COMA

Counterfactual Multi-Agent Policy Gradient

2024. 03. 14

AI Robotics KR, Multi-agent

Minkyoung Kim

Introduction

- Takes an actor-critic approach
  - Centralized critic: estimate the Q-function
  - Decentralized actors: optimize the agent's policy
- Main Idea of COMA
  - Uses a centralized critic
    - The critic is only used during learning, conditioning on the joint action, all state information
    - The actor is needed during execution, conditioning on own action-observation history
  - Uses a counterfactual baseline to address the challenges of multi-agent credit assignment
    - Inspired by *Difference rewards*(2002)
    - Using centralized critic to compute an agent-specific advantage function
  - 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.
    - All Q-values for all agents can be computed in a single batched forward pass.

Background

- Fully cooperative multi-agent task, can be described as a stochastic game G
- $G = < S, U, P, r\ Z, O, n, \gamma >$
  - n agents $a \in A \equiv \{1, \dots, n\}$
  - True state $s \in S$
  - Agent's action $u^a \in U$
  - Joint action $\mathbf{u} \in \boldsymbol{U} \equiv U^n$
  - State transition function $P(s'|s, \boldsymbol{u}) : S \times \boldsymbol{U} \times S \rightarrow [0,1]$
  - Reward function $r(s, \boldsymbol{u}): S \times \boldsymbol{U} \rightarrow \mathbb{R}$
  - Discount factor $\gamma \in [0,1]$
  - Agent observation $\mathrm{z} \in Z$
  - Observation function $\mathrm{O}(s, a): S \times A \rightarrow Z$
  - Each agent has an action-observation history $\tau^a \in T \equiv (Z \times U)^*$
  - Stochastic policy $\pi^a(u^a|\tau^a)$: $T \times U \rightarrow [0,1]$
  - Joint quantities over agents in bold
    - Joint quantities over agents other than a given agent a with the superscript $-a$

Background

- In COMA, train critics $f^c(\cdot, \theta^c)$ on-policy to estimate either Q or V, using a variant of TD(λ)
  - TD(λ) uses a mixture of n-step return $G_t^{(n)} = \sum_{l=1}^{n} \gamma^{l-1} r_{t+1} + \gamma^n f^c(\cdot_{t+n}, \theta^c)$
  - Loss $\mathcal{L}_t(\theta^c) = (y^{(\lambda)} - f^c(\cdot_t, \theta^c))^2$
    - $y^{(\lambda)} = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$
      - n-step return $G_t^{(n)}$ calculated with boostrapped values estimated by target network

Background

- Independent Actor-Critic(IAC)
  - Each agent learn independently, with own actor and critic
    - Similar with independent Q-learning(IQN, 1993)
  - Speed learning by sharing parameters among the agents
    - Learn only one actor and one critic, which are used by all agents
  - Limitation
    - Non-stationary learning
    - Hard to learn to coordinate
    - Multi-agent credit assignment problem
  - Two variants of IAC
    - IAC-V: Each agent's critic estimates $V(\tau^a)$ and follows a gradient based on the TD error
      - TD error: $r_t + \gamma V(s_{t+1}) - V(s)$
    - IAC-Q: Each agent's critic estimates $Q(\tau^a|u^a)$ and follows a gradient based on Advantage function
      - $A(\tau^a|u^a) = Q(\tau^a|u^a) - V(\tau^a)$
      - $V(\tau^a) = \sum_{u^a} \pi(u^a|\tau^a) Q(\tau^a|u^a)$

