Counterfactual Multi-Agent Policy Gradients

Jakob N. Foerster[1,†]
jakob.foerster@cs.ox.ac.uk

Gregory Farquhar[1,†]
gregory.farquhar@cs.ox.ac.uk

Triantafyllos Afouras[1]
afourast@robots.ox.ac.uk

Nantas Nardelli[1]
nantas@robots.ox.ac.uk

Shimon Whiteson[1]
shimon.whiteson@cs.ox.ac.uk

[1]University of Oxford, United Kingdom [†]Equal contribution

Abstract

Many real-world problems, such as network packet routing and the coordination of autonomous vehicles, are naturally modelled as cooperative multi-agent systems. There is a great need for new reinforcement learning methods that can efficiently learn decentralised policies for such systems. To this end, we propose a new multi-agent actor-critic method called *counterfactual multi-agent* (COMA) policy gradients. COMA uses a centralised critic to estimate the Q-function and decentralised actors to optimise the agents' policies. In addition, to address the challenges of multi-agent credit assignment, it uses a *counterfactual baseline* that marginalises out a single agent's action, while keeping the other agents' actions fixed. COMA also uses a critic representation that allows the counterfactual baseline to be computed efficiently in a single forward pass. We evaluate COMA in the testbed of *StarCraft unit micromanagement*, using a decentralised variant with significant partial observability. COMA significantly improves average performance over other multi-agent actor-critic methods in this setting, and the best performing agents are competitive with state-of-the-art centralised controllers that get access to the full state.

1 Introduction

Many complex *reinforcement learning* (RL) problems such as the coordination of autonomous vehicles (Cao et al. 2013), network packet delivery (Ye, Zhang, and Yang 2015), and distributed logistics (Ying and Dayong 2005) are naturally modelled as cooperative multi-agent systems. However, RL methods designed for single agents typically fare poorly on such tasks, since the joint action space of the agents grows exponentially with the number of agents.

To cope with such complexity, it is often necessary to resort to *decentralised policies*, in which each agent selects its own action conditioned only on its local action-observation history. Furthermore, partial observability and communication constraints during execution may necessitate the use of decentralised policies even when the joint action space is not prohibitively large.

Hence, there is a great need for new RL methods that can efficiently learn decentralised policies. In some settings, the learning itself may also need to be decentralised. However, in many cases, learning can take place in a simulator or a laboratory in which extra state information is available and agents can communicate freely. This *centralised training of decentralised policies* is a standard paradigm for multi-agent planning (Oliehoek, Spaan, and Vlassis 2008; Kraemer and Banerjee 2016) and has recently been picked up by the deep RL community (Foerster et al. 2016; Jorge, Kågebäck, and Gustavsson 2016). However, the question of how best to exploit the opportunity for centralised learning remains open.

Another crucial challenge is *multi-agent credit assignment* (Chang, Ho, and Kaelbling 2003): in cooperative settings, joint actions typically generate only global rewards, making it difficult for each agent to deduce its own contribution to the team's success. Sometimes it is possible to design individual reward functions for each agent. However, these rewards are not generally available in cooperative settings and often fail to encourage individual agents to sacrifice for the greater good. This often substantially impedes multi-agent learning in challenging tasks, even with relatively small numbers of agents.

In this paper, we propose a new multi-agent RL method called *counterfactual multi-agent* (COMA) policy gradients, in order to address these issues. COMA takes an *actor-critic* (Konda and Tsitsiklis 2000) approach, in which the *actor*, i.e., the policy, is trained by following a gradient estimated by a *critic*. COMA is based on three main ideas.

First, COMA uses a centralised critic. The critic is only used during learning, while only the actor is needed during execution. Since learning is centralised, we can therefore use a centralised critic that conditions on the joint action and all available state information, while each agent's policy conditions only on its own action-observation history.

Second, COMA uses a *counterfactual baseline*. The idea is inspired by *difference rewards* (Wolpert and Tumer 2002; Tumer and Agogino 2007), in which each agent learns from a shaped reward that compares the global reward to the reward received when that agent's action is replaced with a *default action*. While difference rewards are a powerful way to perform multi-agent credit assignment, they require access to a simulator or estimated reward function, and in general it is unclear how to choose the default action. COMA addresses this by using the centralised critic to compute an agent-specific *advantage function* that compares the estimated return for the current joint action to a counterfactual baseline that marginalises out a single agent's action, while keeping the other agents' actions fixed. This is similar to cal-

culating an *aristocrat utility* (Wolpert and Tumer 2002), but avoids the problem of a recursive interdependence between the policy and utility function because the expected contribution of the counterfactual baseline to the policy gradient is zero. Hence, instead of relying on extra simulations, approximations, or assumptions regarding appropriate default actions, COMA computes a separate baseline for each agent that relies on the centralised critic to reason about counterfactuals in which only that agent's action changes.

Third, COMA uses a critic representation that allows the counterfactual baseline to be computed efficiently. In a single forward pass, it computes the Q-values for all the different actions of a given agent, conditioned on the actions of all the other agents. Because a single centralised critic is used for all agents, all Q-values for all agents can be computed in a single batched forward pass.

We evaluate COMA in the testbed of *StarCraft unit micromanagement*[1], which has recently emerged as a challenging RL benchmark task with high stochasticity, a large state-action space, and delayed rewards. Previous works (Usunier et al. 2016; Peng et al. 2017) have made use of a centralised control policy that conditions on the entire state and can use powerful macro-actions, using StarCraft's built-in planner, that combine movement and attack actions. To produce a meaningfully decentralised benchmark that proves challenging for scenarios with even relatively few agents, we propose a variant that massively reduces each agent's field-of-view and removes access to these macro-actions.

Our empirical results on this new benchmark show that COMA can significantly improve performance over other multi-agent actor-critic methods, as well as ablated versions of COMA itself. In addition, COMA's best agents are competitive with state-of-the-art centralised controllers that are given access to full state information and macro-actions.

