Learning with Opponent Q-Learning Awareness (ICLR 2024)




Milad Aghajohari* Juan Duque* Tim Cooijmans Aaron Courville


With
LOQA
Without
LOQA

A simple game where naïve Multi-Agent RL fails

A simple game where naïve Multi-Agent RL fails

Wait, what do we mean by naïve Multi-Agent RL?

Naïve Multi-Agent RL

$$\theta_{i+1}^{1} = \underset{\theta_1}{\operatorname{argmax}} V^{1}(\theta_i^1, \theta_i^2)$$
$$\theta_{i+1}^{2} = \underset{\theta_2}{\operatorname{argmax}} V^{2}(\theta_i^1, \theta_i^2)$$

Naïve Multi-Agent RL with Shared Rewards

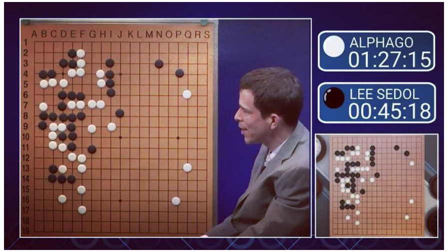
$$\theta^1_{i+1} = \underset{\theta_1}{\operatorname{argmax}}\,(V^1 + V^2)(\theta^1_i, \theta^2_i)$$
$$\theta^2_{i+1} = \underset{\theta_2}{\operatorname{argmax}}\,(V^2 + V^1)(\theta^1_i, \theta^2_i)$$

MARL performance on games

MARL performance on games

Zero-Sum games

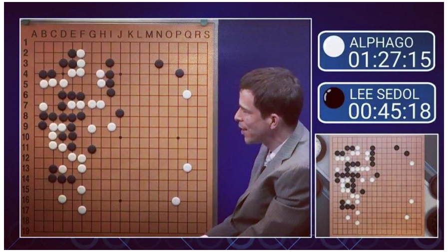
Beats professional players



MARL performance on games

Zero-Sum games

Beats professional players



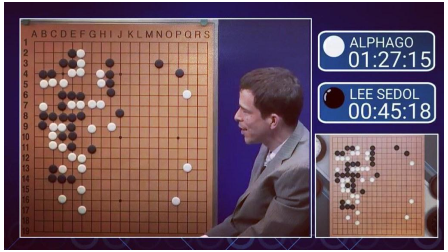
Fully Cooperative games

Cooperates with professional players

MARL performance on games

Zero-Sum games

Beats professional players

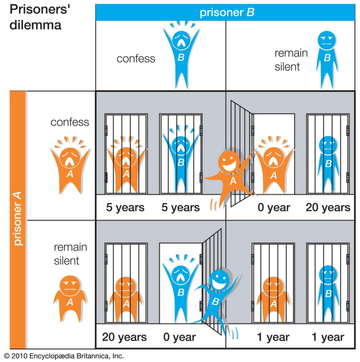


Fully Cooperative games

Cooperates with professional players

General-Sum games

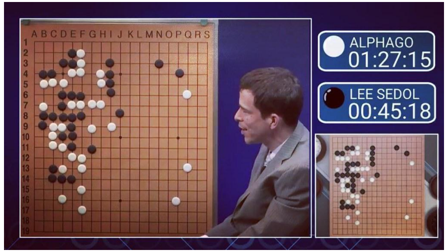
Fails in 2 action matrix game



MARL performance on games

Zero-Sum games

Beats professional players

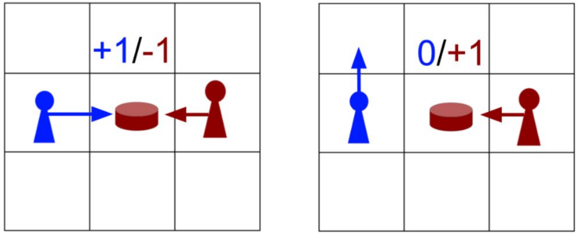


Fully Cooperative games

Cooperates with professional players

General-Sum games

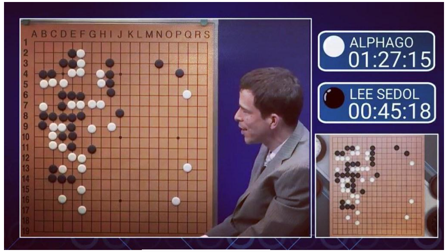
Fails in 3x3 board game



MARL performance on games

Zero-Sum games

Beats professional players

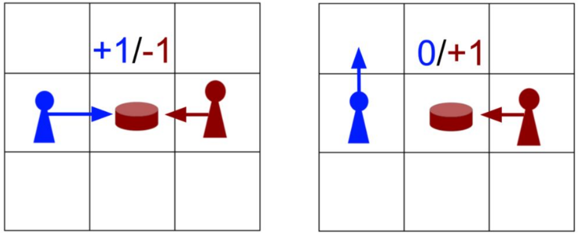


Fully Cooperative games

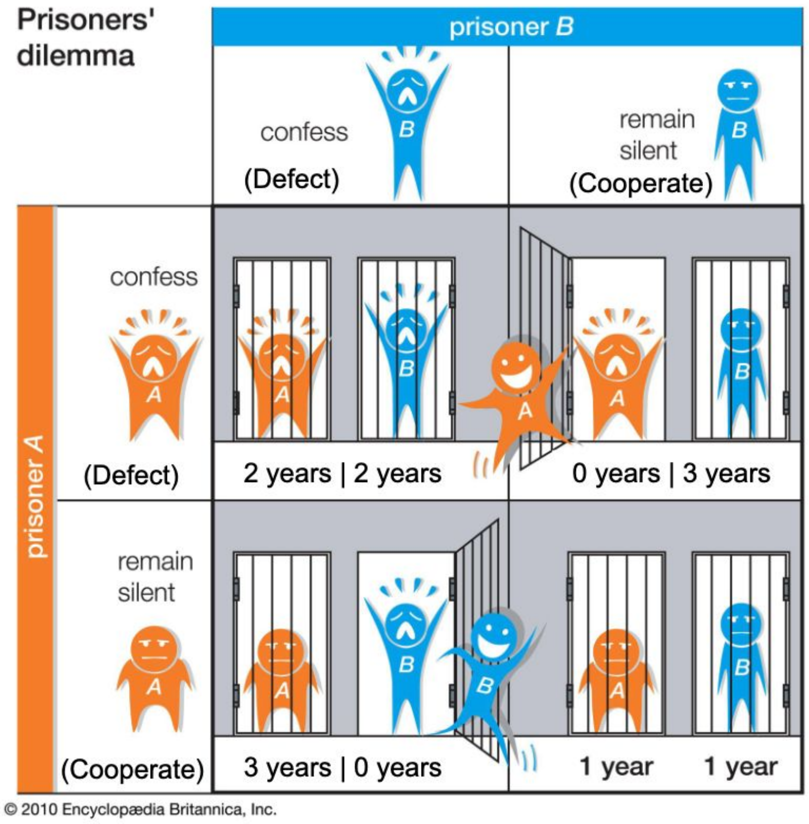
Cooperates with professional players

General-Sum games

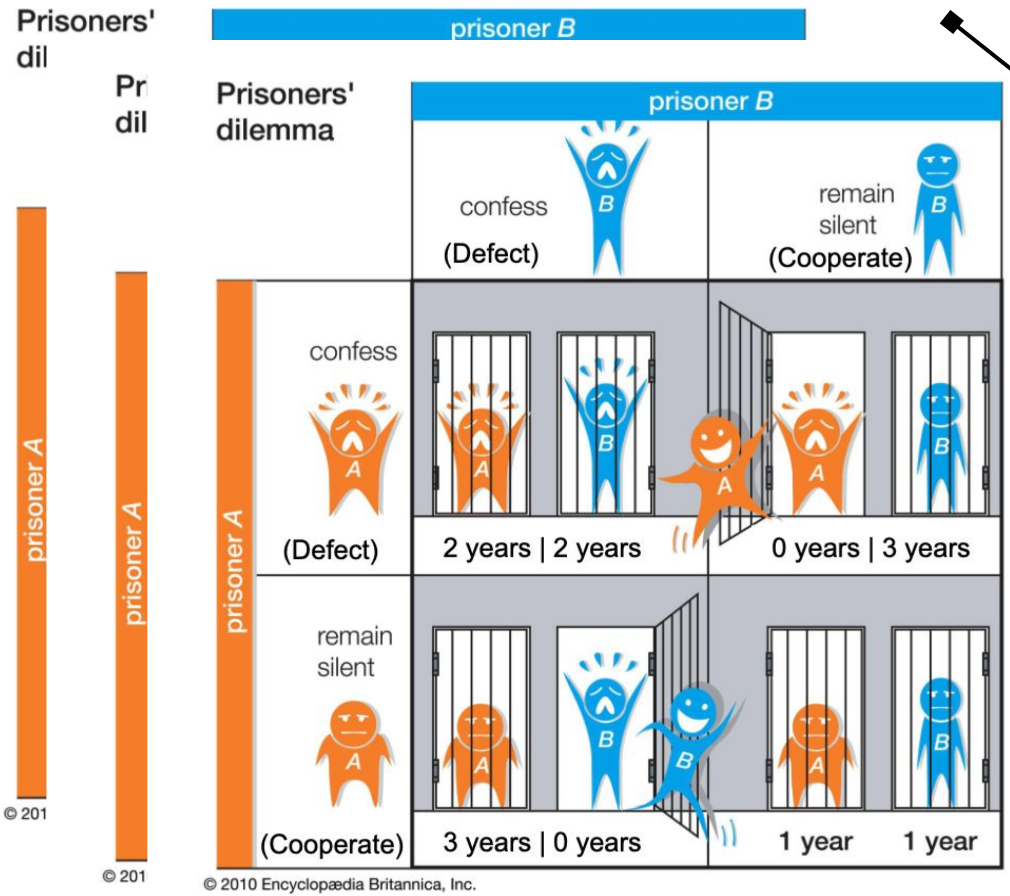
Fails in 3x3 board game



A simple game where naïve Multi-Agent RL fails



Iterated Prisoner's Dilemma

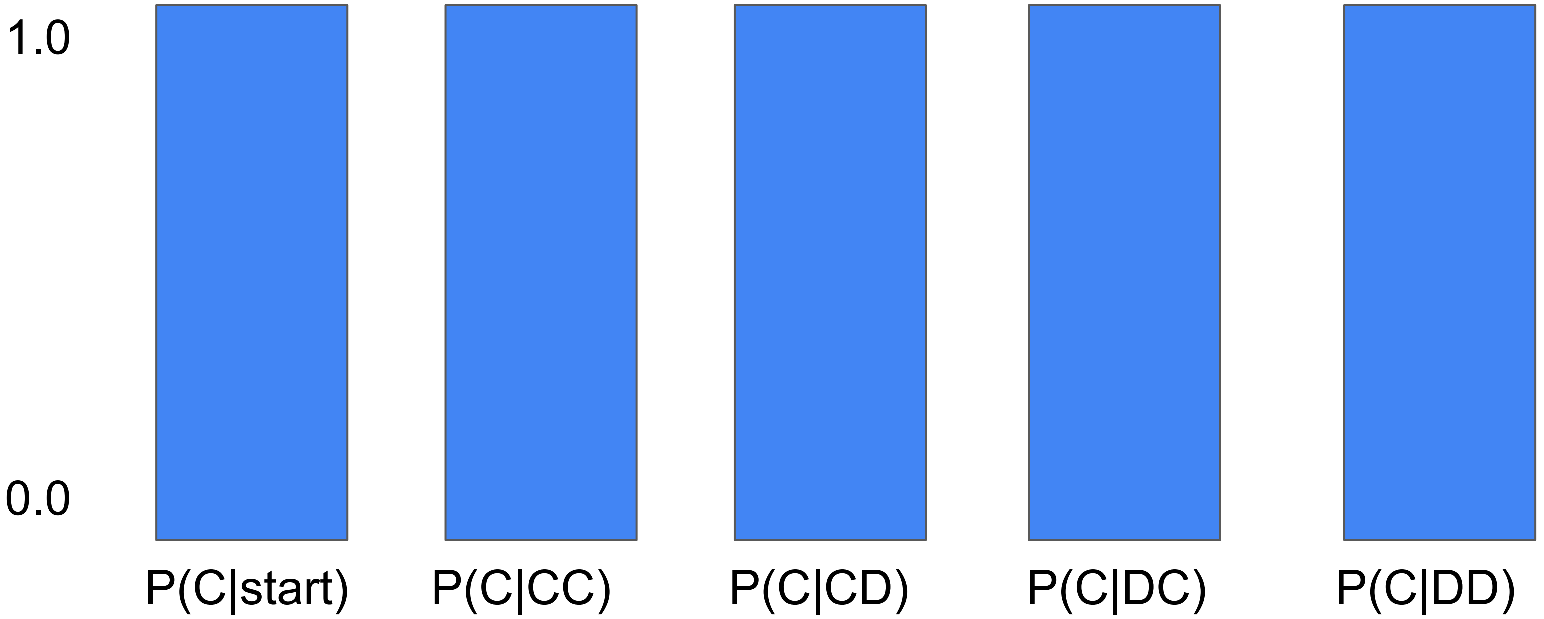


Repeated ∞ times

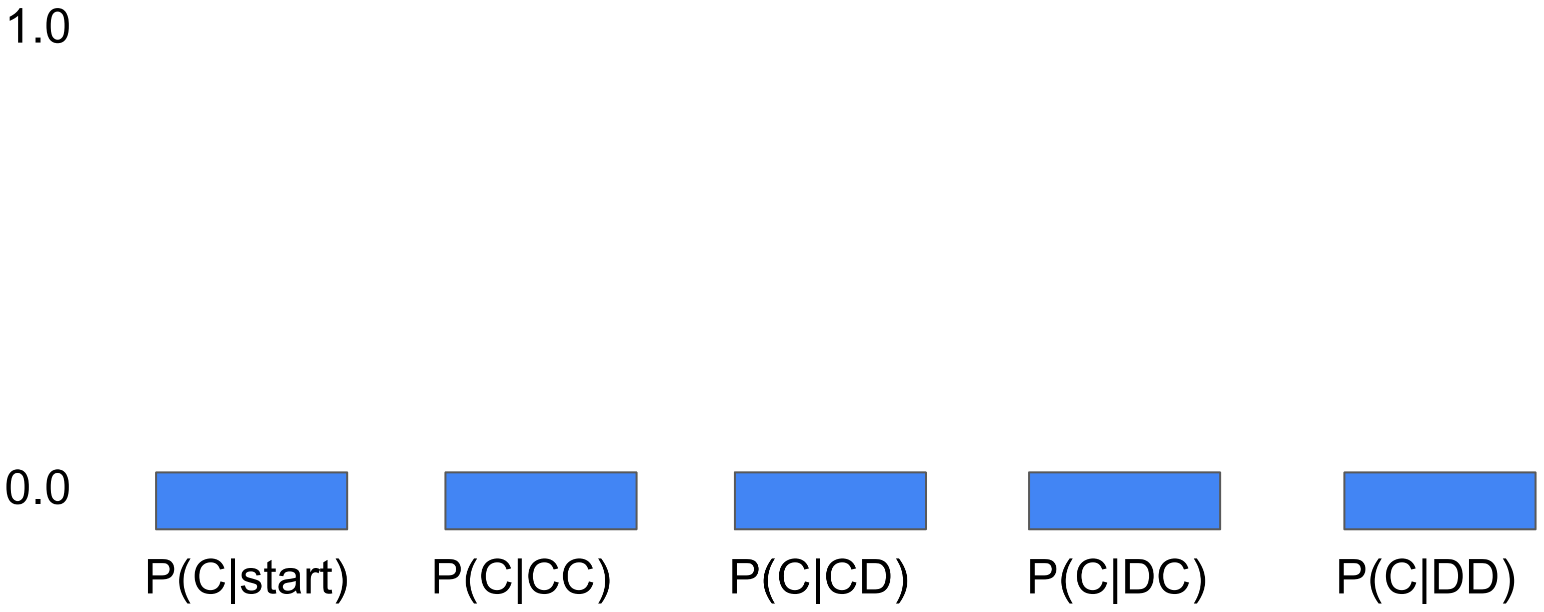
Policy via 5 logits

- One step of history
- There are 5 possible state: start, CC, CD, DC, DD
- The policy assigns the probability of cooperation given the state
- The policy can just be modeled by 5 logits
- P(C|start), P(C|CC), P(C|CD), P(C|DC), P(C|DD)