Methods

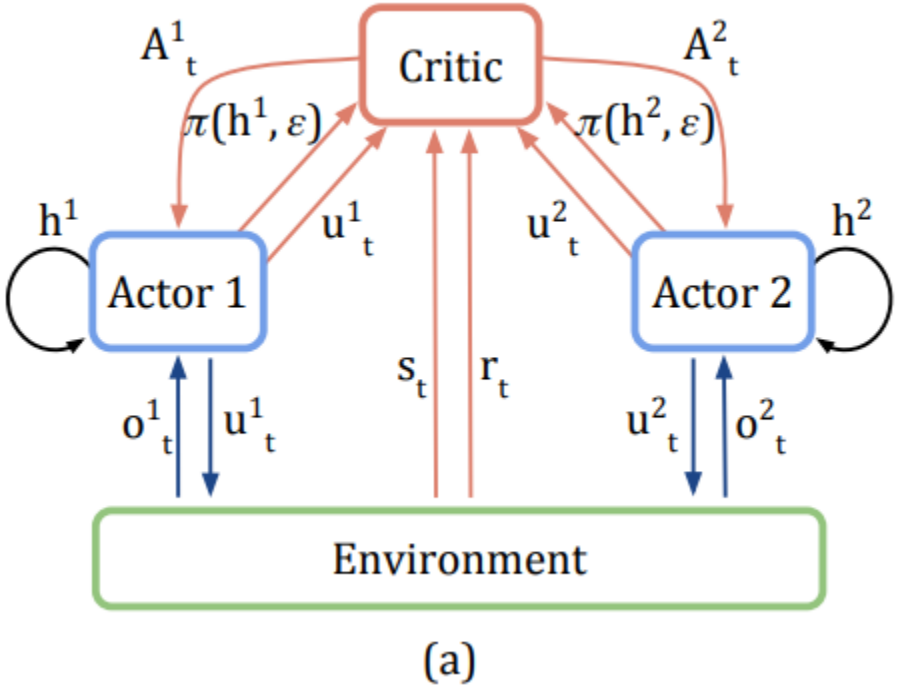
- Counterfactual multi-agent(COMA) policy gradients
  1. Centralization of the critic
     - Stabilize learning to coordinate
  2. Use of a counterfactual baseline
     - Tackle multi-agent credit assignment
  3. Use of a critic representation that allow efficient evaluation of the baseline
     - Scale to large NNs

Methods

1. Centralization of the critic

- In IAC,
  - each actor $\pi(u^a|\tau^a)$ and each critic $Q(\tau^a|u^a), V(\tau^a)$ conditions only on the agent's own action-observation history τ^a

- Critic: used only during learning
  - Using true global state s, joint action-observation histories $\boldsymbol{\tau}$
- Actor: needed during execution
  - Using the agent's own action-observation history τ^a



(a)

Methods

2. Use of a counterfactual baseline(1/2)

- A naïve way of centralized critic would follow a gradient based on TD error estimated from the critic
  - $g_a = \nabla_{\theta^\pi} \log \pi(u_t^a|\tau_t^a)\big(r_t + \gamma V(s_{t+1}) - V(s_t)\big)$
  - Fail to address a key credit assignment problem (TD error considers only global rewards)

- Counterfactual baseline inspired by *difference rewards*(2002)
  - Shaped reward $D^a = r(s, \boldsymbol{u}) - r(s, (\boldsymbol{u}^{-a}, c^a))$
    - Compare global reward to the reward received when the agent's action is replaced with a *default action* c^a

  - Powerful way to perform multi-agent credit assignment!
    - Limitation1: require access to a simulator or estimated reward function $r(s, (\boldsymbol{u}^{-a}, c^a))$
    - Limitation2: unclear how to choose the default action(in general)

Methods

2. Use of a counterfactual baseline(2/2)

- COMA learns a centralized critic, $Q(s, \boldsymbol{u})$ that estimates Q-values for the joint action $\boldsymbol{u}$ conditioned on the central state s

- For each agent a, *compute an advantage function* that compares the Q-value for current action u^a to a counterfactual baseline that marginalises out u^a, while keeping the other agents' actions $\boldsymbol{u}^{-a}$ fixed:
  - $A^a(s, \boldsymbol{u}) = Q(s, \boldsymbol{u}) - \underbrace{\sum_{u'^a} \pi^a(u'^a|\tau^a) Q(s, (\boldsymbol{u}^{-a}, u'^a))}_{\text{Counterfactual baseline}}$

