Robust Multi-Agent Reinforcement Learning by *Minimax Deep Deterministic Policy Gradient*

YI WU, UC BERKELEY

WITH SHIHUI LI, FEI FANG, STUART RUSSELL, XINYUE CUI & HONGHUA DONG

Multi-Agent Reinforcement Learning (MARL)

Deep Reinforcement Learning (DRL) has achieved big advances in multi-agent games

Multi-Agent Reinforcement Learning (MARL)

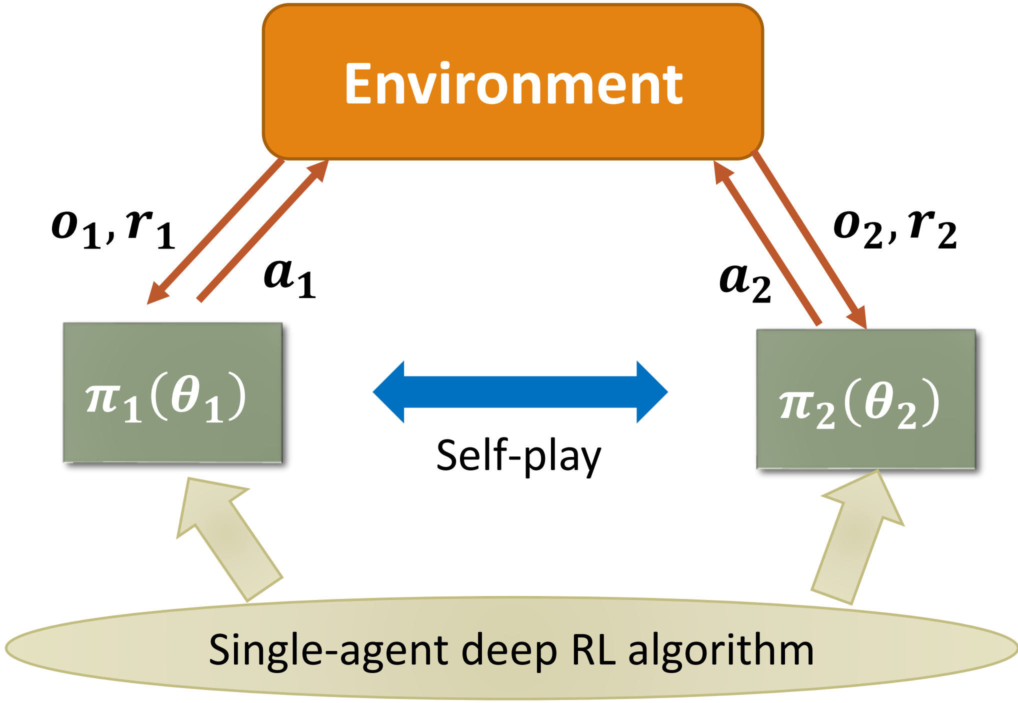
The common technique: *Deep RL + Self-play*

- Train multiple neural policies
- Let them compete against each other and optimize its own reward
- Each policy co-evolves and improves
- Until convergence

An elegant, general and effective solution

A significant issues: Non-stationary problem

- In practice, a lot of expert engineering required



Challenges in MARL

Non-stationary issue:

