Maximizing utility in multi-agent environments by anticipating the behavior of other learners

Angelos Assos (MIT) Yuval Dagan (Tel Aviv) Costis Daskalakis (MIT)

Learning in repeated games

Agents in strategic environments have to make sequential decisions over a time horizon

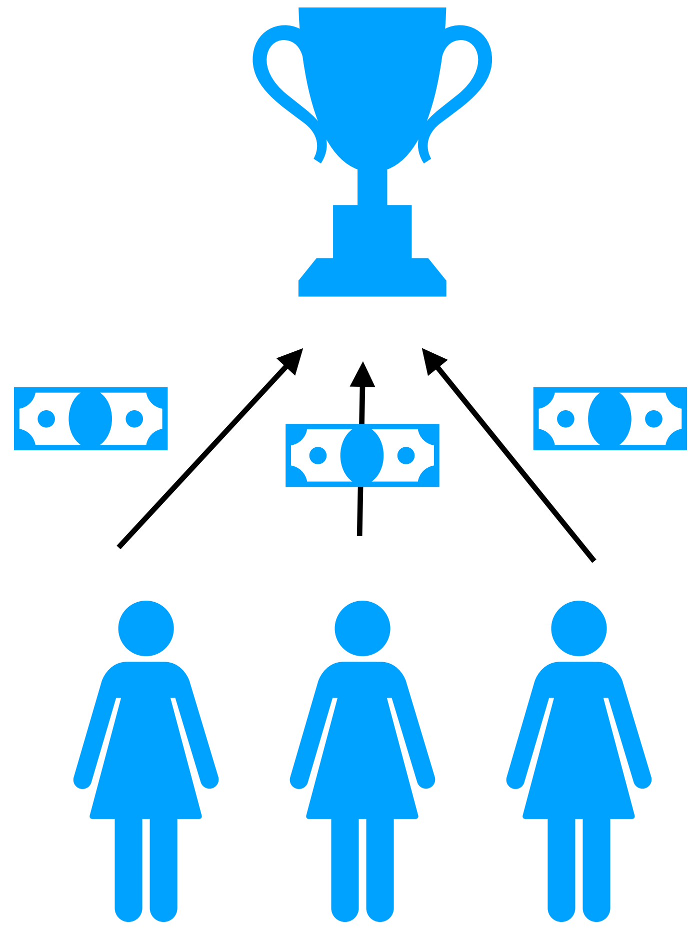
Learning in repeated games

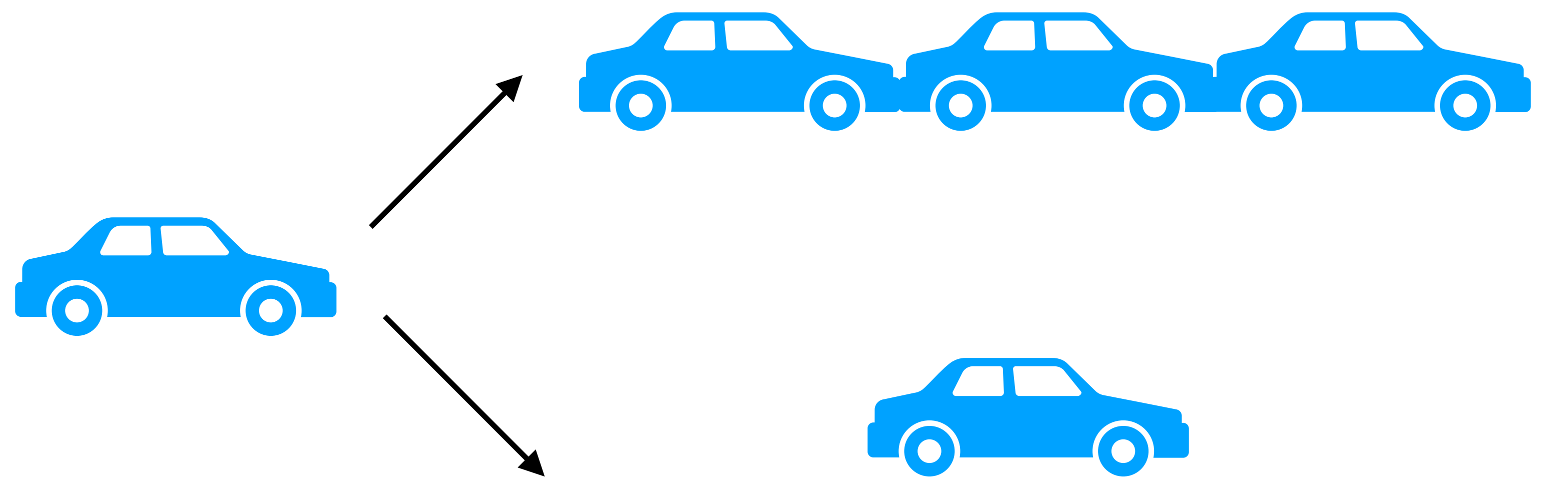
Agents in strategic environments have to make sequential decisions over a time horizon

Examples include

- Repeated Auctions
- Congestion games
- Network routing games

Etc.



