Recent advances in computing Nash equilibria in Markov Games

Ioannis Panageas (UC Irvine)

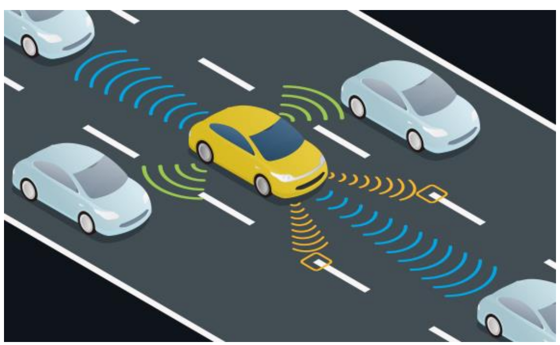
Based on joint works with F.Kalogiannis, I.Anagnostides, S.Leonardos, W.Overman, M.Vlatakis, V.Chatziafratis, G.Piliouras and S.Stavroulakis

Multi-agent systems and RL

Decentralized systems

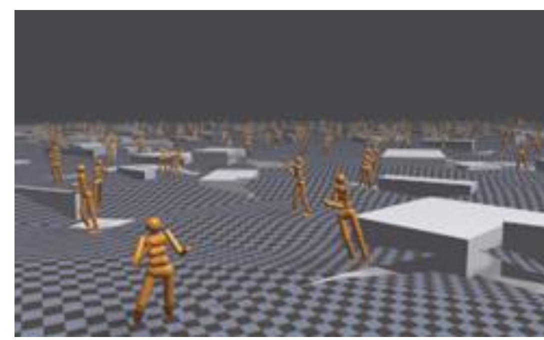
Individual interests (rational agents, cooperation/competition etc)

Distributed optimization



Self-driving cars

Auctions



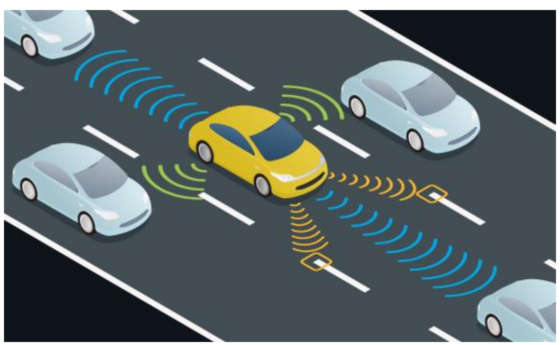
Robotics

Multi-agent systems and RL

Decentralized systems

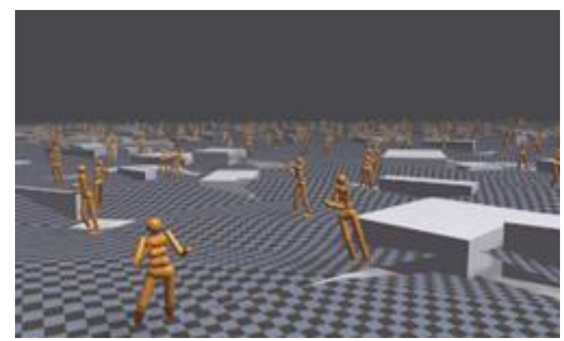
Individual interests (rational agents, cooperation/competition etc)

Distributed optimization



Self-driving cars

Auctions



Robotics

How these systems evolve? Predictions?

Outline

- Basics on single agent RL and policy gradient
- Definitions and basics on two-player zero-sum games
- Potential Markov games
- Adversarial Markov team games
- Polymatrix Markov games
- Open Questions/Future projects

- *Single agent RL*

The framework

A finite Markov Decision Process (MDP) is defined as follows:

- A finite state space $\mathcal{S}$.
- A finite action space $\mathcal{A}$.
- A transition model $\mathbb{P}$ where $\mathbb{P}(s'|s, a)$ is the probability of transitioning into state s' upon taking action a in state s. $\mathbb{P}$ is a matrix of size $(S \cdot A) \times S$.
- Reward function $r : \mathcal{S} \times \mathcal{A} \to [-1, 1]$.
- A discounted factor $\gamma \in [0, 1)$.
- $\boldsymbol{\rho} \in \Delta(\mathcal{S})$, an initial state distribution.

Definitions

Definition (Markovian stationary policy). *Policy is called a function*

$$\pi : \mathcal{S} \to \mathcal{A}.$$

Definition (Value function). *Given a policy π the value function is given by*

$$V^{\pi}(\boldsymbol{\rho}) = \mathbb{E}_{\pi,\mathbb{P}}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 \sim \boldsymbol{\rho}\right]$$

The goal is to solve

$$\max_{\pi} V^{\pi}(\boldsymbol{\rho}).$$

Definitions

Definition (Markovian stationary policy). *Policy is called a function*

$$\pi : \mathcal{S} \to \mathcal{A}.$$

Definition (Value function). *Given a policy π the value function is given by*

$$V^{\pi}(\boldsymbol{\rho}) = \mathbb{E}_{\pi,\mathbb{P}}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 \sim \boldsymbol{\rho}\right]$$

The goal is to solve

$$\max_{\pi} V^{\pi}(\boldsymbol{\rho}).$$

Remarks

- The **max** operator is over all (possibly non-stationary and randomized) policies.
- It suffices to focus on deterministic.
- *V is not concave in π.*

Example

Example (Navigation). *Suppose you are given a grid map. The state of the agent is their current location. The four actions might be moving 1 step along each of east, west, north or south. The transitions in the simplest setting are deterministic. There is a goal g that is trying to reach. Reward is one if the agent reaches the goal and zero otherwise.*

0.729	0.81	0.9	☆
0.656		0.81	0.9
0.590	0.656	0.729	0.81

→	→	→	☆
↑		↑	↑
↑	→	↑	↑

Remark

- What is V?
- What is γ in the example?

Bellman operator

Definition (Bellman Operator). *Let's define the following operator $\mathcal{T}$:*

$$\mathcal{T}\, W(s) = \max_{a \in \mathcal{A}} \{ r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) W(s') \}$$

Set $V^*(s) := \max_{\pi} V^{\pi}(s)$.

Claim (Bellman Operator). *V^* is the unique fixed point of the operator.*

Bellman operator

Definition (Bellman Operator). *Let's define the following operator* $\mathcal{T}$*:*

$$\mathcal{T}\,W(s) = \max_{a \in \mathcal{A}} \{r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) W(s')\}$$

Set $V^*(s) := \max_{\pi} V^{\pi}(s)$.

Claim (Bellman Operator). V^* *is the unique fixed point of the operator.*

Proof. Easy to see V^* is a fixed point. We will show that $\mathcal{T}$ is contracting! (Banach Fixed point Theorem).

Bellman operator

Definition (Bellman Operator). *Let's define the following operator $\mathcal{T}$:*

$$\mathcal{T}\, W(s) = \max_{a \in \mathcal{A}} \{ r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) W(s') \}$$

Set $V^*(s) := \max_{\pi} V^{\pi}(s)$.