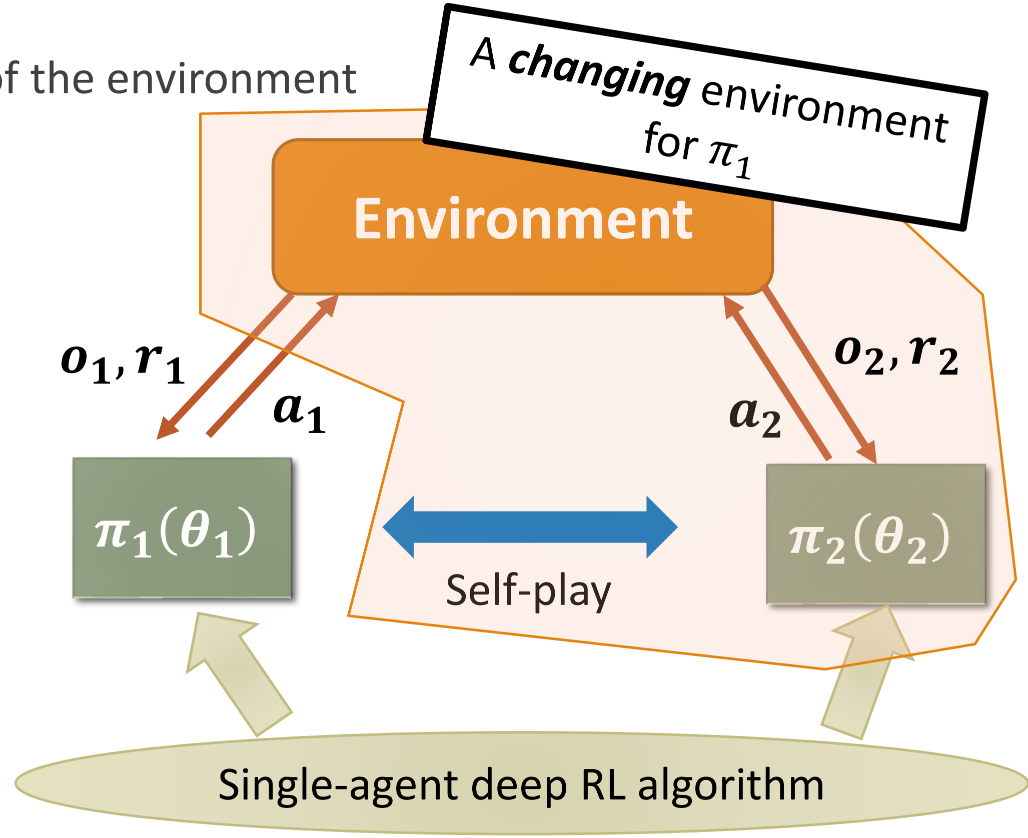
- RL assumes a stationary environment (fixed MDP)
- If directly apply single-agent RL, other agents becomes a part of the environment
- Non-stationery from each agent's own perspective

Challenge#1: unstable training

- Neural networks can hardly converge
- Degenerate to bad local mode

Challenge#2: easier to overfit

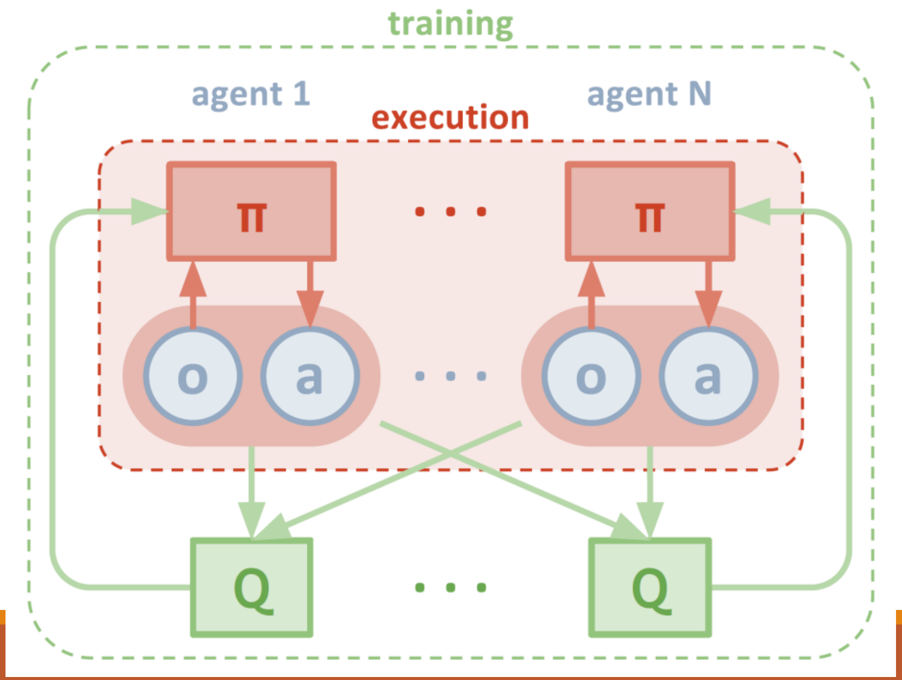
- Learned policy overfits its training partners
- Much worse when testing with a different opponent



Preliminary: MADDPG algorithm

MADDPG, the first general purpose deep MARL algorithm for stabilizing training (Challenge#1)

- **M**ulti-**A**gent **D**eep **D**eterministic **P**olicy **G**radient (MADDPG, Lowe*, Wu*, et.al., NIPS2017)
- Use actor-critic framework
  - Decentralized actors (policies, π) to keep the ***self-play framework***
  - Centralized critic (baseline, Q) to ease learning and ***reduce variance***
- Applies to mixed competitive and cooperative environments



What about overfitting (Challenge#2)?