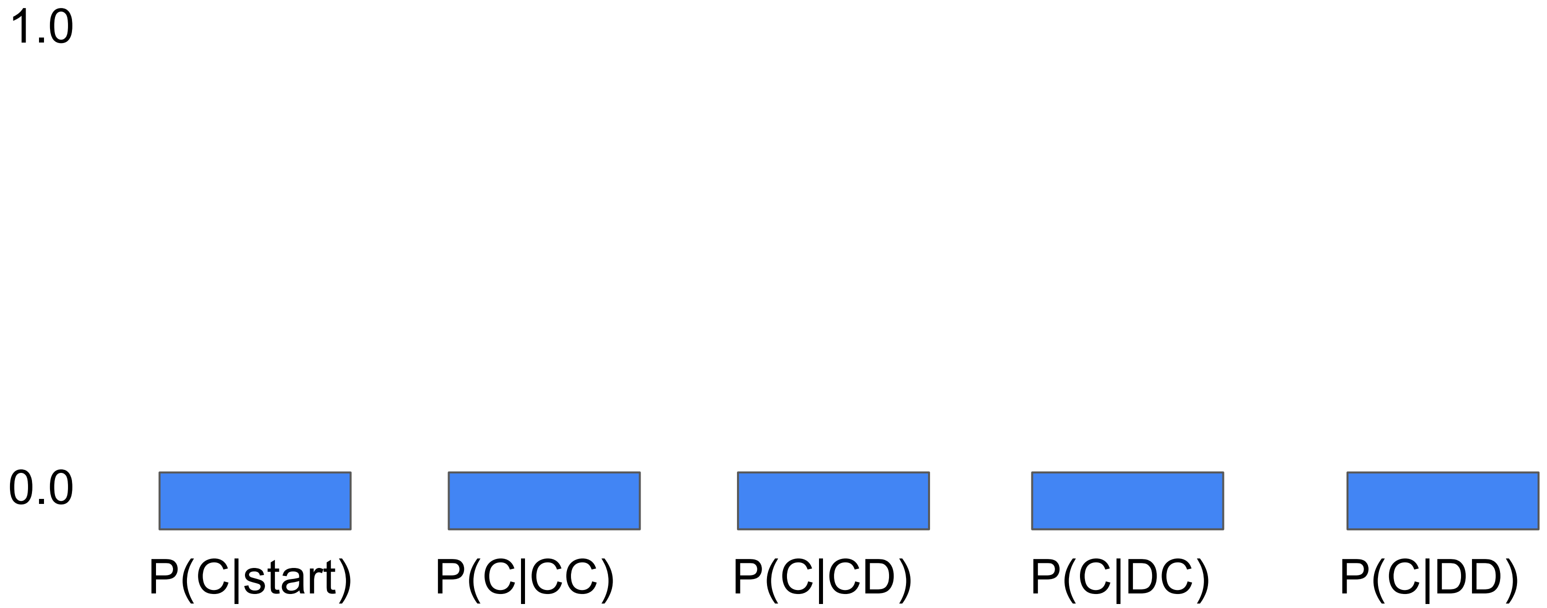
Always Cooperate



Always Defect



Always Defect



Naïve learners learn either AC or AD

1. Naïve learners with no shared reward learn Always Defect.

2. Naïve learners with shared reward learn Always Cooperate.

AD and AC, both are not desirable

- Always Defect exploits Always Cooperate, but does not cooperate well with itself
- Always Cooperates cooperates with itself, but gets exploited

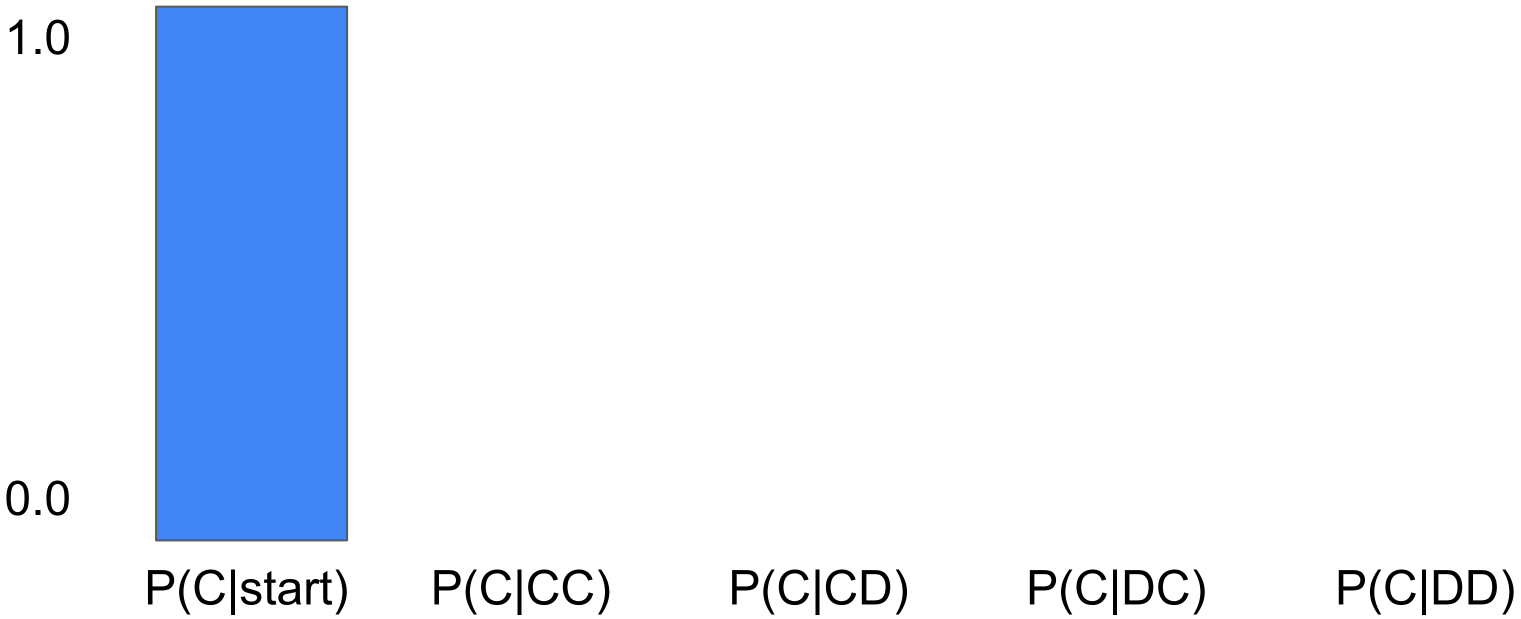
Tit for Tat

1.0

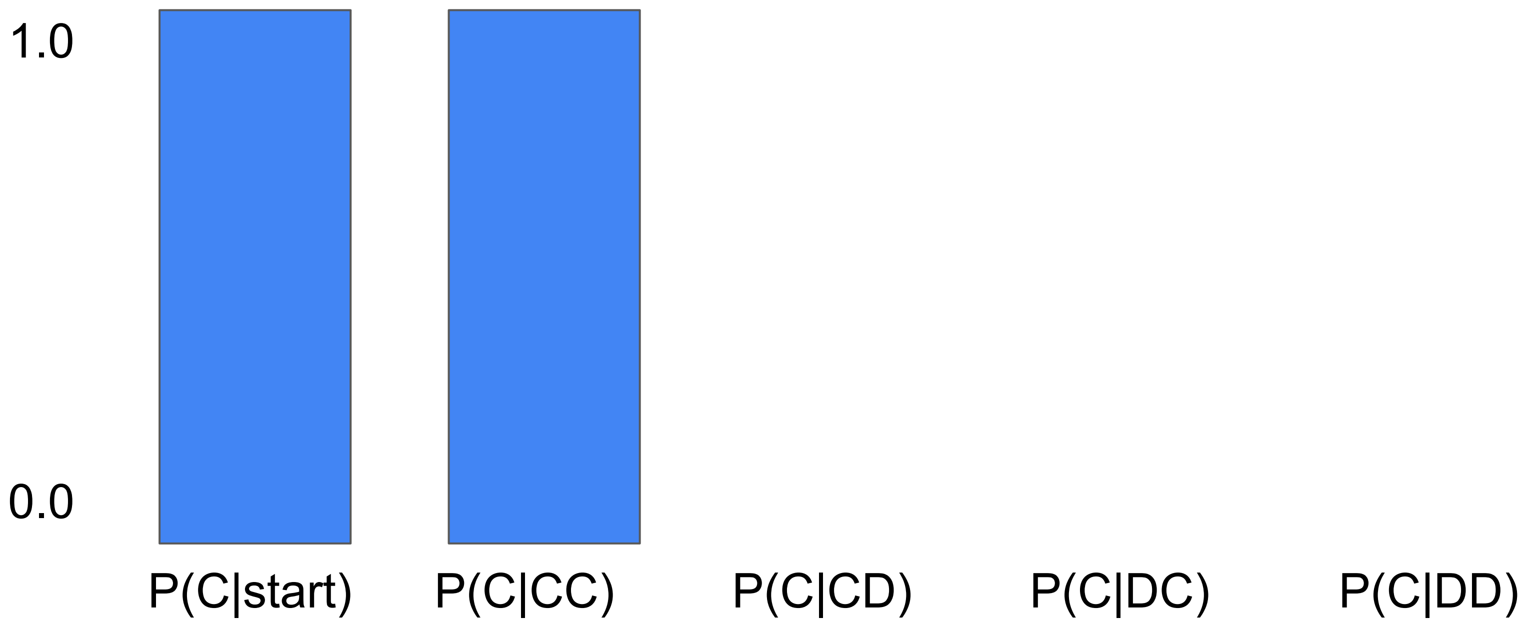
0.0

P(C|start) P(C|CC) P(C|CD) P(C|DC) P(C|DD)

