# L13 Stochastic Games (Markov Decision Processes).

CS 280 Algorithmic Game Theory

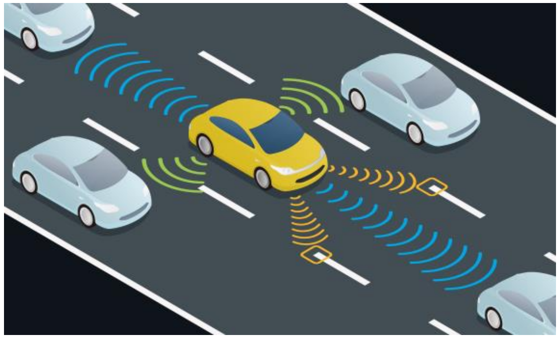
Ioannis Panageas

# Multi-agent systems and RL

Decentralized systems

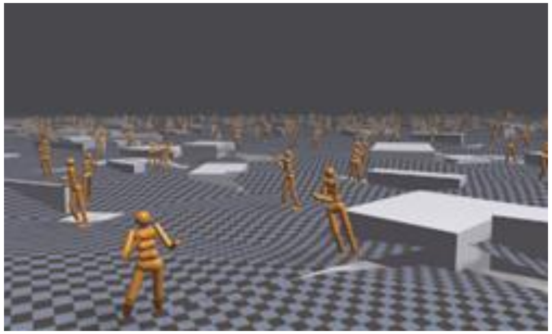
Individual interests (rational agents, cooperation/competition etc)

Distributed optimization



Self-driving cars


Auctions



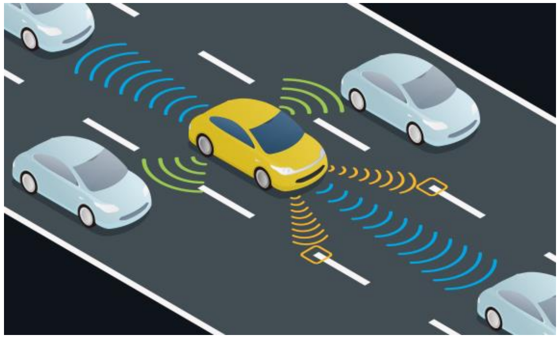
Robotics

# Multi-agent systems and RL

Decentralized systems

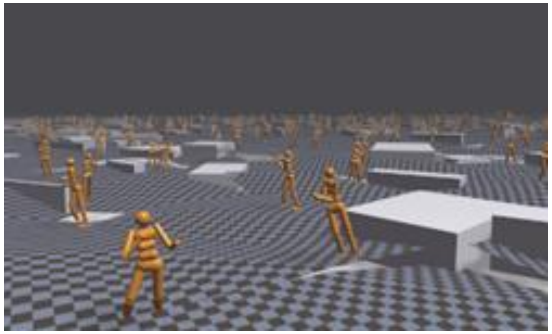
Individual interests (rational agents, cooperation/competition etc)

Distributed optimization



Self-driving cars


Auctions

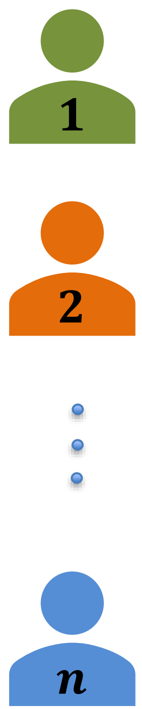


Robotics

**How these systems evolve? Predictions?**

# Markov Games

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




$\boldsymbol{n}$ number of players

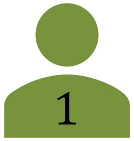
# Markov Games

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


$\boldsymbol{n}$ number of players

# Markov Games

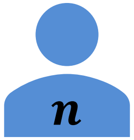
*Markov* games or *stochastic* games are established as a framework for multi-agent reinforcement learning [Littman, 1994].