2 Related Work

Although multi-agent RL has been applied in a variety of settings (Busoniu, Babuska, and De Schutter 2008; Yang and Gu 2004), it has often been restricted to tabular methods and simple environments. One exception is recent work in deep multi-agent RL, which can scale to high dimensional input and action spaces. Tampuu et al. (2015) use a combination of DQN with independent Q-learning (Tan 1993; Shoham and Leyton-Brown 2009) to learn how to play two-player pong. More recently the same method has been used by Leibo et al. (2017) to study the emergence of collaboration and defection in sequential social dilemmas.

Also related is work on the emergence of communication between agents, learned by gradient descent (Das et al. 2017; Mordatch and Abbeel 2017; Lazaridou, Peysakhovich, and Baroni 2016; Foerster et al. 2016; Sukhbaatar, Fergus, and others 2016). In this line of work, passing gradients between agents during training and sharing parameters are two common ways to take advantage of centralised training. However, these methods do not allow for extra state information to be used during learning and do not address the multi-agent credit assignment problem.

Gupta, Egorov, and Kochenderfer (2017) investigate actor-critic methods for decentralised execution with centralised training. However, in their methods both the actors and the critic condition on local, per-agent, observations and actions, and multi-agent credit assignment is addressed only with hand-crafted local rewards.

Most previous applications of RL to StarCraft micromanagement use a centralised controller, with access to the full state, and control of all units, although the architecture of the controllers exploits the multi-agent nature of the problem. Usunier et al. (2016) use a *greedy MDP*, which at each timestep sequentially chooses actions for agents given all previous actions, in combination with zero-order optimisation, while Peng et al. (2017) use an actor-critic method that relies on RNNs to exchange information between the agents.

The closest to our problem setting is that of Foerster et al. (2017), who also use a multi-agent representation and decentralised policies. However, they focus on stabilising experience replay while using DQN and do not make full use of the centralised training regime. As they do not report on absolute win-rates we do not compare performance directly. However, Usunier et al. (2016) address similar scenarios to our experiments and implement a DQN baseline in a fully observable setting. In Section 6 we therefore report our competitive performance against these state-of-the-art baselines, while maintaining decentralised control. Omidshafiei et al. (2017) also address the stability of experience replay in multi-agent settings, but assume a fully decentralised training regime.

(Lowe et al. 2017) concurrently propose a multi-agent policy-gradient algorithm using centralised critics. Their approach does not address multi-agent credit assignment. Unlike our work, it learns a separate centralised critic for each agent and is applied to competitive environments with continuous action spaces.

Our work builds directly off of the idea of *difference rewards* (Wolpert and Tumer 2002). The relationship of COMA to this line of work is discussed in Section 4.

3 Background

We consider a fully cooperative multi-agent task that can be described as a stochastic game G, defined by a tuple $G = \langle S, U, P, r, Z, O, n, \gamma \rangle$, in which n agents identified by $a \in A \equiv \{1, ..., n\}$ choose sequential actions. The environment has a true state $s \in S$. At each time step, each agent simultaneously chooses an action $u^a \in U$, forming a joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$ which induces a transition in the environment according to the state transition function $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \to [0, 1]$. The agents all share the same reward function $r(s, \mathbf{u}) : S \times \mathbf{U} \to \mathbb{R}$ and $\gamma \in [0, 1)$ is a discount factor.

We consider a partially observable setting, in which agents draw observations $z \in Z$ according to the observation function $O(s, a) : S \times A \to Z$. Each agent has an action-observation history $\tau^a \in T \equiv (Z \times U)^*$, on which it conditions a stochastic policy $\pi^a(u^a|\tau^a) : T \times U \to [0, 1]$.

[1] StarCraft and its expansion StarCraft: Brood War are trademarks of Blizzard Entertainment™.

We denote joint quantities over agents in bold, and joint quantities over agents other than a given agent a with the superscript $-a$.

The discounted return is $R_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$. The agents' joint policy induces a value function, i.e., an expectation over R_t, $V^{\boldsymbol{\pi}}(s_t) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{u}_{t:\infty}}[R_t|s_t]$, and an action-value function $Q^{\boldsymbol{\pi}}(s_t, \mathbf{u}_t) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{u}_{t+1:\infty}}[R_t|s_t, \mathbf{u}_t]$. The advantage function is given by $A^{\boldsymbol{\pi}}(s_t, \mathbf{u}_t) = Q^{\boldsymbol{\pi}}(s_t, \mathbf{u}_t) - V^{\boldsymbol{\pi}}(s_t)$.

Following previous work (Oliehoek, Spaan, and Vlassis 2008; Kraemer and Banerjee 2016; Foerster et al. 2016; Jorge, Kågebäck, and Gustavsson 2016), our problem setting allows centralised training but requires decentralised execution. This is a natural paradigm for a large set of multi-agent problems where training is carried out using a simulator with additional state information, but the agents must rely on local action-observation histories during execution. To condition on this full history, a deep RL agent may make use of a recurrent neural network (Hausknecht and Stone 2015), typically with a gated model such as LSTM (Hochreiter and Schmidhuber 1997) or GRU (Cho et al. 2014).

In Section 4, we develop a new multi-agent policy gradient method for tackling this setting. In the remainder of this section, we provide some background on single-agent policy gradient methods (Sutton et al. 1999). Such methods optimise a single agent's policy, parameterised by θ^{π}, by performing gradient ascent on an estimate of the expected discounted total reward $J = \mathbb{E}_{\pi}[R_0]$. Perhaps the simplest form of policy gradient is REINFORCE (Williams 1992), in which the gradient is:

$$g = \mathbb{E}_{s_{0:\infty}, u_{0:\infty}}\left[\sum_{t=0}^{T} R_t \nabla_{\theta^{\pi}} \log \pi(u_t|s_t)\right]. \tag{1}$$

In *actor-critic* approaches (Sutton et al. 1999; Konda and Tsitsiklis 2000; Schulman et al. 2015), the *actor*, i.e., the policy, is trained by following a gradient that depends on a *critic*, which usually estimates a value function. In particular, R_t is replaced by any expression equivalent to $Q(s_t, u_t) - b(s_t)$, where $b(s_t)$ is a baseline designed to reduce variance (Weaver and Tao 2001). A common choice is $b(s_t) = V(s_t)$, in which case R_t is replaced by $A(s_t, u_t)$. Another option is to replace R_t with the *temporal difference* (TD) error $r_t + \gamma V(s_{t+1}) - V(s)$, which is an unbiased estimate of $A(s_t, u_t)$. In practice, the gradient must be estimated from trajectories sampled from the environment, and the (action-)value functions must be estimated with function approximators. Consequently, the bias and variance of the gradient estimate depends strongly on the exact choice of estimator (Konda and Tsitsiklis 2000).