→ If the critic is a deep neural network, those evaluations may themselves be expensive

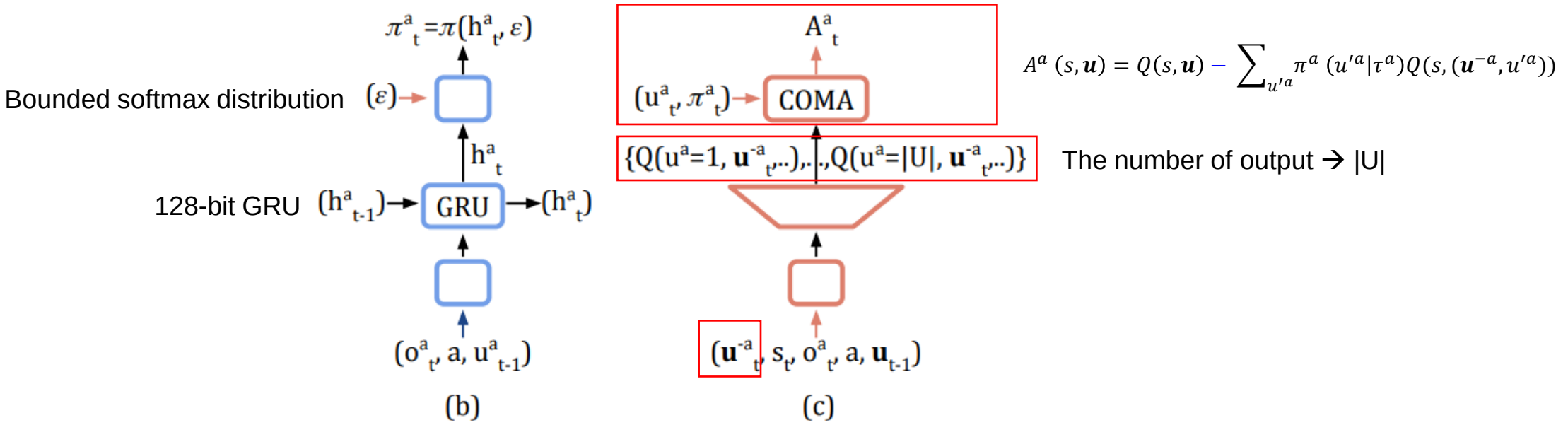
→ The number of output nodes of such a network would equal $|U|^n$, the size of the joint action space

→ Impractical to train!

Methods

3. Use of a critic representation that allow efficient evaluation of the counterfactual baseline

- Actions of the other agents, $\boldsymbol{u}^{-a}$, are part of the input to the network
- The network output a Q-value for each of agent a's action
- The *counterfactual advantage* can be calculated efficiently by a single forward pass of the actor and critic, for each agent.



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

Experiment

- Evaluate COMA in StarCraft unit micromanagement
  - High stochasticity, a large state action space, delayed rewards

- Conditions
  - Partial observability
    - Local observation: distance, relative x, relative y, unit type, shield
    - Global state: regardless of field of view, x-y locations relative to the centre of map, health points, cool down
  - Field of view on the agents == the firing range of ranged unit's weapons
  - Agents cannot distinguish between enemies who are dead and those who are out of range
  - All agents receive the same global reward at each step

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

Experiment

- Central-V: estimated V, follow the gradient based on TD-error
- Central-QV: estimated V and Q, follow the gradient based on Advantage function

	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).