- This work, M3DDPG
- A extension and variant of MADDPG for ***robust policies***

Overview

Goal: learning robust policies in deep MARL

Key Ideas:

- Minimax objective: introduce minimax concept into deep MARL
- Multi-Agent Adversarial Learning (MAAL): use techniques from ***adversarial training*** for tractable and practical approximate computation

The algorithm: **M**ini**M**ax **M**ulti-agent **D**eep **D**eterministic **P**olicy **G**radient (M3DDPG)

- Simple, efficient and improved MADDPG

Minimax MARL Objective

Minimax is a fundamental idea in zero-sum games

Minimax MARL: learning minimax deep policies

Challenge:
How to handle the inner-loop ***minimization?***
Continuous action space?

$$\max_{\theta} \mathbb{E}\left[\sum_t \gamma^t r_t(s_t, \pi(s_t|\theta))\right]$$

Classical RL

$$\max_{\theta} \mathbb{E}[Q(s_0, \pi(s_0|\theta))]$$

$$Q(s,a) = r(s,a) + \gamma Q(s',a')$$

Q-function (Q-learning)

$$\max_{\theta_i} \min_{a_{j\neq i}} \mathbb{E}\left[\sum_t \gamma^t r_t(s_t, \pi_i(s_t|\theta_i), \dots, a_j, \dots)\right]$$

Minimax MARL

$$\max_{\theta_i} \min_{a_{j\neq i}} \mathbb{E}[Q^M(s_0, \pi_i(s_0|\theta_i), \dots, a_j, \dots)]$$

$$Q^M(s, a_i, \dots, a_j, \dots) = r(s, a_1, \dots) + \gamma \min_{a'_{j\neq i}} Q^M(s', a'_1, \dots)$$

Minimax Q-function

Multi-Agent Adversarial Learning

The inner loop minimization is intractable and expensive to compute

- No closed-form solution
- An expensive inner-loop gradient descent optimization process

$$Q^M(s, a_i, a_j) = r(s, a_i, a_j) + \gamma Q^M\left(s', a_i', a_j'^{(*)}\right)$$

$$a_j'^{(*)} = \min_{a_j} Q^M(s', a_i', a_j)$$

Minimax Objective

Optimization

$$\boldsymbol{a_j'^{(*)}} \leftarrow a_j'^{(k+1)} = a_j'^{(k)} - \alpha \nabla_{a_j} Q^M(s', a_i', a_j)\Big|_{a_j = a_j'^{(k)}}$$

One-step relaxation

MAAL

$$\boldsymbol{a_j'^{(*)}} = \pi_j(s') - \alpha \nabla_{a_j} Q^M(s', a_i', a_j)\Big|_{a_j = \pi_j(s')}$$

Multi-agent Adversarial Learning (MAAL)