Chooses action $a_1 \in A_1$


Chooses action $a_2 \in A_2$

⋮




Chooses action $a_n \in A_n$

Environment is at state $s \in S$

$\boldsymbol{n}$ number of players

# Markov Games

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

1 Gets reward $r_1(s, a_1, \ldots, a_n) \in [-1,1]$

2 Gets reward $r_2(s, a_1, \ldots, a_n) \in [-1,1]$

$s' \sim \mathrm{P}(s, a_1, \ldots, a_n)$

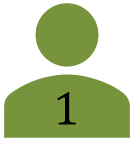
$n$ Gets reward $r_n(s, a_1, \ldots, a_n) \in [-1,1]$

Environment jumps at state $s' \in S$

$\boldsymbol{n}$ number of players

# Markov Games

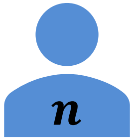
*Markov* games or *stochastic* games are established as a framework for multi-agent reinforcement learning [Littman, 1994].




Gets value $\mathrm{V}_1(\mathrm{s}^0) \coloneqq \sum_{t=0}^{H} r_1(s^t, a_1^t, \dots, a_n^t)$


Gets value $\mathrm{V}_2(\mathrm{s}^0) \coloneqq \sum_{t=0}^{H} r_2(s^t, a_1^t, \dots, a_n^t)$

$\vdots$




Gets value $\mathrm{V}_\mathrm{n}(\mathrm{s}^0) \coloneqq \sum_{t=0}^{H} r_n(s^t, a_1^t, \dots, a_n^t)$

$\boldsymbol{n}$ number of players

# Markov Games

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

Gets value $\mathrm{V}_1(\mathrm{s}^0) \coloneqq \sum_{t=0}^{H} r_1(s^t, a_1^t, \dots, a_n^t)$

Gets value $\mathrm{V}_2(\mathrm{s}^0) \coloneqq \sum_{t=0}^{H} r_2(s^t, a_1^t, \dots, a_n^t)$

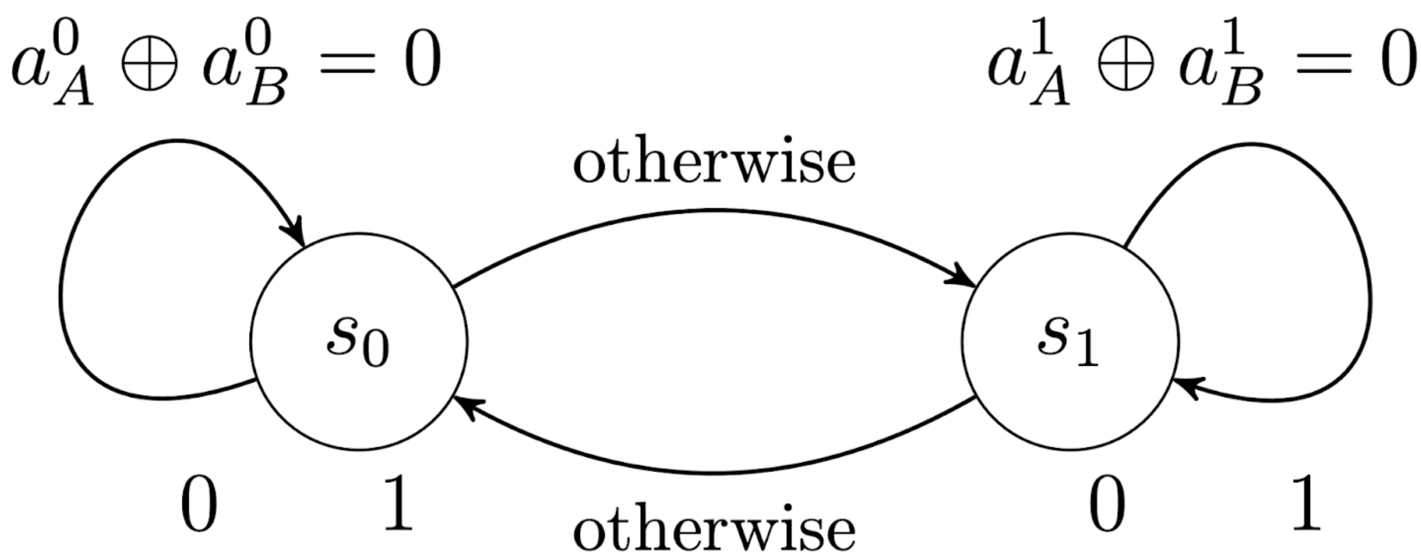
**If H is $\infty$, then we introduce a discount $\gamma$**