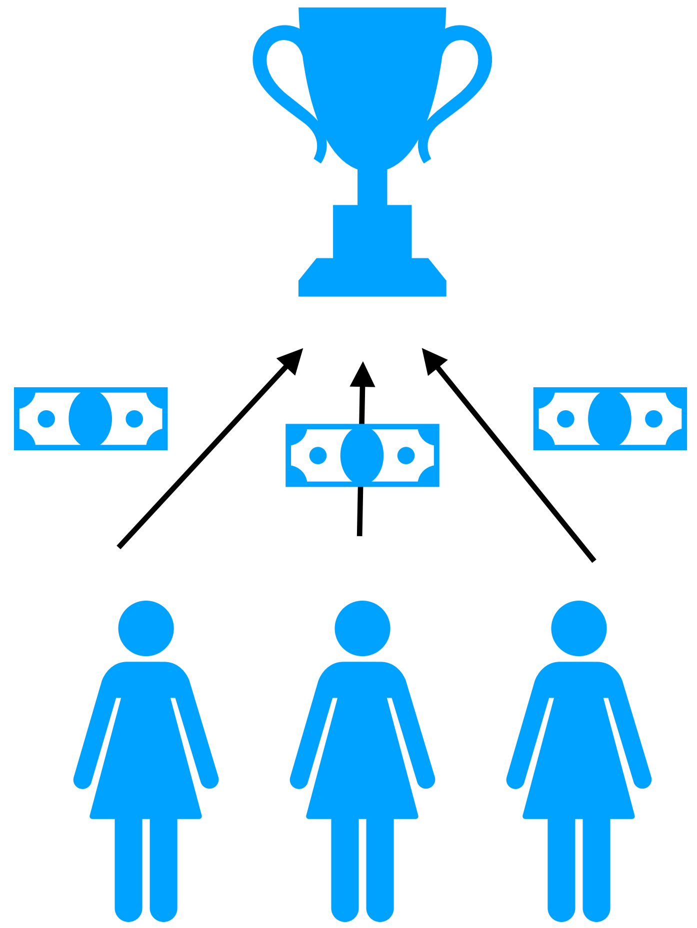


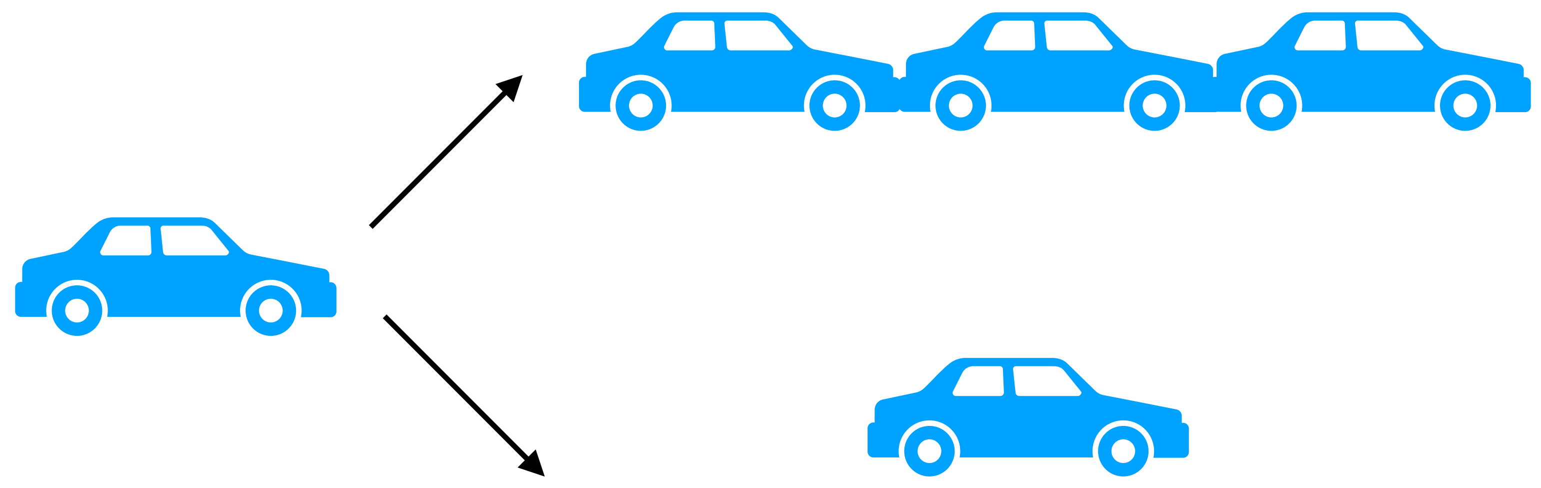
Learning in repeated games

Agents in strategic environments have to make sequential