

Multi-Agent Actor-Critic for Mixed Cooperative- Competitive Environments

AUTHORS:

RYAN LOWE, YI WU, AVIV TAMAR, JEAN HARB, PIETER ABBEEL, IGOR MORDATCH

Cooperative Behaviors

Many RL tasks require cooperation with other agents:

- ▶ Multi-robot control
- ▶ Modelling communication & language
- ▶ Multiplayer games
- ▶ Analyzing social dilemmas

Issues with Classic RL

Other agents are constantly changing

This causes two problems:

1. The environment is not stationary, so we can't effectively replay past experiences
2. For competitive tasks, your opponent is constantly changing, which causes instability

New Contributions

- ▶ A centralized critic with access to all the actors
- ▶ Agents learn approximations to other agents
- ▶ Train an ensemble of policies all at once

Multi-Agent Actor Critic

Some Intuition

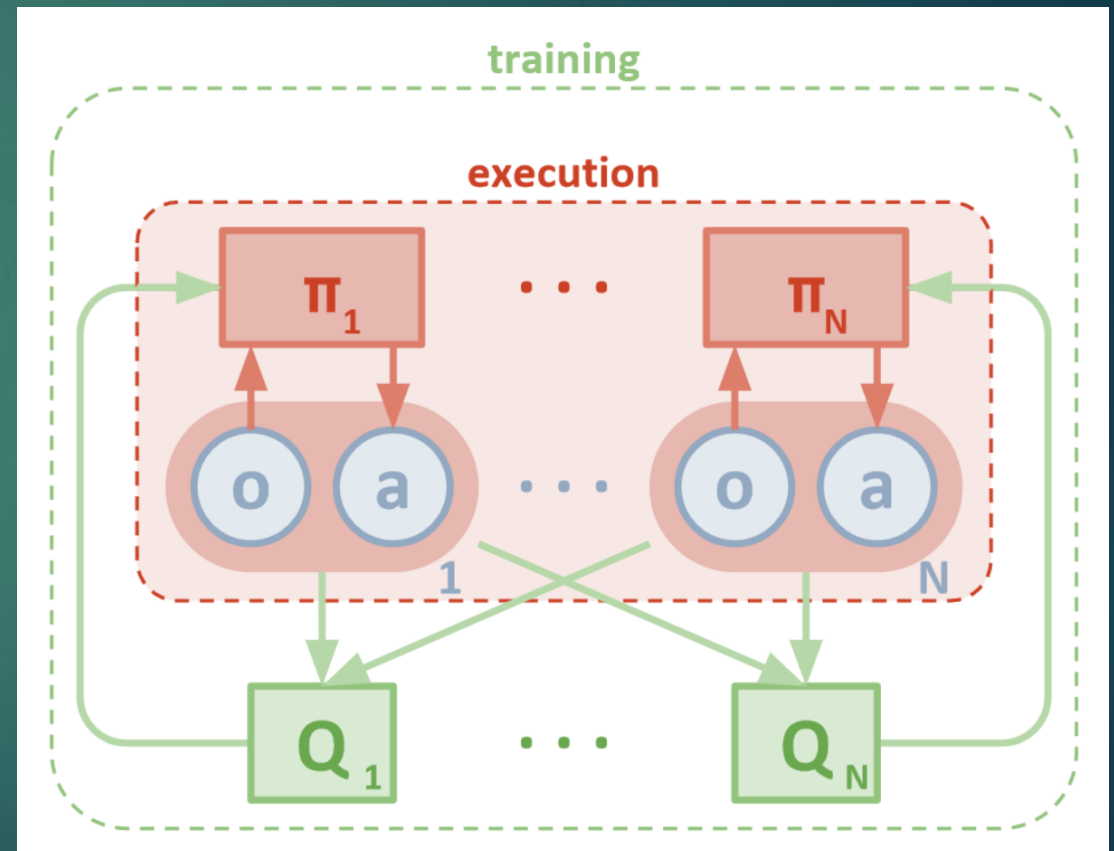
The reason multi-agent setups are hard is because predicting the next state requires predicting the actions of all other agents

What if we just tell each agent the actions of all others?