In this paper, we train critics $f^c(\cdot, \theta^c)$ on-policy to estimate either Q or V, using a variant of TD(λ) (Sutton 1988) adapted for use with deep neural networks. TD(λ) uses a mixture of n-step returns $G_t^{(n)} = \sum_{l=1}^{n} \gamma^{l-1} r_{t+l} + \gamma^n f^c(\cdot_{t+n}, \theta^c)$. In particular, the critic parameters θ^c are updated by minibatch gradient descent to minimise the following loss:

$$\mathcal{L}_t(\theta^c) = (y^{(\lambda)} - f^c(\cdot_t, \theta^c))^2, \tag{2}$$

where $y^{(\lambda)} = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$, and the n-step returns $G_t^{(n)}$ are calculated with bootstrapped values estimated by a *target network* (Mnih et al. 2015) with parameters copied periodically from θ^c.

4 Methods

In this section, we describe approaches for extending policy gradients to our multi-agent setting.

Independent Actor-Critic

The simplest way to apply policy gradients to multiple agents is to have each agent learn independently, with its own actor and critic, from its own action-observation history. This is essentially the idea behind *independent Q-learning* (Tan 1993), which is perhaps the most popular multi-agent learning algorithm, but with actor-critic in place of Q-learning. Hence, we call this approach *independent actor-critic* (IAC).

In our implementation of IAC, we speed learning by sharing parameters among the agents, i.e., we learn only one actor and one critic, which are used by all agents. The agents can still behave differently because they receive different observations, including an agent-specific ID, and thus evolve different hidden states. Learning remains independent in the sense that each agent's critic estimates only a local value function, i.e., one that conditions on u^a, not $\mathbf{u}$. Though we are not aware of previous applications of this specific algorithm, we do not consider it a significant contribution but instead merely a baseline algorithm.

We consider two variants of IAC. In the first, each agent's critic estimates $V(\tau^a)$ and follows a gradient based on the TD error, as described in Section 3. In the second, each agent's critic estimates $Q(\tau^a, u^a)$ and follows a gradient based on the advantage: $A(\tau^a, u^a) = Q(\tau^a, u^a) - V(\tau^a)$, where $V(\tau^a) = \sum_{u^a} \pi(u^a|\tau^a) Q(\tau^a, u^a)$. Independent learning is straightforward, but the lack of information sharing at training time makes it difficult to learn coordinated strategies that depend on interactions between multiple agents, or for an individual agent to estimate the contribution of its actions to the team's reward.

Counterfactual Multi-Agent Policy Gradients

The difficulties discussed above arise because, beyond parameter sharing, IAC fails to exploit the fact that learning is centralised in our setting. In this section, we propose *counterfactual multi-agent* (COMA) policy gradients, which overcome this limitation. Three main ideas underly COMA: 1) centralisation of the critic, 2) use of a counterfactual baseline, and 3) use of a critic representation that allows efficient evaluation of the baseline. The remainder of this section describes these ideas.

First, COMA uses a centralised critic. Note that in IAC, each actor $\pi(u^a|\tau^a)$ and each critic $Q(\tau^a, u^a)$ or $V(\tau^a)$ conditions only on the agent's own action-observation history τ^a. However, the critic is used only during learning and only the actor is needed during execution. Since learning is centralised, we can therefore use a centralised critic that conditions on the true global state s, if it is available, or the joint

action-observation histories $\boldsymbol{\tau}$ otherwise. Each actor conditions on its own action-observation histories τ^a, with parameter sharing, as in IAC. Figure 1a illustrates this setup.

A naive way to use this centralised critic would be for each actor to follow a gradient based on the TD error estimated from this critic:

$$g = \nabla_{\theta^\pi} \log \pi(u|\tau_t^a) \left(r + \gamma V(s_{t+1}) - V(s_t)\right). \quad (3)$$

However, such an approach fails to address a key credit assignment problem. Because the TD error considers only global rewards, the gradient computed for each actor does not explicitly reason about how that particular agent's actions contribute to that global reward. Since the other agents may be exploring, the gradient for that agent becomes very noisy, particularly when there are many agents.

Therefore, COMA uses a *counterfactual baseline*. The idea is inspired by *difference rewards* (Wolpert and Tumer 2002), in which each agent learns from a shaped reward $D^a = r(s, \mathbf{u}) - r(s, (\mathbf{u}^{-a}, c^a))$ that compares the global reward to the reward received when the action of agent a is replaced with a *default action* c^a. Any action by agent a that improves D^a also improves the true global reward $r(s, \mathbf{u})$, because $r(s, (\mathbf{u}^{-a}, c^a))$ does not depend on agent a's actions.

Difference rewards are a powerful way to perform multi-agent credit assignment. However, they typically require access to a simulator in order to estimate $r(s, (\mathbf{u}^{-a}, c^a))$. When a simulator is already being used for learning, difference rewards increase the number of simulations that must be conducted, since each agent's difference reward requires a separate counterfactual simulation. Proper and Tumer (2012) and Colby, Curran, and Tumer (2015) propose estimating difference rewards using function approximation rather than a simulator. However, this still requires a user-specified default action c^a that can be difficult to choose in many applications. In an actor-critic architecture, this approach would also introduce an additional source of approximation error.