Claim (Bellman Operator). *V^* is the unique fixed point of the operator.*

Proof. Easy to see V^* is a fixed point. We will show that $\mathcal{T}$ is contracting! (Banach Fixed point Theorem).

$$\|\mathcal{T}V - \mathcal{T}V'\|_\infty = \left\| \max_a \{ r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|a,s) V(s') \} - \max_{a'} \{ r(s,a') + \gamma \sum_{s'} \mathbb{P}(s'|a',s) V'(s') \} \right\|_\infty$$

Bellman operator

Definition (Bellman Operator). *Let's define the following operator $\mathcal{T}$:*

$$\mathcal{T}\ W(s) = \max_{a \in \mathcal{A}} \{ r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|s,a) W(s') \}$$

Set $V^*(s) := \max_{\pi} V^{\pi}(s)$.

Claim (Bellman Operator). *V^* is the unique fixed point of the operator.*

Proof. Easy to see V^* is a fixed point. We will show that $\mathcal{T}$ is contracting! (Banach Fixed point Theorem).

$$\|\mathcal{T}V - \mathcal{T}V'\|_\infty = \left\| \max_a \{ r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|a,s) V(s') \} - \max_{a'} \{ r(s,a') + \gamma \sum_{s'} \mathbb{P}(s'|a',s) V'(s') \} \right\|_\infty$$

$$\leq \left\| \max_a \{ r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|a,s) V(s') - r(s,a) - \gamma \sum_{s'} \mathbb{P}(s'|a,s) V'(s') \} \right\|_\infty$$

Bellman operator

$||x - y||_\infty \geq |||x||_\infty - ||y||_\infty|$

$$\|\mathcal{T}V - \mathcal{T}V'\|_\infty = \left\| \max_a \{r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|a,s)V(s')\} - \max_{a'} \{r(s,a') + \gamma \sum_{s'} \mathbb{P}(s'|a',s)V'(s')\} \right\|_\infty$$

$$\leq \left\| \max_a \{r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|a,s)V(s') - r(s,a) - \gamma \sum_{s'} \mathbb{P}(s'|a,s)V'(s')\} \right\|_\infty$$

Bellman operator

$$\|\mathcal{T}V - \mathcal{T}V'\|_\infty = \left\|\max_a\{r(s,a) + \gamma\sum_{s'}\mathbb{P}(s'|a,s)V(s')\} - \max_{a'}\{r(s,a') + \gamma\sum_{s'}\mathbb{P}(s'|a',s)V'(s')\}\right\|_\infty$$

$$\leq \left\|\max_a\{r(s,a) + \gamma\sum_{s'}\mathbb{P}(s'|a,s)V(s') - r(s,a) - \gamma\sum_{s'}\mathbb{P}(s'|a,s)V'(s')\}\right\|_\infty$$

$$= \gamma\left\|\max_a\{\mathbb{P}_a(V - V')\}\right\|_\infty$$

Bellman operator

$$||Ax||_\infty \leq ||A||_\infty ||x||_\infty$$

$$\|\mathcal{T}V - \mathcal{T}V'\|_\infty = \left\| \max_a \{r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|a,s)V(s')\} - \max_{a'}\{r(s,a') + \gamma \sum_{s'} \mathbb{P}(s'|a',s)V'(s')\} \right\|_\infty$$

$$\leq \left\| \max_a \{r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|a,s)V(s') - r(s,a) - \gamma \sum_{s'} \mathbb{P}(s'|a,s)V'(s')\} \right\|_\infty$$

$$= \gamma \left\| \max_a \{\mathbb{P}_a(V - V')\} \right\|_\infty$$

$$\leq \gamma \|V - V'\|_\infty \qquad \text{since } \|\mathbb{P}_a\|_\infty = 1.$$

Remarks

- Bellman operator is contracting for infinity norm.
- Applying the operator does not give a polynomial time algorithm. Why?
- Linear programming can give optimal policies in polynomial time.

- *Definition of Markov games and solution concepts*

n-player Markov game: Formal definition

Markov games or *stochastic* games are established as a framework for multi-agent reinforcement learning [Littman, 1994]

- $\mathcal{S}$, a finite state space,
- $\mathcal{N}$, a finite set of agents with $n := |\mathcal{N}|$,
- $\mathcal{A}_k$, a finite action space each player k, and $\mathcal{A} = \times_{k=1}^{n} \mathcal{A}_k$
- $r_k : \mathcal{S} \times \mathcal{A} \to [-1, 1]$, a reward function for each agent k,
- $\mathbb{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ a transition probability function,
- $\gamma \in [0, 1)$, a discount factor,
- $\boldsymbol{\rho} \in \Delta(\mathcal{S})$, an initial state distribution.

Solution Concept

- Every agent k picks a *policy* $\pi_k : \mathcal{S} \to \Delta(\mathcal{A}_k)$ (*do not share randomness*)
- The goal of each agent k is to *maximize* their own value function:

$$V_k^{(\pi_k, \pi_{-k})}(\boldsymbol{\rho}) = \mathbb{E}_{\pi, \mathbb{P}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \;\middle|\; s_0 \sim \boldsymbol{\rho} \right].$$

Solution Concept

- Every agent k picks a *policy* $\pi_k : \mathcal{S} \to \Delta(\mathcal{A}_k)$ (*do not share randomness*)
- The goal of each agent k is to *maximize* their own value function:

$$V_k^{(\pi_k, \pi_{-k})}(\boldsymbol{\rho}) = \mathbb{E}_{\pi, \mathbb{P}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \;\middle|\; s_0 \sim \boldsymbol{\rho} \right].$$

An ϵ-approximate *Nash equilibrium (NE)* $\pi^* = (\pi_1^*, \ldots, \pi_n^*)$ means that no agent can unilaterally increase their expected value more than ϵ,

$$V_k^{\pi^*}(\boldsymbol{\rho}) \geq V_k^{(\pi'_k, \pi^*_{-k})}(\boldsymbol{\rho}) - \epsilon, \;\; \forall k \in \mathcal{N}, \forall \pi'_k.$$

Solution Concept

- Every agent k picks a *policy* $\pi_k : \mathcal{S} \to \Delta(\mathcal{A}_k)$ (*do not share randomness*)
- The goal of each agent k is to *maximize* their own value function:

$$V_k^{(\pi_k, \pi_{-k})}(\boldsymbol{\rho}) = \mathbb{E}_{\pi, \mathbb{P}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \;\middle|\; s_0 \sim \boldsymbol{\rho} \right].$$

An ϵ-approximate *Nash equilibrium (NE)* $\pi^* = (\pi_1^*, \ldots, \pi_n^*)$ means that no agent can unilaterally increase their expected value more than ϵ,

$$V_k^{\pi^*}(\boldsymbol{\rho}) \geq V_k^{(\pi'_k, \pi^*_{-k})}(\boldsymbol{\rho}) - \epsilon, \;\; \forall k \in \mathcal{N}, \forall \pi'_k.$$

Remarks

- Fixing all agents but *i*, induces a classic MDP. Every agent aims at (approximate) best response.
- Generalizes notion of Nash Equilibrium.
- Nash (stationary, Markovian) policy always exist (Fink 64).
- Policies are defined to be *Markovian* and *stationary.*

Policy Gradient Iteration

Definition (Direct Parametrization). *Every agent uses the following:*

$$\pi_k(a \mid s) = x_{k,s,a}$$

with $x_{k,s,a} \geq 0$ *and* $\sum_{a \in A_k} x_{k,s,a} = 1$.

