Partial Reward Decoupling

Ben Freed

April 2019

1 Introduction

As the required complexity of robot-robot interactions in domains such as autonomous transportation and manufacturing increases, so will the need for scalable and effective multi-agent control strategies. Recent approaches to multi-agent reinforcement learning (MARL) have shown great success in controlling highly-coordinated multi-agent teams to accomplish strategy-intensive tasks, often at super-human levels of performance. However, scaling multi-agent approaches to large numbers of agents continues to be a challenge. One reason for this poor scaling is commonly referred to as the *credit assignment problem*: as the team size increases, it becomes increasingly difficult to determine the impact that any one agent's actions had on the group reward, making it more difficult to determine which actions to reinforce and which to discourage. One way to mitigate the credit assignment problem is to use heavily-engineered reward functions. Consider the case in which each agent i earns an individual reward at each timestep, $r_t^{(i)}$, and we want agents to learn a policy that maximizes expected sum of all agents' individual rewards, across all timesteps, *i.e.* $\sum_{i=1}^{K} \sum_{t=0}^{T} r_t^{(i)}$. Rather than optimizing the policy of a particular agent j to maximize sum of all agents' individual rewards, we might instead optimize a weighted sum of individual agent's rewards, *i.e.,* $\sum_{i=1}^{K} \sum_{t=0}^{T} c_i r_t^{(i)}$. The weighting variables, c_i, allow us to manually specify how much we want agent j to care about the rewards of agent i. However, the ideal way to set these weights at the outset is unclear, because agents' actions can intereact with other agents' rewards in potentially very complex ways, and an improper setting may cause agents to learn greedy or competitive policies instead of cooperative policies[1]. For this reason, our goal is to design an algorithm that automatically and dynamically can choose these weightings, so as to mitigate the credit assignment problem, while minimally changing the optimal policies away from the fully-cooperative optimal policies (policies which are optimal when weightings are all set to 1).

2 Background

Consider a MARL problem in which K agents exist in an environment. At each timestep t, each agent i observes the state of the environemnt S_t, and then samples an action $a_t^{(i)}$ from

[1]This problem is studied heavily in the context of game theory, where incentive structures (such as the reward functions used in RL) are analyzed to determine the Nash equilibria of the problem

its policy $\pi_\theta^{(i)}(\mathbf{a_t^{(i)}}|S_t;\theta_i)$, where θ_i denotes the policy parameters for agent i. We refer to the set of all actions chosen by all agents at time t as the joint action a_t. Once all agents have selected an action, each agent i receives a reward $r_t^{(i)} \sim p_{r^{(i)}|S,a}(\cdot|S_t, a_t)$. We refer to the sequence of states, rewards, and actions taken during an episode as trajectory τ, *i.e.*, $\tau = (S_0, a_0, r_0, ..., S_T, a_T, r_T$. The policy parameters define a trajectory distribution (the distribution of trajectories we could take during a rollout), *i.e.*, $\tau \sim p_\tau(\cdot|\theta)$. Our goal is to find the joint policy parameters $\theta = \{\theta_1, ..., \theta_K$ that maximize sum of expected discounted future rewards, *i.e.*,

$$\theta^* = \text{argmax}_\theta J(\theta), \tag{1}$$

$$\text{where} \tag{2}$$

$$J(\theta) = \mathbb{E}_{\tau\sim p_\tau(\cdot|\theta)}\left[\sum_{j=1}^{K}\sum_{t=0}^{T}\gamma^t r_t^{(j)}\right]. \tag{3}$$

