Collaboration in Deep MARL

Jacob Beck

Many tasks are well-modeled as fully-cooperative multi-agent games, for which solutions can be learned using multi-agent reinforcement learning (MARL). From poker to control of autonomous vehicles, many settings place restrictions on information shared between the agents at runtime, but make no such requirements of training. These restrictions have given rise to the paradigm of centralized training and decentralized execution (CTDE), where each agent can share its observations at training, but only has access to its own observations at execution (1). Although decentralized execution requires factoring the joint action across all agents into individual policies for each agent, early factorization during training poses significant problems for learning collaboration. The problem is common to both current policy-gradient approaches and value-based approaches, with solutions in the former remaining unexplored. Here, I will give background information, summarize the problem, and propose a potential direction for research.

Background

A major issue with MARL is that it introduces non-stationarity in the transition function of a Markov Decision Process (MDP). In other words, the MDP for one agent changes as the others learn. A common approach is to learn a single "joint" network architecture designed to factor into individual networks operating on individual observations.

Policy-Gradient Methods. The policy gradient can be written (2):

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\pi,\mathbf{a}\sim\pi}[Q^\pi(s,\mathbf{a})\nabla_\theta \log(\pi_\theta(\mathbf{a}|s))].$$

In the multi-agent setting, given that each agent must select an action without communicating, the individual actions of each agent must be conditionally independent given the state. If $\mathbf{a}$ represents the joint-action vector for all agents, and $\pi^{(i)}$ represents the policy of the ith agent, then the policy gradient decomposes:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\pi,\mathbf{a}\sim\pi}[Q^\pi(s,\mathbf{a})\nabla_\theta \log(\prod_{a_i\in\mathbf{a}} \pi_\theta^{(i)}(a_i|s))]$$

$$= \mathbb{E}_{s\sim\pi,\mathbf{a}\sim\pi}[Q^\pi(s,\mathbf{a})\nabla_\theta \sum_{a_i\in\mathbf{a}} \log(\pi_\theta^{(i)}(a_i|s))]$$

$$= \sum_{a_i\in\mathbf{a}} \mathbb{E}_{s\sim\pi,\mathbf{a}\sim\pi}[Q^\pi(s,\mathbf{a})\nabla_\theta \log(\pi_\theta^{(i)}(a_i|s))]$$

This is convenient and seemingly allows us to treat the multi-agent problem in the same way we would several single agents, updating them all simultaneously.

Problem

The problem we will see with vanilla policy-gradient approaches generalizes to contemporary approaches, so we start here. Consider the following two player collaborative game where: agent A has a policy that plays column-0 with probability p and column-1 with probability (1-p); and agent B has a policy that plays row-0 with probability q and row-1 with probability (1-q):

	p	1-p
q	0	-5
1-q	-5	10

The optimal, fully-collaborative policy of $p = 0$ and $q = 0$ achieves a maximum score of $J = 10$. Now let's consider the derivative of the expected payoff with respect to p, i.e. the policy gradient for player A:

$$\frac{d}{dp}J(p,q) = \frac{d}{dp}(-5(1-p)q - 5(1-q)p + 10(1-p)(1-q))$$

$$= -15 + 20q$$

Note that this is negative if $q < \frac{3}{4}$ and positive if $q > \frac{3}{4}$. This implies a major issue for the policy gradient methods. That is, if agent B is currently playing a policy with $q > \frac{3}{4}$, then agent A will increase p. By symmetry, the same is true for agent B. However, this means that the agents learn to collaborate less and less and will reach the local optimum of $r = 0$. This a terrible outcome given that the agents could have relied on each other to achieve a reward of 10!

Now consider the expected reward for a single joint agent that can choose any joint-action with probabilities $[(1-b-c-d), b, c, d]$ (ignoring other constraints):

$$J = -5b - 5c + 10d$$

Note that the gradient for d, the weight on 10, is now always positive, and policy always converges toward the joint action with the highest payoff. Thus, factorization prevents learning an otherwise learnable optimal policy.