decisions over a time horizon

Examples include

- Repeated Auctions
- Congestion games
- Network routing games





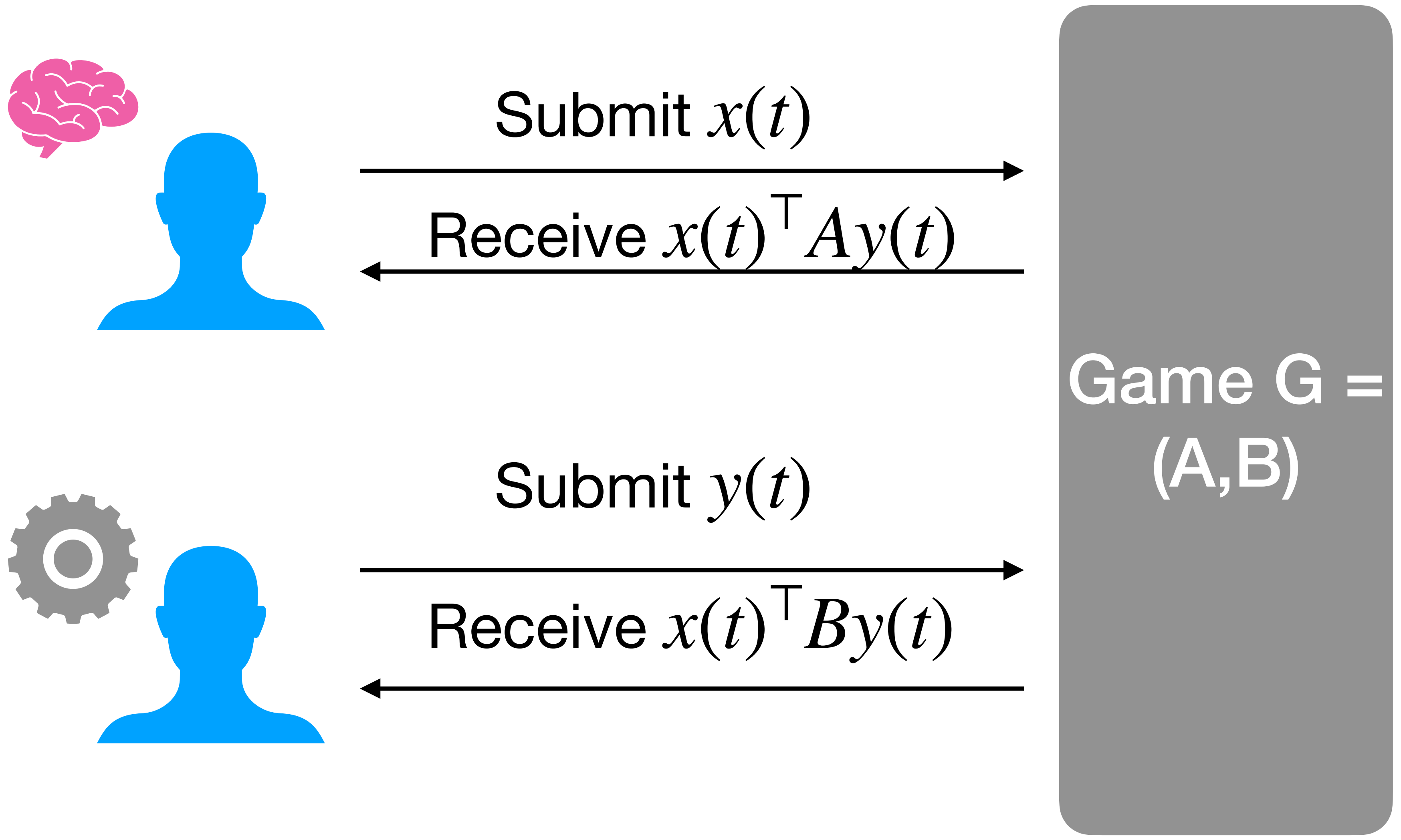
Etc.