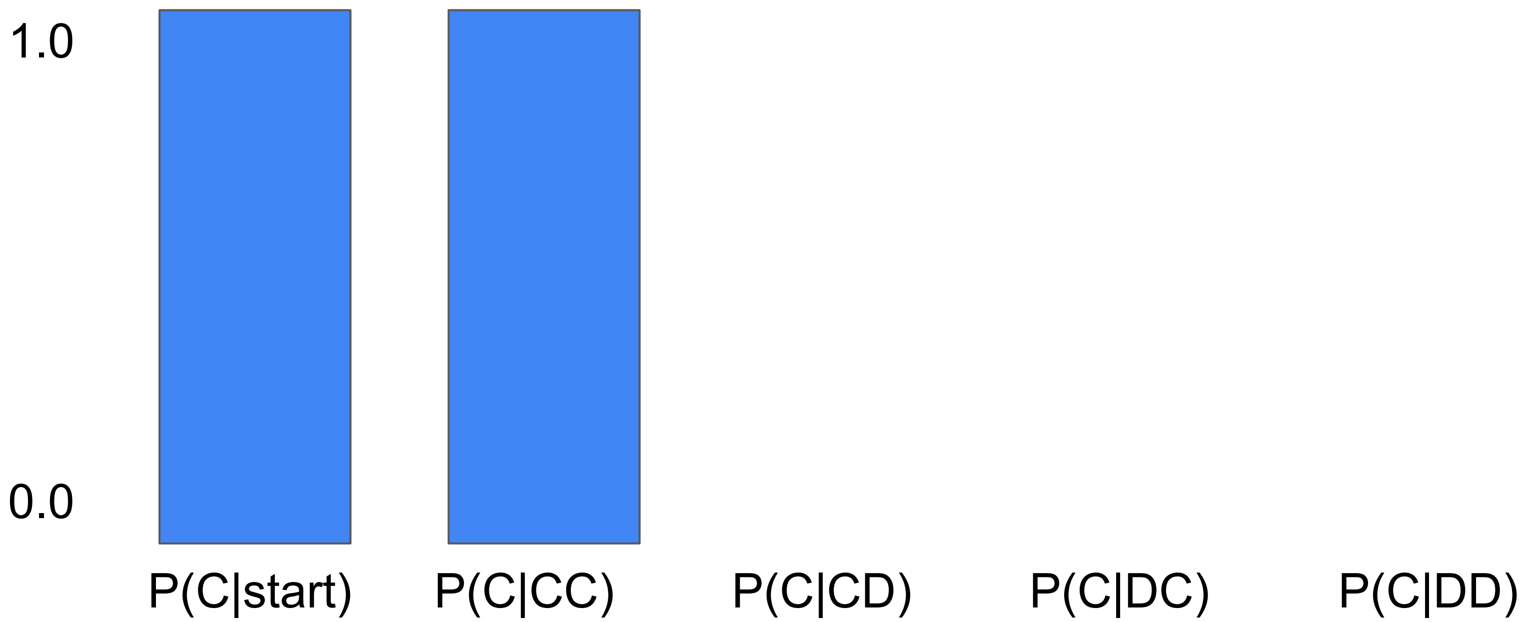
Tit for Tat



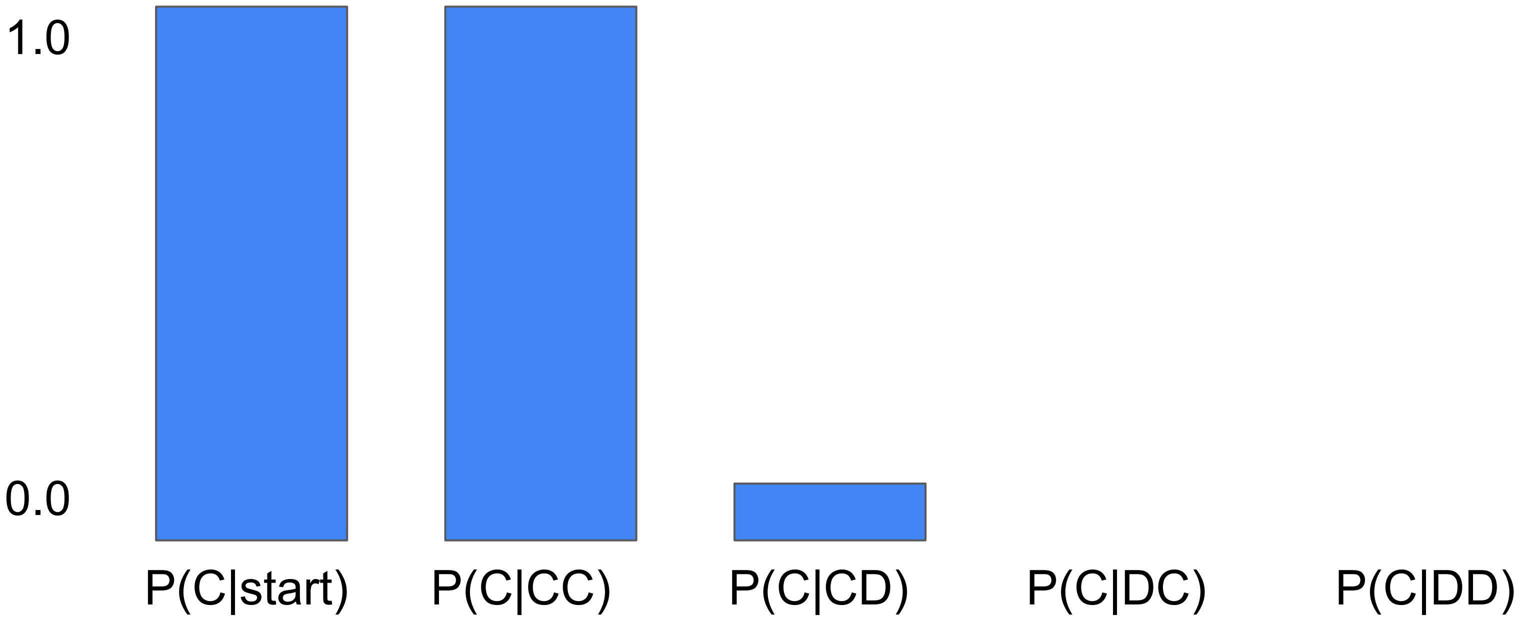
Tit for Tat



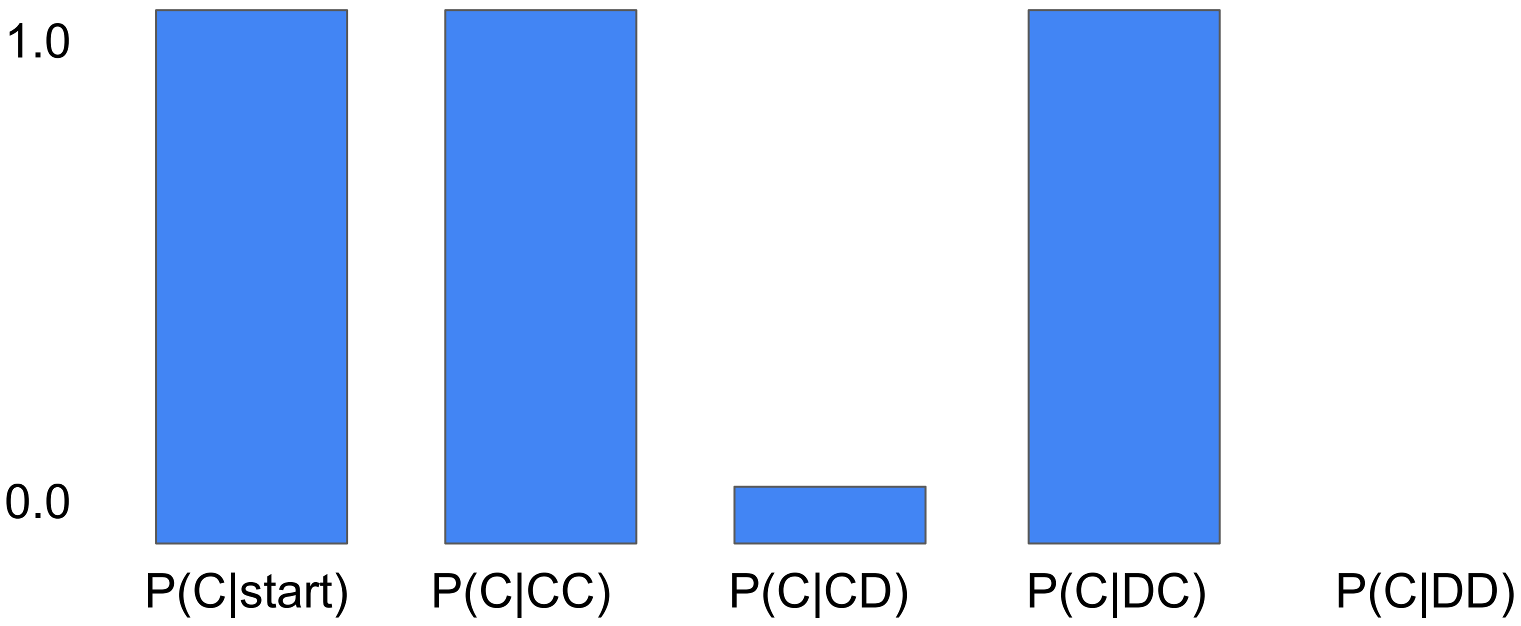
Tit for Tat



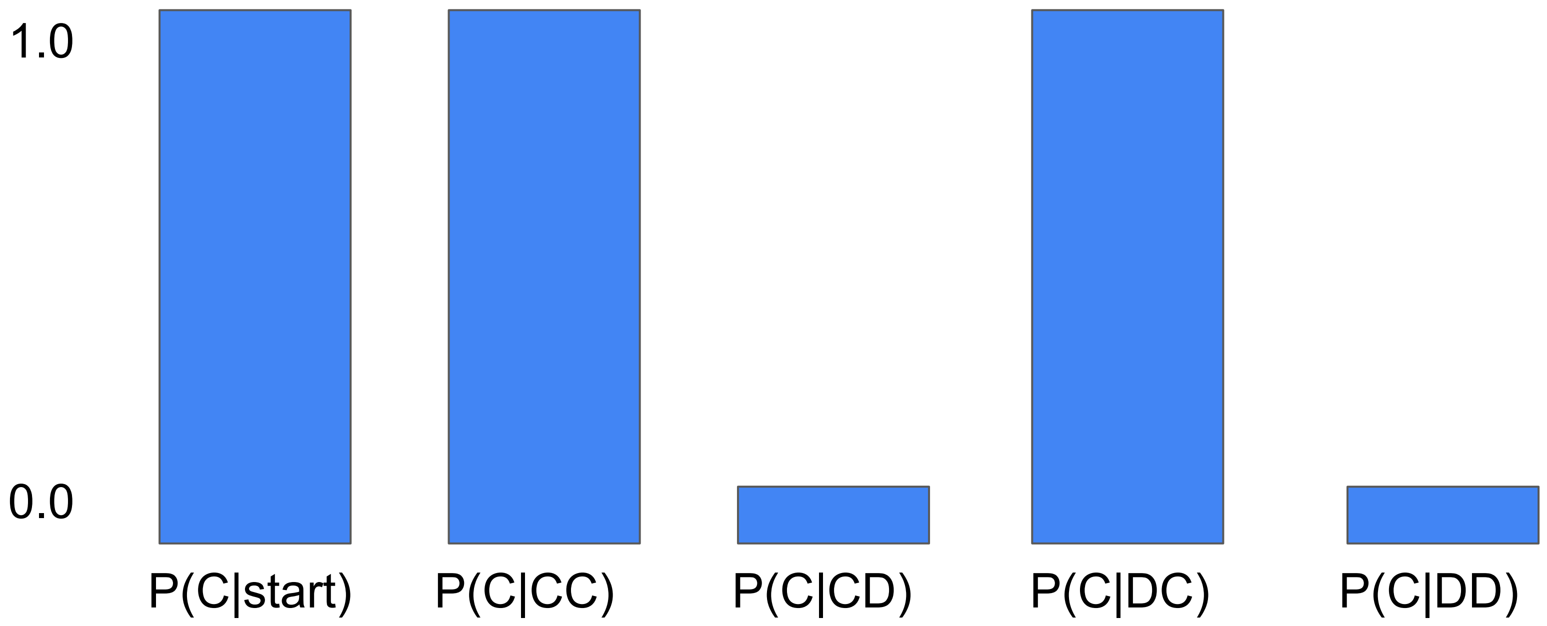
Tit for Tat



Tit for Tat



Tit for Tat



LOLA: Learning with Opponent Learning Awareness

$$\theta^1_{i+1} = \underset{\theta_1}{\operatorname{argmax}}\, V^1(\theta^1_i, \theta^2_i \qquad\qquad\qquad)$$

LOLA: Learning with Opponent Learning Awareness

$$\theta^1_{i+1} = \underset{\theta_1}{\operatorname{argmax}}\, V^1(\theta^1_i, \theta^2_i + \boxed{\Delta\theta^2_i})$$

LOLA: Learning with Opponent Learning Awareness

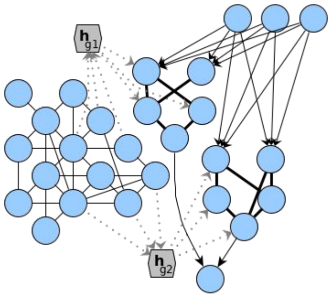
$$\theta^1_{i+1} = \underset{\theta_1}{\operatorname{argmax}} V^1(\theta^1_i, \theta^2_i + \boxed{\Delta\theta^2_i})$$

$$\Delta\theta^2_i \approx \eta . \nabla_{\theta^2_i} V^2(\theta^1_i, \theta^2_i)$$

LOLA: Learning with Opponent Learning Awareness

$$\theta^1_{i+1} = \underset{\theta_1}{\operatorname{argmax}} V^1(\theta^1_i, \theta^2_i + \boxed{\Delta\theta^2_i})$$

$$\Delta\theta^2_i \approx \eta . \nabla_{\theta^2_i} V^2(\theta^1_i, \theta^2_i)$$



LOLA learns TFT in IPD

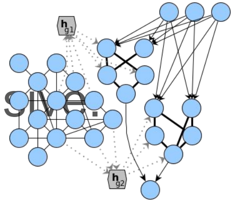
So, what is LOLA’s problem?

The problem with LOLA

1 - Need high learning rate for approximating opponent's optimization.

2 - Large steps do not approximate optimization of neural networks accurately (at all!)

3 - Differentiating through explicit optimization steps of opponent is expensive.

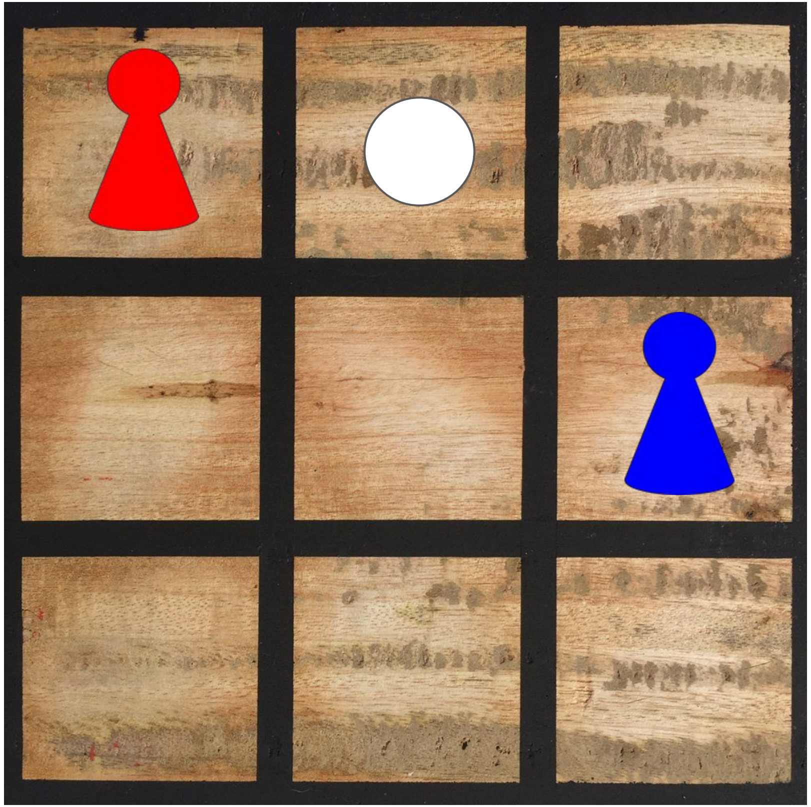


Why LOLA, POLA, M-FOS are not scalable?

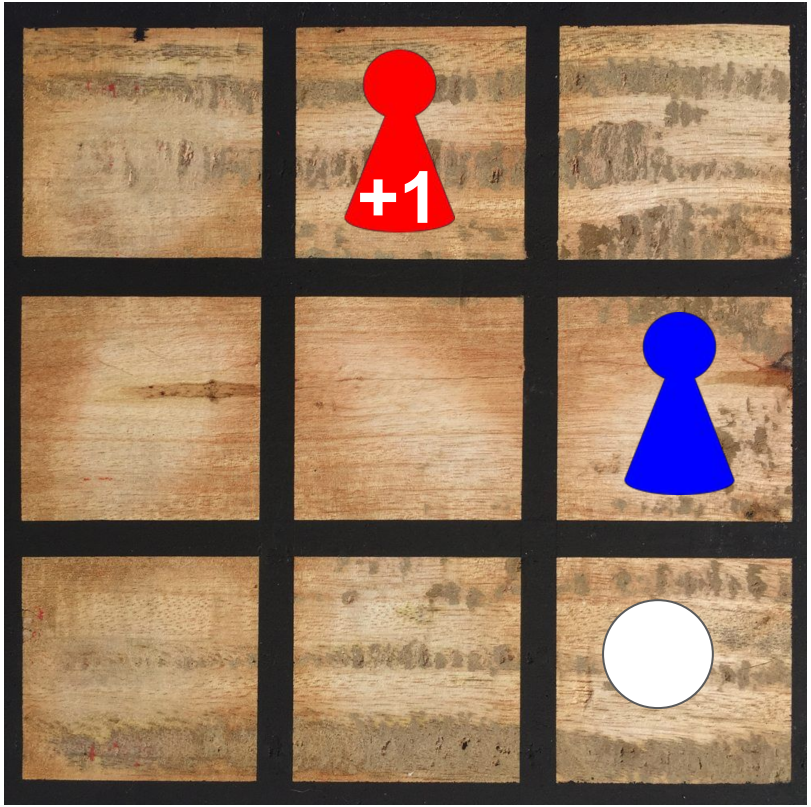
1-Explicit Computation graph of the optimization (LOLA, POLA)

2-Meta games are expensive (M-FOS)