- ▶ Predicting the next state becomes much easier
- ▶ We don't need to keep updating our prediction function as the other agents change policy

Multi-Agent Actor Critic

- ▶ Give each agent's Q-function the actions of every other agent as part of its input
- ▶ The state transition function observed by one agent no longer depends on the policy of each other agent
- ▶ The environment is now stationary



Algorithm Details

Updating the actor:

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{\mathbf{x}, a \sim \mathcal{D}} [\nabla_{\theta_i} \boldsymbol{\mu}_i(a_i | o_i) \nabla_{a_i} Q_i^\mu(\mathbf{x}, a_1, \dots, a_N) |_{a_i = \boldsymbol{\mu}_i(o_i)}],$$

Updating the critic:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x}, a, r, \mathbf{x}'} [(Q_i^\mu(\mathbf{x}, a_1, \dots, a_N) - y)^2], \quad y = r_i + \gamma Q_i^{\mu'}(\mathbf{x}', a'_1, \dots, a'_N) |_{a'_j = \boldsymbol{\mu}'_j(o_j)},$$

Algorithm Details

Updating the actor:

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{\mathbf{x}, a \sim \mathcal{D}} [\nabla_{\theta_i} \boldsymbol{\mu}_i(a_i | o_i) \nabla_{a_i} Q_i^\mu(\mathbf{x}, a_1, \dots, a_N) |_{a_i = \boldsymbol{\mu}_i(o_i)}],$$

Updating the critic:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x}, a, r, \mathbf{x}'} [(Q_i^\mu(\mathbf{x}, a_1, \dots, a_N) - y)^2], \quad y = r_i + \gamma Q_i^{\mu'}(\mathbf{x}', a'_1, \dots, a'_N) |_{a'_j = \boldsymbol{\mu}'_j(o_j)}$$

This requires knowing the exact policy of every other agent


Inferring Policies of Other Agents

Why do we need this?

In a standard reinforcement learning training environment, we always know the policies of other agents

When do we need to infer policies?

- ▶ Distributed training with limited bandwidth
- ▶ Modelling individual learning
- ▶ Learning alongside a human

- 
- ▶ We wish we knew the other agents' policies
 - ▶ Train a neural network to learn their policies!
 - ▶ When in doubt, just add more networks

Solution

Each agent trains another network to model the actions of each other agent in the scene

- ▶ With a small number of agents this works very well
- ▶ This requires a quadratic number of networks, so it can be slow with a large number of agents

Agents with Policy Ensembles

Issues with Competitive Tasks

- ▶ You're training two agents to play Rock-Paper-Scissors
- ▶ Opponent starts out usually playing Rock
- ▶ You learn to usually play Paper
- ▶ Opponent learns to usually play Scissors
- ▶ You learn to usually play Rock
- ▶ Opponent learns to usually play Paper
- ▶ (and so on)