A number of algorithms exist for performing this optimization. A number are based on estimating the policy gradient (the gradient of J with respect to θ) from empirical data. By applying policy gradient theorem [2] in a multi-agent setting we find that the policy gradient for the parameters of a particular agent i is given by

$$\nabla_{\theta_i} J(\theta) = \mathbb{E}_\tau\left[\sum_{t=0}^{T}\nabla_{\theta^{(i)}}\log\pi^{(i)}(a_t^{(i)}|S_t)\sum_{j=1}^{K}\left(\hat{Q}_t^j - V^{j,\pi}(S_t, a_t^{-i})\right)\right] \tag{4}$$

where $\hat{Q}_t^j$ is the empirical discounted future reward sum for agent j from time t onward, *i.e.*, $\hat{Q}_t^j = \sum_{\ell=t}^{T}\gamma^{\ell-t}r_\ell^j$, and $V^{j,\pi}(S_t, a_t^{-i})$ is the expected future discounted reward sum for agent j from time t onward, given the state of the environment and all agents' actions except that of agent i at time t, and given that agents follow joint policy π for the remainder of the episode.

In the actor-critic algorithm, this gradient is approximated by the following simple estimator:

$$\nabla_{\theta_i} J(\theta) \approx g_i = \nabla_{\theta^{(i)}}\log\pi^{(i)}(a_t^{(i)}|S_t)\sum_{j=1}^{K}\left(\hat{Q}_t^j - V^{j,\pi}(S_t, a_t^{-i})\right), \tag{5}$$

for a state S_t sampled from an empirical trajectory. It can be easily verified using the law of large numbers that g_i is an unbiased gradient estimator, *i.e.*, $\mathbb{E}_\tau[g_i(\tau,\theta)] = \nabla_{\theta_i} J(\theta)$. In practice, $V^{j,\pi}$ is typically estimated using a value function approximator (referred to as the critic) [1].

3 The Credit Assignment Problem

The credit assignment problem can be easily demonstrated in the case of the actor-critic estimator, if we make some simplifying assumptions. Assume that, at timestep t, agent i takes an action that does not impact the future rewards of another agent, k, given the current state and actions of all other agents. That is, $\sum_{\ell=t}^{T} \gamma^{\ell-t} r_\ell^k \perp a_t^{(i)} | S_t, a_t^{(-i)}$. An example of when this may occur in a real-life application is a manufacturing problem in which agents i and k build two separate sub-components, and individual rewards are assigned based on how well each of them builds their particular subcomponent. Even if the two subcomponents are later combined into one larger component by a third agent, agent l, and agent l is rewarded based on how well the final product comes out, agent k's reward is fully independent of agent i's actions (and vice versa) because agent i's and agent k's actions only impact their own reward, and the reward of agent l. Splitting the inner summation in the actor-critic policy gradient estimator into terms involving the rewards from agents other than k, and the rewards from agent k, we obtain

$$g_i = \nabla_{\theta^{(i)}} \log \pi^{(i)}(a_t^{(i)}|S_t) \sum_{j \neq k} \left(\hat{Q}_t^j - V^{j,\pi}(S_t, a_t^{-i}) \right) + \nabla_{\theta^{(i)}} \log \pi^{(i)}(a_t^{(i)}|S_t) \left(\hat{Q}_t^k - V^{k,\pi}(S_t, a_t^{-i}) \right). \tag{6}$$

By our independence assumption, the second term, on average, contributes nothing to the gradient. That is to say, the second term can be dropped completely, and the estimator will remain unbiased. **Proof:** taking the expectation of g_i with respect to future rewards, given state S_t and joint action a_t, we obtain:

$$\begin{aligned} \mathbb{E}[g_i] = \mathbb{E}\left[\nabla_{\theta^{(i)}} \log \pi^{(i)}(a_t^{(i)}|S_t) \sum_{j \neq k} \left(\hat{Q}_t^j - V^{j,\pi}(S_t, a_t^{-i}) \right) \right] \quad (7) \\ + \mathbb{E}\left[\nabla_{\theta^{(i)}} \log \pi^{(i)}(a_t^{(i)}|S_t) \left(\hat{Q}_t^k - V^{k,\pi}(S_t, a_t^{-i}) \right) \right]. \quad (8) \end{aligned}$$

