Difference Rewards Policy Gradients

Jacopo Castellini[1], Sam Devlin[2], Frans A. Oliehoek[3], Rahul Savani[1]

[1]University of Liverpool, [2]Microsoft Research Cambridge, [3]Delft University of Technology

3-4 May 2021

Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) is widely used to solve *cooperative multi-agent systems* (MASs) with a shared reward signal.

Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) is widely used to solve *cooperative multi-agent systems* (MASs) with a shared reward signal.

Centralized Training-Decentralized Execution (CTDE) framework:

Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) is widely used to solve *cooperative multi-agent systems* (MASs) with a shared reward signal.

Centralized Training-Decentralized Execution (CTDE) framework:

- CT: information are shared among agents during training phase (i.e. centralized critic),

Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) is widely used to solve *cooperative multi-agent systems* (MASs) with a shared reward signal.

Centralized Training-Decentralized Execution (CTDE) framework:

- CT: information are shared among agents during training phase (i.e. centralized critic),
- DE: during execution, each agent has to rely only on its local information (i.e. decentralized policies).

Multi-Agent Policy Gradients

Multi-Agent Policy Gradients (MAPG) [Peshkin et al., 2000] algorithms have become one of the most popular MARL techniques under CTDE.

Multi-Agent Policy Gradients

Multi-Agent Policy Gradients (MAPG) [Peshkin et al., 2000] algorithms have become one of the most popular MARL techniques under CTDE.

They improve the parameters θ^i of agent i policy $\pi(s, a^i; \theta^i)$ by following the gradients:

$$\nabla_{\theta^i} J(\theta^i) = \mathbb{E}_{a^i \sim \pi_{\theta^i}(\cdot|s)} \left[\nabla_{\theta^i} \log \pi_{\theta^i}(a^i|s) G_t \right],$$

where $G_t = \sum_{l=0}^{T} \gamma^l r_{t+l}$ is the return.

Multi-Agent Policy Gradients

Multi-Agent Policy Gradients (MAPG) [Peshkin et al., 2000] algorithms have become one of the most popular MARL techniques under CTDE.

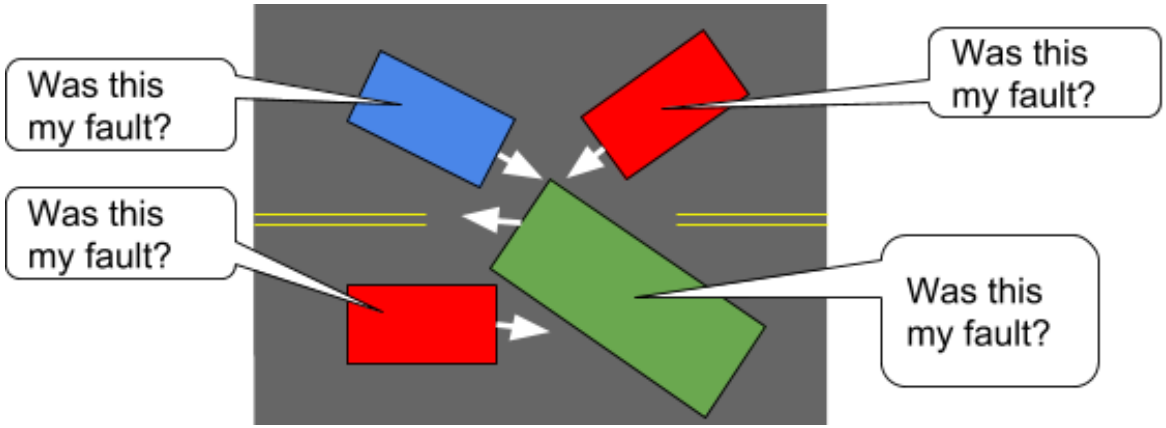
They improve the parameters θ^i of agent i policy $\pi(s, a^i; \theta^i)$ by following the gradients:

$$\nabla_{\theta^i} J(\theta^i) = \mathbb{E}_{a^i \sim \pi_{\theta^i}(\cdot|s)} \left[\nabla_{\theta^i} \log \pi_{\theta^i}(a^i|s) G_t \right],$$

where $G_t = \sum_{l=0}^{T} \gamma^l r_{t+l}$ is the return.