**e.g.,** $\mathrm{V}_1(\mathrm{s}^0) \coloneqq \sum_{t=0}^{\infty} \gamma^t r_1(s^t, a_1^t, \dots, a_n^t)$

Gets value $\mathrm{V}_\mathrm{n}(\mathrm{s}^0) \coloneqq \sum_{t=0}^{H} r_n(s^t, a_1^t, \dots, a_n^t)$

$\boldsymbol{n}$ number of players

# An example




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

# $n$-player Markov game: Formal definition

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

- $\mathcal{N}$, a finite set of agents with $n := |\mathcal{N}|$,
- $\mathcal{S}$, a finite state space,
- $\mathcal{A}_k$, 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,
- $\rho \in \Delta(\mathcal{S})$, an initial state distribution.

# $n$-player Markov game: Formal definition

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

- $\mathcal{N}$, a finite set of agents with $n := |\mathcal{N}|$,
- $\mathcal{S}$, a finite state space,
- $\mathcal{A}_k$, 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 $k$,
- $P : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ a transition probability function,
- $\gamma \in [0, 1)$, a discount factor,
- $\rho \in \Delta(\mathcal{S})$, an initial state distribution.

# $n$-player Markov game: Formal definition

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

- $\mathcal{N}$, a finite set of agents with $n := |\mathcal{N}|$,
- $\mathcal{S}$, a finite state space,
- $\mathcal{A}_k$, 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,
- $\rho \in \Delta(\mathcal{S})$, an initial state distribution.

# $n$-player Markov game: Formal definition

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

- $\mathcal{N}$, a finite set of agents with $n := |\mathcal{N}|$,
- $\mathcal{S}$, a finite state space,
- $\mathcal{A}_k$, 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,
- $\rho \in \Delta(\mathcal{S})$, an initial state distribution.

# $n$-player Markov game: Formal definition

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

- $\mathcal{N}$, a finite set of agents with $n := |\mathcal{N}|$,
- $\mathcal{S}$, a finite state space,
- $\mathcal{A}_k$, 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,
- $\rho \in \Delta(\mathcal{S})$, an initial state distribution.

# $n$-player Markov game: Formal definition

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

- $\mathcal{N}$, a finite set of agents with $n := |\mathcal{N}|$,
- $\mathcal{S}$, a finite state space,
- $\mathcal{A}_k$, 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.

- *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 $\pi$ 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 $\pi$ 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 $\pi$.*

# 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 $\gamma$ 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

$$\boxed{||Ax||_\infty \le ||A||_\infty ||x||_\infty}$$

$$\begin{aligned}
\|\mathcal{T}V - \mathcal{T}V'\|_\infty &= \left\| \max_a \{r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|a,s)V(s')\} - \max_{a'} \{r(s,a') + \gamma \sum_{s'} \mathbb{P}(s'|a',s)V'(s')\} \right\|_\infty \\
&\le \left\| \max_a \{r(s,a) + \gamma \sum_{s'} \mathbb{P}(s'|a,s)V(s') - r(s,a) - \gamma \sum_{s'} \mathbb{P}(s'|a,s)V'(s')\} \right\|_\infty \\
&= \gamma \left\| \max_a \{\mathbb{P}_a(V - V')\} \right\|_\infty \\
&\le \gamma \left\| V - V' \right\|_\infty \qquad \text{since } \|\mathbb{P}_a\|_\infty = 1.
\end{aligned}$$