A key insight underlying COMA is that a centralised critic can be used to implement difference rewards in a way that avoids these problems. COMA learns a centralised critic, $Q(s, \mathbf{u})$ that estimates Q-values for the joint action $\mathbf{u}$ conditioned on the central state s. For each agent a we can then compute an advantage function that compares the Q-value for the current action u^a to a counterfactual baseline that marginalises out u^a, while keeping the other agents' actions $\mathbf{u}^{-a}$ fixed:

$$A^a(s, \mathbf{u}) = Q(s, \mathbf{u}) - \sum_{u'^a} \pi^a(u'^a|\tau^a) Q(s, (\mathbf{u}^{-a}, u'^a)). \quad (4)$$

Hence, $A^a(s, u^a)$ computes a separate baseline for each agent that uses the centralised critic to reason about counterfactuals in which only a's action changes, learned directly from agents' experiences instead of relying on extra simulations, a reward model, or a user-designed default action.

This advantage has the same form as the *aristocrat utility* (Wolpert and Tumer 2002). However, optimising for an aristocrat utility using value-based methods creates a self-consistency problem because the policy and utility function depend recursively on each other. As a result, prior work focused on difference evaluations using default states and actions. COMA is different because the counterfactual baseline's expected contribution to the gradient, as with other policy gradient baselines, is zero. Thus, while the baseline does depend on the policy, its expectation does not. Consequently, COMA can use this form of the advantage without creating a self-consistency problem.

While COMA's advantage function replaces potential extra simulations with evaluations of the critic, those evaluations may themselves be expensive if the critic is a deep neural network. Furthermore, in a typical representation, the number of output nodes of such a network would equal $|U|^n$, the size of the joint action space, making it impractical to train. To address both these issues, COMA uses a critic representation that allows for efficient evaluation of the baseline. In particular, the actions of the other agents, $\mathbf{u}_t^{-a}$, are part of the input to the network, which outputs a Q-value for each of agent a's actions, as shown in Figure 1c. Consequently, the counterfactual advantage can be calculated efficiently by a single forward pass of the actor and critic, for each agent. Furthermore, the number of outputs is only $|U|$ instead of ($|U|^n$). While the network has a large input space that scales linearly in the number of agents and actions, deep neural networks can generalise well across such spaces.

In this paper, we focus on settings with discrete actions. However, COMA can be easily extended to continuous actions spaces by estimating the expectation in (4) with Monte Carlo samples or using functional forms that render it analytical, e.g., Gaussian policies and critic.

The following lemma establishes the convergence of COMA to a locally optimal policy. The proof follows directly from the convergence of single-agent actor-critic algorithms (Sutton et al. 1999; Konda and Tsitsiklis 2000), and is subject to the same assumptions.

Lemma 1. *For an actor-critic algorithm with a compatible TD(1) critic following a COMA policy gradient*

$$g_k = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_a \nabla_{\theta_k} \log \pi^a(u^a|\tau^a) A^a(s, \mathbf{u}) \right] \quad (5)$$

at each iteration k,

$$\liminf_k ||\nabla J|| = 0 \quad w.p.\ 1. \quad (6)$$

Proof. The COMA gradient is given by

$$g = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_a \nabla_\theta \log \pi^a(u^a|\tau^a) A^a(s, \mathbf{u}) \right], \quad (7)$$

$$A^a(s, \mathbf{u}) = Q(s, \mathbf{u}) - b(s, \mathbf{u}^{-a}), \quad (8)$$

where θ are the parameters of all actor policies, e.g. $\theta = \{\theta^1, \ldots, \theta^{|A|}\}$, and $b(s, \mathbf{u}^{-a})$ is the counterfactual baseline defined in equation 4.

First consider the expected contribution of the this baseline $b(s, \mathbf{u}^{-a})$:

$$g_b = -\mathbb{E}_{\boldsymbol{\pi}} \left[\sum_a \nabla_\theta \log \pi^a(u^a|\tau^a) b(s, \mathbf{u}^{-a}) \right], \quad (9)$$

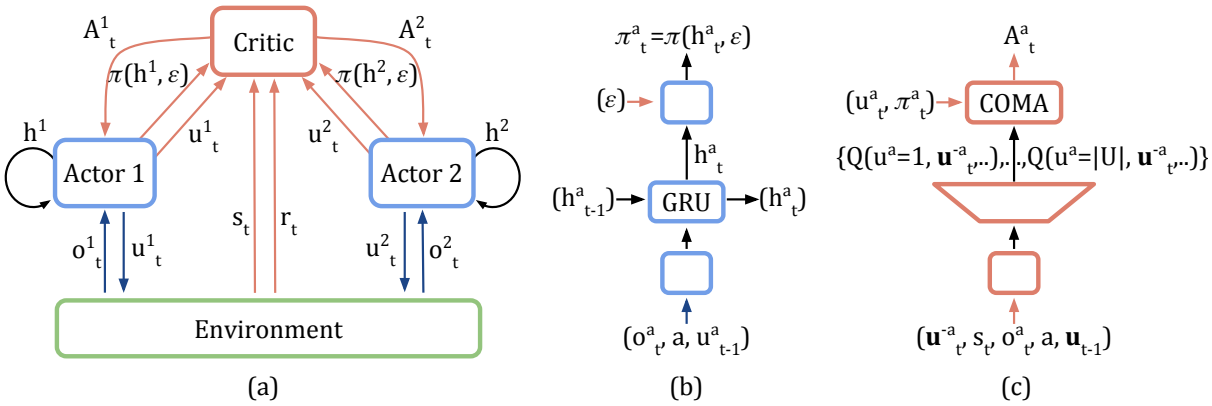


Figure 1: In (a), information flow between the decentralised actors, the environment and the centralised critic in COMA; red arrows and components are only required during centralised learning. In (b) and (c), architectures of the actor and critic.