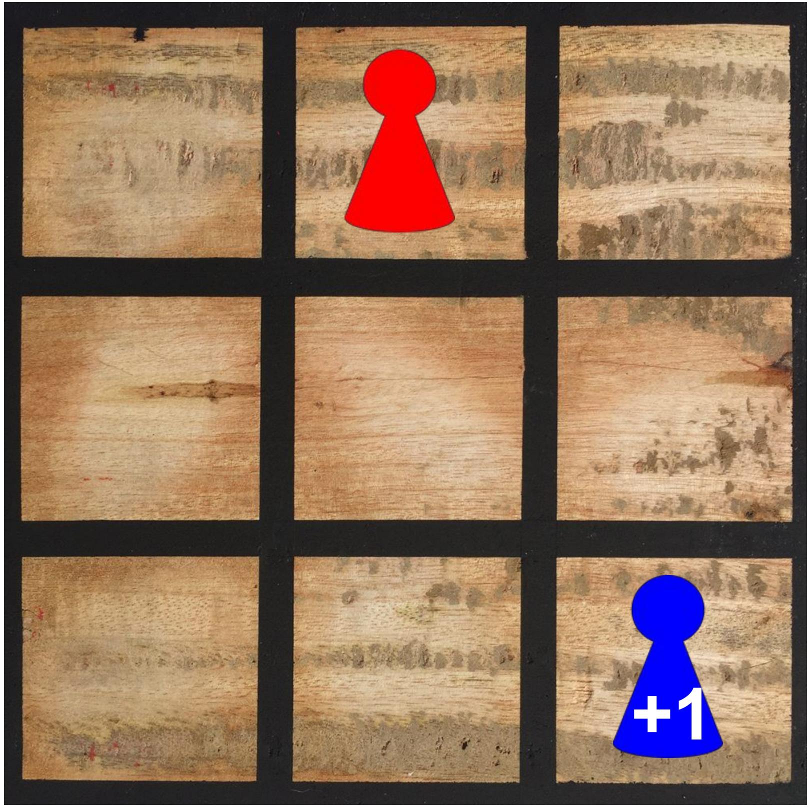
The Coin Game



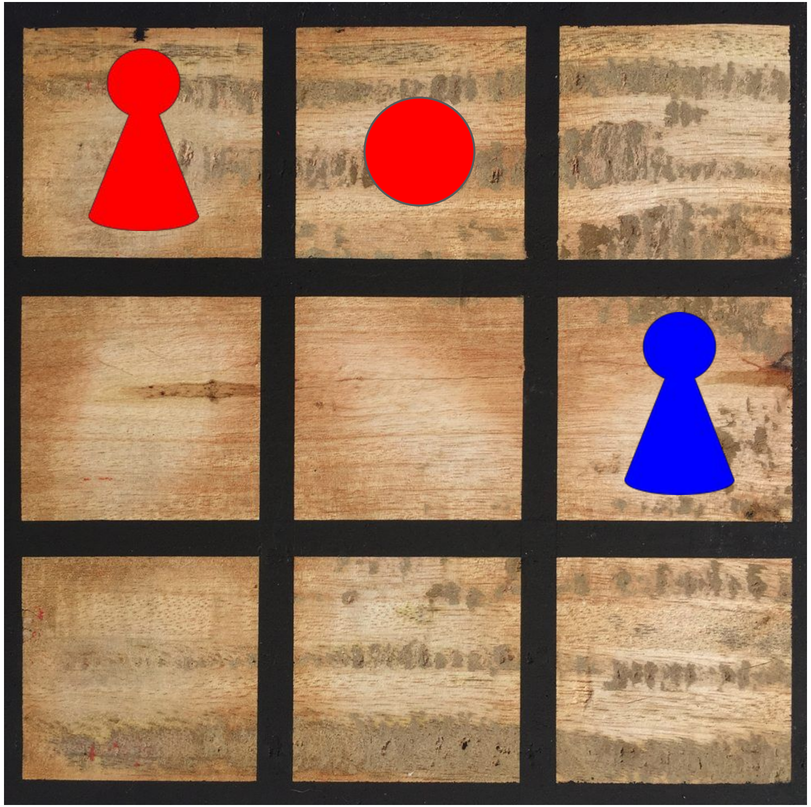
The Coin Game



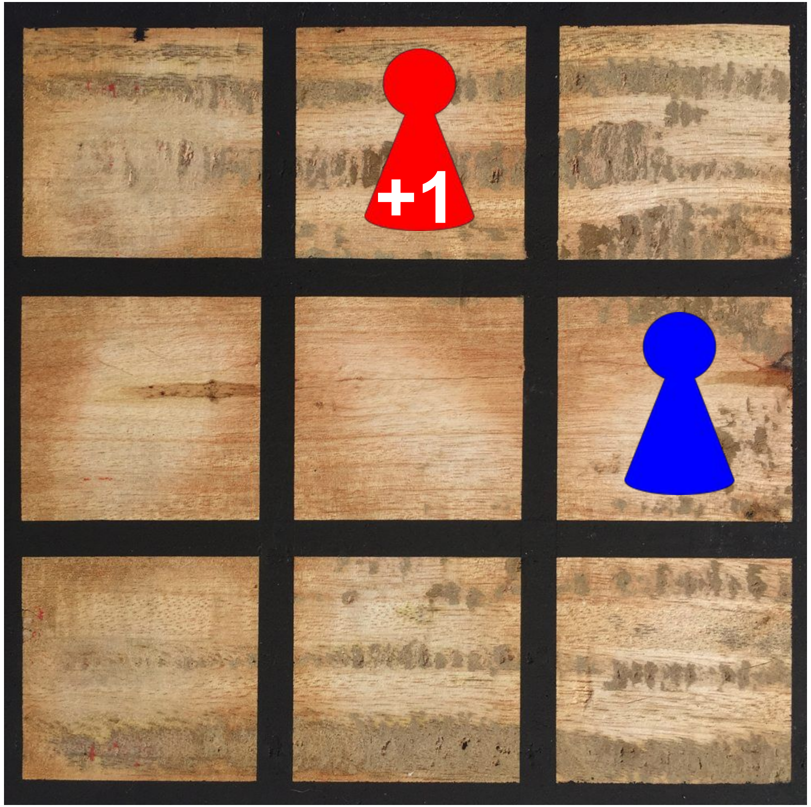
The Coin Game



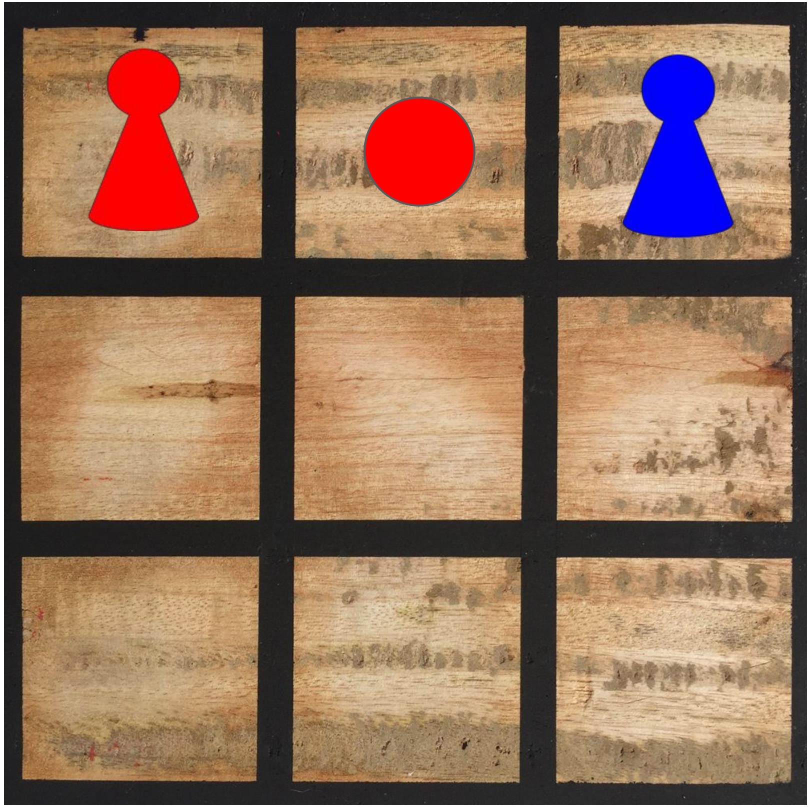
The Coin Game



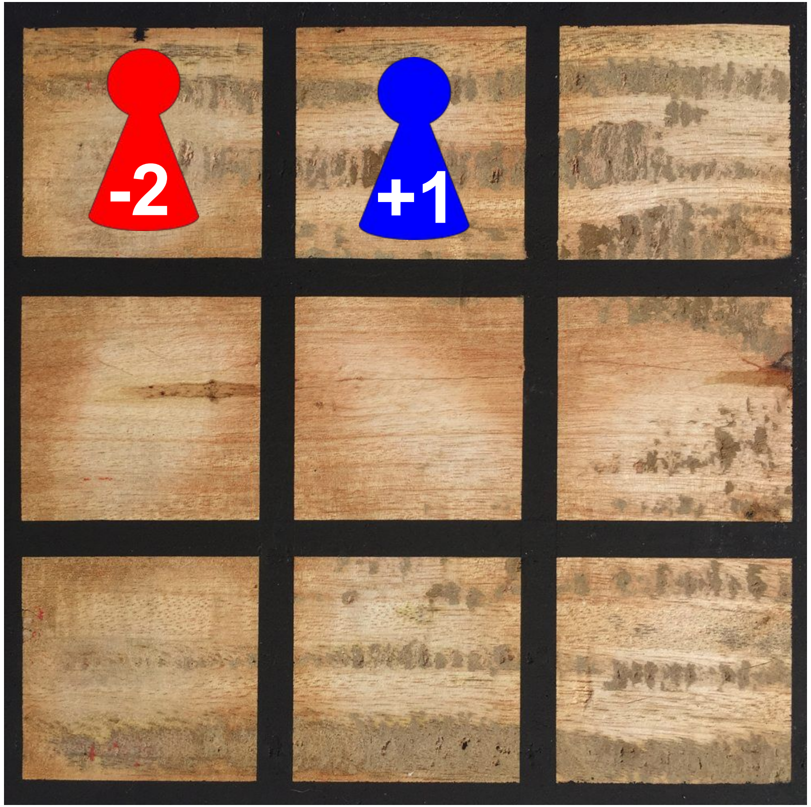
The Coin Game



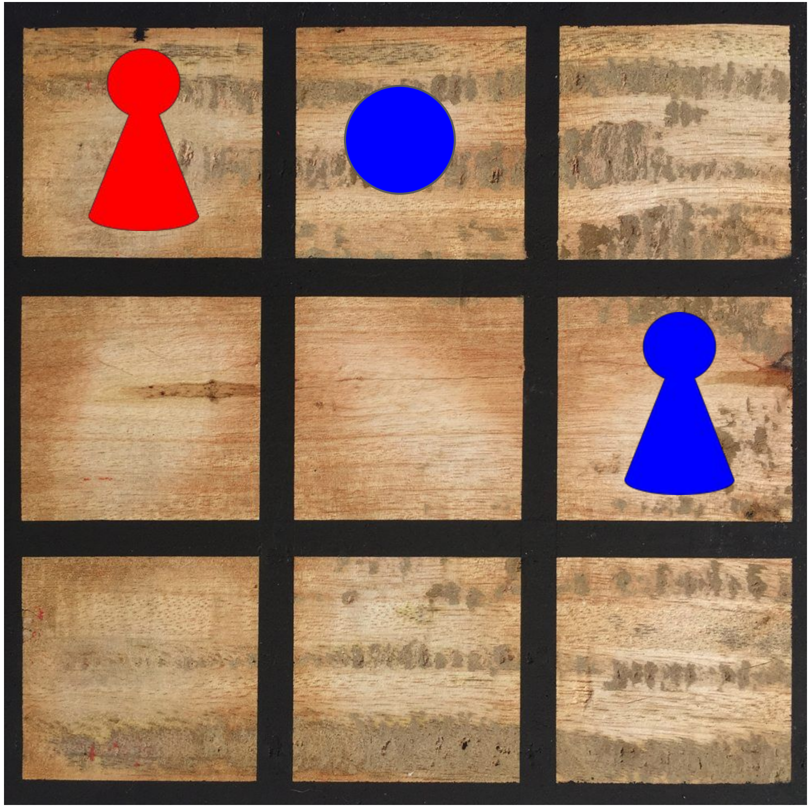
The Coin Game



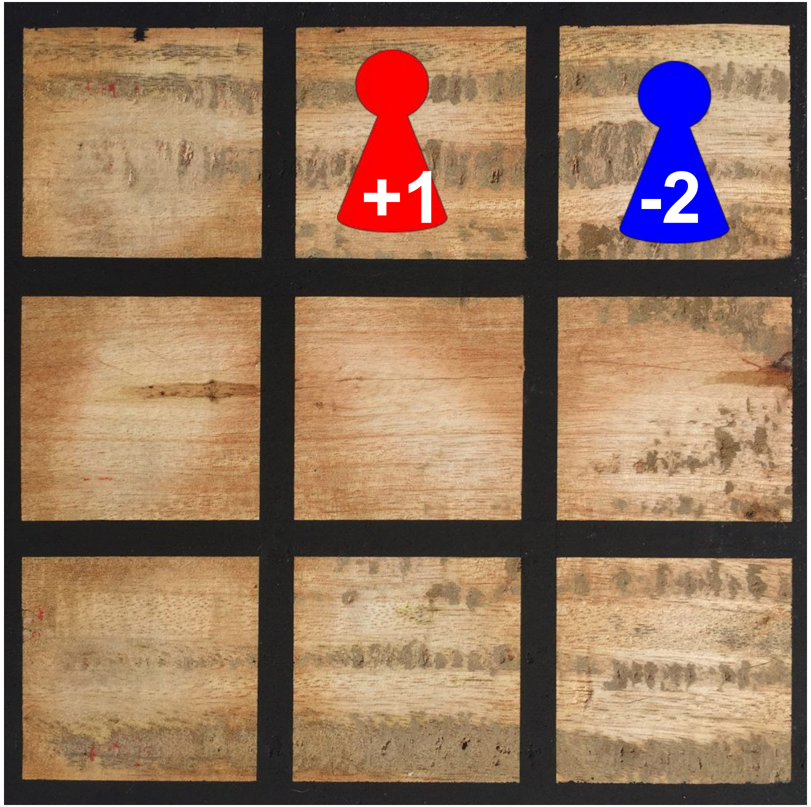
The Coin Game



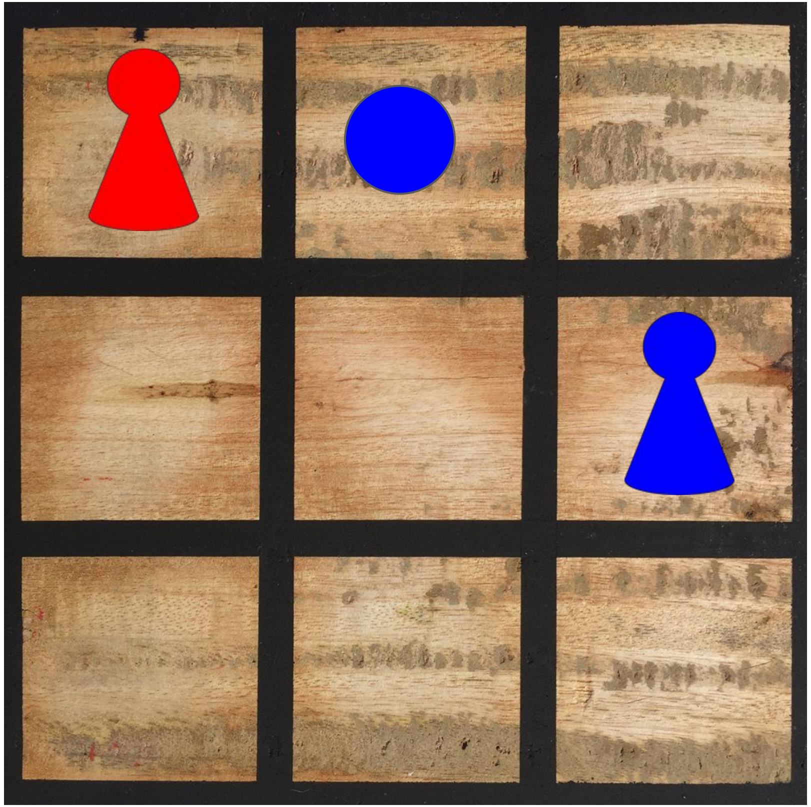
Always Defect



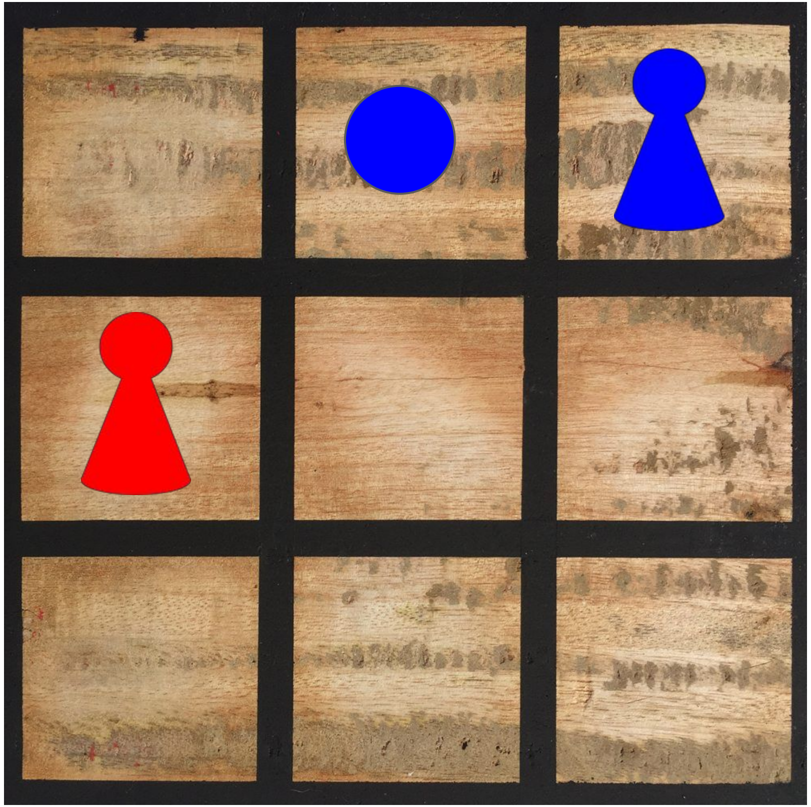
Always Defect



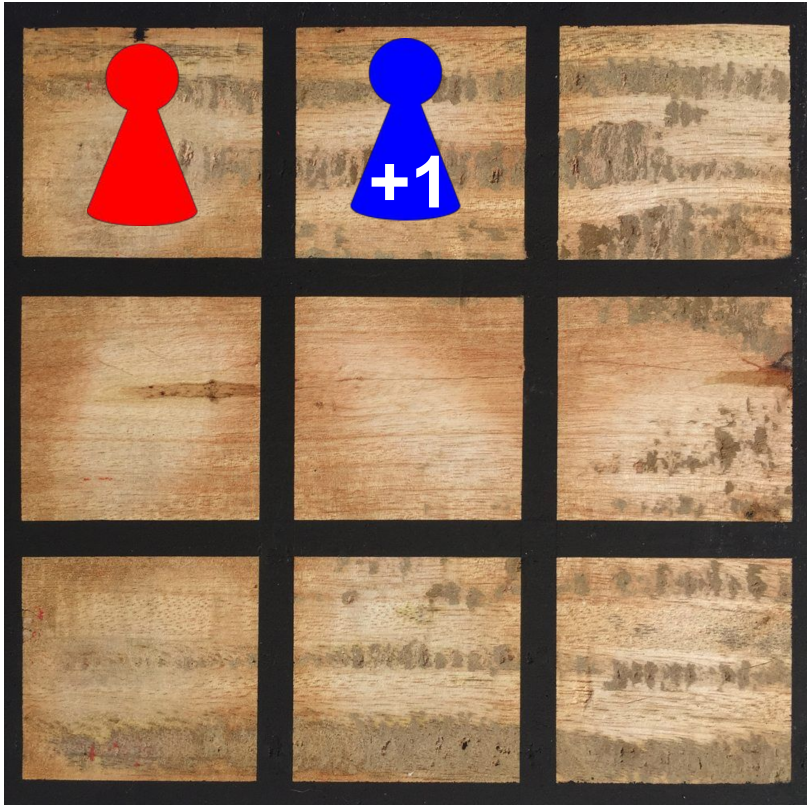
Always Cooperate



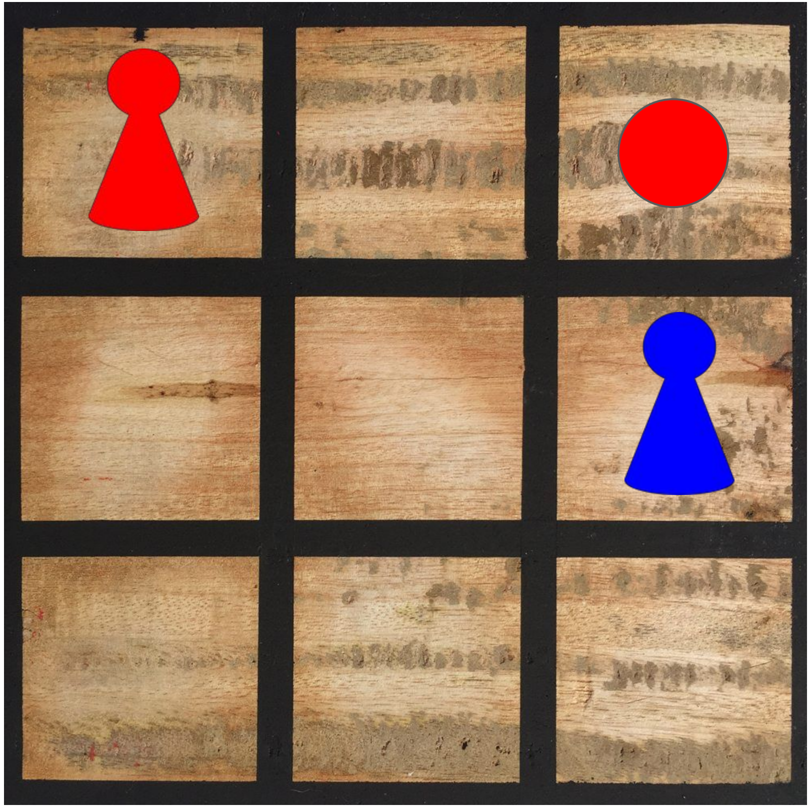
Always Cooperate



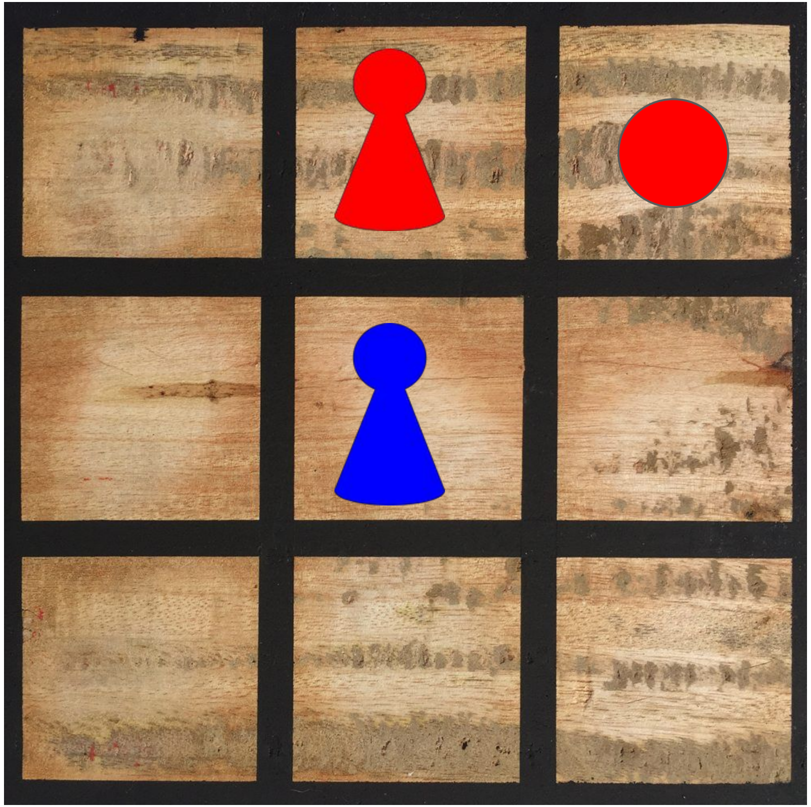
Always Cooperate



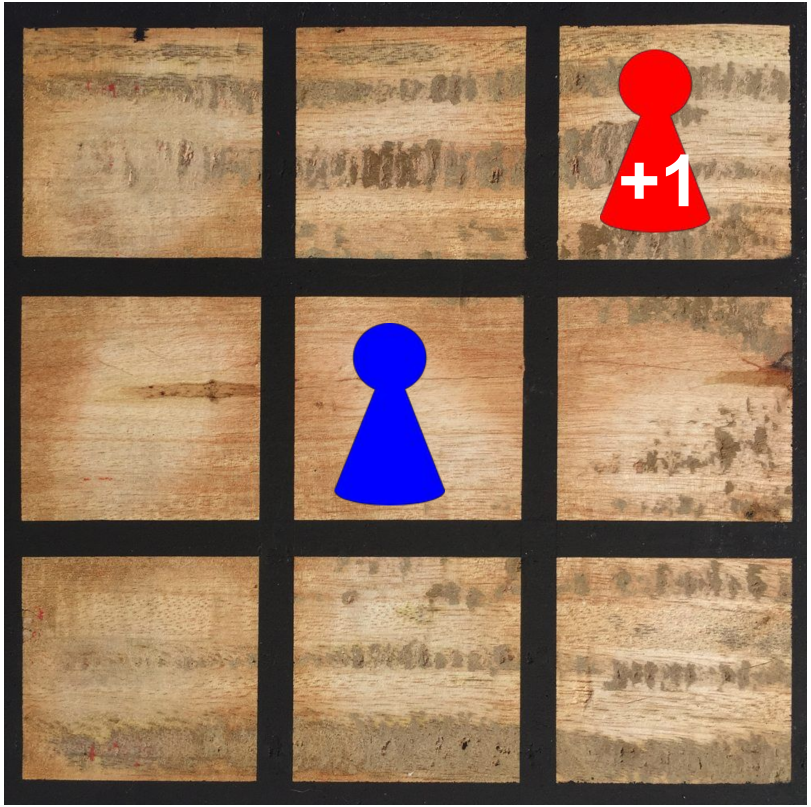
Tit for Tat



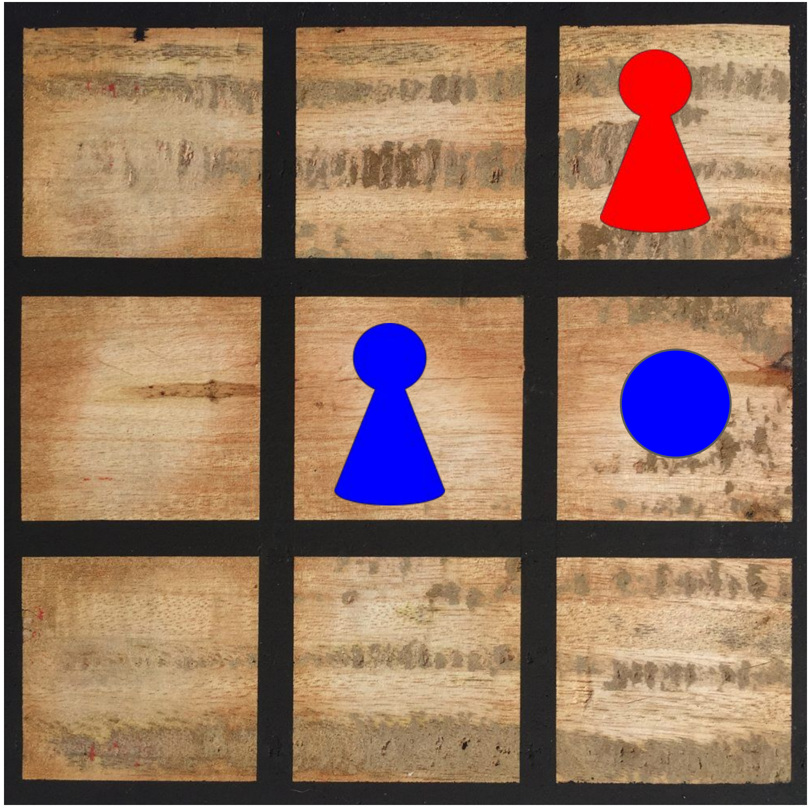
Tit for Tat



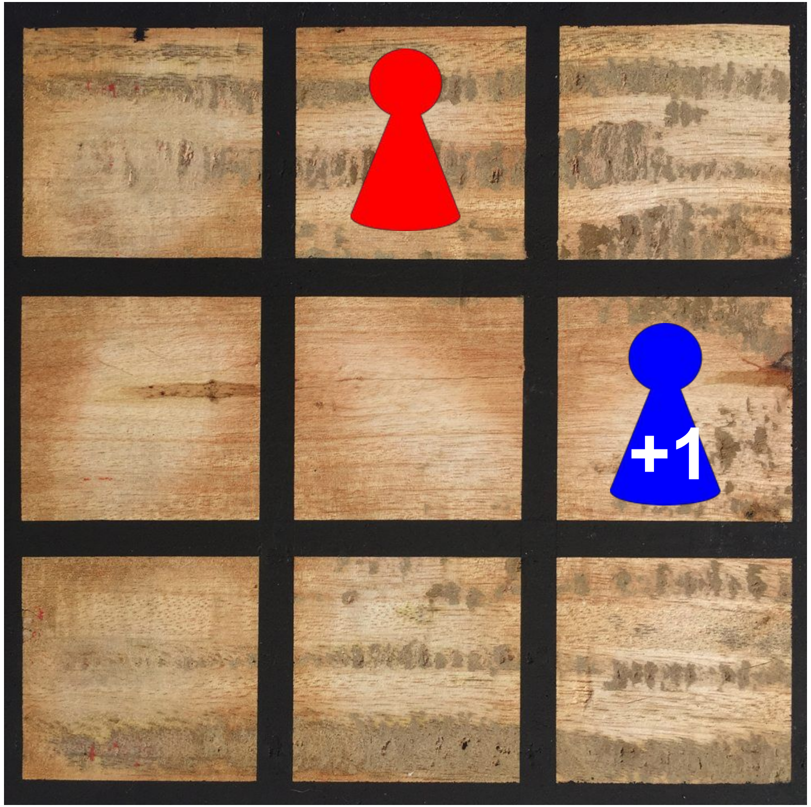
Tit for Tat



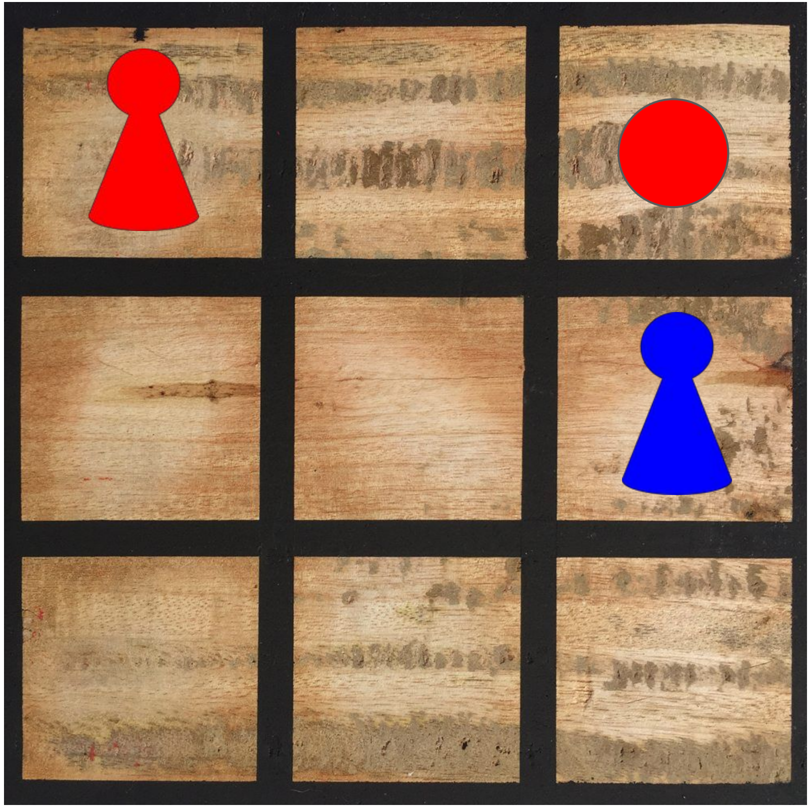
Tit for Tat



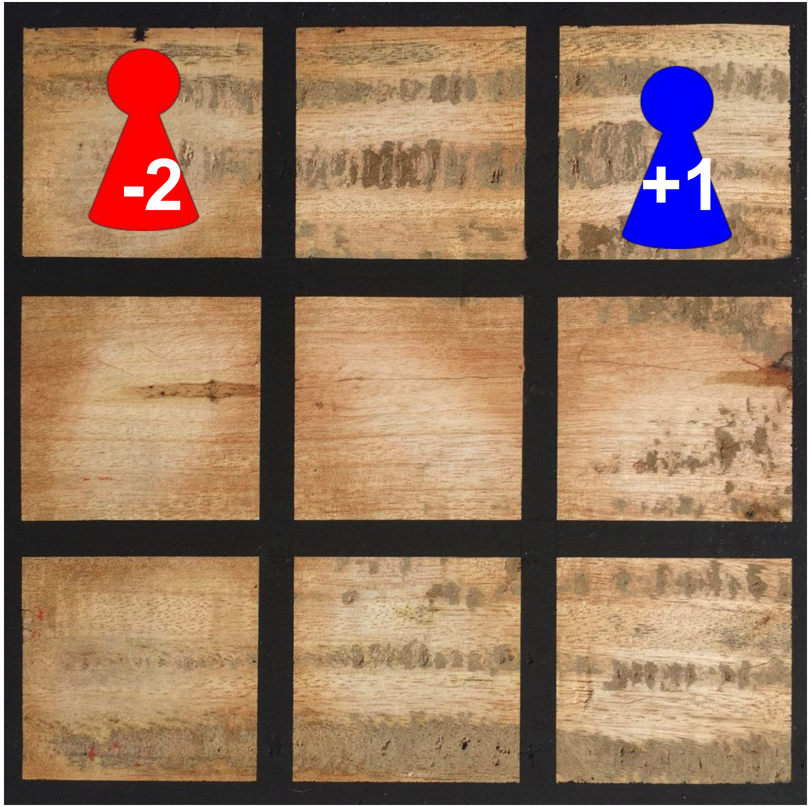
Tit for Tat



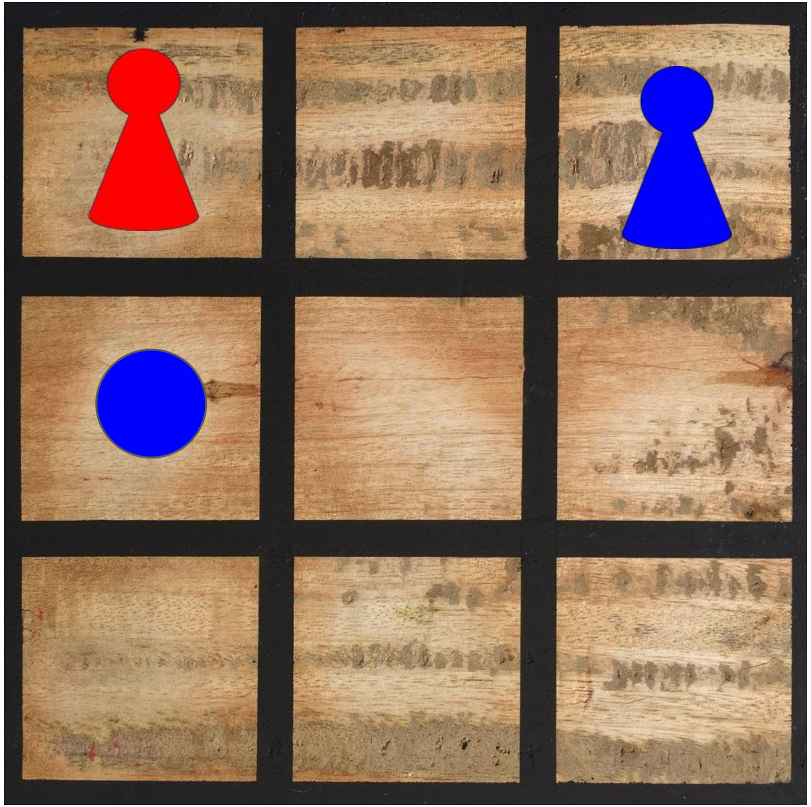
Tit for Tat



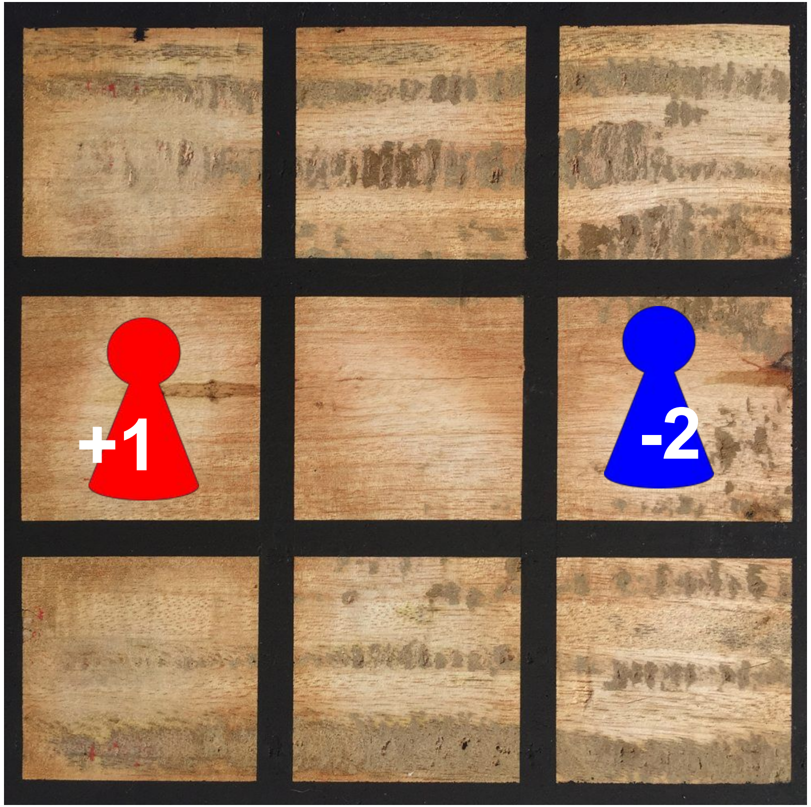
Tit for Tat



Tit for Tat



Tit for Tat



Naive RL fails on the coin game

Naïve learners without shared reward learn Always Defect, to always take the coin no matter the color. This is not cooperative.

Naïve learners with shared reward learn to Always Cooperate, to always not to take the opponent’s coin. This is exploitable.

Learning with Opponent Q-Learning Awareness (LOQA)

Key observation: The rewards that the agent observes are dependent on the policy that the opponent plays and vice-versa.

Suppose we observe a trajectory,

$$\tau := s_0,\ a_0,\ b_0,\ r_0^1,\ r_0^2,\ s_1, \ldots$$

The probability of τ is given by

$$\mathbb{P}(\tau) = \mu(s_0)\pi^1(a_0|s_0;\theta^1)\pi^2(b_0|s_0;\theta^2)P(s_1|s_0,a_0,b_0)\ldots$$

Initial distribution over states