Policy Gradient Iteration

Definition (Direct Parametrization). *Every agent uses the following:*

$$\pi_k(a \mid s) = x_{k,s,a}$$

with $x_{k,s,a} \geq 0$ *and* $\sum_{a \in A_k} x_{k,s,a} = 1$.

Definition (Policy Gradient Ascent). *PGA is defined iteratively:*

$$x_k^{(t+1)} := \Pi_{\Delta(A_k)^S}(x_k^{(t)} + \eta \nabla_{x_k} V_k^{x^{(t)}}(\rho),$$

where Π denotes projection on product of simplices.

- *Two-player zero sum Markov games*

2-player zero-sum Markov games

– $\mathcal{N} = \{1, 2\}$, i.e., $n = 2$,

– $\mathcal{A}, \mathcal{B}$, the finite action space of players $1, 2$ respectively.

– $r_2 = -r_1$,

– rest the same.

Conventions

- We call player 2 the maximizer and player 1 the minimizer.
- The value of maximizer is $V^{(\pi_1, \pi_2)}(\rho)$.

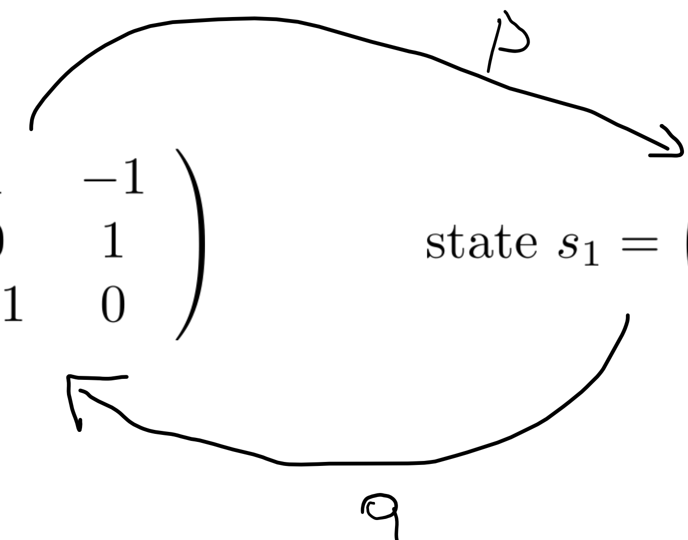
2-player zero-sum Markov games

- $\mathcal{N} = \{1, 2\}$, i.e., $n = 2$,
- $\mathcal{A}, \mathcal{B}$, the finite action space of players $1, 2$ respectively.
- $r_2 = -r_1$,
- rest the same.

Conventions

- We call player 2 the maximizer and player 1 the minimizer.
- The value of maximizer is $V^{(\pi_1, \pi_2)}(\rho)$.

$$\text{state } s_0 = \begin{pmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{pmatrix} \qquad \text{state } s_1 = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$



2-player zero-sum Markov games

A crucial property:

Theorem (Shapley 53). *In any two-player zero-sum Markov game*

$$\min_{\pi_1} \max_{\pi_2} V^{\pi_1,\pi_2}(\boldsymbol{\rho}) = \max_{\pi_2} \min_{\pi_1} V^{\pi_1,\pi_2}(\boldsymbol{\rho})$$

2-player zero-sum Markov games

A crucial property:

Theorem (Shapley 53). *In any two-player zero-sum Markov game*

$$\min_{\pi_1} \max_{\pi_2} V^{\pi_1,\pi_2}(\boldsymbol{\rho}) = \max_{\pi_2} \min_{\pi_1} V^{\pi_1,\pi_2}(\boldsymbol{\rho})$$

Remark

- The game has a unique value V^* (recall Von Neumann for normal form two player zero-sum games).
- The theorem implies it does not matter who plays first.
- The function is **not** convex-concave!
- The proof of Shapley uses a contraction argument.
- The complexity of finding a Nash equilibrium is *unknown*.

2-player zero-sum Markov games

Proof. Similar to Bellman, different operator.

Let val(.) be the operator applied to a payoff matrix that returns the value of the corresponding zero-sum game.

$$\text{e.g., val}\left(\begin{bmatrix}-1, 1 \\ 1, -1\end{bmatrix}\right) = 0.$$

2-player zero-sum Markov games

Proof. Similar to Bellman, different operator.

Let val(.) be the operator applied to a payoff matrix that returns the value of the corresponding zero-sum game.

$$\text{e.g., } \mathrm{val}\left(\begin{bmatrix} -1, 1 \\ 1, -1 \end{bmatrix} \right) = 0.$$

Fact: $|val(A) - val(B)| \leq max_{i,j} |A_{ij} - B_{ij}|$

Given a value vector $V(s)$, we define the operator $\mathcal{T}$

$$\mathcal{T}V(s) := \mathrm{val}(r_2(s,.,.) + \gamma \sum_{s'} \mathbb{P}(s'|s,.,.)V(s')).$$

2-player zero-sum Markov games

$$
\begin{aligned}
\|\mathcal{T}V - \mathcal{T}V'\|_\infty &= \left\| \mathrm{val}\{r(s,.,.) + \gamma \sum_{s'} \mathbb{P}(s'|s,.,.)V(s')\} - \mathrm{val}\{r(s,.,.) + \gamma \sum_{s'} \mathbb{P}(s'|s,.,.)V'(s')\} \right\|_\infty \\
&\leq \left\| \max_{a,b}\{r(s,a,b) + \gamma \sum_{s'} \mathbb{P}(s'|s,a,b)V(s') - r(s,a,b) - \gamma \sum_{s'} \mathbb{P}(s'|s,a,b)V'(s')\} \right\|_\infty \\
&= \gamma \left\| \max_{a,b}\{\mathbb{P}_{a,b}(V - V')\} \right\|_\infty \\
&\leq \gamma \|V - V'\|_\infty
\end{aligned}
$$

2-player zero-sum Markov games

$$\|\mathcal{T}V - \mathcal{T}V'\|_\infty = \left\| \mathrm{val}\{r(s,.,.) + \gamma \sum_{s'} \mathbb{P}(s'|s,.,.)V(s')\} - \mathrm{val}\{r(s,.,.) + \gamma \sum_{s'} \mathbb{P}(s'|s,.,.)V'(s')\} \right\|_\infty$$

$$\leq \left\| \max_{a,b}\{r(s,a,b) + \gamma \sum_{s'} \mathbb{P}(s'|s,a,b)V(s') - r(s,a,b) - \gamma \sum_{s'} \mathbb{P}(s'|s,a,b)V'(s')\} \right\|_\infty$$

$$= \gamma \left\| \max_{a,b}\{\mathbb{P}_{a,b}(V - V')\} \right\|_\infty$$

$$\leq \gamma \|V - V'\|_\infty$$

Remarks

- Bellman operator is contracting for infinity norm.
- Applying the operator does not give a polynomial time algorithm. Why?