Given the factorization of the joint policy into the product of marginals, our policy gradient succumbs to non-stationarity in the following way: each agent assumes that the other is part of an immutable environment and "pessimisticfally" moves towards policies that do not rely on changing that environment. Thus agents are susceptible to local sub-optimal Nash equilibria. This makes learning to collaborate difficult without an initialization already sufficiently collaborative.

Related Work

One of the notable recent policy-gradient methods is COMA (3). COMA approximates Q in the policy gradient above, and then subtracts a function from it, called a baseline, in order to minimize variance of the estimate. In particular, the baseline they choose to subtract is $\mathbb{E}[R|\mathbf{a}^-]$, where $\mathbf{a}^-$ is the action of every other agent. This helps with credit assignment, since the baseline for each agent is correlated with the actions of the other agents, but it does not change the policy gradient for a given agent. Although this reduces variance of Q estimates, the factorization of each agent leaves COMA susceptible to the specified problem above.

The other notable recent policy-gradient methods is MADDPG (4). MADDPG uses a deterministic policy gradient in order to learn a deterministic policy. Essentially, a critic is learned that approximates Q_{jt} for a given action, state, and the actions of all other agents. Conditioning on the actions of all agents allows the transitions to be

stationary for the learning of Q_{jt}. However, this Q_{jt} is then used to calculate a policy gradient for each agent so that the agents can be separated at runtime. Here, the gradient for each policy is determined in expectation over the actions of other agents filling a buffer. Thus, whatever policy for the other agents is in the buffer will determine the policy for each agent. Again, if the other agents aren't collaborating in the buffer, the gradient will not encourage collaboration.

In addition to policy-gradient approaches, value-based methods have the same problem for a slightly different reason. The first value-based method we consider is QMix (5). The solution in QMix is to learn a joint Q_{jt} function that takes in all observations and actions. However, unlike the other methods which use value networks as a throw-away-component to train policy networks, the goal of QMix is to retain and use this joint Q network at runtime. In order to do so, QMix learns a joint Q_{jt}function that is forced to be a monotonic function of each individual agent's Q-function. Thus maximizing the individual Q functions at runtime produces an action that maximizes the joint Q_{jt}. However, this monotonicity is more restrictive than it needs to be to enforce this factorization of max's. In particular, as noted in the paper, it will fail to represent the same matrix game I have posed. To be precise, QMix can only represent payoff matrices that can, for some monotonic function f be written:

$0 \stackrel{?}{=} f(a_0, b_0)$	$-5 \stackrel{?}{=} f(a_1, b_0)$
$-5 \stackrel{?}{=} f(a_0, b_1)$	$10 \stackrel{?}{=} f(a_1, b_1)$

Note that there is not a monotonic function f that can represent the values in the matrix above, since $f(a_0, b_0) > f(a_1, b_0)$ implies $a_0 > a_1$, which implies $f(a_0, b_1) > f(a_1, b_1)$, which is false. Similar statements can be made for $b_0 > b_1$, which means that the bottom right in our given matrix would have to have the lowest payoff, but it has the highest. Again, we can see that this will succumb to local Nash equilibria.

Next, we consider the value-based QTran (6). First, it is important to note that there exists a precursor to QMix called VDN (7), which composes the joint Q-value from a summation of the individual Q-values, which has a strictly narrower solution set than QMix. Much like VDN, QTran predicts joint Q values by summing up individual Q values, into what I will call Q_s, but also learns true joint Q_{jt} values, conditioned on all observations and actions, and a true joint V_{jt} function. Instead of using these to train a policy network, QTran uses the joint value functions to ensure that i) $Q_s + V_{jt}$ matches the actual joint Q_{jt} optimal action and ii) $Q_s + V_{jt}$ is an underestimate for the non-optimal actions. They prove that by doing this, choosing the argmax of the individual Q functions will maximize the joint Q_{jt} function. Thus, although some matrices still cannot be accurately represented after discarding the joint training network, it should not matter. QTran is insightful but has 2 major drawbacks: First, It requires adding to the normal TD loss for Q-Learning, a loss to encourage condition i) to be met and a loss to encourage condition ii) to be met. It is not clear how to choose the two hyper-parameters representing the loss weightings, how to tell if the conditions are satisfied, and how bad the policy will be if the conditions are not completely satisfied at termination, which they likely will not be in every possible state. Second, it requires Q-learning instead of policy gradient methods. Policy gradient methods can be useful: they allow for using stochasticity to resolve states aliased by partial observably, and are often much simpler to learn and represent.