Transition dynamics

Learning with Opponent Q-Learning Awareness (LOQA)

Key observation: The rewards that the agent observes are dependent on the policy that the opponent plays and vice-versa.

In reinforcement learning we aim to optimize the expected return of the agent given by:

$$V^1(\mu) = \mathbb{E}_\tau \left[R^1(\tau)\right] = \mathbb{E}_\tau \left[\sum_{t=1}^{\infty} \gamma^t r^1(s_t, a_t, b_t)\right]$$

Hence, we can differentiate the value and Q functions of the opponent w.r.t. the parameters of the agent (and vice-versa) using the reinforce estimator:

$$\nabla_{\theta_1} V^2(\mu) = \mathbb{E}_\tau \left[R^2(\tau) \sum_{t=1}^{\infty} \nabla_{\theta_1} \log \pi^1(a_t|s_t)\right]$$

Differentiable approximation of the opponent's policy

Q(b1|s) = 1.0

Q(b2|s) = 2.0

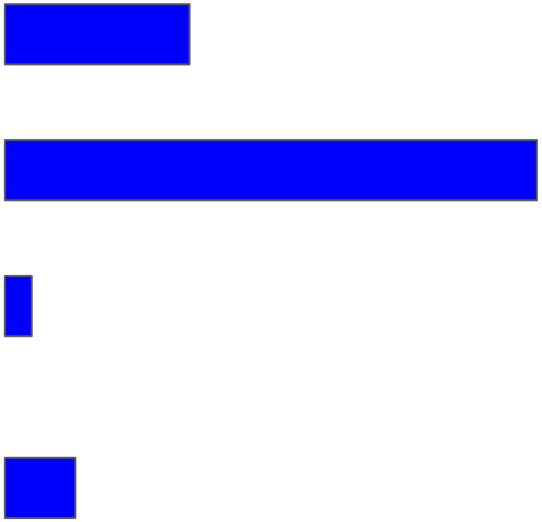
Q(b3|s) = -1.0

Q(b4|s) = 0.2

Intuition: The policy will be optimized to increase probability of b2

Differentiable approximation of the opponent’s policy

Approximated Optimized Opponent’s Policy



$Q(b1|s) = 1.0$

$Q(b2|s) = 2.0$

$Q(b3|s) = -1.0$

$Q(b4|s) = 0.2$

Differentiable approximation of the opponent's policy

Q(b1|s) = 1.0

Q(b2|s) = 2.0

Q(b3|s) = -1.0

Q(b4|s) = 0.2

We make an approximation to the optimized policy by a softmax over q-values

Differentiable approximation of the opponent's policy

$$\pi^2(b_t|s_t) \approx \frac{\exp(Q^2(s_t, b_t))}{\sum_{b'} \exp(Q^2(s_t, b'))}$$

Q(b1|s) = 1.0

Q(b2|s) = 2.0

Q(b3|s) = -1.0

Q(b4|s) = 0.2

We make an approximation to the optimized policy by a softmax over q-values

Differentiable approximation of the opponent's policy

$$\pi^2(b_t|s_t) \approx \frac{\exp(Q^2(s_t, b_t))}{\sum_{b'} \exp(Q^2(s_t, b'))}$$

Q(b1|s) = 1.0

Q(b2|s) = 2.0

This softmax is differentiable w.r.t agent parameters as action-values are differentiable.