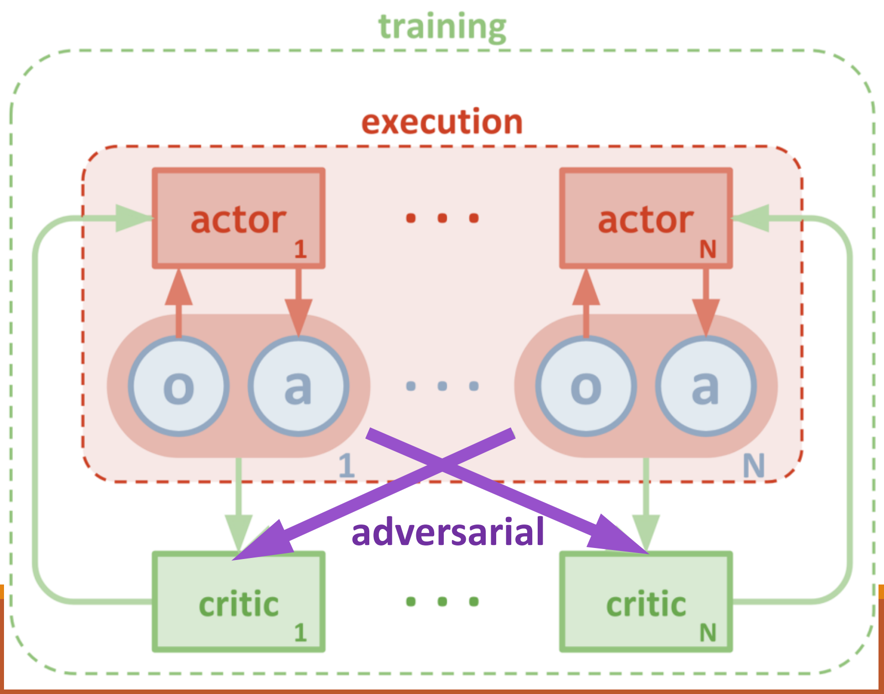
- Key idea: replace the inner-loop minimization by ***a one-step gradient descent***
- Motivated by recent successes of adversarial training (Goodfellow, et. al, ICLR 2014), and meta-learning (MAML, Finn, et. al., ICML 2017)
- Fully differentiable, efficient approximation, effective in practice

M3DDPG

MADDPG + Minimax $\approx$ Minimax MADDPG (M3DDPG)

Adversarial Learning

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\mathbf{x},a\sim\mathcal{D}}\left[\begin{array}{l}\nabla_{\theta_i}\boldsymbol{\mu}_i(o_i)\nabla_{a_i}Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x},a_1^\star,\ldots,a_i,\ldots a_N^\star)\Big| \\ \quad a_i=\boldsymbol{\mu}_i(o_i) \\ \quad a_j^\star=a_j+\hat{\epsilon}_j,\quad \forall j\neq i \\ \quad \hat{\epsilon}_j=-\alpha_j\nabla_{a_j}Q^{\boldsymbol{\mu}}_{\mathrm{M},i}(\mathbf{x},a_1,\ldots,a_N)\end{array}\right]$$



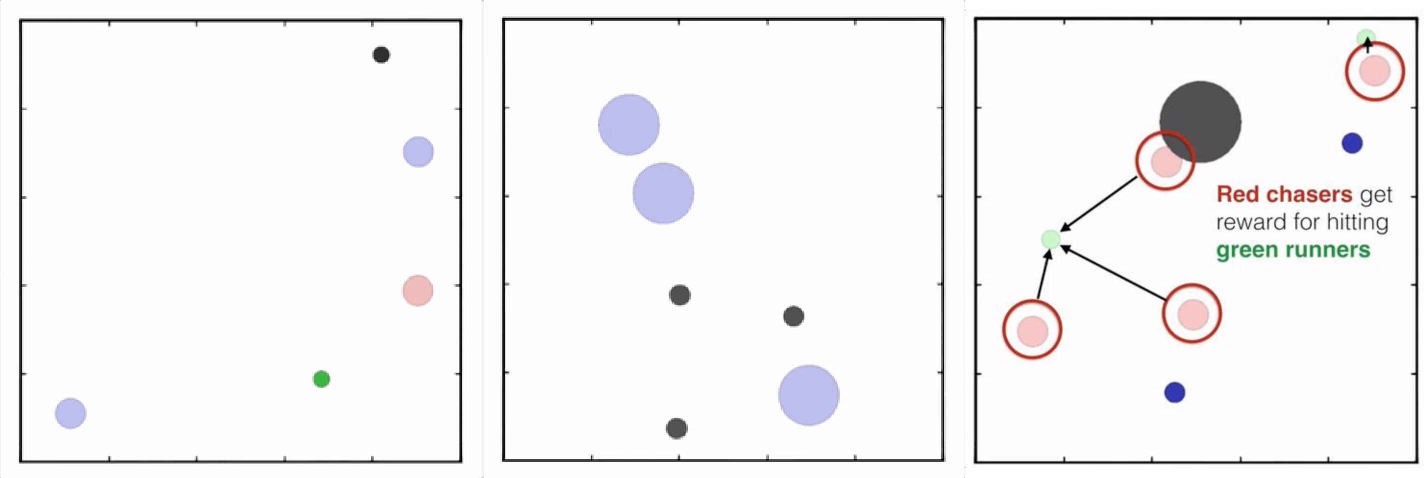
M3DDPG

- Fast, efficient approximate algorithm for robust learning
- ***Trade-off between objective and practical computation***