A lot of times agents use famous learning algorithms to determine what action to take.

Motivating question: Can strategic agents take advantage of these algorithms?

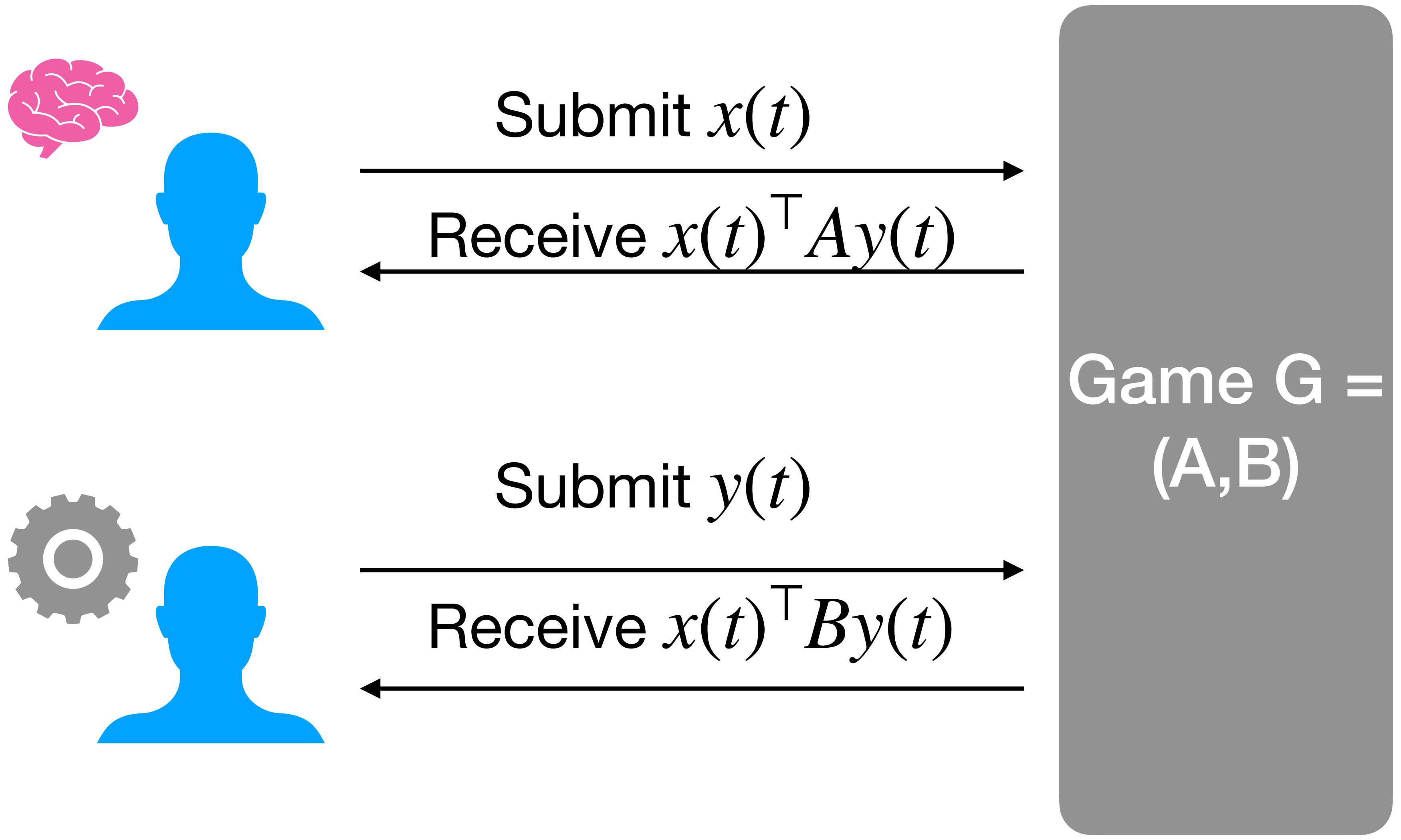
Setting

- Two player, repeated, normal form game played for time T.
- One player is called the *learner* and uses an algorithm throughout the game.
- Other player is called *optimizer,* knows the *learner's* algorithm and tries to take advantage of that to maximize their own utility.
- *Optimizer* and *learner* have n and m actions from action spaces $\mathscr{A}$ and $\mathscr{B}$ and utility matrices A, B respectively.