Q(b3|s) = -1.0

Q(b4|s) = 0.2

Differentiable approximation of the opponent’s policy

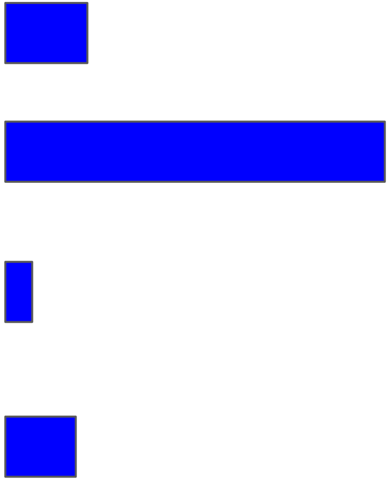
$$\pi^2(b_t|s_t) \approx \frac{\exp(Q^2(s_t, b_t))}{\sum_{b'} \exp(Q^2(s_t, b'))}$$

Q(b1|s) = 1.0

Q(b2|s) = 2.0

Q(b3|s) = -1.0

Q(b4|s) = 0.2



Differentiable approximation of the opponent's policy

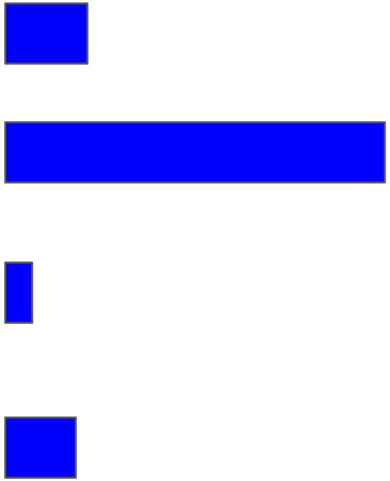
$$\pi^2(b_t|s_t) \approx \frac{\exp(Q^2(s_t, b_t))}{\sum_{b'} \exp(Q^2(s_t, b'))}$$

Q(b1|s) = 1.0

Q(b2|s) = 1.5

Q(b3|s) = -1.0

Q(b4|s) = 0.2



Differentiable approximation of the opponent's policy

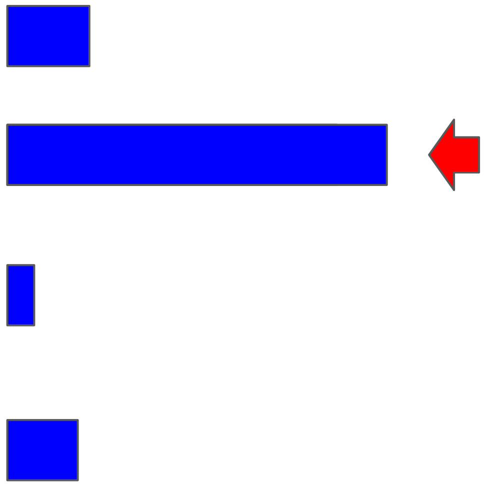
$$\pi^2(b_t|s_t) \approx \frac{\exp(Q^2(s_t, b_t))}{\sum_{b'} \exp(Q^2(s_t, b'))}$$

Q(b1|s) = 1.0

Q(b2|s) = 1.5

Q(b3|s) = -1.0

Q(b4|s) = 0.2



Differentiable approximation of the opponent’s policy

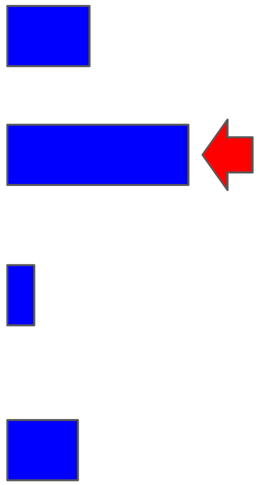
$$\pi^2(b_t|s_t) \approx \frac{\exp(Q^2(s_t, b_t))}{\sum_{b'} \exp(Q^2(s_t, b'))}$$

Q(b1|s) = 1.0

Q(b2|s) = 1.5

Q(b3|s) = -1.0

Q(b4|s) = 0.2



Approximating the opponent's policy

Key assumption: The policy of the opponent can be approximated as a softmax over the Q-Values.

We first approximate the Q value with Monte Carlo rollouts:

$$\hat{Q}^2(s_t, b_t) := \mathbb{E}_{\tau \sim \mathrm{mc}}\left[R^2(\tau) | s = s_t, b = b_t\right]$$

Then we approximate the opponent's policy:

Differentiable w.r.t θ^1

$$\hat{\pi}^2(b_t | s_t) := \frac{\underline{\exp(\hat{Q}^2(s_t, b_t))}}{\underline{\exp(\hat{Q}^2(s_t, b_t))} + \underline{\sum_{b' \neq b_t} \exp(Q^2(s_t, b'))}}$$

Non differentiable w.r.t θ^1

LOQA is an Actor Critic Algorithm

<u>LOQA's actor-critic update:</u> Given that we assume control over the opponent's policy, a new term emerges in LOQA's policy gradient.

$$\nabla_{\theta^1} V^1(\mu) = \mathbb{E}_\tau \left[\sum_{t=0}^{T} A^1(s_t, a_t, b_t) \left(\underbrace{\nabla_{\theta^1} \log \pi^1(a_t|s_t)}_{\mathbf{1}} + \underbrace{\nabla_{\theta^1} \log \hat{\pi}^2(b_t|s_t)}_{\mathbf{2}} \right) \right]$$

1. <u>Regular Advantage policy gradient</u>

2. <u>LOQA's opponent shaping component:</u> Induces the opponent to select actions that are beneficial to the agent.

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

  Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

  $L^1_Q \leftarrow 0, L^2_Q \leftarrow 0$

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

  Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

  $L^1_Q \leftarrow 0, L^2_Q \leftarrow 0$

  for $t = 1, 2, \ldots, T-1$ **do**

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

$L^1_Q \leftarrow 0, L^2_Q \leftarrow 0$

for $t = 1, 2, \ldots, T-1$ **do**

$L^1_Q \leftarrow L^1_Q +$ HUBER_LOSS$(r_t + \gamma Q^1_{\text{target}}(s_{t+1}, a_{t+1}) - Q^1(s_t, a_t))$

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

$L^1_Q \leftarrow 0, L^2_Q \leftarrow 0$

for $t = 1, 2, \ldots, T-1$ **do**

$L^1_Q \leftarrow L^1_Q +$ HUBER_LOSS$(r_t + \gamma Q^1_{\text{target}}(s_{t+1}, a_{t+1}) - Q^1(s_t, a_t))$

$L^2_Q \leftarrow L^2_Q +$ HUBER_LOSS$(r_t + \gamma Q^2_{\text{target}}(s_{t+1}, b_{t+1}) - Q^2(s_t, a_t))$

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

  Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

  $L^1_Q \leftarrow 0, L^2_Q \leftarrow 0$

  for $t = 1, 2, \ldots, T-1$ **do**

    $L^1_Q \leftarrow L^1_Q + \text{HUBER\_LOSS}(r_t + \gamma Q^1_{\text{target}}(s_{t+1}, a_{t+1}) - Q^1(s_t, a_t))$

    $L^2_Q \leftarrow L^2_Q + \text{HUBER\_LOSS}(r_t + \gamma Q^2_{\text{target}}(s_{t+1}, b_{t+1}) - Q^2(s_t, a_t))$

  end for

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

  Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

  $L^1_Q \leftarrow 0, L^2_Q \leftarrow 0$

  for $t = 1, 2, \ldots, T-1$ **do**

    $L^1_Q \leftarrow L^1_Q + \text{HUBER\_LOSS}(r_t + \gamma Q^1_{\text{target}}(s_{t+1}, a_{t+1}) - Q^1(s_t, a_t))$

    $L^2_Q \leftarrow L^2_Q + \text{HUBER\_LOSS}(r_t + \gamma Q^2_{\text{target}}(s_{t+1}, b_{t+1}) - Q^2(s_t, a_t))$

  end for

  Optimize L^1_Q w.r.t. ϕ^1 and L^2_Q w.r.t. ϕ^2 with optimizer of choice

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

  Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

  $L^1_Q \leftarrow 0$, $L^2_Q \leftarrow 0$

  for $t = 1, 2, \ldots, T-1$ **do**

    $L^1_Q \leftarrow L^1_Q +$ HUBER_LOSS$(r_t + \gamma Q^1_{\text{target}}(s_{t+1}, a_{t+1}) - Q^1(s_t, a_t))$

    $L^2_Q \leftarrow L^2_Q +$ HUBER_LOSS$(r_t + \gamma Q^2_{\text{target}}(s_{t+1}, b_{t+1}) - Q^2(s_t, a_t))$

  end for

  Optimize L^1_Q w.r.t. ϕ^1 and L^2_Q w.r.t. ϕ^2 with optimizer of choice

  Compute advantage estimates $\{A^1_1, \ldots, A^1_T\}$, $\{A^2_1, \ldots, A^2_T\}$

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

  Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

  $L^1_Q \leftarrow 0$, $L^2_Q \leftarrow 0$

  for $t = 1, 2, \ldots, T-1$ **do**

    $L^1_Q \leftarrow L^1_Q +$ HUBER_LOSS$(r_t + \gamma Q^1_{\text{target}}(s_{t+1}, a_{t+1}) - Q^1(s_t, a_t))$

    $L^2_Q \leftarrow L^2_Q +$ HUBER_LOSS$(r_t + \gamma Q^2_{\text{target}}(s_{t+1}, b_{t+1}) - Q^2(s_t, a_t))$

  end for

  Optimize L^1_Q w.r.t. ϕ^1 and L^2_Q w.r.t. ϕ^2 with optimizer of choice

  Compute advantage estimates $\{A^1_1, \ldots, A^1_T\}$, $\{A^2_1, \ldots, A^2_T\}$

  $L^1_a \leftarrow$ LOQA_ACTOR_LOSS$(\tau, \pi^1, \gamma, \{A^1_1, \ldots, A^1_T\})$

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

$L_Q^1 \leftarrow 0$, $L_Q^2 \leftarrow 0$

for $t = 1, 2, \ldots, T-1$ **do**

$L_Q^1 \leftarrow L_Q^1 + \text{HUBER\_LOSS}(r_t + \gamma Q^1_{\text{target}}(s_{t+1}, a_{t+1}) - Q^1(s_t, a_t))$

$L_Q^2 \leftarrow L_Q^2 + \text{HUBER\_LOSS}(r_t + \gamma Q^2_{\text{target}}(s_{t+1}, b_{t+1}) - Q^2(s_t, a_t))$

end for

Optimize L_Q^1 w.r.t. ϕ^1 and L_Q^2 w.r.t. ϕ^2 with optimizer of choice

Compute advantage estimates $\{A_1^1, \ldots, A_T^1\}$, $\{A_1^2, \ldots, A_T^2\}$

$L_a^1 \leftarrow \text{LOQA\_ACTOR\_LOSS}(\tau, \pi^1, \gamma, \{A_1^1, \ldots, A_T^1\})$

$L_a^2 \leftarrow \text{LOQA\_ACTOR\_LOSS}(\tau, \pi^2, \gamma, \{A_1^2, \ldots, A_T^2\})$

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

  Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

  $L^1_Q \leftarrow 0$, $L^2_Q \leftarrow 0$

  for $t = 1, 2, \ldots, T-1$ **do**

    $L^1_Q \leftarrow L^1_Q + \text{HUBER\_LOSS}(r_t + \gamma Q^1_{\text{target}}(s_{t+1}, a_{t+1}) - Q^1(s_t, a_t))$

    $L^2_Q \leftarrow L^2_Q + \text{HUBER\_LOSS}(r_t + \gamma Q^2_{\text{target}}(s_{t+1}, b_{t+1}) - Q^2(s_t, a_t))$

  end for

  Optimize L^1_Q w.r.t. ϕ^1 and L^2_Q w.r.t. ϕ^2 with optimizer of choice

  Compute advantage estimates $\{A^1_1, \ldots, A^1_T\}$, $\{A^2_1, \ldots, A^2_T\}$

  $L^1_a \leftarrow \text{LOQA\_ACTOR\_LOSS}(\tau, \pi^1, \gamma, \{A^1_1, \ldots, A^1_T\})$

  $L^2_a \leftarrow \text{LOQA\_ACTOR\_LOSS}(\tau, \pi^2, \gamma, \{A^2_1, \ldots, A^2_T\})$

  Optimize L^1_a w.r.t. θ^1 and L^2_a w.r.t. θ^2 with optimizer of choice

Actual steps of the LOQA algorithm step by step

Algorithm 1 LOQA

Initialize: Discount factor γ, agent action-value parameters ϕ^1, target action-value parameters ϕ^1_{target}, actor parameters θ^1, opponent action-value parameters ϕ^2, target action-value parameters ϕ^2_{target}, actor parameters θ^2

for iteration$= 1, 2, \ldots$ **do**

  Run policies π^1 and π^2 for T timesteps in environment and collect trajectory τ

  $L^1_Q \leftarrow 0$, $L^2_Q \leftarrow 0$

  for $t = 1, 2, \ldots, T-1$ **do**

    $L^1_Q \leftarrow L^1_Q + \text{HUBER\_LOSS}(r_t + \gamma Q^1_{\text{target}}(s_{t+1}, a_{t+1}) - Q^1(s_t, a_t))$

    $L^2_Q \leftarrow L^2_Q + \text{HUBER\_LOSS}(r_t + \gamma Q^2_{\text{target}}(s_{t+1}, b_{t+1}) - Q^2(s_t, a_t))$

  end for

  Optimize L^1_Q w.r.t. ϕ^1 and L^2_Q w.r.t. ϕ^2 with optimizer of choice

  Compute advantage estimates $\{A^1_1, \ldots, A^1_T\}$, $\{A^2_1, \ldots, A^2_T\}$

  $L^1_a \leftarrow \text{LOQA\_ACTOR\_LOSS}(\tau, \pi^1, \gamma, \{A^1_1, \ldots, A^1_T\})$

  $L^2_a \leftarrow \text{LOQA\_ACTOR\_LOSS}(\tau, \pi^2, \gamma, \{A^2_1, \ldots, A^2_T\})$

  Optimize L^1_a w.r.t. θ^1 and L^2_a w.r.t. θ^2 with optimizer of choice

end for

Actual steps of the LOQA algorithm step by step

Algorithm 2 LOQA_ACTOR_LOSS

Input: Trajectory τ of length T, actor policy π^i, $\boxed{\text{opponent action-value function } Q^{-i}}$, discount factor γ, advantages $\{A_1^i, \dots, A_T^i\}$

Actual steps of the LOQA algorithm step by step

Algorithm 2 LOQA_ACTOR_LOSS

Input: Trajectory τ of length T, actor policy π^i, opponent action-value function Q^{-i}, discount factor γ, advantages $\{A_1^i, \ldots, A_T^i\}$

$L_a \leftarrow 0$

Actual steps of the LOQA algorithm step by step

Algorithm 2 LOQA_ACTOR_LOSS

Input: Trajectory τ of length T, actor policy π^i, opponent action-value function Q^{-i}, discount factor γ, advantages $\{A_1^i, \ldots, A_T^i\}$

$L_a \leftarrow 0$

for $t = 1, 2, \ldots, T-1$ **do**

Actual steps of the LOQA algorithm step by step

Algorithm 2 LOQA_ACTOR_LOSS

Input: Trajectory τ of length T, actor policy π^i, opponent action-value function Q^{-i}, discount factor γ, advantages $\{A_1^i, \ldots, A_T^i\}$

$L_a \leftarrow 0$

for $t = 1, 2, \ldots, T-1$ **do**

$\hat{Q}^{-i}(s_t, b_t) \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r^{-i}(s_k, a_k, b_k)$ ▷ r^{-i} made differentiable using DiCE

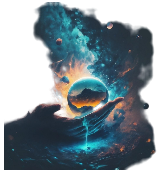
Actual steps of the LOQA algorithm step by step

Algorithm 2 LOQA_ACTOR_LOSS

Input: Trajectory τ of length T, actor policy π^i, opponent action-value function Q^{-i}, discount factor γ, advantages $\{A_1^i, \ldots, A_T^i\}$

$L_a \leftarrow 0$

for $t = 1, 2, \ldots, T-1$ **do**

$\hat{Q}^{-i}(s_t, b_t) \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r^{-i}(s_k, a_k, b_k)$  $\triangleright$ r^{-i} made differentiable using DiCE

Actual steps of the LOQA algorithm step by step

Algorithm 2 LOQA_ACTOR_LOSS

Input: Trajectory τ of length T, actor policy π^i, opponent action-value function Q^{-i}, discount factor γ, advantages $\{A_1^i, \ldots, A_T^i\}$

$L_a \leftarrow 0$

for $t = 1, 2, \ldots, T-1$ **do**

  $\hat{Q}^{-i}(s_t, b_t) \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r^{-i}(s_k, a_k, b_k)$ ▷ r^{-i} made differentiable using DiCE

  Compute $\hat{\pi}^{-i}$ using $\hat{Q}^{-i}(s_t, b_t)$ and $Q^{-i}(s_t, b_t)$ according to equation (2)