Some facts about Policy Gradient

Definition (Policy Gradient Ascent). *PGA is defined iteratively:*

$$x_k^{(t+1)} := \Pi_{\Delta(A_k)^S}(x_k^{(t)} + \eta \nabla_{x_k} V_k^{x^{(t)}}(\rho)),$$

where Π denotes projection on product of simplices.

Theorem (Policy Gradient Ascent [Agarwal et al 2020]). *It can be shown for one agent that after $O(1/\epsilon^2)$ iterations, an ϵ-optimal policy can be reached.*

Theorem (Policy Gradient Descent/Ascent [Daskalakis et al 2020]). *It can be shown a two-time scale Policy Gradient Descent/Ascent can give an ϵ-Nash equilibrium in poly$(1/\epsilon)$ time.*

Remarks

- No guarantees for more than two players (only very specific settings).
- Can we find other classes of Markov games that PGA converges?
- In general, approximating even stationary CCE is PPAD-complete [Daskalakis et al 2022].

- *Potential Markov games*

Markov Potential Games

Definition (Markov Potential Game). *A Markov Decision Process (MDP) is called a* Markov Potential Game (MPG) *if there exists a (state-dependent) function* $\Phi_s : \Pi \to \mathbb{R}$ *for* $s \in S$ *so that*

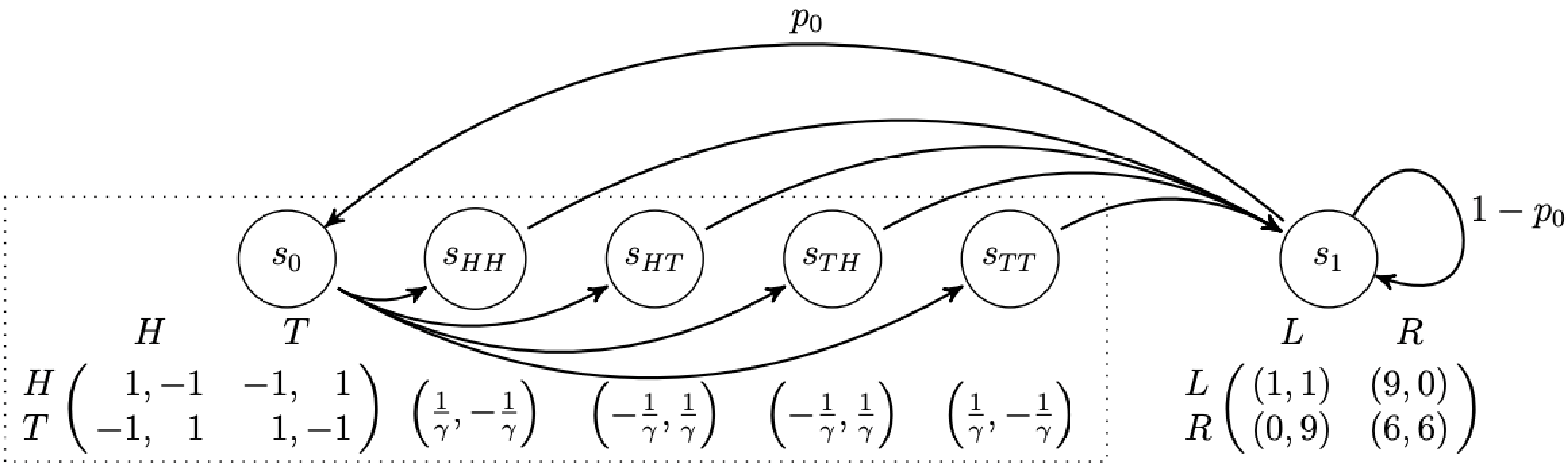
$$\Phi^{(\pi_k, \pi_{-k})}(s) - \Phi^{(\pi'_k, \pi_{-k})}(s) = V_k^{(\pi_k, \pi_{-k})}(s) - V_k^{(\pi'_k, \pi_{-k})}(s),$$

for all agents $k \in \mathcal{N}$, *all states* $s \in \mathcal{S}$ *and all policies* $\pi_k, \pi'_k \in P_k, \pi_{-k} \in P_{-k}$.

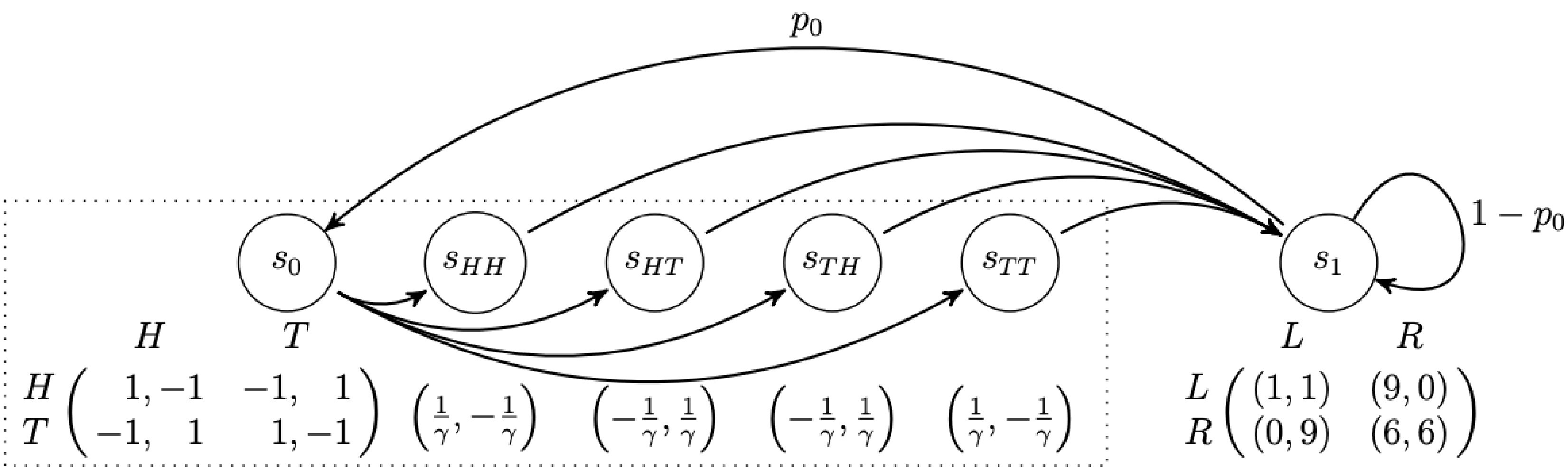
Remarks

- This notion generalizes the Potential Games in Game Theory.
- Potential Games capture routing (congestion games), important class.
- Deterministic Nash policies always exist!
- Each state a potential game does not imply MPG. Might have also zero sum game states!