where the expectation $\mathbb{E}_{\boldsymbol{\pi}}$ is with respect to the state-action distribution induced by the joint policy $\boldsymbol{\pi}$. Now let $d^{\boldsymbol{\pi}}(s)$ be the discounted ergodic state distribution as defined by Sutton et al. (1999):

$$
\begin{aligned}
g_b &= -\sum_s d^{\boldsymbol{\pi}}(s) \sum_a \sum_{\mathbf{u}^{-a}} \boldsymbol{\pi}(\mathbf{u}^{-a}|\boldsymbol{\tau}-a) \cdot \\
&\quad \sum_{u^a} \pi^a(u^a|\tau^a) \nabla_\theta \log \pi^a(u^a|\tau^a) b(s, \mathbf{u}^{-a}) \qquad (10) \\
&= -\sum_s d^{\boldsymbol{\pi}}(s) \sum_a \sum_{\mathbf{u}^{-a}} \boldsymbol{\pi}(\mathbf{u}^{-a}|\boldsymbol{\tau}-a) \cdot \\
&\quad \sum_{u^a} \nabla_\theta \pi^a(u^a|\tau^a) b(s, \mathbf{u}^{-a}) \qquad (11) \\
&= -\sum_s d^{\boldsymbol{\pi}}(s) \sum_a \sum_{\mathbf{u}^{-a}} \boldsymbol{\pi}(\mathbf{u}^{-a}|\boldsymbol{\tau}-a) b(s, \mathbf{u}^{-a}) \nabla_\theta 1 \\
&= 0. \qquad (12)
\end{aligned}
$$

Clearly, the per-agent baseline, although it reduces variance, does not change the expected gradient, and therefore does not affect the convergence of COMA.

The remainder of the expected policy gradient is given by:

$$
\begin{aligned}
g &= \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_a \nabla_\theta \log \pi^a(u^a|\tau^a) Q(s, \mathbf{u}) \right] \qquad (13) \\
&= \mathbb{E}_{\boldsymbol{\pi}} \left[\nabla_\theta \log \prod_a \pi^a(u^a|\tau^a) Q(s, \mathbf{u}) \right]. \qquad (14)
\end{aligned}
$$

Writing the joint policy as a product of the independent actors:

$$
\boldsymbol{\pi}(\mathbf{u}|s) = \prod_a \pi^a(u^a|\tau^a), \qquad (15)
$$

yields the standard single-agent actor-critic policy gradient:

$$
g = \mathbb{E}_{\boldsymbol{\pi}} \left[\nabla_\theta \log \boldsymbol{\pi}(\mathbf{u}|s) Q(s, \mathbf{u}) \right]. \qquad (16)
$$

Konda and Tsitsiklis (2000) prove that an actor-critic following this gradient converges to a local maximum of the expected return $J^{\boldsymbol{\pi}}$, given that:

1. the policy $\boldsymbol{\pi}$ is differentiable,
2. the update timescales for Q and $\boldsymbol{\pi}$ are sufficiently slow, and that $\boldsymbol{\pi}$ is updated sufficiently slower than Q, and
3. Q uses a representation compatible with $\boldsymbol{\pi}$,

amongst several further assumptions. The parameterisation of the policy (i.e., the single-agent joint-action learner is decomposed into independent actors) is immaterial to convergence, as long as it remains differentiable. Note however that COMA's centralised critic is essential for this proof to hold. □

5 Experimental Setup

In this section, we describe the StarCraft problem to which we apply COMA, as well as details of the state features, network architectures, training regimes, and ablations.

Decentralised StarCraft Micromanagement. StarCraft is a rich environment with stochastic dynamics that cannot be easily emulated. Many simpler multi-agent settings, such as Predator-Prey (Tan 1993) or Packet World (Weyns, Helleboogh, and Holvoet 2005), by contrast, have full simulators with controlled randomness that can be freely set to any state in order to perfectly replay experiences. This makes it possible, though computationally expensive, to compute difference rewards via extra simulations. In StarCraft, as in the real world, this is not possible.

In this paper, we focus on the problem of *micromanagement* in StarCraft, which refers to the low-level control of individual units' positioning and attack commands as they fight enemies. This task is naturally represented as a multi-agent system, where each StarCraft unit is replaced by a decentralised controller. We consider several scenarios with symmetric teams formed of: 3 marines (3m), 5 marines (5m), 5 wraiths (5w), or 2 dragoons with 3 zealots (2d_3z). The enemy team is controlled by the StarCraft AI, which uses reasonable but suboptimal hand-crafted heuristics.

We allow the agents to choose from a set of discrete actions: `move[direction]`, `attack[enemy_id]`, `stop`, and `noop`. In the StarCraft game, when a unit selects an attack action, it first moves into attack range before firing, using the game's built-in pathfinding to choose a

route. These powerful *attack-move* macro-actions make the control problem considerably easier.

To create a more challenging benchmark that is meaningfully decentralised, we impose a restricted field of view on the agents, equal to the firing range of ranged units' weapons, shown in Figure 2. This departure from the standard setup for centralised StarCraft control has three effects.

Figure 2: Starting position with example local field of view for the 2d_3z map.

First, it introduces significant partial observability. Second, it means units can only attack when they are in range of enemies, removing access to the StarCraft macro-actions. Third, agents cannot distinguish between enemies who are dead and those who are out of range and so can issue invalid attack commands at such enemies, which results in no action being taken. This substantially increases the average size of the action space, which in turn increases the difficulty of both exploration and control.

Under these difficult conditions, scenarios with even relatively small numbers of units become much harder to solve. As seen in Table 1, we compare against a simple hand-coded heuristic that instructs the agents to run forwards into range and then focus their fire, attacking each enemy in turn until it dies. This heuristic achieves a 98% win rate on 5m with a full field of view, but only 66% in our setting. To perform well in this task, the agents must learn to cooperate by positioning properly and focussing their fire, while remembering which enemy and ally units are alive or out of view.

All agents receive the same global reward at each time step, equal to the sum of damage inflicted on the opponent units minus half the damage taken. Killing an opponent generates a reward of 10 points, and winning the game generates a reward equal to the team's remaining total health plus 200. This damage-based reward signal is comparable to that used by Usunier et al. (2016). Unlike (Peng et al. 2017), our approach does not require estimating local rewards.

State Features. The actor and critic receive different input features, corresponding to local observations and global state, respectively. Both include features for allies and enemies. *Units* can be either allies or enemies, while *agents* are the decentralised controllers that command ally units.

The local observations for every agent are drawn only from a circular subset of the map centred on the unit it controls and include for each unit within this field of view: `distance`, `relative x`, `relative y`, `unit type` and `shield`.[2] All features are normalised by their maximum values. We do not include any information about the units' current target.