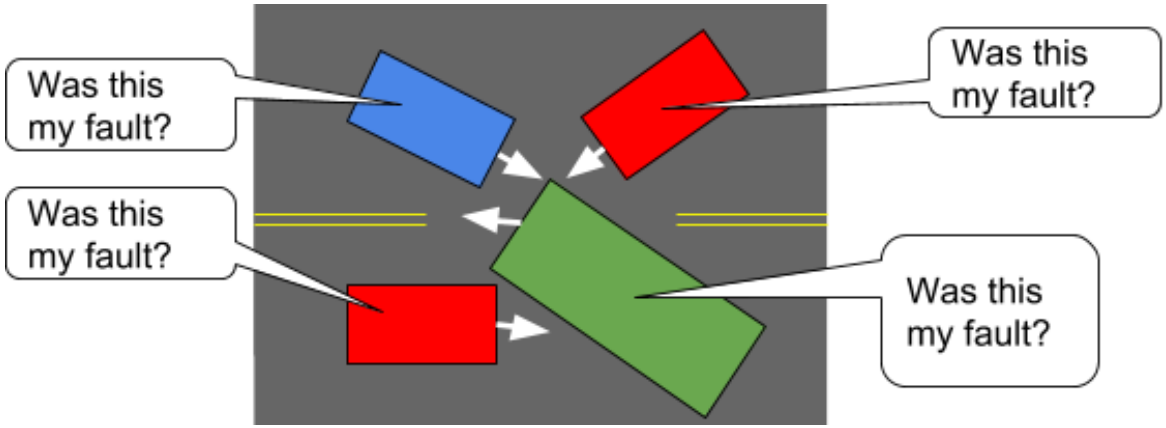
Remark: as $G_t \approx Q^{\pi_\theta}(s, a)$, one can instead learn a *centralized critic* Q_ω to approximate it.

Multi-Agent Credit Assignment



Multi-agent credit assignment: how can an agent determine how its actions are affecting the overall performance of the team?

Multi-Agent Credit Assignment



Multi-agent credit assignment: how can an agent determine how its actions are affecting the overall performance of the team?

MAPG algorithms do not usually address this key problem in MARL...

Difference Rewards

Difference rewards (DR) [Wolpert and Tumer, 2001] $\rightarrow$ Provide the agents with an individual *shaped reward* to use during learning.

Difference Rewards

Difference rewards (DR) [Wolpert and Tumer, 2001] $\rightarrow$ Provide the agents with an individual *shaped reward* to use during learning.

A possible version is called *aristocrat utility*:

$$\Delta^i = R(s, a) - \mathbb{E}_{b^i \sim \pi_{\theta^i}(\cdot|s)} \left[R(s, \langle a^{-i}, b^i \rangle) \right].$$

Difference Rewards

Difference rewards (DR) [Wolpert and Tumer, 2001] $\rightarrow$ Provide the agents with an individual *shaped reward* to use during learning.

A possible version is called *aristocrat utility*:

$$\Delta^i = R(s, a) - \mathbb{E}_{b^i \sim \pi_{\theta^i}(\cdot|s)} \left[R(s, \langle a^{-i}, b^i \rangle) \right].$$

Problem: usually rewards $R(s, \langle a^{-i}, b^i \rangle)$ for each local action b^i are not known...

DR: An Example

		Agent Y	
		a_1	a_2
Agent X	a_1	10	10
	a_2	5	0

$\pi^X = \pi^Y = (0.5, 0.5)$

At time step t:

- $a_t = \langle a_2, a_1 \rangle$
- $r_t = 5$

DR: An Example

Agent X \ Agent Y	a_1	a_2
a_1	10	10
a_2	5	0

$$\pi^X = \pi^Y = (0.5, 0.5)$$

At time step t:

- $a_t = \langle a_2, a_1 \rangle$
- $r_t = 5$

$$\Delta R_t^X = r_t - \sum_{b^X \in \{a_1, a_2\}} \pi^X(b^X) R(\langle b^X, a_1 \rangle)$$

$$= 5 - (0.5 \cdot 10 + 0.5 \cdot 5) = -2.5$$

DR: An Example

Agent X \ Agent Y	a_1	a_2
a_1	10	10
a_2	5	0

$\pi^X = \pi^Y = (0.5, 0.5)$

At time step t:

- $a_t = \langle a_2, a_1 \rangle$
- $r_t = 5$

$$\Delta R_t^X = r_t - \sum_{b^X \in \{a_1, a_2\}} \pi^X(b^X) R(\langle b^X, a_1 \rangle)$$

$$= 5 - (0.5 \cdot 10 + 0.5 \cdot 5) = -2.5$$

$$\Delta R_t^Y = r_t - \sum_{b^Y \in \{a_1, a_2\}} \pi^Y(b^Y) R(\langle a_2, b^Y \rangle)$$

$$= 5 - (0.5 \cdot 5 + 0.5 \cdot 0) = 2.5$$

Counterfactual Multi-Agent Policy Gradients

COMA [Foerster et al., 2018] is combining MAPG with DR applied on the Q-function (approximated by a centralized critic $Q_\omega(s, a)$):

$$\nabla_{\theta^i} J(\theta^i) = \mathbb{E}_{a^i \sim \pi_{\theta^i}(\cdot|s)} \left[\nabla_{\theta^i} \log \pi_{\theta^i}(a^i|s) \Delta Q^i(s, a) \right],$$

where $\Delta Q^i(s, a) = Q_\omega(s, a) - \mathbb{E}_{b_i \sim \pi_{\theta_i}(\cdot|s)} [Q_\omega(s, \langle a_{-i}, b_i \rangle)]$.

Counterfactual Multi-Agent Policy Gradients

COMA [Foerster et al., 2018] is combining MAPG with DR applied on the Q-function (approximated by a centralized critic $Q_\omega(s, a)$):

$$\nabla_{\theta^i} J(\theta^i) = \mathbb{E}_{a^i \sim \pi_{\theta^i}(\cdot|s)} \left[\nabla_{\theta^i} \log \pi_{\theta^i}(a^i|s) \Delta Q^i(s, a) \right],$$

where $\Delta Q^i(s, a) = Q_\omega(s, a) - \mathbb{E}_{b_i \sim \pi_{\theta_i}(\cdot|s)} [Q_\omega(s, \langle a_{-i}, b_i \rangle)]$.

1. Learning $Q_\omega(s, a)$ is difficult because of bootstrapping and the moving target problem,
2. It depends on the (exponentially many) joint actions a,
3. Does not exploit knowledge of the reward function $R(s, a)$.

In this work...

Exploiting CTDE, we combine decentralized policy gradients with DR to learn in fully cooperative multi-agent scenarios [Castellini et al., 2021].

1. *Dr.Reinforce*: when knowledge of the reward function is available,

In this work...

Exploiting CTDE, we combine decentralized policy gradients with DR to learn in fully cooperative multi-agent scenarios [Castellini et al., 2021].

1. *Dr.Reinforce*: when knowledge of the reward function is available,
2. *Dr.ReinforceR*: a *centralized reward network* $R_\psi(s, a)$ is additionally learned to compute such DR when that is not the case,
3. Learning R_ψ is a regression problem that does not involve bootstrapping or moving targets, so easier to learn than Q_ω.

Dr.Reinforce

When the reward function $R(s, a)$ is known in advance, we define the *difference return* for agent i as:

$$\Delta G^i = \sum_{t=0}^{T} \gamma^t \Delta R^i(s, a),$$

where ΔR^i is the aristocrat utility:

$$\Delta R^i(s, a) = r_t - \sum_{b^i \in A^i} \pi_{\theta^i}(b^i|s) R(s, \langle b^i, a^{-i} \rangle).$$

Dr.Reinforce (cont'd)

We then apply decentralized policy gradients using our difference return ΔG^i as:

$$\nabla_{\theta^i} J(\theta^i) = \nabla_{\theta^i} \log \pi_{\theta^i}(a^i|s) \Delta G^i.$$

This way, each agent receives a different gradient that takes into account its own contribution to the team performance.

Proof of converge in [Castellini et al., 2021].

Dr.ReinforceR

When $R(s, a)$ is unknown, we learn an additional *centralized reward network* $R_\psi(s, a)$ as:

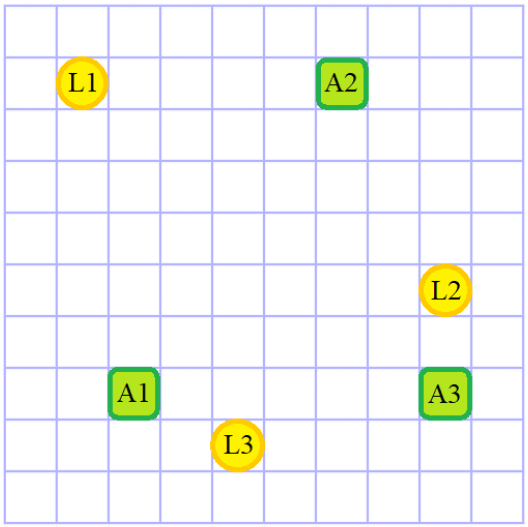
$$\mathcal{L}_t(\psi) = \frac{1}{2}\left(r_t - R_\psi(s, a)\right)^2.$$