An example of a MPG

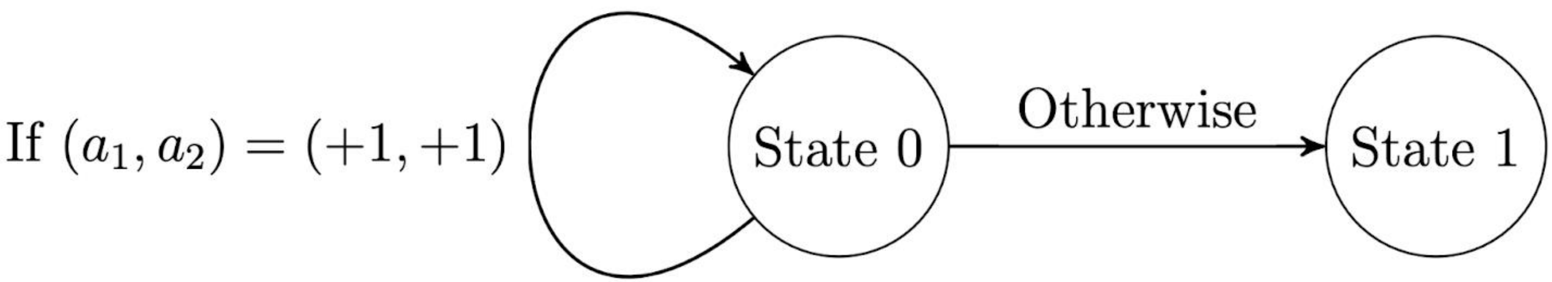


An example of a MPG



Subgames can be zero sum!

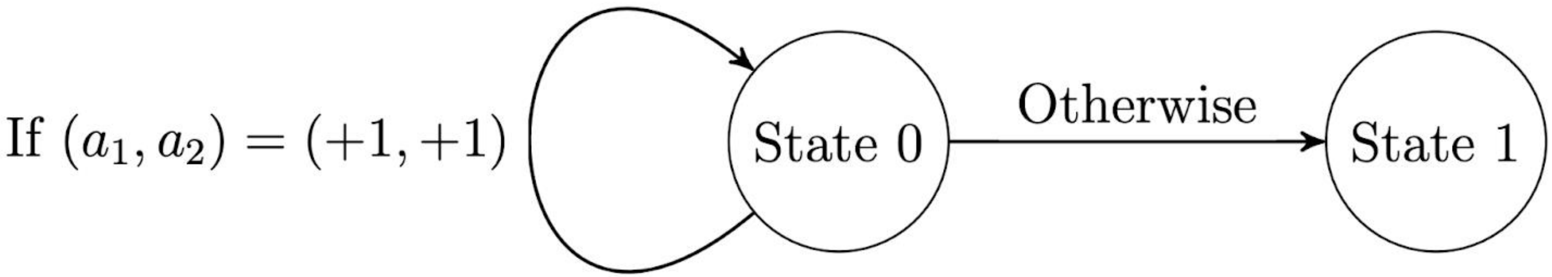
An example of an ``almost'' MPG



$$(R_0^1, R_0^2) = \begin{array}{c} \\ +1 \\ -1 \end{array} \begin{array}{c} \begin{array}{cc} +1 & -1 \end{array} \\ \begin{pmatrix} 5, 2 & -1, -2 \\ -5, -4 & 1, 4 \end{pmatrix} \end{array}$$

$$(R_1^1, R_1^2) = (0, 0)$$

An example of an ``almost” MPG

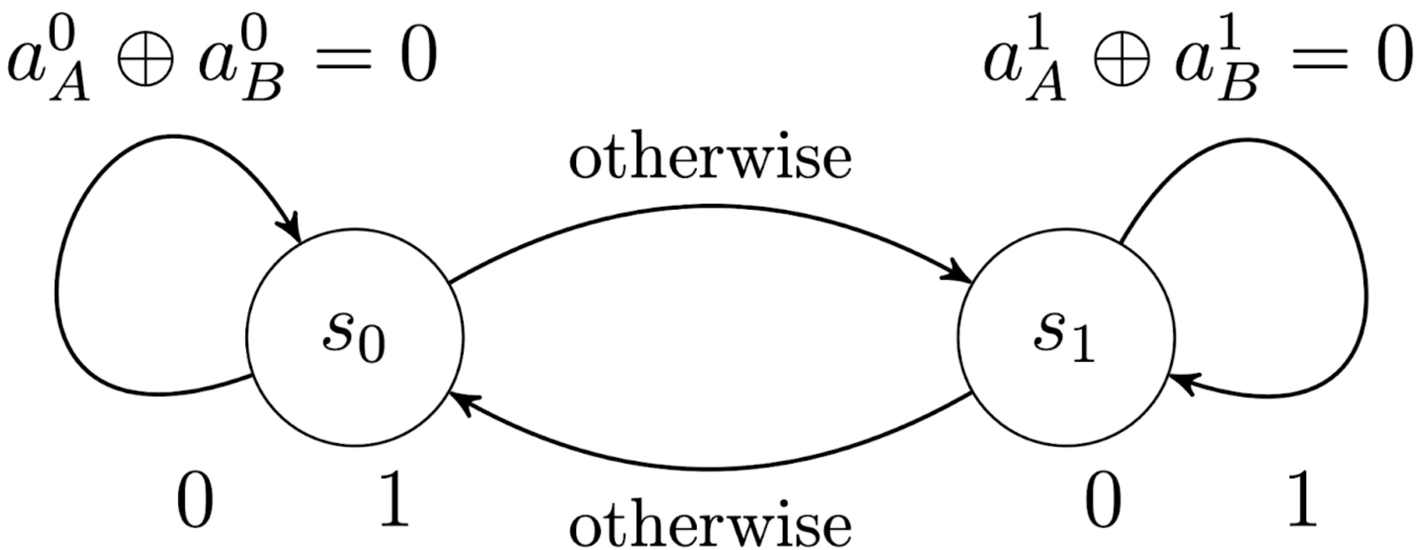


$$(R_0^1, R_0^2) = \begin{array}{c} \\ +1 \\ -1 \end{array} \begin{array}{c} \begin{array}{cc} +1 & -1 \end{array} \\ \begin{pmatrix} 5, 2 & -1, -2 \\ -5, -4 & 1, 4 \end{pmatrix} \end{array}$$