Actual steps of the LOQA algorithm step by step

Algorithm 2 LOQA_ACTOR_LOSS

Input: Trajectory τ of length T, actor policy π^i, opponent action-value function Q^{-i}, discount factor γ, advantages $\{A_1^i, \ldots, A_T^i\}$

$L_a \leftarrow 0$

for $t = 1, 2, \ldots, T-1$ **do**

$\hat{Q}^{-i}(s_t, b_t) \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r^{-i}(s_k, a_k, b_k)$ $\triangleright$ r^{-i} made differentiable using DiCE

Compute $\hat{\pi}^{-i}$ using $\hat{Q}^{-i}(s_t, b_t)$ and $Q^{-i}(s_t, b_t)$ according to equation (2)

$L_a \leftarrow L_a + A_t^i \left[\log \pi^i(a_t|s_t) + \boxed{\log \hat{\pi}^{-i}(b_t|s_t)}\right]$

Actual steps of the LOQA algorithm step by step

Algorithm 2 LOQA_ACTOR_LOSS

Input: Trajectory τ of length T, actor policy π^i, opponent action-value function Q^{-i}, discount factor γ, advantages $\{A_1^i, \dots, A_T^i\}$

$L_a \leftarrow 0$

for $t = 1, 2, \dots, T-1$ **do**

  $\hat{Q}^{-i}(s_t, b_t) \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r^{-i}(s_k, a_k, b_k)$ $\triangleright$ r^{-i} made differentiable using DiCE

  Compute $\hat{\pi}^{-i}$ using $\hat{Q}^{-i}(s_t, b_t)$ and $Q^{-i}(s_t, b_t)$ according to equation (2)

  $L_a \leftarrow L_a + A_t^i \left[\log \pi^i(a_t|s_t) + \log \hat{\pi}^{-i}(b_t|s_t)\right]$

end for

Actual steps of the LOQA algorithm step by step

Algorithm 2 LOQA_ACTOR_LOSS

Input: Trajectory τ of length T, actor policy π^i, opponent action-value function Q^{-i}, discount factor γ, advantages $\{A_1^i, \ldots, A_T^i\}$

$L_a \leftarrow 0$

for $t = 1, 2, \ldots, T-1$ **do**

  $\hat{Q}^{-i}(s_t, b_t) \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r^{-i}(s_k, a_k, b_k)$ $\triangleright$ r^{-i} made differentiable using DiCE

  Compute $\hat{\pi}^{-i}$ using $\hat{Q}^{-i}(s_t, b_t)$ and $Q^{-i}(s_t, b_t)$ according to equation (2)

  $L_a \leftarrow L_a + A_t^i \left[\log \pi^i(a_t|s_t) + \log \hat{\pi}^{-i}(b_t|s_t)\right]$

end for

return: L_a

Actual steps of the LOQA algorithm step by step

Algorithm 3 LOQA_ACTOR_LOSS with loaded-DiCE

Input: Trajectory τ of length T, actor policy π^i, opponent action-value function Q^{-i}, discount factor γ, action discount factor λ, advantages $\{A_1^i, \dots, A_T^i\}$

$L_a \leftarrow 0$

for $t = 1, 2, \dots, T-1$ **do**

$\hat{Q}^{-i}(s_t, b_t) \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r^{-i}(s_k, a_k, b_k)$

Compute $\hat{\pi}^{-i}$ using $\hat{Q}^{-i}(s_t, b_t)$ and $Q^{-i}(s_t, b_t)$ according to equation (2)

$L_a \leftarrow L_a + A_t^i \left[\log \pi^i(a_t|s_t) + \log \hat{\pi}^{-i}(b_t|s_t)\right]$

end for

return: L_Q

end function

For curious minds, and anyone who actually wants to implement LOQA

Actual steps of the LOQA algorithm step by step

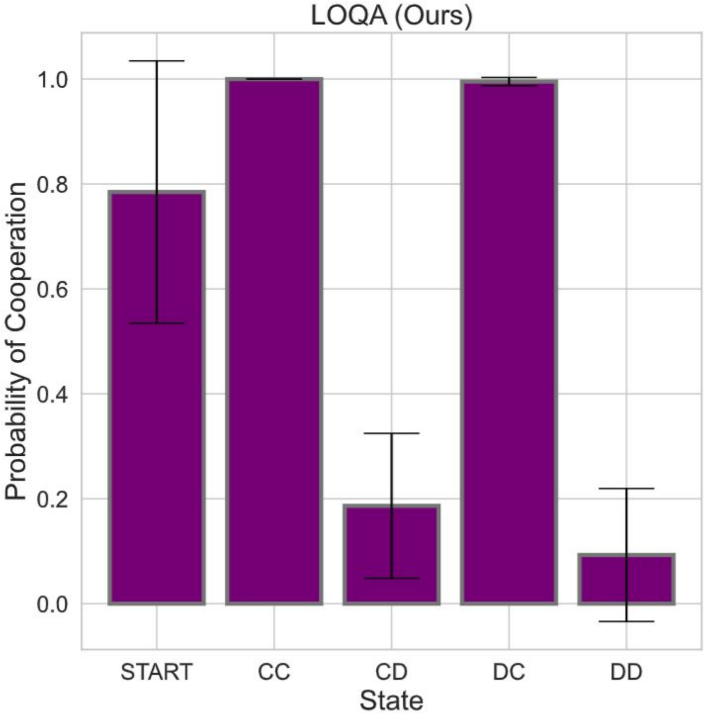
Algorithm 3 LOQA_ACTOR_LOSS with loaded-DiCE

Input: Trajectory τ of length T, actor policy π^i, opponent action-value function Q^{-i}, discount factor γ, action discount factor λ, advantages $\{A_1^i, \dots, A_T^i\}$

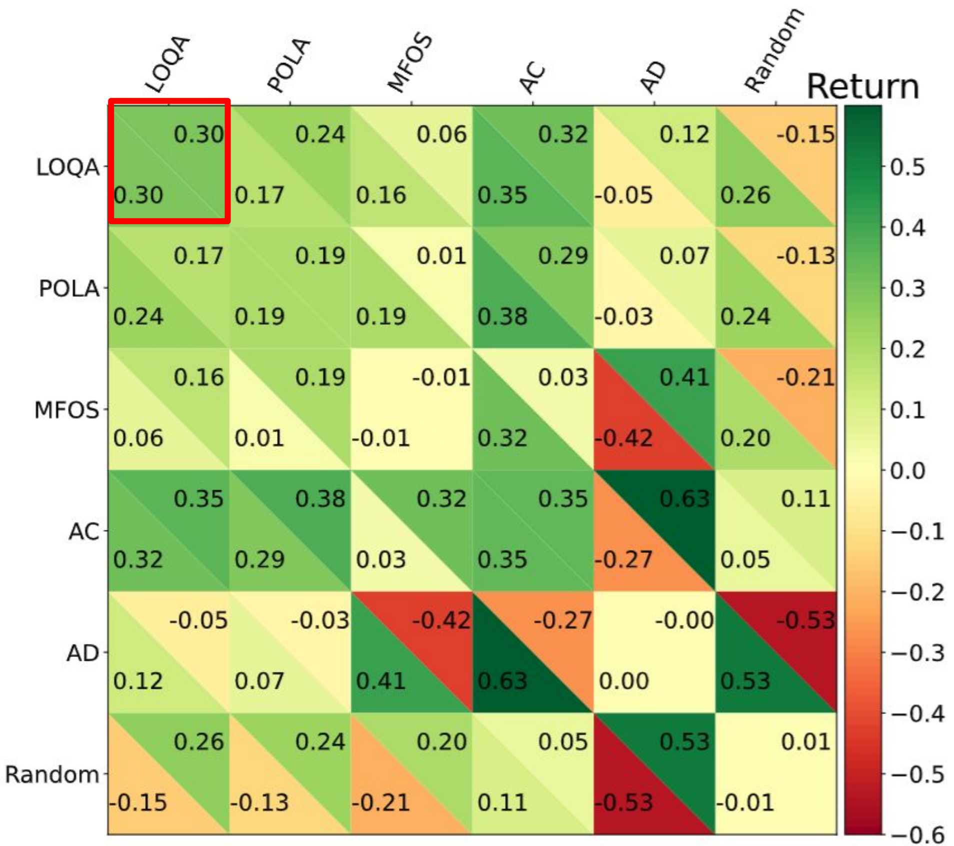
$L_a \leftarrow 0$
for $t = 1, 2, \dots, T-1$ **do**
  $w \leftarrow 0$
  $\hat{Q}^{-i}(s_t, b_t) \leftarrow \sum_{k=t}^{T} \gamma^{k-t} r^{-i}(s_k, a_k, b_k)$
  for $k = 2, \dots, T$ **do**
    $w \leftarrow \lambda \cdot w + \log\left(\pi^i(a_t|s_t)\right)$
    $v \leftarrow \lambda \cdot w - \log\left(\pi^i(a_t|s_t)\right)$
    $\hat{Q}^{-i}(s_t, b_t) \leftarrow \hat{Q}^{-i}(s_t, b_t) + A_k^i(f(w) - f(v))$
  end for
  Compute $\hat{\pi}^{-i}$ using $\hat{Q}^{-i}(s_t, b_t)$ and $Q^{-i}(s_t, b_t)$ according to equation (2)
  $L_a \leftarrow L_a + A_t^i \left[\log \pi^i(a_t|s_t) + \log \hat{\pi}^{-i}(b_t|s_t)\right]$
end for
return: L_Q