Stable Training with Ensembles

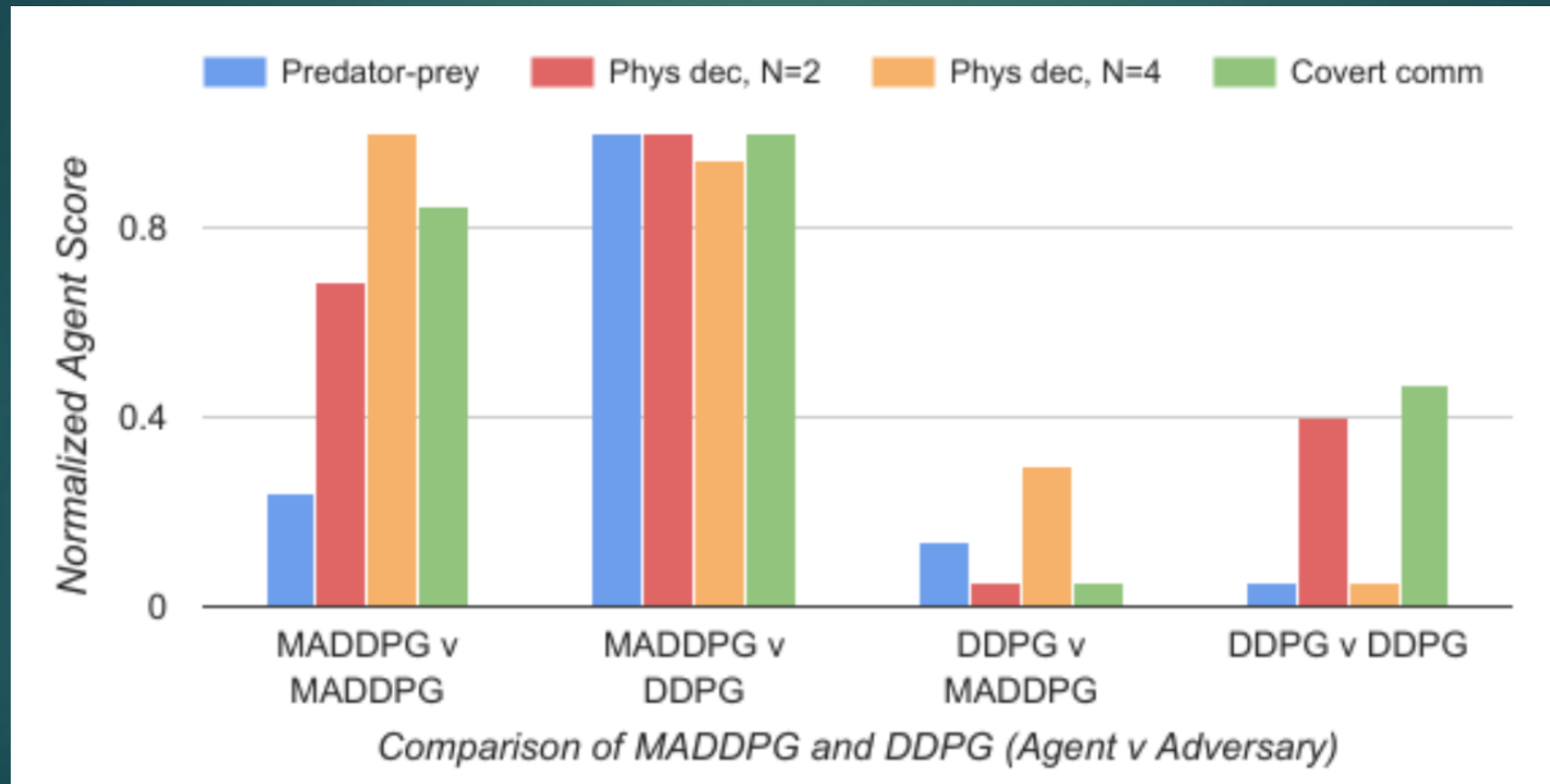
We learn a strategy that's good against our current opponent, but that might not be good in general