$$(R_1^1, R_1^2) = (0, 0)$$

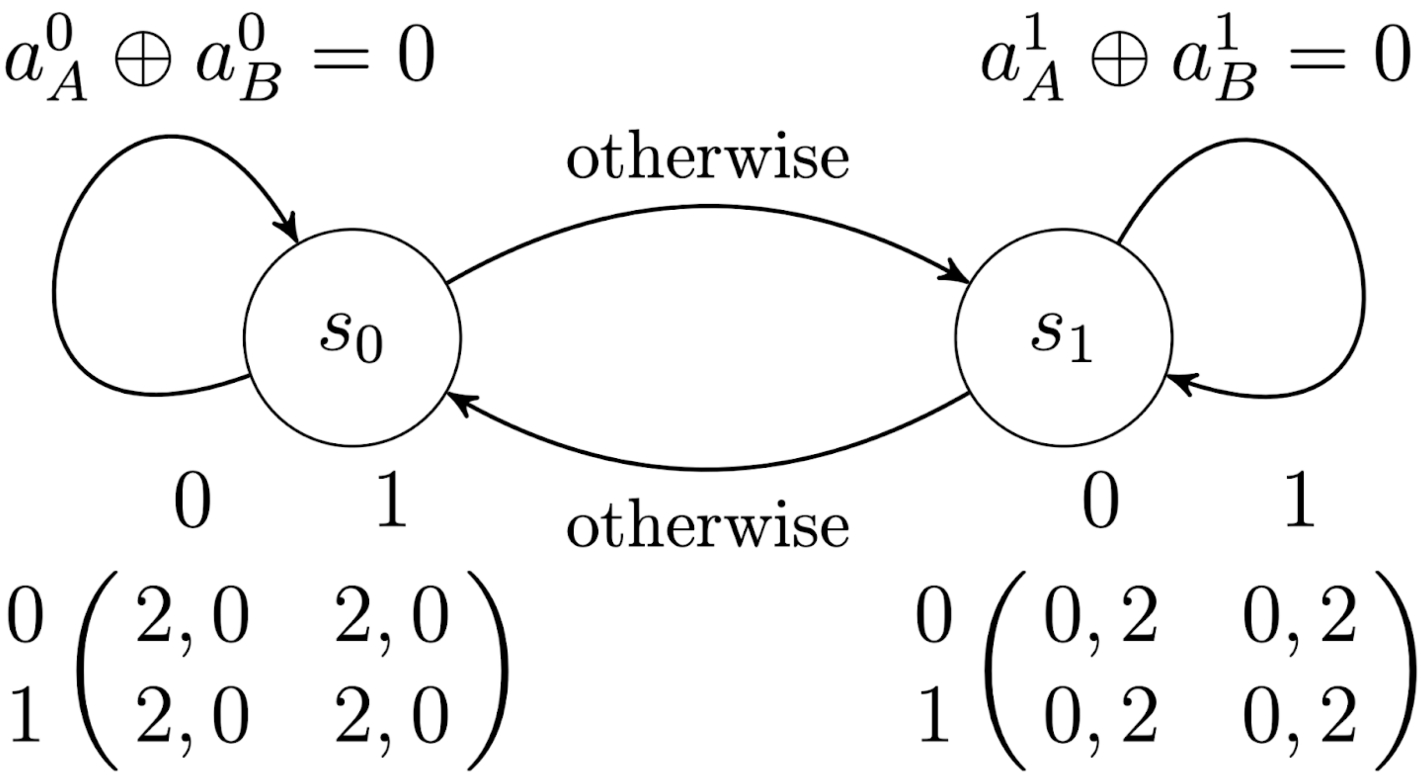
Ordinal MPG!

Not Markov Potential Game



$$\begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} 2,0 & 2,0 \\ 2,0 & 2,0 \end{pmatrix} \end{array} \qquad \begin{array}{c} \\ 0 \\ 1 \end{array} \begin{array}{c} \begin{array}{cc} 0 & 1 \end{array} \\ \begin{pmatrix} 0,2 & 0,2 \\ 0,2 & 0,2 \end{pmatrix} \end{array}$$

Not Markov Potential Game



Transitions can create competition!

Main Result

Theorem (PGA for Markov Potential Games)**.** *Suppose all agents run policy gradient iteration independently and update simultaneously. It can be shown that after* $O(1/\epsilon^2)$ *iterations, an* ϵ*-Nash policy can be reached.*

Remarks

- This result can be generalized (different rates, i.e., $O\left(\frac{1}{\epsilon^6}\right)$) if agents do not have access to exact gradients. Stochastic variant + greedy parametrization.
- It matches the result for single-agent.
- The running time depends polynomially on the sum of cardinalities of the players' actions spaces and not on the product.

Proof Steps I

Lemma (Key Lemma 1). *Policy gradient on values of agents is equivalent to projected gradient ascent on* Φ. *Formally it holds*

$$\nabla_{x_k} \Phi = \nabla_{x_k} V_k.$$

Remarks

- This is true by definition of Φ. Note that we do not know Φ!
- Policy gradient is Projected Gradient Ascent on Φ

Proof Steps I

Lemma (Key Lemma 1). *Policy gradient on values of agents is equivalent to projected gradient ascent on* Φ. *Formally it holds*

$$\nabla_{x_k}\Phi = \nabla_{x_k} V_k.$$

Remarks

- This is true by definition of Φ. Note that we do not know Φ!
- Policy gradient is Projected Gradient Ascent on Φ

Lemma (Key Lemma 2). *Stationary points for* Φ *are exactly Nash policies!*

Remarks

- This is a technical lemma, it uses the gradient domination property.

Gradient domination (PL condition) $f(x^*) - f(x) = O(G(x))$ where G(x) is a scalar notion of first-order stationarity (e.g. $||\nabla f||$).

- It holds for approximate stationary points too.

Proof Steps II

Lemma (Theorem (e.g., [Ghadimi et al 2013])). *Gradient descent reaches an ϵ-stationary point after $O(\frac{1}{\epsilon^2})$ steps for functions f with Lipschitz gradient.*

Intuition: A standard descent lemma gives:

$$f\left(x - \frac{1}{L}\nabla f(x)\right) - f(x) \leq -C(\|\nabla f(x)\|_2^2).$$

Proof Steps II

Lemma (Theorem (e.g., [Ghadimi et al 2013])). *Gradient descent reaches an ϵ-stationary point after $O(\frac{1}{\epsilon^2})$ steps for functions f with Lipschitz gradient.*

Intuition: A standard descent lemma gives:

$$f(x_{t+1}) - f(x_t) \leq -C(\|\nabla f(x_t)\|_2^2).$$

Assume that $\|\nabla f(x_t)\|_2 > \epsilon$ for $t = 1, ..., T$. We get that

$$f(x_T) - f(x_{T-1}) + f(x_{T-1}) - f(x_{T-2}) + ... + f(x_1) - f(x_0) < -C\epsilon^2 T.$$

Proof Steps II

Lemma (Theorem (e.g., [Ghadimi et al 2013])). *Gradient descent reaches an ϵ-stationary point after $O(\frac{1}{\epsilon^2})$ steps for functions f with Lipschitz gradient.*