function $f(x)$
  return: $\exp(x - \text{STOP\_GRADIENT}(x))$
end function

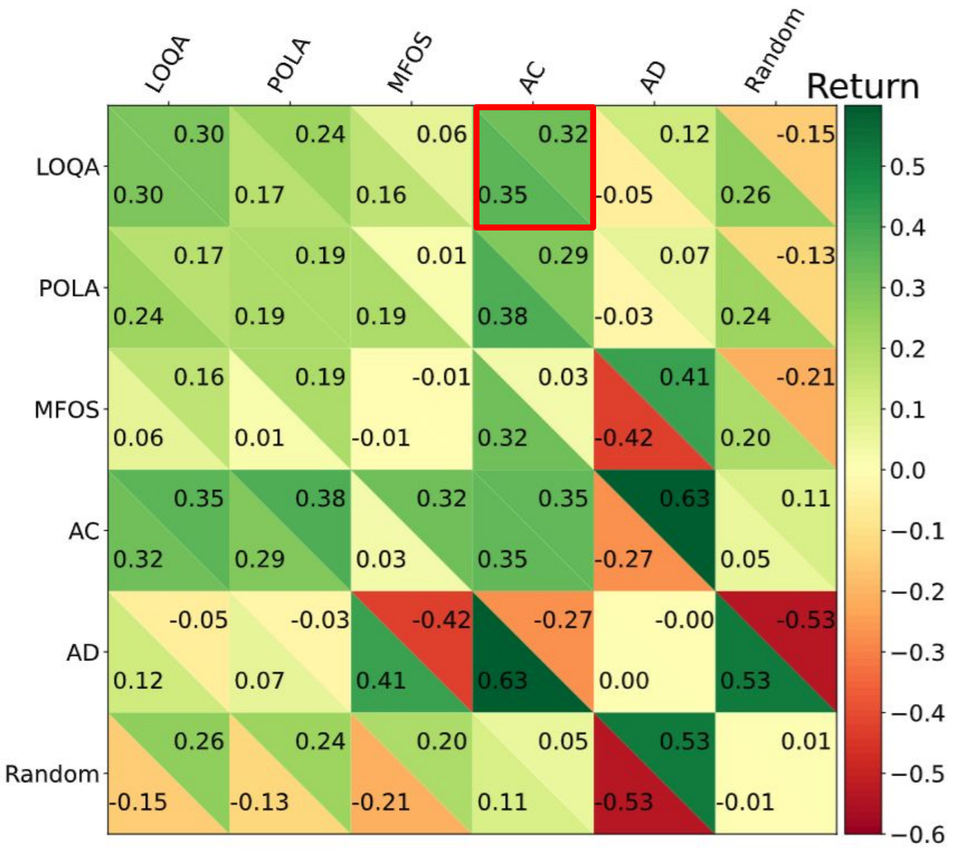
IPD experiments



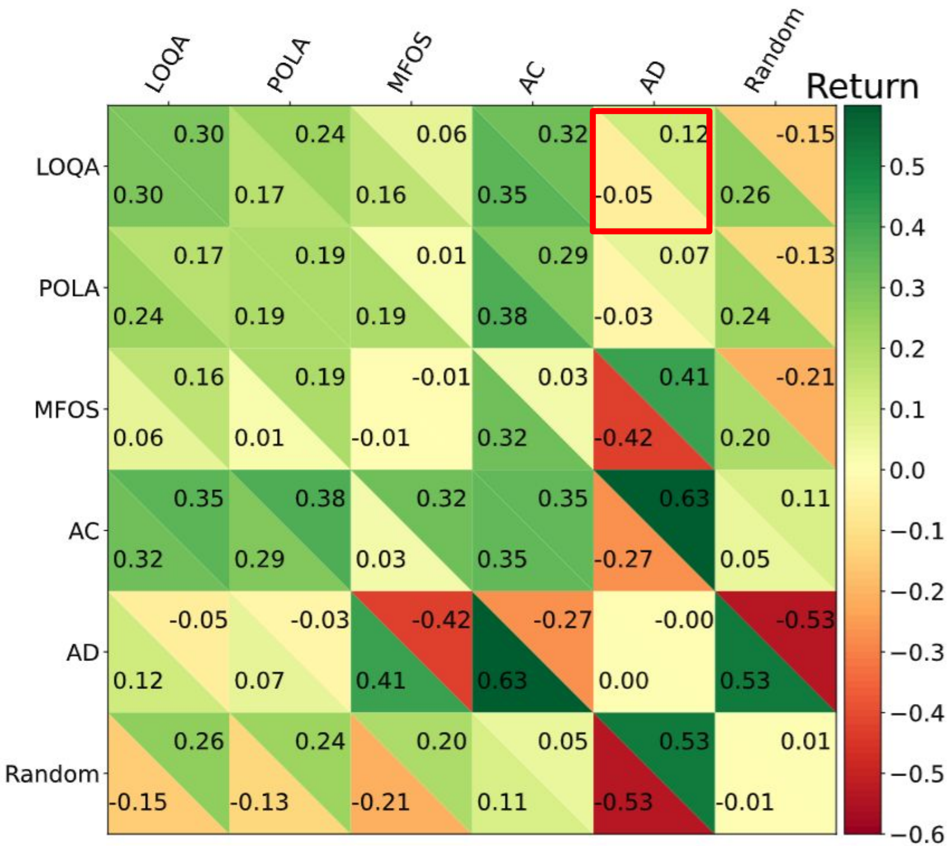
Coin Game League



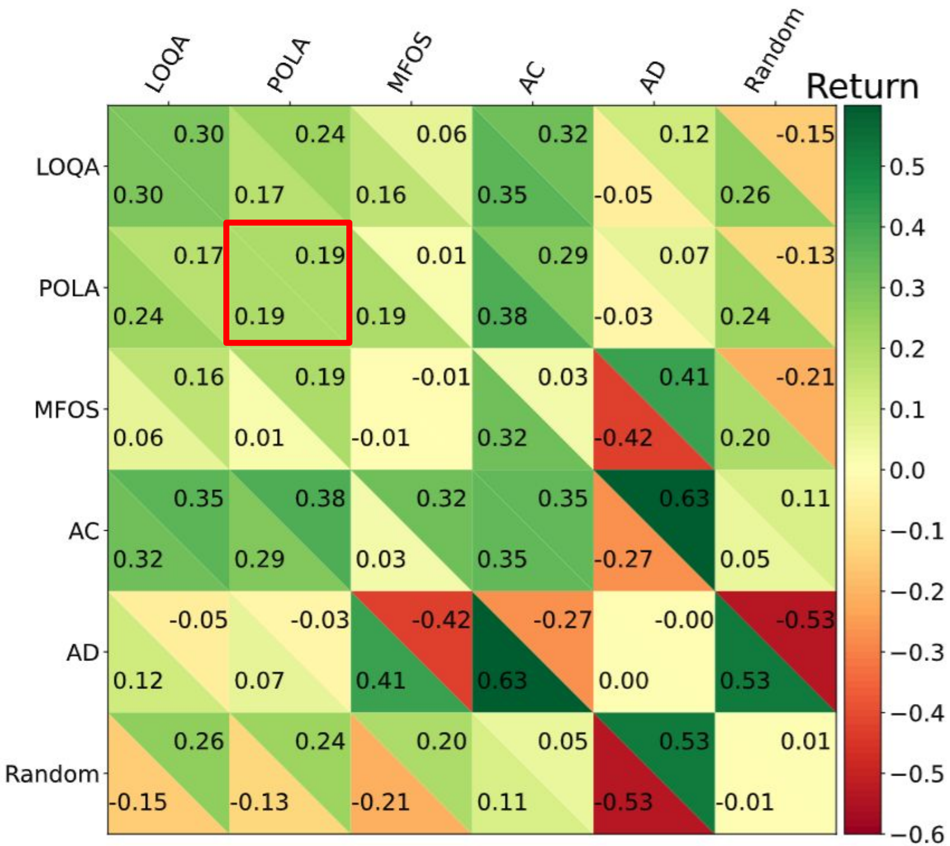
Coin Game League



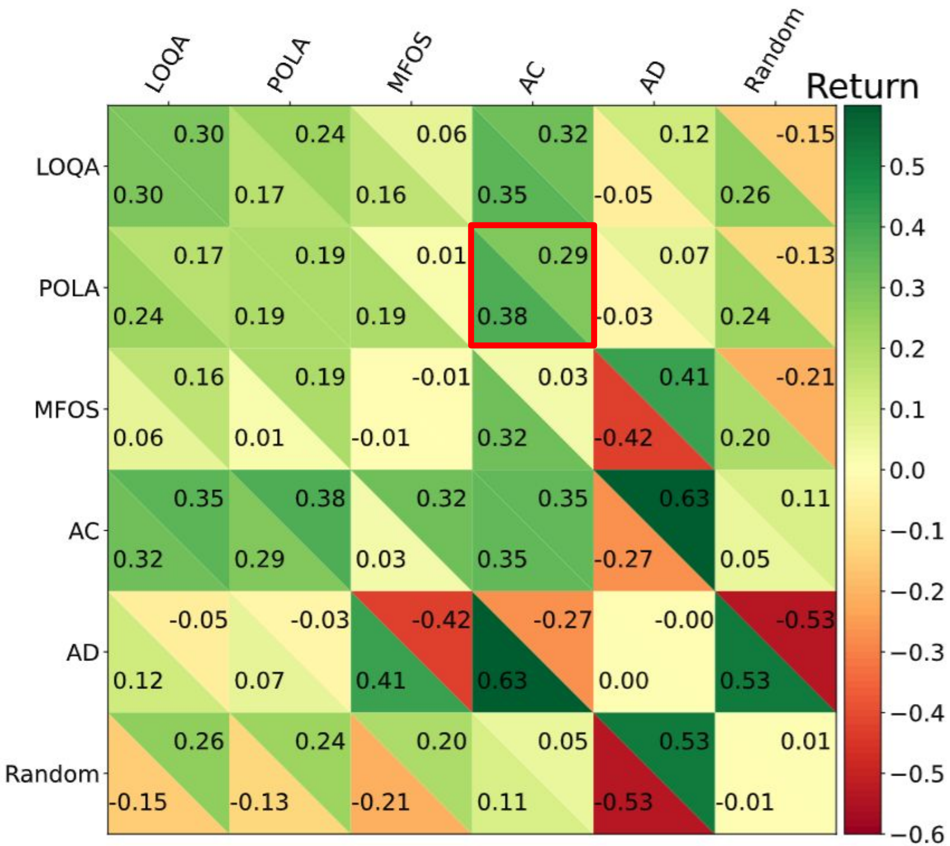
Coin Game League



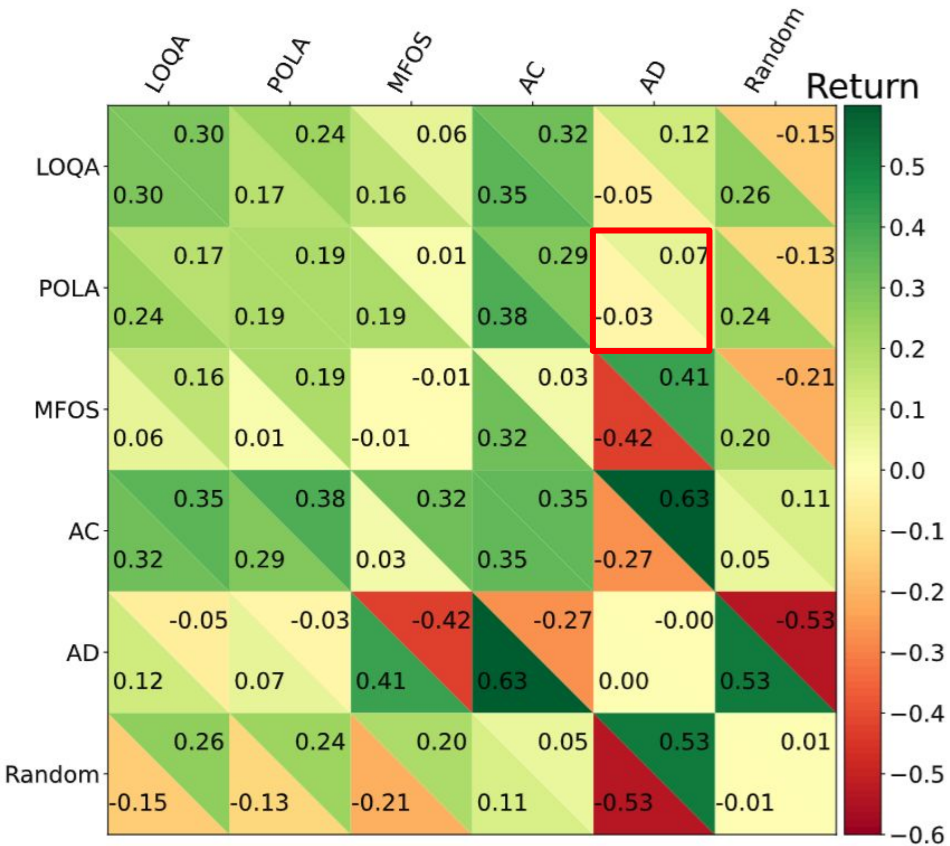
Coin Game League



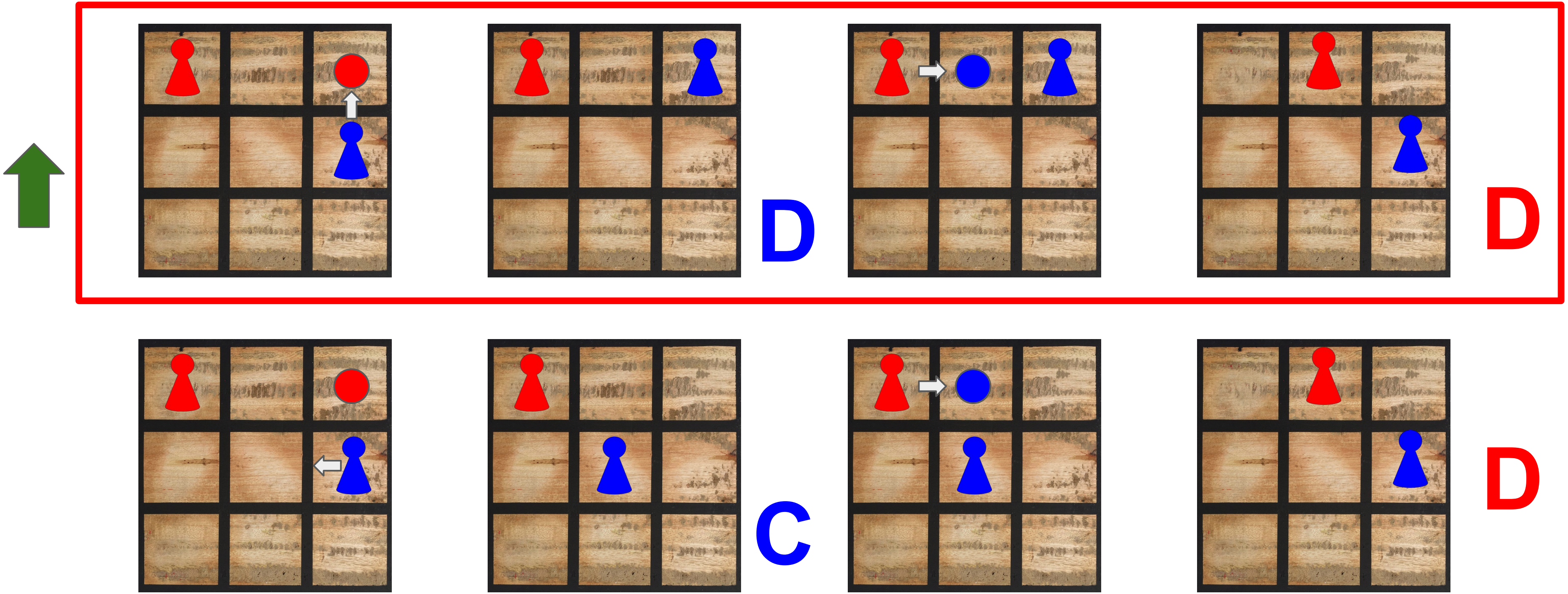
Coin Game League



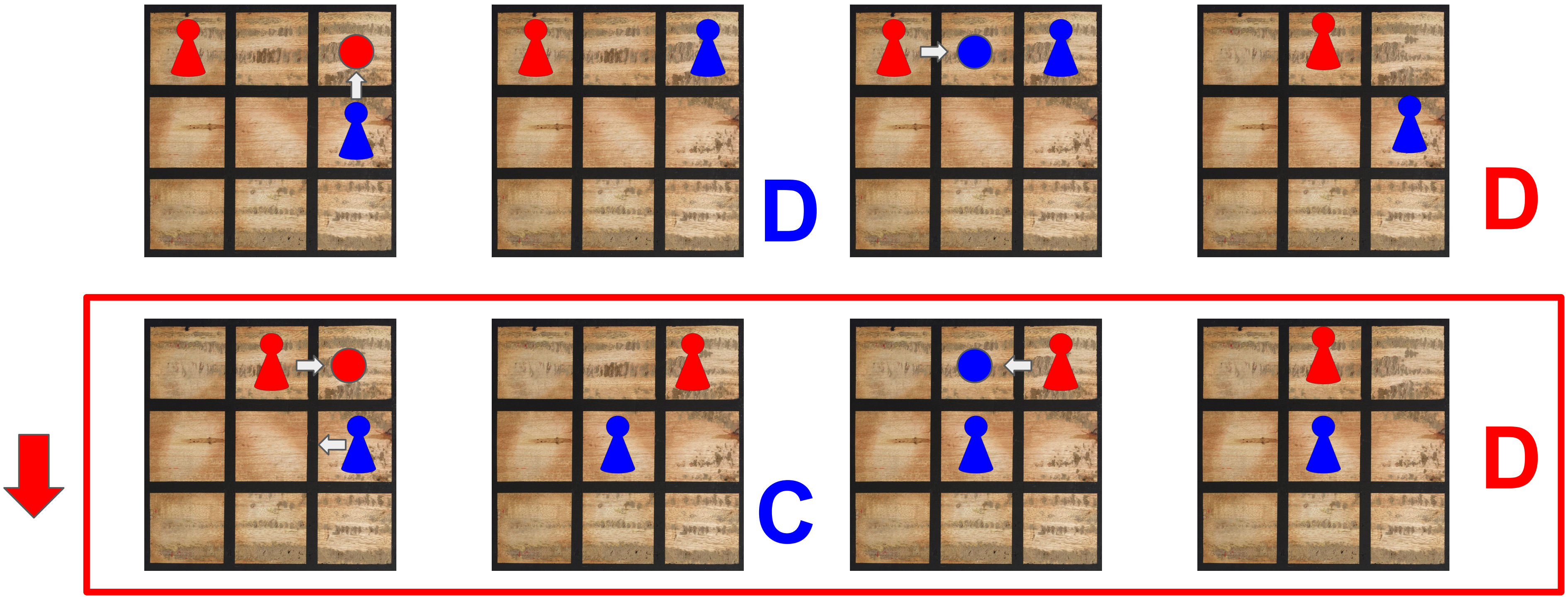
Coin Game League



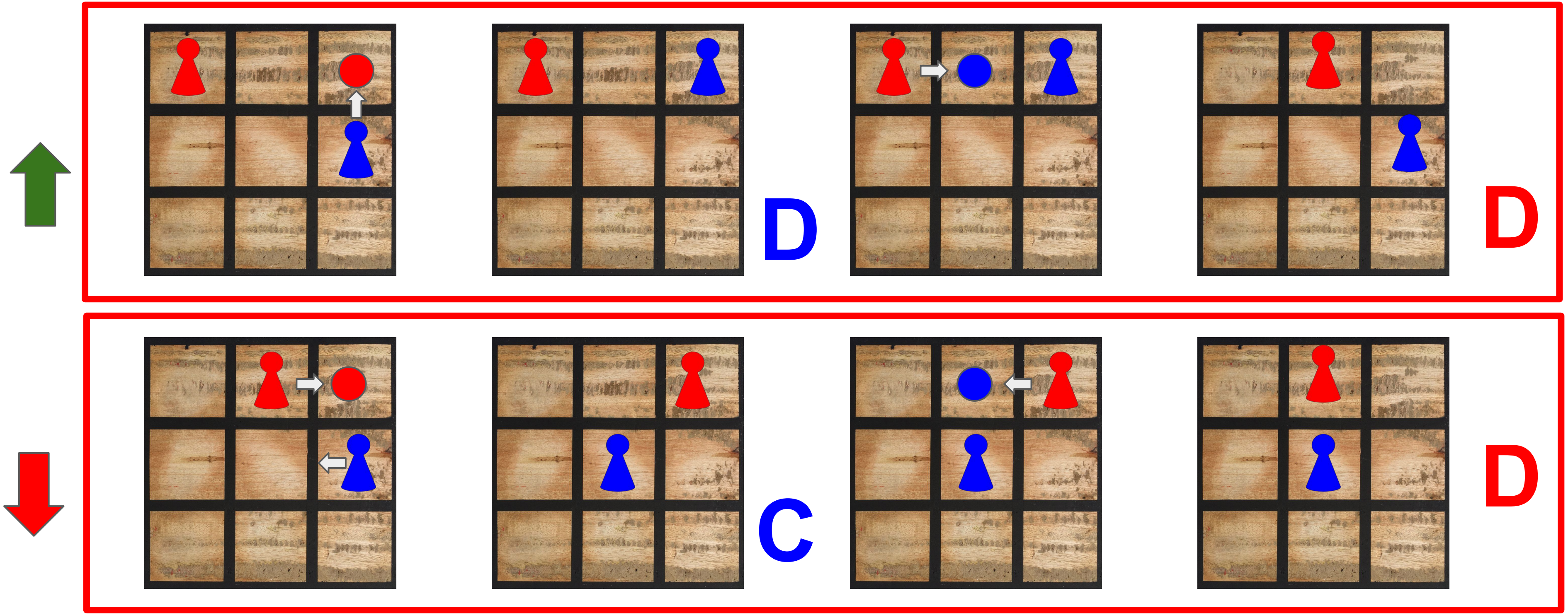
How does LOQA solve coin game intuitively?



How does LOQA solve coin game intuitively?



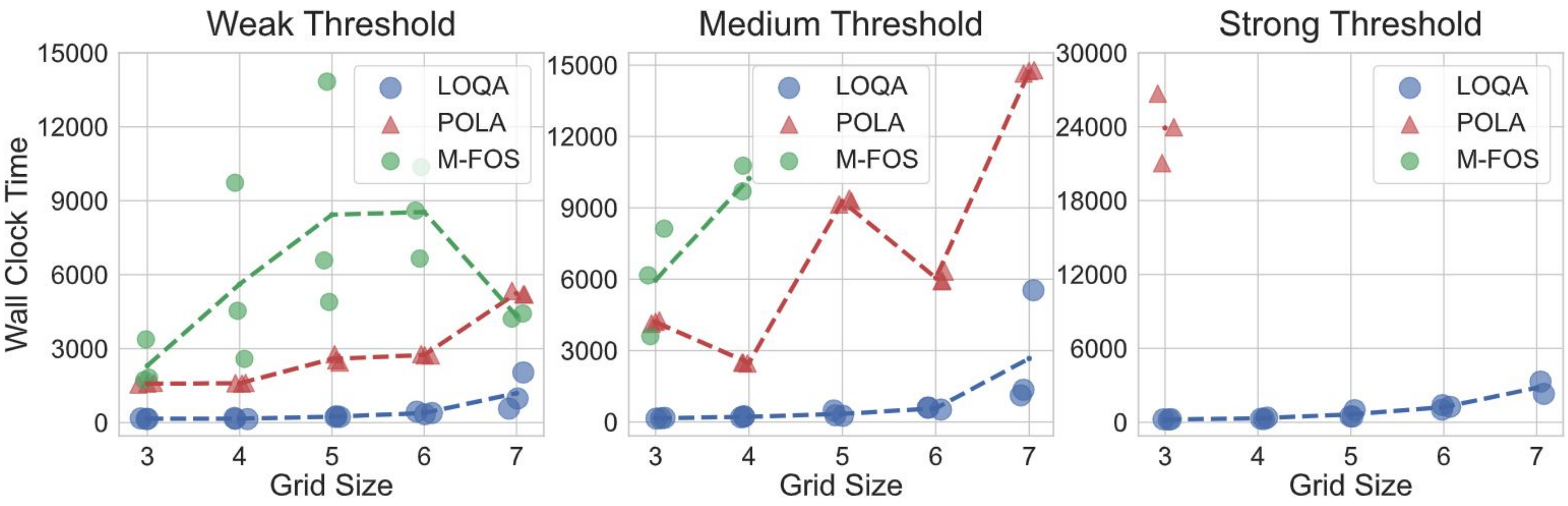
How does LOQA solve coin game intuitively?



How does LOQA solve coin game intuitively?

1. Increase probability of defecting after opponent defects.
2. Decrease probability of defecting after opponent cooperates.

LOQA's scalability (most important plot)




Back to first slide, Why the world is so much better with LOQA?
Reciprocity-based Cooperation

So, is the problem solved?

Not yet, there are problems:

So, is the problem solved?

Not yet, there are problems:

1. Number of players > 2

So, is the problem solved?

Not yet, there are problems:

1. Number of players > 2
2. Access to opponent’s q-value function is not granted, sometimes you don’t know the reward of the opponent

So, is the problem solved?

Not yet, there are problems:

1. Number of players > 2
2. Access to opponent’s q-value function is not granted, sometimes you don’t know the reward of the opponent
3. Continuous action values

So, is the problem solved?

Not yet, there are problems:

1. Number of players > 2
2. Access to opponent’s q-value function is not granted, sometimes you don’t know the reward of the opponent
3. Continuous action values
4. Scalability to more complex environments (MeltingPot, RICE-N, Pure-Diplomacy, etc)

Thank You!