Solution: learn against many opponents at once

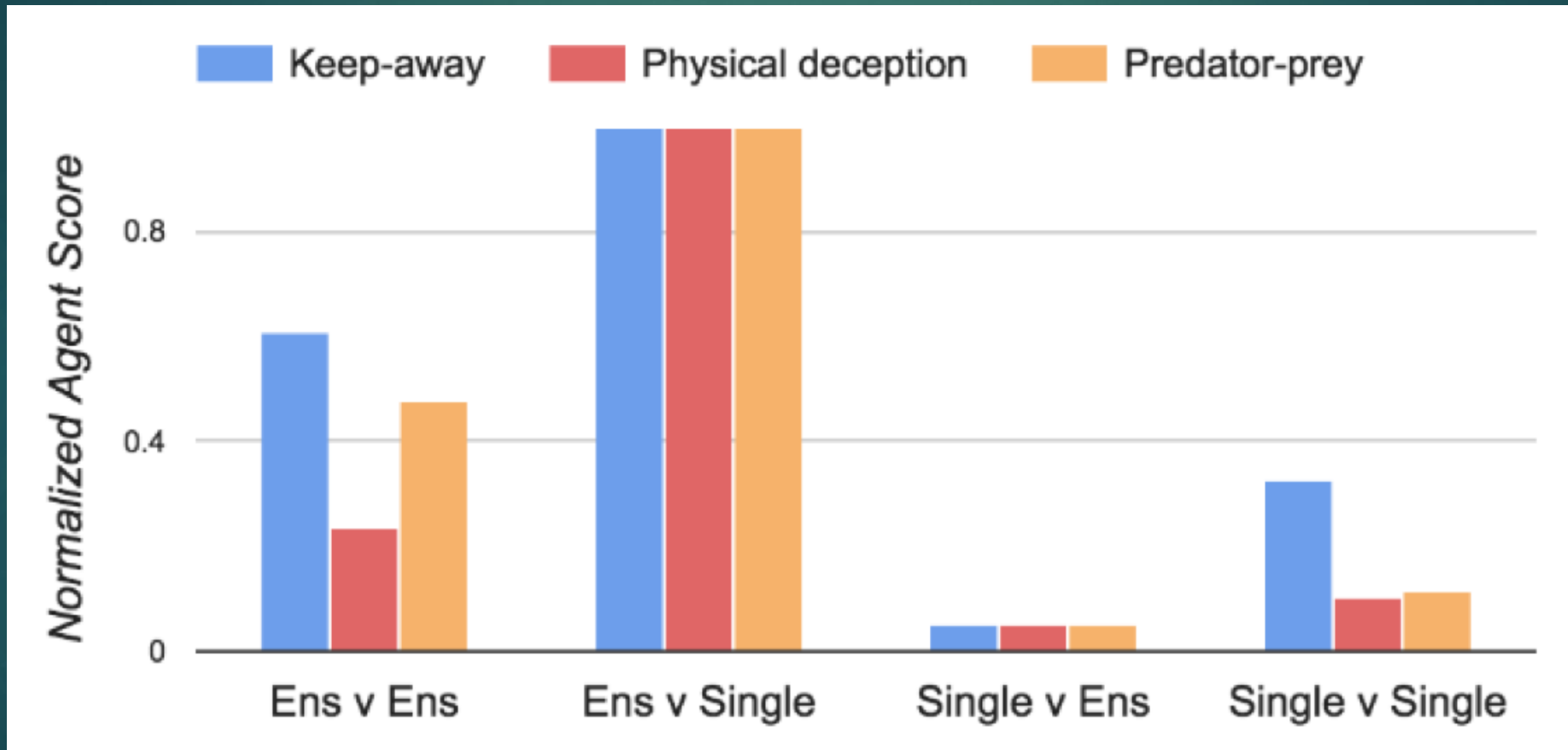
- ▶ Train an ensemble of K different sub-policies
- ▶ Pick a random policy for each episode
- ▶ Keep the other policies in our replay buffer