Experiments

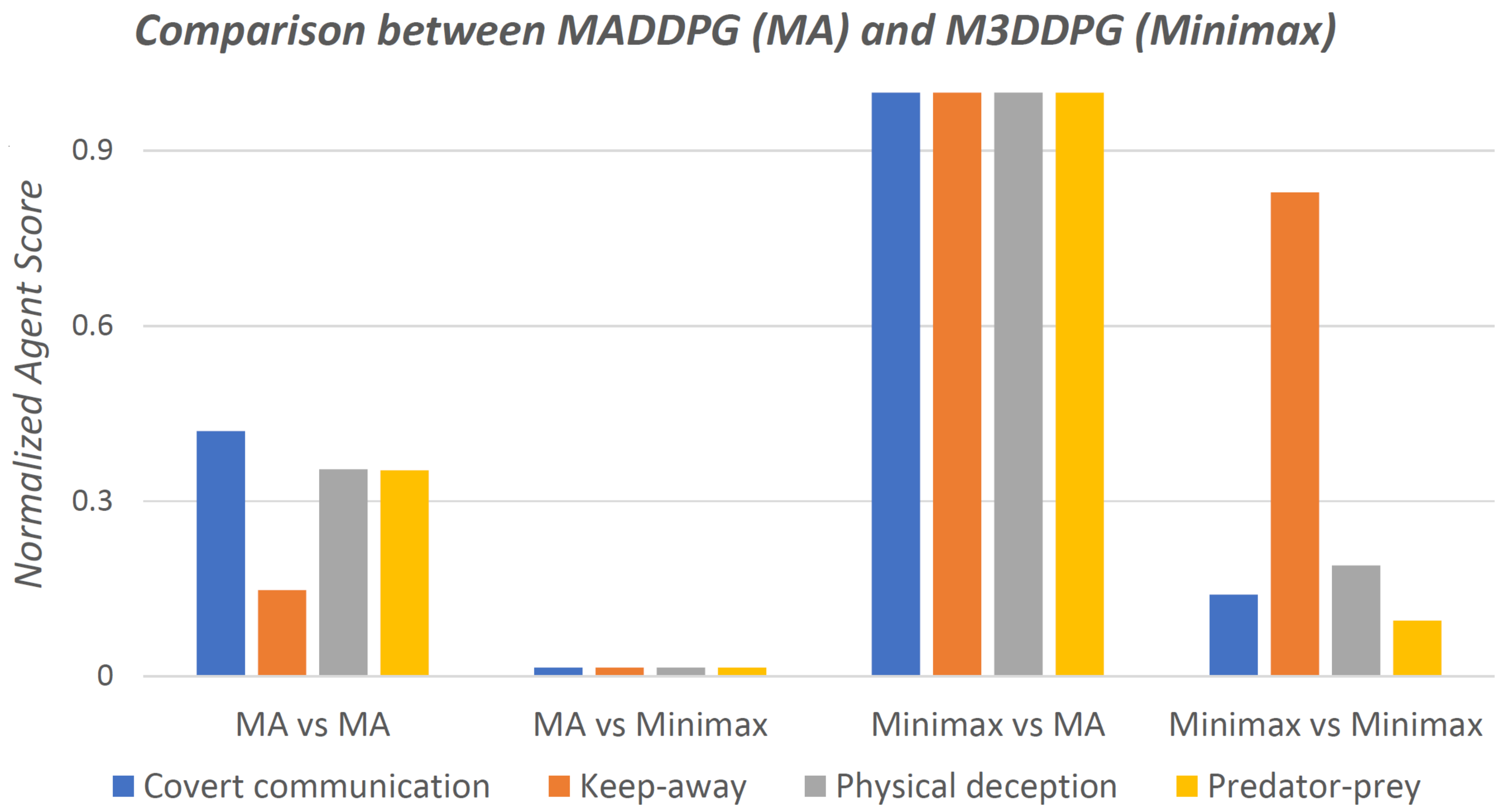
Environment and tasks

- The particle world environment from MADDPG (demo below)