Moving the expectation of the second term inside,

$$\begin{aligned} \mathbb{E}[g_i] = \mathbb{E}\left[\nabla_{\theta^{(i)}} \log \pi^{(i)}(a_t^{(i)}|S_t) \sum_{j \neq k} \left(\hat{Q}_t^j - V^{j,\pi}(S_t, a_t^{-i}) \right) \right] \quad (9) \\ + \nabla_{\theta^{(i)}} \log \pi^{(i)}(a_t^{(i)}|S_t) \left(\mathbb{E}\left[\hat{Q}_t^k \middle| S_t, a_t \right] - V^{k,\pi}(S_t, a_t^{-i}) \right). \quad (10) \end{aligned}$$

Because $\hat{Q}_t^k$is independent of $a_t^{(i)}$ as per the independence assumption, we have that $\mathbb{E}\left[\hat{Q}_t^k \middle| S_t, a_t \right] = \mathbb{E}\left[\hat{Q}_t^k \middle| S_t, a_t^{-i} \right] = V^{k,\pi}(S_t, a_t^{-i})$. Therefore,

$$\begin{aligned} \mathbb{E}[g_i] = \mathbb{E}\left[\nabla_{\theta^{(i)}} \log \pi^{(i)}(a_t^{(i)}|S_t) \sum_{j \neq k} \left(\hat{Q}_t^j - V^{j,\pi}(S_t, a_t^{-i}) \right) \right] \quad (11) \\ + \nabla_{\theta^{(i)}} \log \pi^{(i)}(a_t^{(i)}|S_t) \left(V^{k,\pi}(S_t, a_t^{-i}) - V^{k,\pi}(S_t, a_t^{-i}) \right) \quad (12) \\ = \mathbb{E}\left[\nabla_{\theta^{(i)}} \log \pi^{(i)}(a_t^{(i)}|S_t) \sum_{j \neq k} \left(\hat{Q}_t^j - V^{j,\pi}(S_t, a_t^{-i}) \right) \right]. \quad (13) \end{aligned}$$

For each agent i, for each timestep, we therefore acknowledge that there exists a set of other agents, which we term the *relevant set of agent i at time t*, for which we should consider the individual rewards in the policy gradient estimation, to ensure unbiased gradient estimates. This set is a function of the state and actions of all agents other than i.

4 Automatic Identification of the Relevant Set

It is common to learn a value funciton in policy-gradient-based methods, such as actor critic, that predicts future expected reward given the currrent state (and potentially actions), under the current joint policy. In the last section, we showed that the relevant set for agent i is the set of all other agents j for which agent j's future rewards are dependent on agent i's action at timestep t, given the state of the environment, and the rewards of all other agents in the relevant set. Another way to say this is that agent j's state-action value function (Q-function) depends on the action of agent i. Agent j's state action value function is given by

$$Q^{j,\pi}(S_t, a_t) = \mathbb{E}\left[\sum_{\ell=t}^{T} \gamma^{\ell-t} r_t^{(j)} \middle| S_t, a_t\right]. \tag{14}$$

Therefore, one way to determine if agent j is in the relevant set of agent i, is to approximate $Q^{j,\pi}$ with a function approximator $Q_\phi^{j,\pi}$ (parameterized by ϕ), and determine if this approximation depends on $a_t^{(j)}$. This approximation can be learned using standard value function approximation techniques, such as minimization of mean squared error with the actual return of agent j, or minimization of Bellman error.

We can adopt a specific parameterization that allows for rapid and efficient determination of dependence of the Q-function approximation on specific actions. For example, consider a simple case in which action inputs to the Q-function are passed through "gates", which can be switched on or off depending on the state of the environment and/or actions of other agents (for example, a gate could represent a multiplication with a scalar, and when that scalar is set to zero, dependence on the corresponding action is perfectly switched off. A slight penalty for non-zero gate scalar values could be imposed to encourage any gates corresponding to actions that do not impact the Q function to be fully switched off. In theory, if we assume the approximate Q-funciton converges to the true Q-funciton, this penalty could be set arbitrarily low so as to not affect the gates of any actions that actually affect the Q-values.

References

[1] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

[2] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.