When then compute ΔG^i in $\nabla_{\theta^i} J(\theta^i)$ by replacing the terms ΔR^i with:

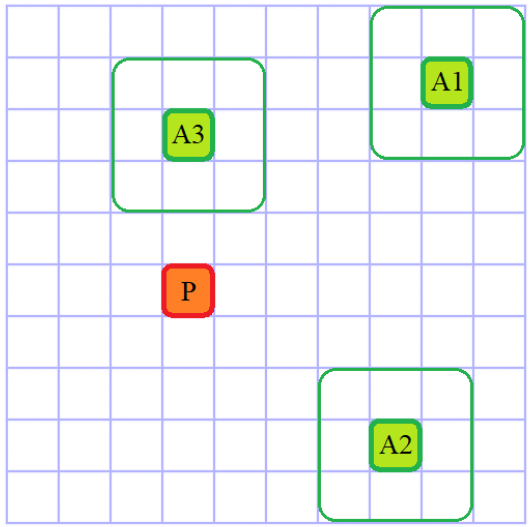
$$\Delta R^i_\psi(s, a) = r_t - \sum_{b^i \in A^i} \pi_{\theta^i}(b^i|s) R_\psi(s, \langle b^i, a^{-i} \rangle).$$

Experiments

We compared against COMA and other MAPG algorithms, as well as with a variation of the DR method in [Colby et al., 2014], on two standard cooperative benchmarks:

