}\pi^i(a^i|s')Q^{\vec{\pi}}(s',\vec{a})da^i da^{-i} ds' \\
&+\int_S \gamma P^{\vec{\pi}}(s'|s,1)\int_S \gamma P^{\vec{\pi}}(s''|s',1)\nabla_{\theta^i}V^{\vec{\pi}}(s'')\, ds''\, ds' \\
=&\int_S \sum_{t=0}^{\infty}\gamma^t P^{\vec{\pi}}(s'|s,t) \\
&\int_{A^{-i}}\pi^{-i}(a^{-i}|s')\int_{A^i}\nabla_{\theta^i}\pi^i(a^i|s')Q^{\vec{\pi}}(s',\vec{a})\, da^i\, da^{-i}\, ds' \\
=&\int_S \sum_{t=0}^{\infty}\gamma^t P^{\vec{\pi}}(s'|s,t) \\
&\int_{A^i}\nabla_{\theta^i}\pi^i(a^i|s')\int_{A^{-i}}\pi^{-i}(a^{-i}|s')Q^{\vec{\pi}}(s',\vec{a})\, da^{-i}\, da^i\, ds'
\end{aligned}$$

In the final line we use Fubini's theorem and exchange the order of integration using the regularity condition so that $||\nabla_{\theta^i}V^{\vec{\pi}}(s)||$ is bounded. We then take the expectation over the possible start states s:

$$\begin{aligned}
\nabla_{\theta^i}J(\theta) =&\nabla_{\theta^i}\int_S P(s)V^{\vec{\pi}}(s)\, ds \quad = \int_S P(s)\nabla_{\theta^i}V^{\vec{\pi}}(s)\, ds \\
&= \int_S\int_S\sum_{t=0}^{\infty}\gamma^t P(s)P^{\vec{\pi}}(s'|s,t)\, ds\int_{A_i}\nabla_{\theta^i}\pi^i(a^i|s') \\
&\quad\int_{A_{-i}}\pi^{-i}(a^{-i}|s')Q^{\vec{\pi}}(s',\vec{a})\, da^{-i}\, da^i\, ds' \\
&= \int_S \rho^{\vec{\pi}}(s')\int_{A^i}\nabla_{\theta^i}\pi^i(a^i|s') \\
&\quad\int_{A^{-i}}\pi^{-i}(a^{-i}|s')Q^{\vec{\pi}}(s',\vec{a})\, da^{-i}\, da^i\, ds'
\end{aligned}$$

□

References

Banerjee, B., and Peng, J. 2003. Adaptive policy gradient in multiagent learning. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, 686–692. New York, NY, USA: ACM.

Degris, T.; White, M.; and Sutton, R. S. 2012. Linear off-policy actor-critic. In *In International Conference on Machine Learning*. Citeseer.

Foerster, J.; Assael, Y. M.; de Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2137–2145.

Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2017a. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926*.

Foerster, J.; Nardelli, N.; Farquhar, G.; Afouras, T.; Torr, P. H. S.; Kohli, P.; and Whiteson, S. 2017b. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, 1146–1155.

Fox, R.; Pakman, A.; and Tishby, N. 2016. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'16, 202–211.

Greenwald, A.; Hall, K.; and Serrano, R. 2003. Correlated Q-learning. In *AAAI Spring Symposium*, volume 3, 242–249.

Haarnoja, T.; Tang, H.; Abbeel, P.; and Levine, S. 2017. Reinforcement learning with deep energy-based policies. *ICML*.

Hu, J., and Wellman, M. P. 2003. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research* 4:1039–1069.

Jie, T., and Abbeel, P. 2010. On a connection between importance sampling and the likelihood ratio policy gradient. In *Advances in Neural Information Processing Systems*, 1000–1008.

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*.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, volume 94, 157–163.

Littman, M. L. 2001. Friend-or-foe Q-learning in general-sum games. In *ICML*, volume 1, 322–328.

Liu, Q., and Wang, D. 2016. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances In Neural Information Processing Systems*, 2378–2386.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*.

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*.

Nair, R.; Tambe, M.; Yokoo, M.; Pynadath, D.; and Marsella, S. 2003. Taming decentralized pomdps: Towards efficient policy computation for multiagent settings. In *IJCAI*, 705–711.

Oliehoek, F. A.; Spaan, M. T.; and Vlassis, N. 2008. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research* 32:289–353.

Panait, L.; Luke, S.; and Wiegand, R. P. 2006. Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Transactions on Evolutionary Computation* 10(6):629–645.

Peshkin, L.; Kim, K.-E.; Meuleau, N.; and Kaelbling, L. P. 2000. Learning to cooperate via policy search. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, 489–496. Morgan Kaufmann Publishers Inc.

Peters, J., and Schaal, S. 2008. Natural actor-critic. *Neurocomputing* 71(7):1180–1190.

Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; and Riedmiller, M. 2014. Deterministic policy gradient algorithms. In *ICML*.

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; Mansour, Y.; et al. 1999. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

Wei, E., and Luke, S. 2016. Lenient learning in independent-learner stochastic cooperative games. *Journal of Machine Learning Research* 17(84):1–42.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8(3-4):229–256.