Intuition: A standard descent lemma gives:

$$f(x_{t+1}) - f(x_t) \leq -C(\|\nabla f(x_t)\|_2^2).$$

Assume that $\|\nabla f(x_t)\|_2 > \epsilon$ for $t = 1, ..., T$. We get that

$$f(x_T) - \cancel{f(x_{T-1})} + \cancel{f(x_{T-1})} - \cancel{f(x_{T-2})} + \ldots + \cancel{f(x_1)} - f(x_0) < -C\epsilon^2 T.$$

If $f(x) \geq f\text{min}$ then $T = O\left(\frac{1}{\epsilon^2}\right)$

Proof Steps II

Lemma (Theorem (e.g., [Ghadimi et al 2013])). *Gradient descent reaches an ϵ-stationary point after $O(\frac{1}{\epsilon^2})$ steps for functions f with Lipschitz gradient.*

Remarks

- We can follow the same analysis if Φ has Lipschitz gradient.
- We show that $\nabla_{x_k} V_k$ is Lipschitz (constant depends on number of agents, number of actions and discount factor γ).
- For the stochastic variant, we need an unbiased estimator with bounded variance.

- *Adversarial team Markov games*

Adversarial team Markov games

- $\mathcal{N} = \{1, \ldots, n\} \cup \{n+1\}$, i.e., n agents and one adversary agent,
- $\mathcal{A}_k$ denotes the finite action space of player k,
- $\mathcal{B}$ denotes the action space of the adversary,
- $r_i = r_j$ for $i, j \in [n]$,
- letting r_{adv} be the adversary's reward; the game is team zero-sum, *i.e.*,

$$\sum_{k=1}^{n} r_k(s, \boldsymbol{a}, b) + r_{\text{adv}}(s, \boldsymbol{a}, b) = 0,$$

- rest the same.

Adversarial team Markov games

- $\mathcal{N} = \{1, \ldots, n\} \cup \{n+1\}$, i.e., n agents and one adversary agent,
- $\mathcal{A}_k$ denotes the finite action space of player k,
- $\mathcal{B}$ denotes the action space of the adversary,
- $r_i = r_j$ for $i, j \in [n]$,
- letting r_{adv} be the adversary's reward; the game is team zero-sum, *i.e.*,

$$\sum_{k=1}^{n} r_k(s, \boldsymbol{a}, b) + r_{\text{adv}}(s, \boldsymbol{a}, b) = 0,$$

- rest the same.

Can we compute an approximate NE ?

Main result

Definition (IPGMAX)**.** *The independent policy gradient on the max function is as follows, for T steps do:*

$$\text{The adversary best-responds to } \boldsymbol{x}\text{:} \tag{1}$$

$$\boldsymbol{y}^\star \leftarrow \arg\max_{\boldsymbol{y}\in\Delta(\mathcal{B})^S} V_{\boldsymbol{\rho}}(\boldsymbol{x},\boldsymbol{y}) \tag{2}$$

$$\text{Every agent } k \text{ independently updates their strategy:} \tag{3}$$

$$\boldsymbol{x}_k \leftarrow \Pi_{\Delta(A_k)^S}\left\{\boldsymbol{x}_k - \eta\nabla_{\boldsymbol{x}_k} V_{\boldsymbol{\rho}}\left(\boldsymbol{x},\boldsymbol{y}^\star\right)\right\} \tag{4}$$

Get a point $\hat{\boldsymbol{x}}$ which is approximate stationary for $\max_{\boldsymbol{y}\in\Delta(\mathcal{B})^S} V_{\boldsymbol{\rho}}(\boldsymbol{x},\boldsymbol{y})$, solve a constrained linear program to get a $\hat{\boldsymbol{y}}$.

Main result

Definition (IPGMAX). *The independent policy gradient on the max function is as follows, for T steps do:*

$$\text{The adversary best-responds to } \boldsymbol{x}: \tag{1}$$

$$\boldsymbol{y}^\star \leftarrow \arg \max_{\boldsymbol{y} \in \Delta(\mathcal{B})^S} V_\rho(\boldsymbol{x}, \boldsymbol{y}) \tag{2}$$

$$\text{Every agent } k \text{ independently updates their strategy:} \tag{3}$$

$$\boldsymbol{x}_k \leftarrow \Pi_{\Delta(A_k)^S} \left\{ \boldsymbol{x}_k - \eta \nabla_{\boldsymbol{x}_k} V_\rho \left(\boldsymbol{x}, \boldsymbol{y}^\star \right) \right\} \tag{4}$$

Get a point $\hat{\boldsymbol{x}}$ *which is approximate stationary for* $\max_{\boldsymbol{y} \in \Delta(\mathcal{B})^S} V_\rho(\boldsymbol{x}, \boldsymbol{y})$, *solve a constrained linear program to get a* $\hat{\boldsymbol{y}}$.

Remark

- Thought experiment on RPS. What would this algorithm return? We need to do extra work to find a NE.

Main result

Theorem (IPGMAX + LP). *Running IPGMAX for* $O\left(\frac{poly(\sum_k |\mathcal{A}_k|+|\mathcal{B}|)}{\epsilon^4}\right)$ *iterations and in the end a LP, yields an* ϵ*-NE.*

Remarks

- We need $\frac{1}{\epsilon^4}$ iterations instead of $\frac{1}{\epsilon^2}$. Why?

Main result

Theorem (IPGMAX + LP). *Running IPGMAX for* $O\left(\frac{poly(\sum_k |\mathcal{A}_k| + |\mathcal{B}|)}{\epsilon^4}\right)$ *iterations and in the end a LP, yields an* ϵ*-NE.*

Remarks

- We need $\frac{1}{\epsilon^4}$ iterations instead of $\frac{1}{\epsilon^2}$. Why? max function is non-smooth.
- We show that an approximate stationary $\hat{x}$ point of $\max_y V(x,y)$ can always be extended to a $(\hat{x}, \hat{y})$ which is an approximate NE.

This includes proving existence of Lagrange multipliers for some non-convex program.

Main result

Theorem (IPGMAX + LP). *Running IPGMAX for* $O\left(\frac{poly(\sum_k |\mathcal{A}_k| + |\mathcal{B}|)}{\epsilon^4}\right)$ *iterations and in the end a LP, yields an* ϵ*-NE.*

Remarks

- We need $\frac{1}{\epsilon^4}$ iterations instead of $\frac{1}{\epsilon^2}$. Why? max function is non-smooth.
- We show that an approximate stationary $\hat{x}$ point of $\max_y V(x, y)$ can always be extended to a $(\hat{x}, \hat{y})$ which is an approximate NE.

This includes proving existence of Lagrange multipliers for some non-convex program.

- To get $\hat{y}$, we somehow create the dual which is linear.