## 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.

# Value Iteration

Idea: We build a sequence of value functions. Let $V_0$ be any vector, then iterate the application of the optimal Bellman operator so that given $V_k$ at iteration $k$ we compute

$$V_{k+1} = TV_k.$$

# Value Iteration

Idea: We build a sequence of value functions. Let $V_0$ be any vector, then iterate the application of the optimal Bellman operator so that given $V_k$ at iteration $k$ we compute

$$V_{k+1} = TV_k.$$

The policy will be given at every iteration as

$$\pi_k = \arg\max_a (1-\gamma) r(s,a) + \gamma \sum_{s'} P(s'|s,a) V_k(s')$$

$$\boxed{\text{After } k = \frac{\log(1/\epsilon)}{\log(1/\gamma)} \text{ we have error } \epsilon.}$$

# Policy Iteration

Idea: We build a sequence of policies. Let $\pi_0$ be any stationary policy. At each iteration k we perform the two following steps:

1. **Policy evaluation** given $\pi_k$, compute $V^{\pi_k}$.
2. **Policy improvement**: we compute the *greedy* policy $\pi_{k+1}$ from $V^{\pi_k}$ as:

$$\pi_{k+1}(x) \in \arg\max_{a \in A} \Big[ r(x,a) + \gamma \sum_y p(y|x,a) V^{\pi_k}(y) \Big].$$

The iterations continue until $V^{\pi_k} = V^{\pi_{k+1}}$.

- *Markov games: Solution concepts*

# Solution Concept: Nash equilibrium

- Every agent $k$ picks a policy $\pi_k$ : **4** possibilities

1. Markovian and stationary.
2. Markovian and **non**-stationary.
3. **Non**-Markovian and stationary.
4. **Non**-Markovian and **non**-stationary.

# Solution Concept: Nash equilibrium

- Every agent $k$ picks a policy $\pi_k$ : **4** possibilities

1. Markovian and stationary.
2. Markovian and **non**-stationary.
3. **Non**-Markovian and stationary.
4. **Non**-Markovian and **non**-stationary.

- The goal of each agent is to maximize their own value.

# Solution Concept: Nash equilibrium

- Every agent $k$ picks a policy $\pi_k$.
- The goal of each agent is to maximize their own value.

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

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

# Solution Concept: Nash equilibrium

- Every agent $k$ picks a policy $\pi_k$.
- The goal of each agent is to maximize their own value.

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

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

Remarks

- Agents do not share randomness.

# Solution Concept: Nash equilibrium

- Every agent $k$ picks a policy $\pi_k$.
- The goal of each agent is to maximize their own value.

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

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

Remarks

- Agents do not share randomness.
- Fixing all agents but $i$, induces a classic MDP. Every agent aims at (approximate) best response.

# Solution Concept: Nash equilibrium

- Every agent $k$ picks a policy $\pi_k$.
- The goal of each agent is to maximize their own value.

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

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

Remarks

- Agents do not share randomness.
- Fixing all agents but $i$, induces a classic MDP. Every agent aims at (approximate) best response.
- Generalizes notion of Nash Equilibrium.
- Nash policies always exist (Fink 64).

# The bad news

- Markov games generalize normal form games.

➡ Inherit *computational* intractability

# The bad news

- Markov games generalize normal form games.

⇒ Inherit *computational* intractability

[Daskalakis, Goldberg, Papadimitriou 06]
[Chen, Deng 06]
[Rubinstein 15] **PPAD-hard**

# The bad news

- Markov games generalize normal form games.

➡ Inherit *computational* intractability

[Daskalakis, Goldberg, Papadimitriou 06]
[Chen, Deng 06]
[Rubinstein 15]

**PPAD-hard**

**Specific classes of games?**

- *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 $V^{(\pi_1, \pi_2)}(\rho)$.

$p$

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

$q$

# 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*

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

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.

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

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

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

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

# 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

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

Remarks

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

# 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 $\Pi$ denotes projection on product of simplices.

# 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 $\Pi$ 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 $\epsilon$-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 $\epsilon$-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].