The global state representation consists of similar features, but for all units on the map regardless of fields of view. Absolute distance is not included, and x-y locations are given relative to the centre of the map rather than to a particular agent. The global state also includes `health points` and `cooldown` for all agents. The representation fed to the centralised Q-function critic is the concatenation of the global state representation with the local observation of the agent whose actions are being evaluated. Our centralised critic that estimates $V(s)$, and is therefore agent-agnostic, receives the global state concatenated with all agents' observations. The observations contain no new information but include the egocentric distances relative to that agent.

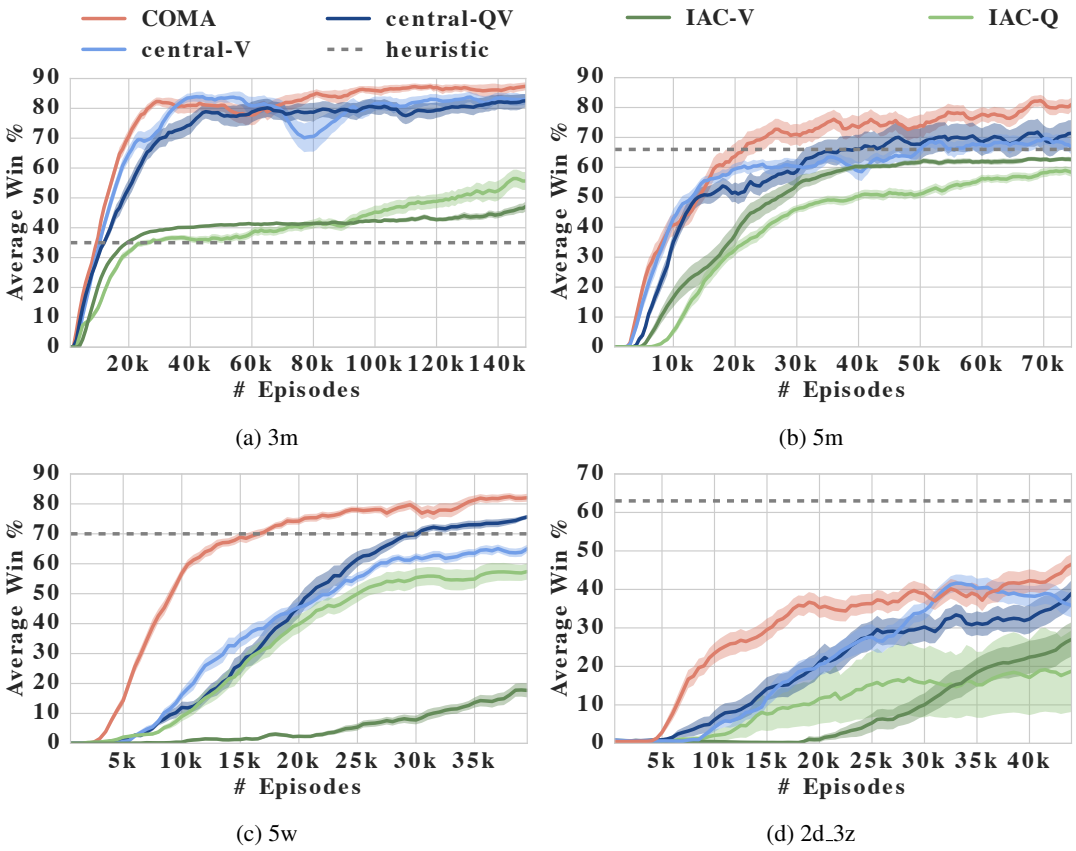
Architecture & Training. The actor consists of 128-bit *gated recurrent units* (GRUs) (Cho et al. 2014) that use fully connected layers both to process the input and to produce the output values from the hidden state, h_t^a. The IAC critics use extra output heads appended to the last layer of the actor network. Action probabilities are produced from the final layer, $\mathbf{z}$, via a bounded softmax distribution that lower-bounds the probability of any given action by $\epsilon/|U|$: $P(u) = (1-\epsilon)\text{softmax}(\mathbf{z})_u + \epsilon/|U|)$. We anneal ϵ linearly from 0.5 to 0.02 across 750 training episodes. The centralised critic is a feedforward network with multiple ReLU layers combined with fully connected layers. Hyperparameters were coarsely tuned on the 5m scenario and then used for all other maps. We found that the most sensitive parameter was TD(λ), but settled on $\lambda = 0.8$, which worked best for both COMA and our baselines. Our implementation uses TorchCraft (Synnaeve et al. 2016) and Torch 7 (Collobert, Kavukcuoglu, and Farabet 2011). Pseudocode and further details on the training procedure are in the appendix.

We experimented with critic architectures that are factored at the agent level and further exploit internal parameter sharing. However, we found that the bottleneck for scalability was not the centralisation of the critic, but rather the difficulty of multi-agent exploration. Hence, we defer further investigation of factored COMA critics to future work.

Ablations. We perform ablation experiments to validate three key elements of COMA. First, we test the importance of centralising the critic by comparing against two IAC variants, IAC-Q and IAC-V. These critics take the same decentralised input as the actor, and share parameters with the actor network up to the final layer. IAC-Q then outputs $|U|$ Q-values, one for each action, while IAC-V outputs a single state-value. Note that we still share parameters between agents, using the egocentric observations and ID's as part of the input to allow different behaviours to emerge. The cooperative reward function is still shared by all agents.

Second, we test the significance of learning Q instead of

[2] After firing, a unit's `cooldown` is reset, and it must drop before firing again. Shields absorb damage until they break, after which units start losing health. Dragoons and zealots have shields but marines do not.



(a) 3m

(b) 5m

(c) 5w

(d) 2d_3z

Figure 3: Win rates for COMA and competing algorithms on four different scenarios. COMA outperforms all baseline methods. Centralised critics also clearly outperform their decentralised counterparts. The legend at the top applies across all plots.

V. The method *central-V* still uses a central state for the critic, but learns $V(s)$, and uses the TD error to estimate the advantage for policy gradient updates.

Third, we test the utility of our counterfactual baseline. The method *central-QV* learns both Q and V simultaneously and estimates the advantage as $Q - V$, replacing COMA's counterfactual baseline with V. All methods use the same architecture and training scheme for the actors, and all critics are trained with TD(λ).