- *Polymatrix Markov games*

(normal form) Polymatrix games

A polymatrix game is defined using a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where

– every agent i coincides with a vertex $v_i \in \mathcal{V}$,

– for every agent i, there is a finite action-space $\mathcal{A}_i$,

– every agent i has a utility function $u_i : \times_{i=1}^{n} \mathcal{A}_i \to [-1, 1]$,

– every edge $(i, j) \in \mathcal{E}$ stands for a two-player (general-sum) game (u_{ij}, u_{ji}).

(normal form) Polymatrix games

A polymatrix game is defined using a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where

- every agent i coincides with a vertex $v_i \in \mathcal{V}$,
- for every agent i, there is a finite action-space $\mathcal{A}_i$,
- every agent i has a utility function $u_i : \times_{i=1}^{n} \mathcal{A}_i \to [-1, 1]$,
- every edge $(i, j) \in \mathcal{E}$ stands for a two-player (general-sum) game (u_{ij}, u_{ji}).

In a *polymatrix game* the utility of every agent i is separable as a sum of pair-wise interactions dictated by the graph,

$$u_i(\boldsymbol{a}) = \sum_{j \in \mathrm{neighb}(i)} u_{ij}(a_i, a_j),$$

$$\text{where} \quad \boldsymbol{a} = (a_1, \ldots, a_i, a_j, \ldots, a_n) \in \times_{i=1}^{n} \mathcal{A}_i.$$

(normal form) Polymatrix games

A polymatrix game is defined using a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where

– every agent i coincides with a vertex $v_i \in \mathcal{V}$,

– for every agent i, there is a finite action-space $\mathcal{A}_i$,

– every agent i has a utility function $u_i : \times_{i=1}^{n} \mathcal{A}_i \to [-1, 1]$,

– every edge $(i, j) \in \mathcal{E}$ stands for a two-player (general-sum) game (u_{ij}, u_{ji}).

In a *polymatrix game* the utility of every agent i is separable as a sum of pair-wise interactions dictated by the graph,

$$u_i(\boldsymbol{a}) = \sum_{j \in \text{neighb}(i)} u_{ij}(a_i, a_j),$$

$$\text{where} \quad \boldsymbol{a} = (a_1, \ldots, a_i, a_j, \ldots, a_n) \in \times_{i=1}^{n} \mathcal{A}_i.$$

Finally it is called zero-sum if $\sum_i u_i = 0$

(normal form) Polymatrix games

Computing NE is easy (in P).

The solutions of the following linear program are Nash equilibria.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} w_i && \text{(1a)} \\ \text{subject to} \quad & w_i \geq u_i(a_i, \boldsymbol{x}_{-i}), \qquad \forall i \in [n], \forall a_i \in \mathcal{A}_i, && \text{(1b)} \\ & \boldsymbol{x}_i \in \Delta(\mathcal{A}_i), \ \forall i \in [n]. && \text{(1c)} \end{aligned}$$

(normal form) Polymatrix games

Computing NE is easy (in P).

The solutions of the following linear program are Nash equilibria.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} w_i && && \text{(1a)} \\ \text{subject to} \quad & w_i \geq u_i(a_i, \boldsymbol{x}_{-i}), && \forall i \in [n], \forall a_i \in \mathcal{A}_i, && \text{(1b)} \\ & \boldsymbol{x}_i \in \Delta(\mathcal{A}_i), \ \forall i \in [n]. && && \text{(1c)} \end{aligned}$$

Remarks

- There is more. The above coincides (slightly) with LP for CCE.

(normal form) Polymatrix games

Computing NE is easy (in P).

The solutions of the following linear program are Nash equilibria.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} w_i && && \text{(1a)} \\ \text{subject to} \quad & w_i \geq u_i(a_i, \boldsymbol{x}_{-i}), && \forall i \in [n], \forall a_i \in \mathcal{A}_i, && \text{(1b)} \\ & \boldsymbol{x}_i \in \Delta(\mathcal{A}_i), \ \forall i \in [n]. && && \text{(1c)} \end{aligned}$$

Remarks

- There is more. The above coincides (slightly) with LP for CCE.

Equilibrium collapse. Marginals of CCEs are NE!!!!

Polymatrix Markov games

A Markov game s.t for every state s, there exists a graph $\mathcal{G}_s(\mathcal{V}_s, \mathcal{E}_s)$ such that,

- the vertices $\mathcal{V}_s$ coincide with the agents,
- the reward function of each agent depends on *pair-wise* interactions with each neighbors,

$$r_i(s, \boldsymbol{a}) = \sum_{j \in \text{neighbors}(i)} r_{ij}(s, a_i, a_j).$$

- the sum of rewards at each state is 0,

$$\sum_{i=1}^{n} r_i(s, \boldsymbol{a}) = 0,$$

Polymatrix Markov games

A Markov game s.t for every state s, there exists a graph $\mathcal{G}_s(\mathcal{V}_s, \mathcal{E}_s)$ such that,

- the vertices $\mathcal{V}_s$ coincide with the agents,
- the reward function of each agent depends on *pair-wise* interactions with each neighbors,

$$r_i(s, \boldsymbol{a}) = \sum_{j \in \text{neighbors}(i)} r_{ij}(s, a_i, a_j).$$

- the sum of rewards at each state is 0,

$$\sum_{i=1}^{n} r_i(s, \boldsymbol{a}) = 0,$$

- *Assumption of Switching control*: at every state there is a single player that controls the probability of transtion to a new state.

Main result

Unfortunately we do not have a LP as before but we have equilibrium collapse.

Theorem (Equilibrium collapse)**.** *Let a coarse correlated equilibrium of the switching control, polymatrix zero-sum Markov game, $\boldsymbol{\sigma}$. Then the marginal product strategy profile, $\boldsymbol{x}^{\sigma}$,*

$$\boldsymbol{x}_{i,s}(a_i) = \sum_{\boldsymbol{a}_{-i} \in \mathcal{A}_{-i}} \sigma_s(a_i, \boldsymbol{a}_{-i})$$

is a Nash equilibrium of the game.

Main result

Unfortunately we do not have a LP as before but we have equilibrium collapse.

The corresponding program looks as follows:

$$\text{minimize} \quad \sum_{i=1}^{n} \sum_{s \in \mathcal{S}} \rho(s) w_i(s) \tag{1a}$$

$$\text{subject to} \quad w_i(s) \geq r_i(s, a_i, \boldsymbol{x}_{-i,s}) + \gamma \sum_{s' \in \mathcal{S}} \mathbb{P}(s' | s, a_i, \boldsymbol{x}_{-i,s}) w_i(s') \ \forall i \in [n], \forall s \in \mathcal{S}, \forall a_i \in \mathcal{A}_i, \tag{1b}$$

$$\boldsymbol{x}_{i,s} \in \Delta(\mathcal{A}_i), \ \forall i \in [n], \forall s \in \mathcal{S}. \tag{1c}$$

Remarks

- Any algorithm that gives approximate Markovian CCEs, gives approximate Markovian NE!