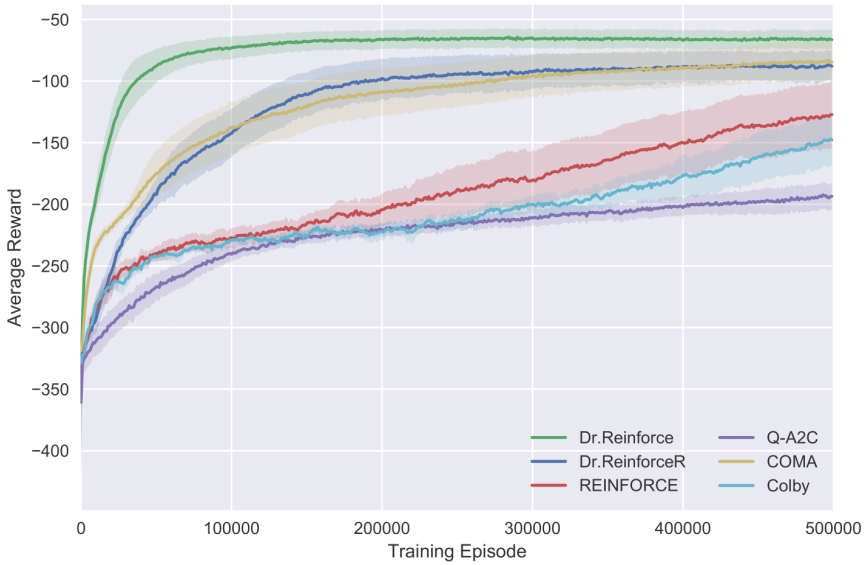


Multi-Rover

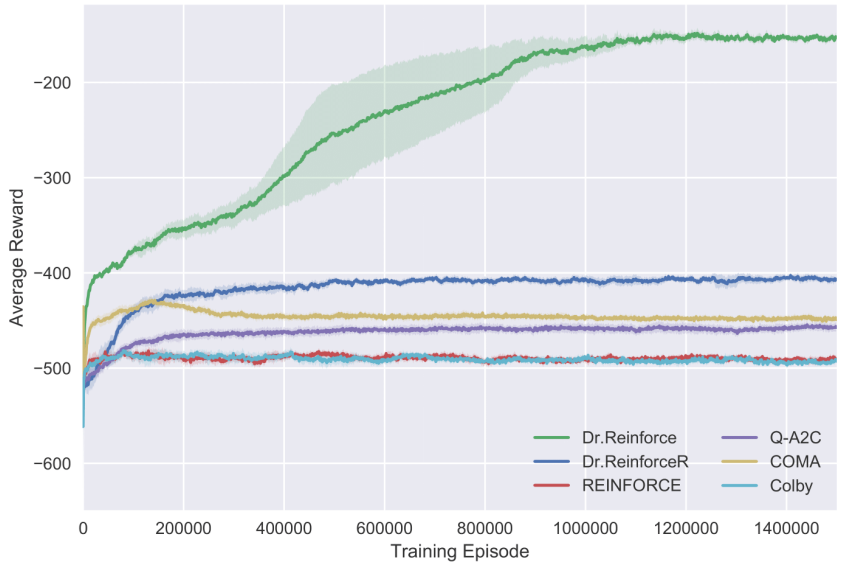


Predator-Prey

Results (Multi-Rover)



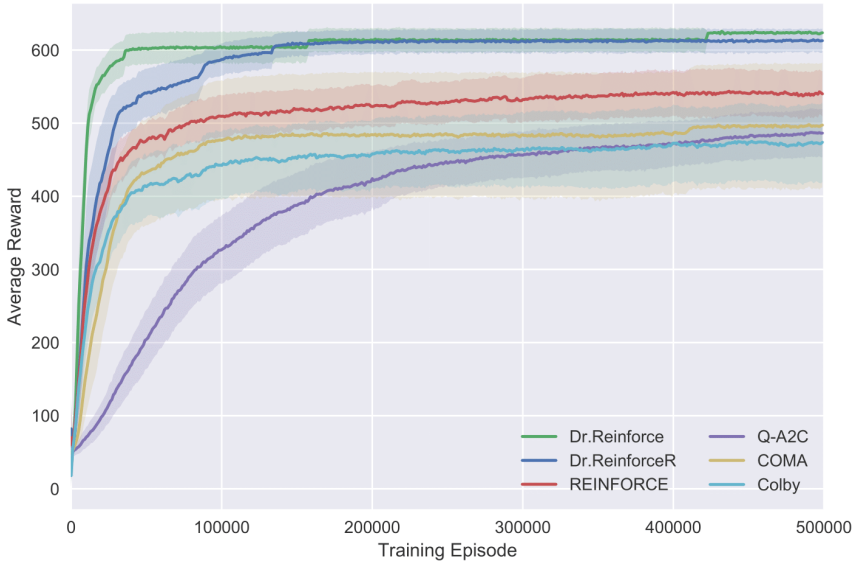
$N = 3$



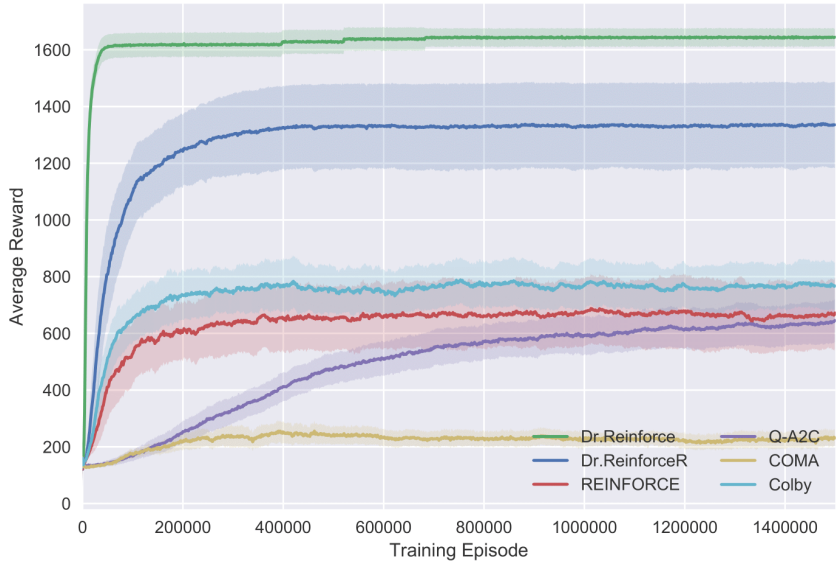
$N = 8$

With few agents both are doing fine, but with more both get worse, and Dr.ReinforceR quickly stop matching Dr.Reinforce upper bound.