Experiments and Results

Multi-Agent Actor Critic vs. Baseline



Ensemble vs. Single

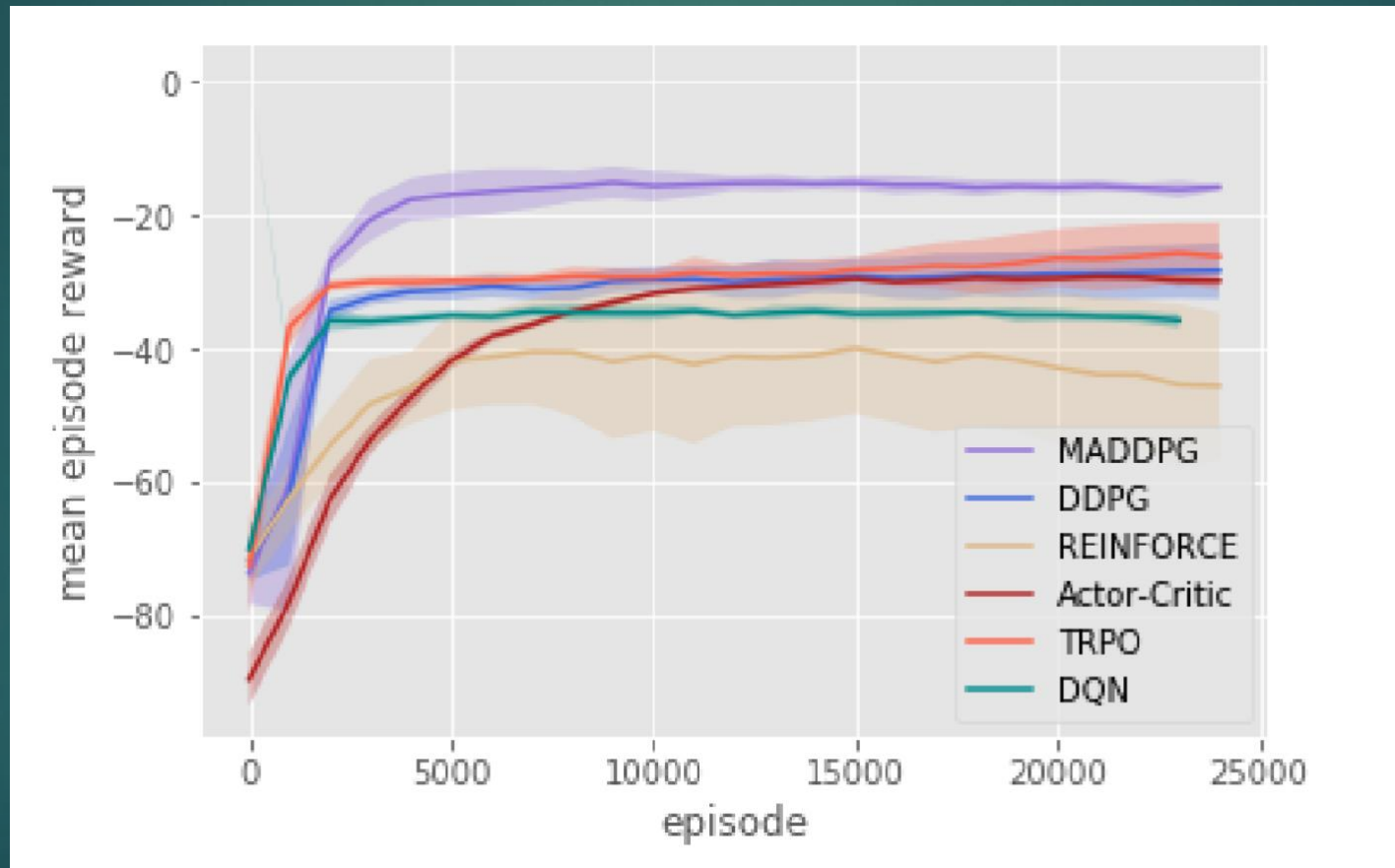


Cooperative Environment

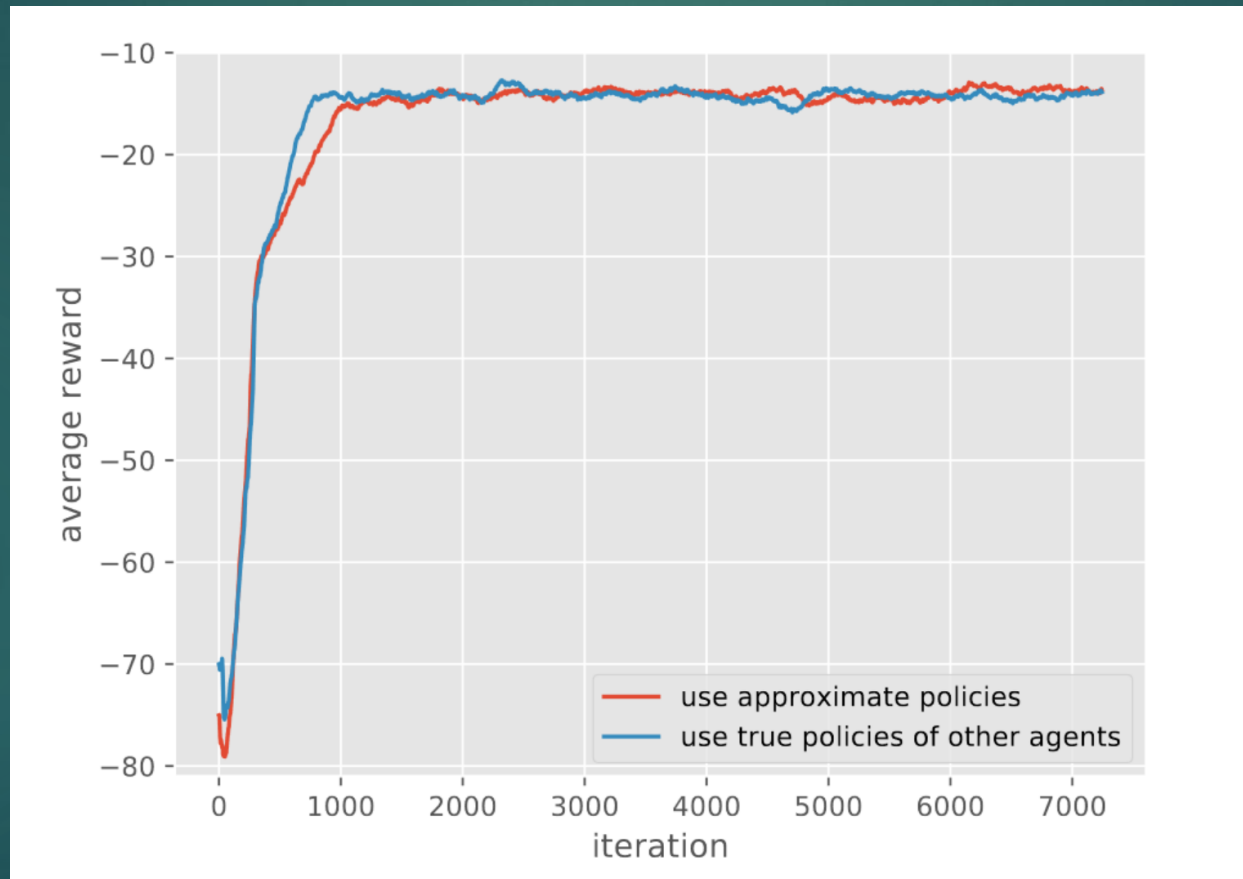
There are two agents, and three squares. Each episode, one square is marked as the right one, and both agents win if the second agent moves to the right square.

- ▶ The first agent can only see which square was picked
- ▶ The second agent can only see where the squares are
- ▶ The first agent can send messages to the second

Multi-Agent Actor Critic vs. Others



True Policies vs. Inferred Policies



Conclusion

Conclusion

Pros:

- ▶ Significantly improves multi-agent training
- ▶ Applies to all kinds of cooperative and competitive tasks
- ▶ Ensemble training gives another improvement

Cons:

- ▶ Doesn't apply to tasks with variable numbers of agents
- ▶ Doesn't scale to a large number of agents

References

- ▶ Lowe, Ryan, et al. [Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments](#). Advances in Neural Information Processing Systems 30 (2017): 6379-6390.