Finally, it is worth noting MACKRL (8), which allows for collaboration by deciding to either 1) utilize a copy of the same joint network for each agent to condition on the same common knowledge (CK), producing the same joint actions, or 2) defer to individual policies to condition on private information. However, MACKRL must know CK in advance and cannot make decisions based on both CK and private knowledge without breaking coordination; it can only use one or the other. This fails, for example, if a multi-dimensional output requires a collaborative and private dimension. Thus MACKRL will only solve collaboration in the special case where each agent does not need to use their private information.

Proposed Method

The goal of this proposal is a policy-gradient method that does not get stuck in local sub-optimal Nash equilibria, despite eventually being factored into separate policy networks. Although there are many potential approaches, I focus on an elegant approach that constitutes a reasonable starting place for research.

Minimize Mutual Information All of the related work examined learns a joint value network of some kind. We can also consider learning a joint policy network, and extracting the necessary information from this. We need the joint action distribution to factor as a product of the marginals given each agent's observation, but we do not need this to be the case right away. Therefore, my proposed solution is the following: learn a joint policy: $\pi_{jt}(\mathbf{s})$ and add to the loss one additional term $\lambda D(\pi_{jt}(\mathbf{S} = \mathbf{s}); \pi_0(S_0 = s_0)\pi_1(S_1 = s_1))$, where D is the Kullback–Leibler divergence. The second term in this loss can be viewed as the weighted mutual information, which is minimized when the two distributions are identical. Thus, if we start λ as 0, our network will learn the collaborative joint policy. Then we can increase λ until we are sure that the policy is factored. By doing this, we are not simply learning a joint policy and then trying to factor it, but rather learning the optimal joint policy that must factorize. In order to make the network amenable to factorization, I suggest sharing most of the parameters between the networks. Finally, instead of hoping that the factorization worked, we can architect the output of the joint network so that it is simply a convex combination of a joint network and the product of the marginals. Thus, we can slowly change our action distribution from entirely the arbitrary joint distribution to entirely the product of the marginals. By the end, we ensure that 1) We wind up back at a policy gradient approach, but after a collaborative initialization; 2) The necessary constraint – conditional independence – is satisfied.

Conclusion

In conclusion, I have laid out an issue with current collaborative deep CTDE MARL approaches, shown how it is not tackled in the policy-gradient setting, and proposed a starting place for such research. Although this is a starting place, it is worth noting that there is likely a wide space of solution in this area. Potential research could additionally explore a modification to the CDTE framework in which a shared random seed is permitted at runtime, based on a known and globally observable source of stochasticity.

A common theme throughout all the related work has been to learn a some function of the joint action space. Although the size of the joint action space increases exponentially with the number of agents, learning some function of the joint actions seems necessary to learn policies that depend on the actions of other agents. That is, we may have made the search space larger, but only because in general for optimal collaborative solutions, the space needs to be that large.

Bibliography

1. Frans A. Oliehoek, Matthijs T. J. Spaan, and Nikos A. Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *AAAI*, 2008.
2. Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, 1999.
3. Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI*, 2017.
4. Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *NIPS*, 2017.
5. Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *ICML*, 2018.
6. Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *ICML*, 2019.
7. Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning. In *AAMAS*, 2017.
8. Christian Schroeder de Witt, Jakob Foerster, Gregory Farquhar, Philip Torr, Wendelin Boehmer, and Shimon Whiteson. Multi-agent common knowledge reinforcement learning. *NIPS*, 2019.