6 Results

Figure 3 shows average win rates as a function of episode for each method and each StarCraft scenario. For each method, we conducted 35 independent trials and froze learning every 100 training episodes to evaluate the learned policies across 200 episodes per method, plotting the average across episodes and trials. Also shown is one standard deviation in performance.

The results show that COMA is superior to the IAC baselines in all scenarios. Interestingly, the IAC methods also eventually learn reasonable policies in 5m, although they need substantially more episodes to do so. This may seem counterintuitive since in the IAC methods, the actor and critic networks share parameters in their early layers (see Section 5), which could be expected to speed learning. However, these results suggest that the improved accuracy of policy evaluation made possible by conditioning on the global state outweighs the overhead of training a separate network.

Furthermore, COMA strictly dominates central-QV, both in training speed and in final performance across all settings. This is a strong indicator that our counterfactual baseline is crucial when using a central Q-critic to train decentralised policies.

Learning a state-value function has the obvious advantage of not conditioning on the joint action. Still, we find that COMA outperforms the central-V baseline in final performance. Furthermore, COMA typically achieves good policies faster, which is expected as COMA provides a shaped training signal. Training is also more stable than central-V, which is a consequence of the COMA gradient tending to zero as the policy becomes greedy. Overall, COMA is the best performing and most consistent method.

Usunier et al. (2016) report the performance of their best agents trained with their state-of-the-art centralised controller labelled GMEZO (greedy-MDP with episodic zero-order optimisation), and for a centralised DQN controller, both given a full field of view and access to attack-move macro-actions. These results are compared in Table 1 against the best agents trained with COMA for each map. Clearly,

	Local Field of View (FoV)							Full FoV, Central Control		
map	heur.	IAC-V	IAC-Q	cnt-V	cnt-QV	COMA mean	COMA best	heur.	DQN	GMEZO
3m	35	47 (3)	56 (6)	83 (3)	83 (5)	**87** (3)	98	74	-	-
5m	66	63 (2)	58 (3)	67 (5)	71 (9)	**81** (5)	95	98	99	100
5w	70	18 (5)	57 (5)	65 (3)	76 (1)	**82** (3)	98	82	70	74[3]
2d_3z	**63**	27 (9)	19 (21)	36 (6)	39 (5)	47 (5)	65	68	61	90

Table 1: Mean win percentage averaged across final 1000 evaluation episodes for the different maps, for all methods and the hand-coded heuristic in the decentralised setting with a limited field of view. The highest mean performances are in bold, while the values in parentheses denote the 95% confidence interval, for example $87(3) = 87 \pm 3$. Also shown, maximum win percentages for COMA (decentralised), in comparison to the heuristic and published results (evaluated in the centralised setting).

in most settings these agents achieve performance comparable to the best published win rates despite being restricted to decentralised policies and local fields of view.

7 Conclusions & Future Work

This paper presented COMA policy gradients, a method that uses a centralised critic in order to estimate a counterfactual advantage for decentralised policies in mutli-agent RL. COMA addresses the challenges of multi-agent credit assignment by using a counterfactual baseline that marginalises out a single agent's action, while keeping the other agents' actions fixed. Our results in a decentralised *StarCraft unit micromanagement* benchmark show that COMA significantly improves final performance and training speed over other multi-agent actor-critic methods and remains competitive with state-of-the-art centralised controllers under best-performance reporting. Future work will extend COMA to tackle scenarios with large numbers of agents, where centralised critics are more difficult to train and exploration is harder to coordinate. We also aim to develop more sample-efficient variants that are practical for real-world applications such as self-driving cars.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement number 637713). It was also supported by the Oxford-Google DeepMind Graduate Scholarship, the UK EPSRC CDT in Autonomous Intelligent Machines and Systems, and a generous grant from Microsoft for their Azure cloud computing services. We would like to thank Nando de Freitas, Yannis Assael, and Brendan Shillingford for helpful comments and discussion. We also thank Gabriel Synnaeve, Zeming Lin, and the rest of the TorchCraft team at FAIR for their work on the interface.

[3] 5w DQN and GMEZO benchmark performances are of a policy trained on a larger map and tested on 5w

References

Busoniu, L.; Babuska, R.; and De Schutter, B. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews* 38(2):156.

Cao, Y.; Yu, W.; Ren, W.; and Chen, G. 2013. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics* 9(1):427–438.

Chang, Y.-H.; Ho, T.; and Kaelbling, L. P. 2003. All learning is local: Multi-agent learning in global reward games. In *NIPS*, 807–814.

Cho, K.; van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Colby, M. K.; Curran, W.; and Tumer, K. 2015. Approximating difference evaluations with local information. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 1659–1660. International Foundation for Autonomous Agents and Multiagent Systems.

Collobert, R.; Kavukcuoglu, K.; and Farabet, C. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*.

Das, A.; Kottur, S.; Moura, J. M.; Lee, S.; and Batra, D. 2017. Learning cooperative visual dialog agents with deep reinforcement learning. *arXiv preprint arXiv:1703.06585*.

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.; Nardelli, N.; Farquhar, G.; Torr, P.; Kohli, P.; Whiteson, S.; et al. 2017. Stabilising experience replay for deep multi-agent reinforcement learning. In *Proceedings of The 34th International Conference on Machine Learning*.

Gupta, J. K.; Egorov, M.; and Kochenderfer, M. 2017. Coop-

erative multi-agent control using deep reinforcement learning.

Hausknecht, M., and Stone, P. 2015. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Jorge, E.; Kågebäck, M.; and Gustavsson, E. 2016. Learning to play guess who? and inventing a grounded language as a consequence. *arXiv preprint arXiv:1611.03218*.

Konda, V. R., and Tsitsiklis, J. N. 2000. Actor-critic algorithms. In *Advances in neural information processing systems*, 1008–1014.

Kraemer, L., and Banerjee, B. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* 190:82–94.

Lazaridou, A.; Peysakhovich, A.; and Baroni, M. 2016. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*.

Leibo, J. Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; and Graepel, T. 2017. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Mordatch, I., and Abbeel, P. 2017. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*.