Setting

- Two player, repeated, normal form game played for time T.
- One player is called the *learner* and uses an algorithm throughout the game.
- Other player is called *optimizer,* knows the *learner's* algorithm and tries to take advantage of that to maximize their own utility.
- *Optimizer* and *learner* have n and m actions from action spaces $\mathscr{A}$ and $\mathscr{B}$ and utility matrices A, B respectively.



Questions we address

Against specific learning algorithms …

- In zero-sum games (where $A + B = 0$), what should the *optimizer* do to maximize their own utility?
- In general-sum games (where $A + B \neq 0$), is the best play for the *optimizer* efficiently computable?

Maximizing utility in zero-sum games

Continuous time setting

Optimizer strategy: any $x : [0,T] \to \Delta(\mathscr{A})$

Learner strategy: $y : [0,T] \to \Delta(\mathscr{B})$

, where:

$$y_i(t) = \frac{exp(\eta \int_0^t x(s)^\top Be_i ds)}{\sum_{i=1}^m exp(\eta \int_0^t x(s)^\top Be_i ds}$$

a.k.a. replicator dynamics, the continuous time analog
of MWU.

Maximizing utility in zero-sum games

Continuous time setting

Optimizer strategy: any $x : [0,T] \to \Delta(\mathscr{A})$

Learner strategy: $y : [0,T] \to \Delta(\mathscr{B})$

, where:

$$y_i(t) = \frac{exp(\eta \int_0^t x(s)^\top Be_i ds)}{\sum_{i=1}^m exp(\eta \int_0^t x(s)^\top Be_i ds}$$

a.k.a. replicator dynamics, the continuous time analog of MWU.

Theorem 1: The rewards of the optimizer depend only on the total time played each action.

Maximizing utility in zero-sum games

Continuous time setting

Optimizer strategy: any $x : [0,T] \to \Delta(\mathscr{A})$

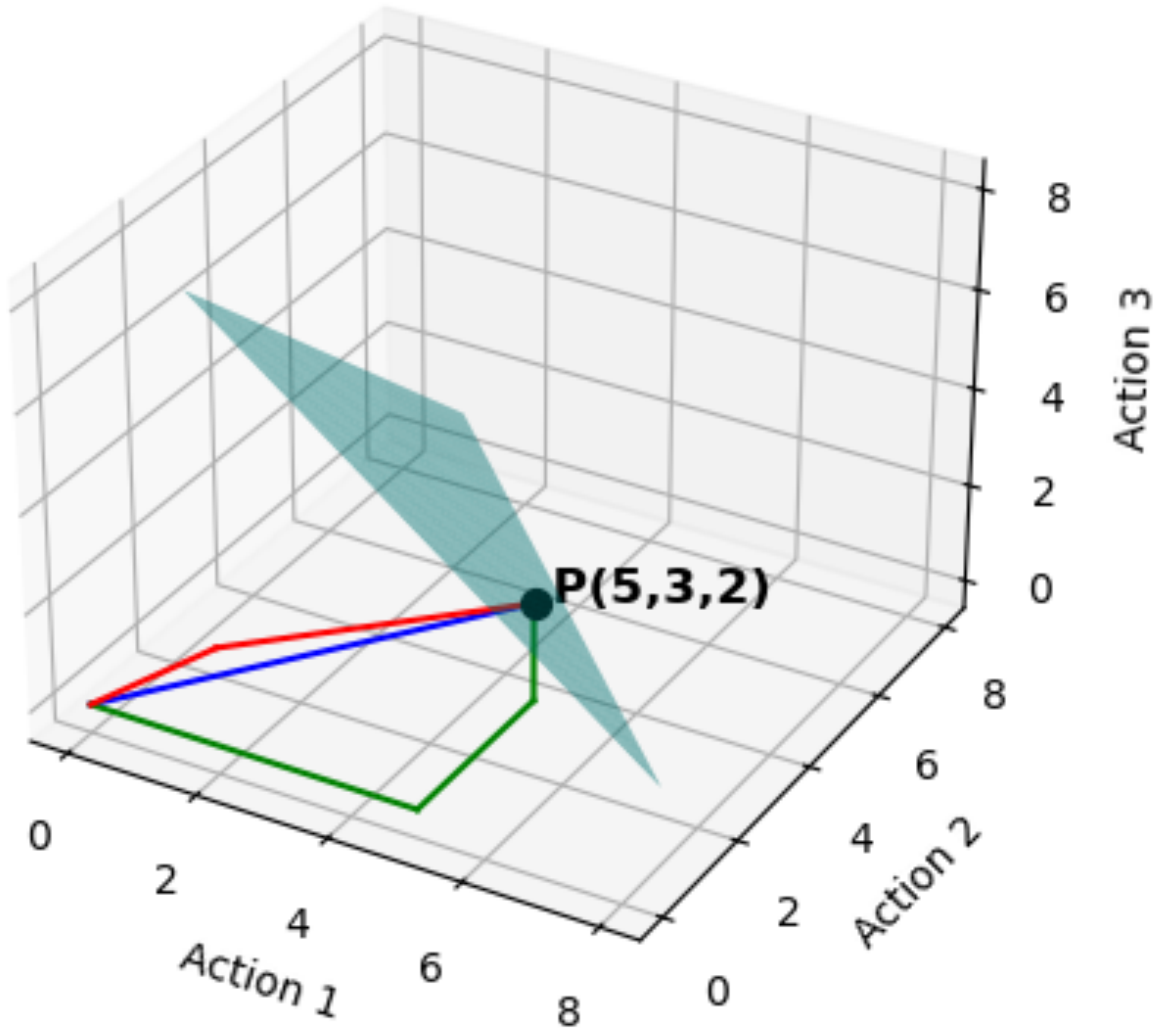
Learner strategy: $y : [0,T] \to \Delta(\mathscr{B})$