Test#1: competition between M3DDPG and MADDPG

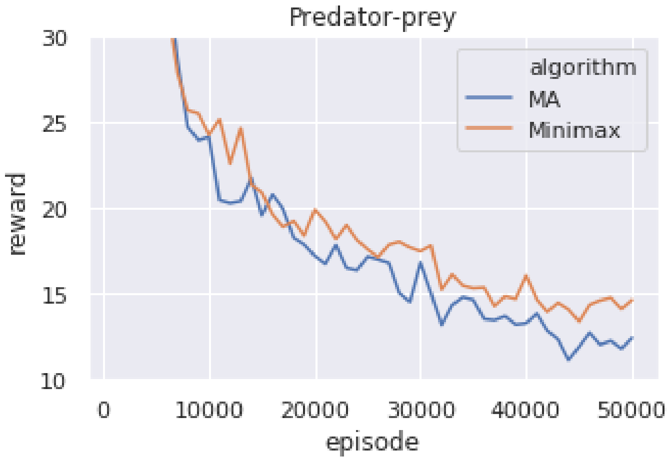
Test#2: performance against the best response

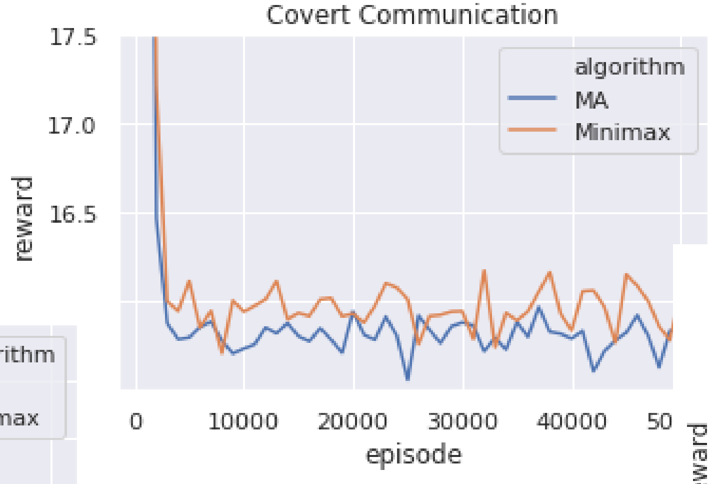

Comparison between MADDPG (MA) and M3DDPG (Minimax)
Normalized Agent Score
0.9
0.6
0.3
0
MA vs MA
MA vs Minimax
Minimax vs MA
Minimax vs Minimax
Covert communication
Keep-away
Physical deception
Predator-prey

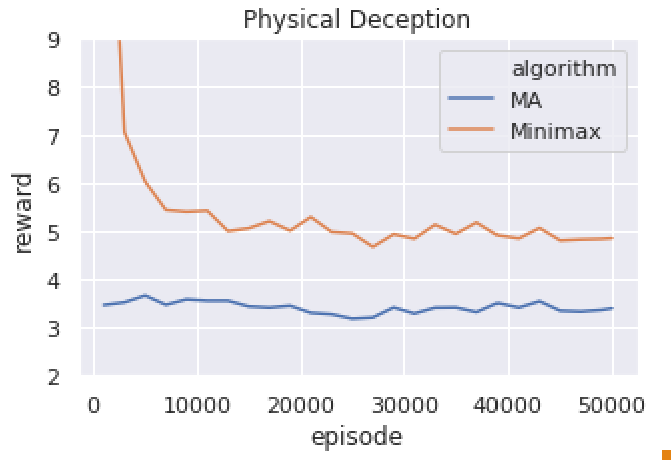
Worst Case Performance (BR)

Evaluate the performance of M3DDPG & MADDPG against its ***best response*** policy

- Fixed the policies learned by M3DDPG & MADDPG
- Retrain a single opponent (reduce to single-agent RL setting, *easy*)







Conclusion

Take-home message

- Minimax MARL + adversarial learning → M3DDPG (Minimax MADDPG)
- A fully differentiable, fast, general algorithm for robust deep MARL
- Interpretation: adversarial training extension of MADDPG

GitHub:

- https://github.com/dadadidodi/m3ddpg

Thanks!