Oliehoek, F. A.; Spaan, M. T. J.; and Vlassis, N. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. 32:289–353.

Omidshafiei, S.; Pazis, J.; Amato, C.; How, J. P.; and Vian, J. 2017. Deep decentralized multi-task multi-agent rl under partial observability. *arXiv preprint arXiv:1703.06182*.

Peng, P.; Yuan, Q.; Wen, Y.; Yang, Y.; Tang, Z.; Long, H.; and Wang, J. 2017. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*.

Proper, S., and Tumer, K. 2012. Modeling difference rewards for multiagent learning. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, 1397–1398. International Foundation for Autonomous Agents and Multiagent Systems.

Schulman, J.; Moritz, P.; Levine, S.; Jordan, M. I.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *CoRR* abs/1506.02438.

Shoham, Y., and Leyton-Brown, K. 2009. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. New York: Cambridge University Press.

Sukhbaatar, S.; Fergus, R.; et al. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, 2244–2252.

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*, volume 99, 1057–1063.

Sutton, R. S. 1988. Learning to predict by the methods of temporal differences. *Machine learning* 3(1):9–44.

Synnaeve, G.; Nardelli, N.; Auvolat, A.; Chintala, S.; Lacroix, T.; Lin, Z.; Richoux, F.; and Usunier, N. 2016. Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint arXiv:1611.00625*.

Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; and Vicente, R. 2015. Multiagent cooperation and competition with deep reinforcement learning. *arXiv preprint arXiv:1511.08779*.

Tan, M. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, 330–337.

Tumer, K., and Agogino, A. 2007. Distributed agent-based air traffic flow management. In *Proceedings of the 6th international joint conference on Autonomous agents and multi-agent systems*, 255. ACM.

Usunier, N.; Synnaeve, G.; Lin, Z.; and Chintala, S. 2016. Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. *arXiv preprint arXiv:1609.02993*.

Weaver, L., and Tao, N. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, 538–545. Morgan Kaufmann Publishers Inc.

Weyns, D.; Helleboogh, A.; and Holvoet, T. 2005. The packet-world: A test bed for investigating situated multi-agent systems. In *Software Agent-Based Applications, Platforms and Development Kits*. Springer. 383–408.

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

Wolpert, D. H., and Tumer, K. 2002. Optimal payoff functions for members of collectives. In *Modeling complexity in economic and social systems*. World Scientific. 355–369.

Yang, E., and Gu, D. 2004. Multiagent reinforcement learning for multi-robot systems: A survey. Technical report, tech. rep.

Ye, D.; Zhang, M.; and Yang, Y. 2015. A multi-agent framework for packet routing in wireless sensor networks. *sensors* 15(5):10026–10047.

Ying, W., and Dayong, S. 2005. Multi-agent framework for third party logistics in e-commerce. *Expert Systems with Applications* 29(2):431–436.

Appendix

Training Details and Hyperparameters

Training is performed in batch mode, with a batch size of 30. Due to parameter sharing, all agents can be processed in parallel, with each agent for each episode and time step occupying one batch entry. The training cycle progresses in three steps (completion of all three steps constitutes as one episode in our graphs): 1) *collect data*: collect $\frac{30}{n}$ episodes; 2) *train critic*: for each time step, apply a gradient step to the feed-forward critic, starting at the end of the episode; and 3) *train actor*: fully unroll the recurrent part of the actor, aggregate gradients in the backward pass across all time steps, and apply a gradient update.

We use a target network for the critic, which updates every 150 training steps for the feed-forward centralised critics and every 50 steps for the recurrent IAC critics. The feed-forward critic receives more learning steps, since it performs a parameter update for each timestep. Both the actor and the critic networks are trained using RMS-prop with learning rate 0.0005 and alpha 0.99, without weight decay. We set gamma to 0.99 for all maps.

Although tuning the skip-frame in StarCraft can improve absolute performance (Peng et al. 2017), we use a default value of 7, since the main focus is a relative evaluation between COMA and the baselines.

Algorithm

Algorithm 1 Counterfactual Multi-Agent (COMA) Policy Gradients

Initialise θ_1^c, $\hat{\theta}_1^c$, θ^π
for each training episode e **do**
  Empty buffer
  for $e_c = 1$ **to** $\frac{\text{BatchSize}}{n}$ **do**
    $s_1 =$ initial state, $t = 0$, $h_0^a = \mathbf{0}$ for each agent a
    while $s_t \neq$ terminal **and** $t < T$ **do**
      $t = t + 1$
      for each agent a **do**
        $h_t^a = \text{Actor}\left(o_t^a, h_{t-1}^a, u_{t-1}^a, a, u; \theta_i\right)$
        Sample u_t^a from $\pi(h_t^a, \epsilon(e))$
      Get reward r_t and next state s_{t+1}
    Add episode to buffer
  Collate episodes in buffer into single batch
  for $t = 1$ **to** T **do** // from now processing all agents in parallel via single batch
    Batch unroll RNN using states, actions and rewards
    Calculate TD(λ) targets y_t^a using $\hat{\theta}_i^c$
  for $t = T$ **down to** 1 **do**
    $\Delta Q_t^a = y_t^a - Q\left(s_j^a, \mathbf{u}\right)$
    $\Delta\theta^c = \nabla_{\theta^c}(\Delta Q_t^a)^2$ // calculate critic gradient
    $\theta_{i+1}^c = \theta_i^c - \alpha\Delta\theta^c$ // update critic weights
    Every C steps reset $\hat{\theta}_i^c = \theta_i^c$
  for $t = T$ **down to** 1 **do**
    $A^a(s_t^a, \mathbf{u}) = Q(s_t^a, \mathbf{u}) - \sum_u Q(s_t^a, u, \mathbf{u}^{-a})\pi(u|h_t^a)$ // calculate COMA
    $\Delta\theta^\pi = \Delta\theta^\pi + \nabla_{\theta^\pi} \log \pi(u|h_t^a) A^a(s_t^a, \mathbf{u})$ // accumulate actor gradients
  $\theta_{i+1}^\pi = \theta_i^\pi + \alpha\Delta\theta^\pi$ // update actor weights