, where:

$$y_i(t) = \frac{exp(\eta \int_0^t x(s)^\top Be_i ds)}{\sum_{i=1}^m exp(\eta \int_0^t x(s)^\top Be_i ds}$$

a.k.a. replicator dynamics, the continuous time analog of MWU.

Theorem 1: The rewards of the optimizer depend only on the total time played each action.



Maximizing utility in zero-sum games

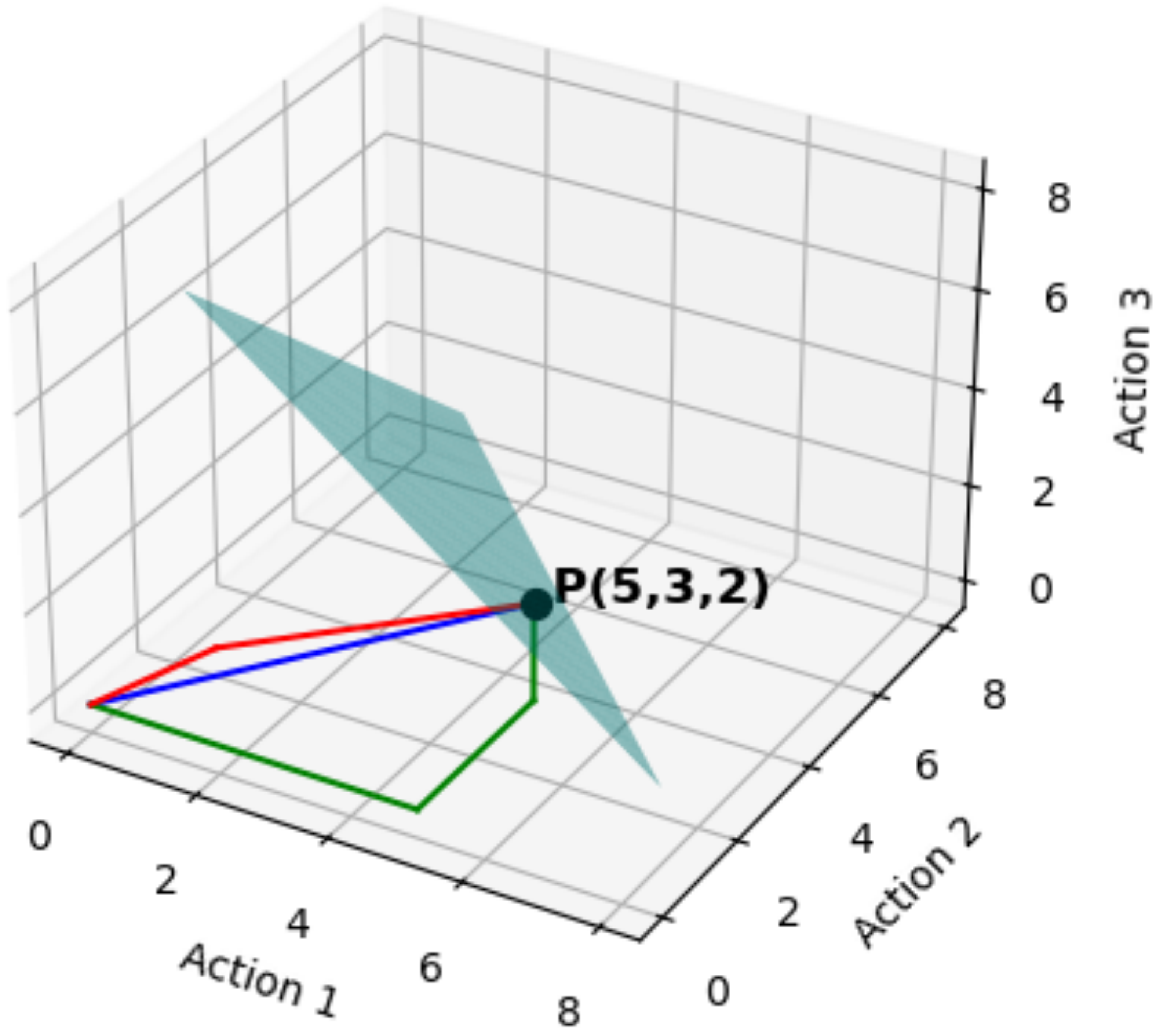
<u>Continuous time setting</u>

<u>Theorem 1:</u> The rewards of the optimizer depend only on the total time played each action.

<u>Corollary 1:</u> Optimal rewards can be achieved by a constant strategy i.e.

$$x(t) = x^*, x^* \in \Delta(\mathscr{A}), \forall t \in [0,T]$$

Moreover, this strategy can be efficiently computed in polynomial time.



Maximizing utility in zero-sum games

Discrete time setting

Optimizer strategy: any $x : \{1,\ldots,T\} \to \Delta(\mathscr{A})$

Learner strategy: $y : \{1,\ldots,T\} \to \Delta(\mathscr{B})$

, where:

$$y_i(t) = \frac{exp(\eta \sum_{s=1}^{t-1} x(s)^\top Be_i)}{\sum_{i=1}^{m} exp(\eta \sum_{s=1}^{t-1} x(s)^\top Be_i}$$

a.k.a. MWU or Hedge.

Maximizing utility in zero-sum games

Discrete time setting

Optimizer strategy: any $x : \{1,\ldots,T\} \to \Delta(\mathscr{A})$

Learner strategy: $y : \{1,\ldots,T\} \to \Delta(\mathscr{B})$

, where:

$$y_i(t) = \frac{exp(\eta \sum_{s=1}^{t-1} x(s)^\top Be_i)}{\sum_{i=1}^{m} exp(\eta \sum_{s=1}^{t-1} x(s)^\top Be_i}$$

a.k.a. MWU or Hedge.

$R_{cont}(A, B, T)$: optimal rewards for the optimizer in the continuous game.

$R_{disc}(A, B, T)$: optimal rewards for the optimizer in the discrete game.

Theorem 2: The following are true:

1. $R_{cont}(A, -A, T) \leq R_{disc}(A, -A, T) \leq R_{cont}(A, -A, T) + \frac{\eta T}{2}$

2. There are classes of games for which $R_{disc}(A, -A, T) = R_{cont}(A, -A, T) + \Omega(\eta T)$

Computational Barrier in general-sum games

Learner is purely best responding to the history:

$$y(t) = \text{arg max}_{y \in \Delta(\mathscr{B})} \sum_{s=1}^{t-1} x(s)^\top B y$$

a.k.a. fictitious play or MWU with $\eta \to \infty$.

Computational Barrier in general-sum games

Learner is purely best responding to the history:

$$y(t) = \text{arg max}_{y \in \Delta(\mathscr{B})} \sum_{s=1}^{t-1} x(s)^\top B y$$

a.k.a. fictitious play or MWU with $\eta \to \infty$.

<u>**OCDP**</u> instance defined by (A, B, n, m, k, T):

- A, B matrices and n, m number of actions for *learner* and *optimizer respectively*.
- T total rounds of the game.

Instance is 'YES' if the optimizer can achieve total reward more than k and 'NO' otherwise.