Experiment

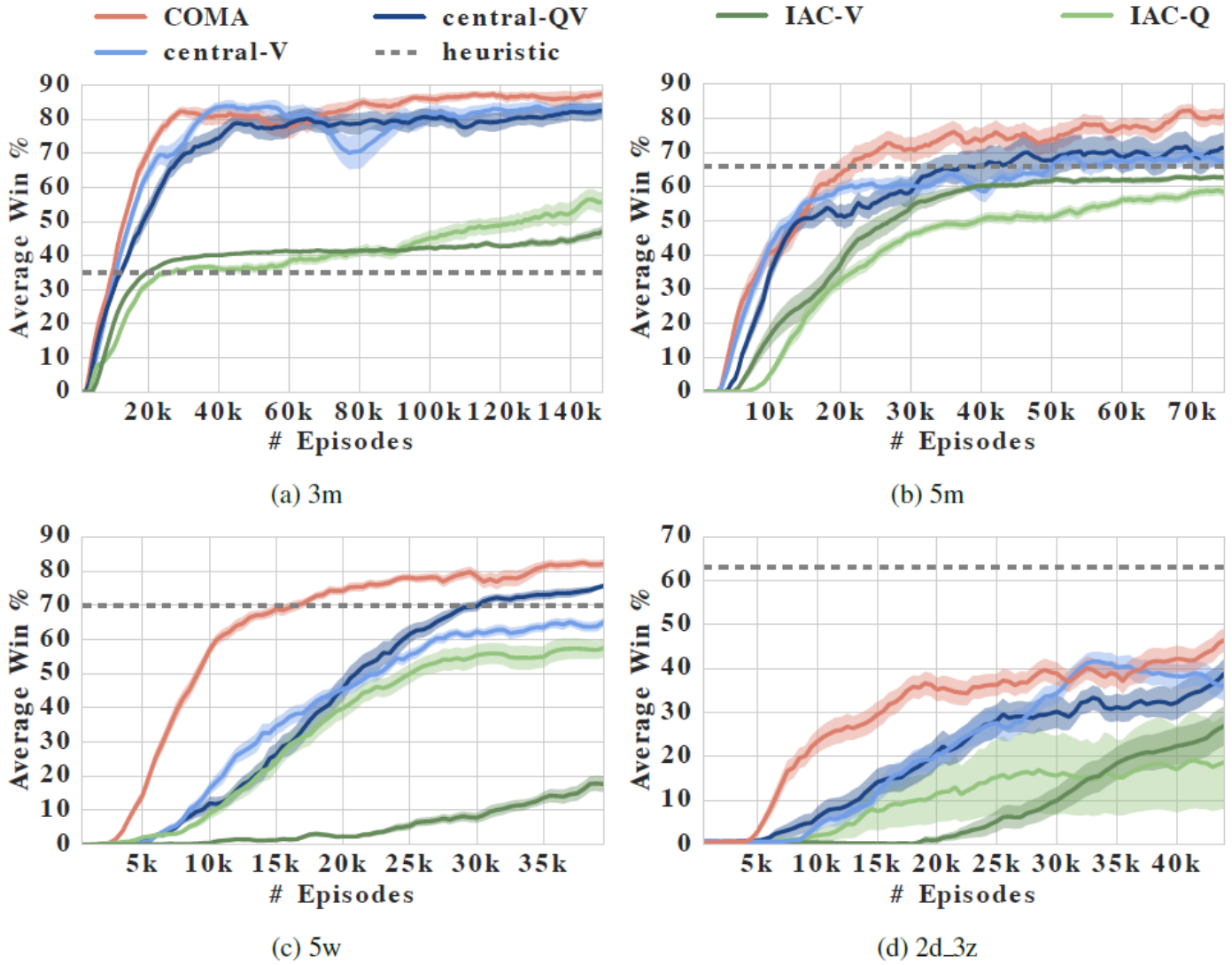


Figure 3: Win rates for COMA and competing algorithms on four different scenarios. COMA outperforms all baseline methods. Centralised critics also clearly outperform their decentralised counterparts. The legend at the top applies across all plots.

Reference

- Counterfactual Multi-Agent Policy Gradients, 2017, https://arxiv.org/pdf/1705.08926.pdf
- Counterfactual Multi-Agent Policy Gradients, Microsoft Research Cambridge, AI Summer School 2017, https://youtu.be/3OVvjE5B9LU?si=JvD1XfvLCIXYhpiB
- https://www.microsoft.com/en-us/research/uploads/prod/2018/03/Shimon-Whiteson.pdf

Appendix

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