Results (Predator-Prey)



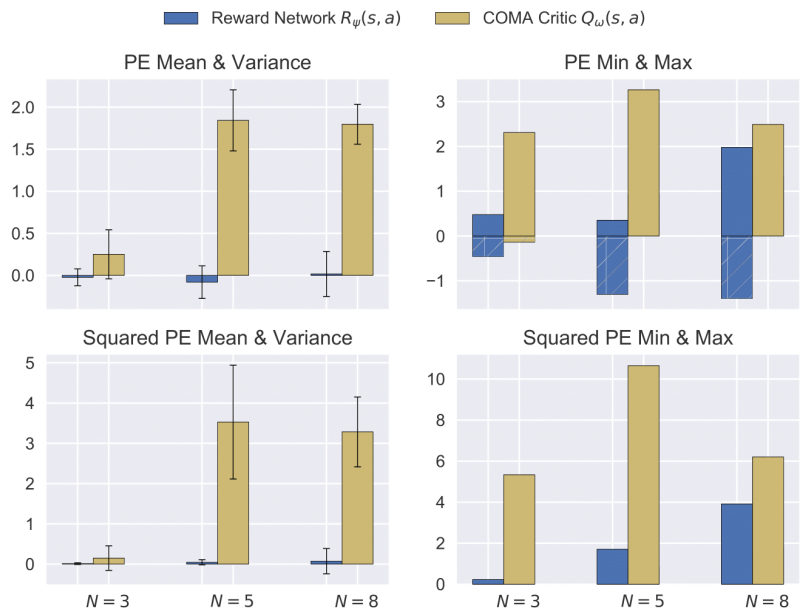
$N = 3$



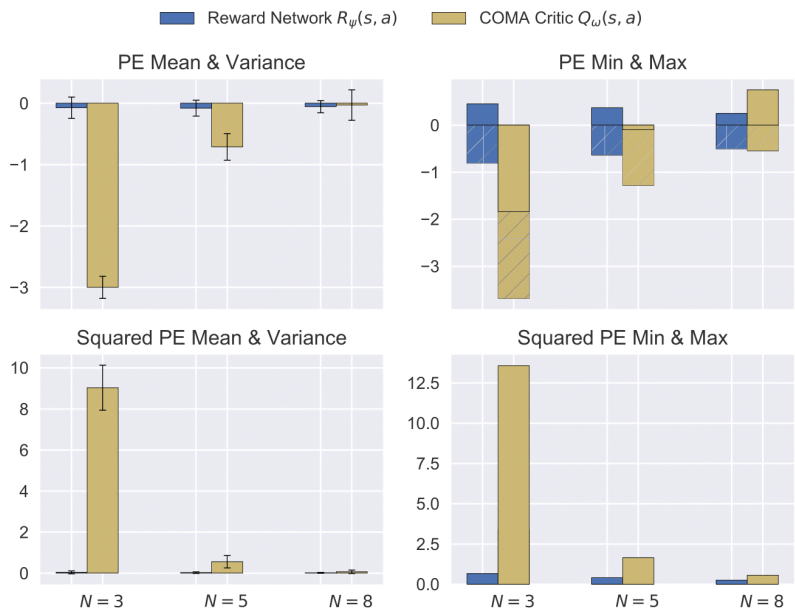
$N = 8$

While COMA is deteriorating with more agents, Dr.ReinforceR is doing almost on-par with Dr.Reinforce!

Analysis



Multi-Rover



Predator-Prey

Trained R_ψ has better *prediction error* (PE) than $Q_\omega \rightarrow$ Better generalization!

Conclusions

1. *Dr.Reinforce*, combining MAPG benefits with DR, outperforms baselines and SOTA algorithms when the reward function is known,
2. *Dr.ReinforceR*, a practical methodology when that is not the case,
3. Learning R_ψ rather than Q_ω is an easier regression problem and results in lower PE and better generalization.

Download

Download the full paper from here:

References I

Castellini, J., Devlin, S., Oliehoek, F. A., and Savani, R. (2021).

Difference rewards policy gradients.

Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems.

Colby, M. K., Curran, W., Rebhuhn, C., and Tumer, K. (2014).

Approximating difference evaluations with local knowledge.

In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems*.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. (2018).

Counterfactual multi-agent policy gradients.

In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*.

References II

Peshkin, L., Kim, K.-E., Meuleau, N., and Kaelbling, L. P. (2000).

Learning to cooperate via policy search.

In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*.

Wolpert, D. H. and Tumer, K. (2001).

Optimal payoff functions for members of collectives.

Advances in Complex Systems.