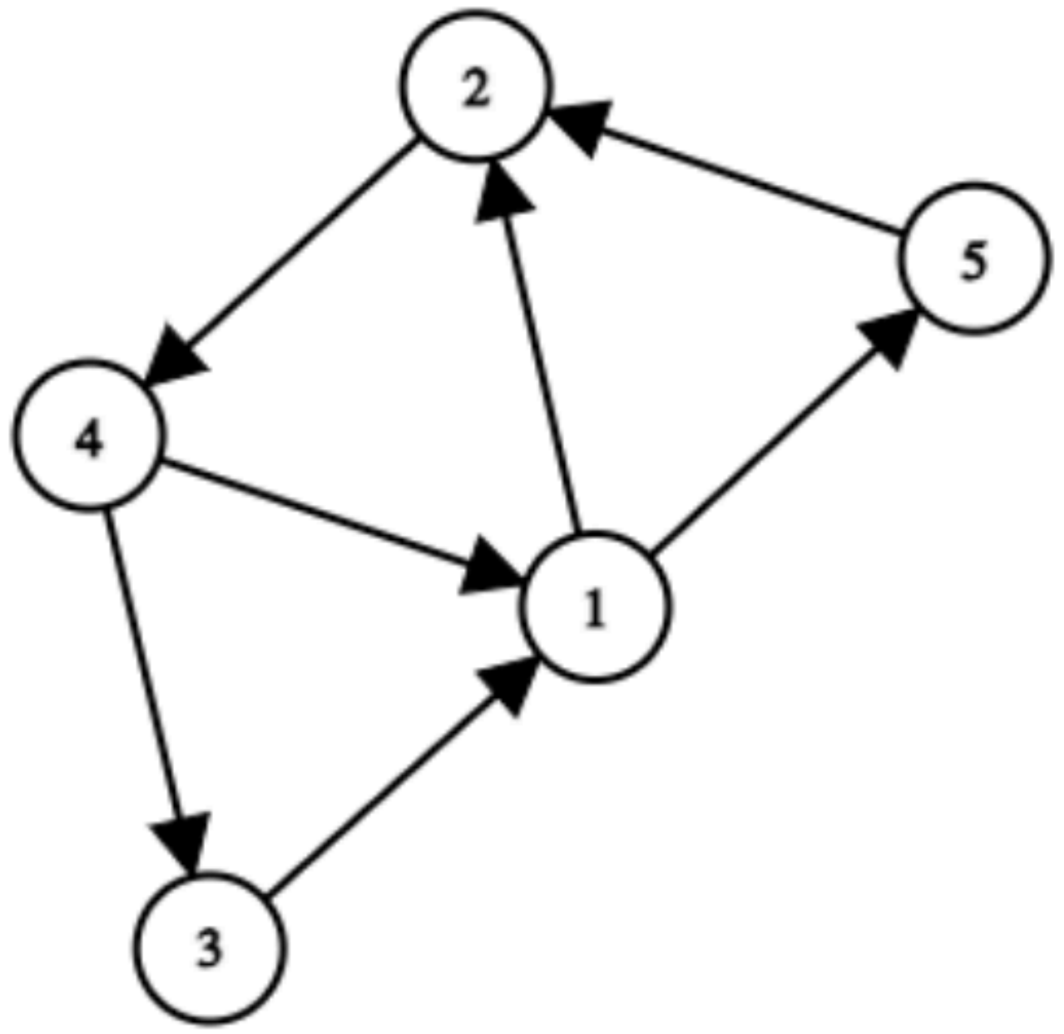
Computational Barrier in general-sum games

Theorem 2: OCDP is NP hard.

Proof sketch: Reduction from Hamiltonian Cycle.

Hamiltonian Cycle instance

$G = (V, E)$



OCDP instance

$T = k = |V| + 1$

A =

	v_1	v_2	v_3	v_4	v_5	v_{in_1}	v_{in_2}	v_{in_3}	v_{in_4}	v_{in_5}
e_1	1	0	0	0	0	0	0	0	0	0
e_2	0	0	0	0	1	0	0	0	0	0
e_3	1	0	0	0	0	0	0	0	0	0
e_4	0	1	0	0	0	0	0	0	0	0
e_5	0	0	0	1	0	0	0	0	0	0
e_6	0	0	0	1	0	0	0	0	0	0
e_7	0	0	1	0	0	0	0	0	0	0

B =

	v_1	v_2	v_3	v_4	v_5	v_{in_1}	v_{in_2}	v_{in_3}	v_{in_4}	v_{in_5}
e_1	-1	0	0	0	1	0.85	0	0	0	0
e_2	0	1	0	0	-4	0	0	0	0	0.85
e_3	-1	1	0	0	0	0.85	0	0	0	0
e_4	0	-4	0	1	0	0	0.85	0	0	0
e_5	1	0	0	-4	0	0	0	0	0.85	0
e_6	0	0	1	-4	0	0	0	0	0.85	0
e_7	1	0	-4	0	0	0	0	0.85	0	0

Summary

In short, our results:

1. In zero sum games, we show exactly how the optimizer should play against a MWU learner.

2. In general sum games, we provide the first known computational lower bound for computing optimal strategies against a best responding learner.

Summary

In short, our results:

1. In zero sum games, we show exactly how the optimizer should play against a MWU learner.

2. In general sum games, we provide the first known computational lower bound for computing optimal strategies against a best responding learner.

Thank you!