DeepMind

Multiagent Reinforcement Learning

Marc Lanctot

Joint work with many, many collaborators!

Workshop Plan

10:00 - 10:15	Workshop Intro
10:15 - 12:00	Introduction to Mulitagent Reinforcement Learning (MARL)
12:00 - 12:30	Break for Lunch
12:30 - 2:30	Adapting RL to Zero-Sum Games
2:30 - 3:00	Coffee Break
3:00 - 4:00	Practical Session: RL & Games with OpenSpiel

Joint work with many great collaborators!

Many, many great collaborators!

DeepMind

1 Workshop Intro

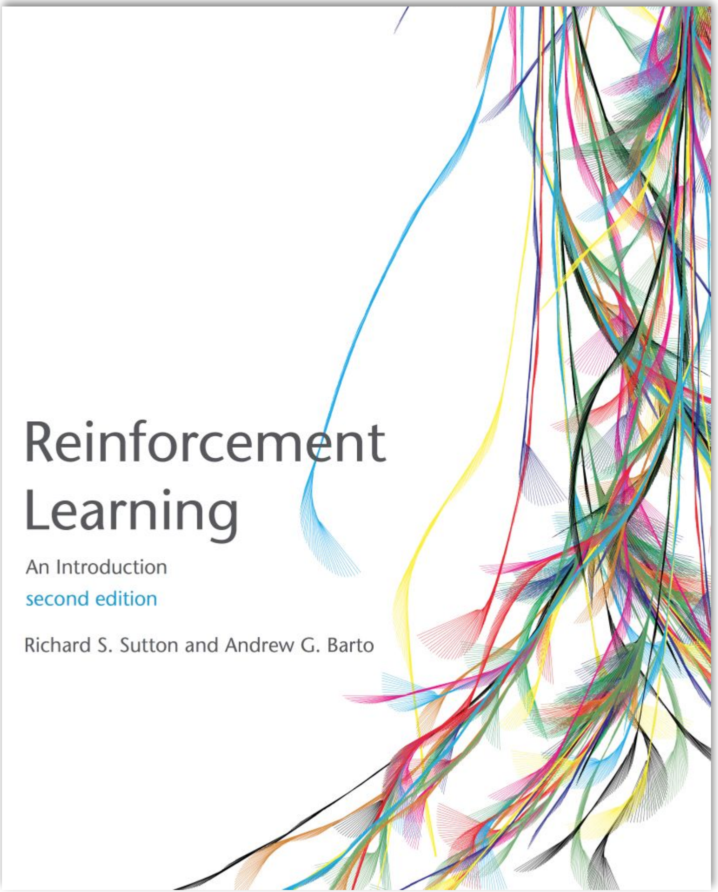
Reinforcement Learning

Reinforcement Learning:

An Introduction

Sutton & Barto '18

http://incompleteideas.net/book/the-book.html



Workshop Topics: Survey

Workshop Topics: Survey

- Unbiased estimator

Workshop Topics: Survey

- Unbiased estimator
- Importance sampling

Workshop Topics: Survey

- Unbiased estimator
- Importance sampling
- Markov decision process

Workshop Topics: Survey

- Unbiased estimator
- Importance sampling
- Markov decision process
- Nash equilibrium **or** Minimax Theorem

Workshop Topics: Survey

- Unbiased estimator
- Importance sampling
- Markov decision process
- Nash equilibrium **or** Minimax Theorem
- Regret Minimization
    - AKA No-regret learning, Hannan/universal consistency

Workshop Topics: Survey

- Unbiased estimator
- Importance sampling
- Markov decision process
- Nash equilibrium **or** Minimax Theorem
- Regret Minimization
    - AKA No-regret learning, Hannan/universal consistency
- Generalized Policy Iteration

Workshop Topics: Survey

- Unbiased estimator
- Importance sampling
- Markov decision process
- Nash equilibrium **or** Minimax Theorem
- Regret Minimization
  - AKA No-regret learning, Hannan/universal consistency
- Generalized Policy Iteration
- Reinforcement Learning (RL)

Workshop Topics: Survey

- Unbiased estimator
- Importance sampling
- Markov decision process
- Nash equilibrium **or** Minimax Theorem
- Regret Minimization
  - AKA No-regret learning, Hannan/universal consistency
- Generalized Policy Iteration
- Reinforcement Learning (RL)
- Difference between value-based RL and policy gradients

Workshop Topics: Survey

- Unbiased estimator
- Importance sampling
- Markov decision process
- Nash equilibrium **or** Minimax Theorem
- Regret Minimization
    - AKA No-regret learning, Hannan/universal consistency
- Generalized Policy Iteration
- Reinforcement Learning (RL)
- Difference between value-based RL and policy gradients
- Monte Carlo tree search **or** minimax search

Workshop Topics: Survey

- Unbiased estimator
- Importance sampling
- Markov decision process
- Nash equilibrium **or** Minimax Theorem
- Regret Minimization
    - AKA No-regret learning, Hannan/universal consistency
- Generalized Policy Iteration
- Reinforcement Learning (RL)
- Difference between value-based RL and policy gradients
- Monte Carlo tree search **or** minimax search
- Function approximation **or** neural network

Workshop Topics: Survey

- Unbiased estimator
- Importance sampling
- Markov decision process
- Nash equilibrium **or** Minimax Theorem
- Regret Minimization
  - AKA No-regret learning, Hannan/universal consistency
- Generalized Policy Iteration
- Reinforcement Learning (RL)
- Difference between value-based RL and policy gradients
- Monte Carlo tree search **or** minimax search
- Function approximation **or** neural network
- Proof by induction

Participate: Welcome Game & Research Topics

1. Let’s play a multiplayer game!
2. Research topic / interest survey

First rule: no Internet (wifi / cell phone / laptop etc.) for the next 5 min!

Game: Guess ⅔ of the Average

1. Write down a real number between 0 and 100.
2. Winner: closest value to ⅔ of the mean of all values

Ok to take a minute or so to decide… but no talking!

Research Topic / Interest Survey

1. Tell me what you do or are generally interested in.
2. .... in no more than 10 words!

Motivations: Research in Multiagent RL

	Single Agent	Multiple (e.g. 2) Agents
Large Problems	Approximate Solution Methods	Approximate Solution Methods
Small Problems	Tabular Solution Methods	Tabular Solution Methods

Motivations: Research in Multiagent RL

Sutton & Barto ‘98, ‘18

	Single Agent	Multiple (e.g. 2) Agents
Large Problems	Approximate Solution Methods	Approximate Solution Methods
Small Problems	Tabular Solution Methods	Tabular Solution Methods

Motivations: Research in Multiagent RL

First era of multiagent RL

	Single Agent	Multiple (e.g. 2) Agents
Large Problems	Approximate Solution Methods	Approximate Solution Methods
Small Problems	Tabular Solution Methods	Tabular Solution Methods

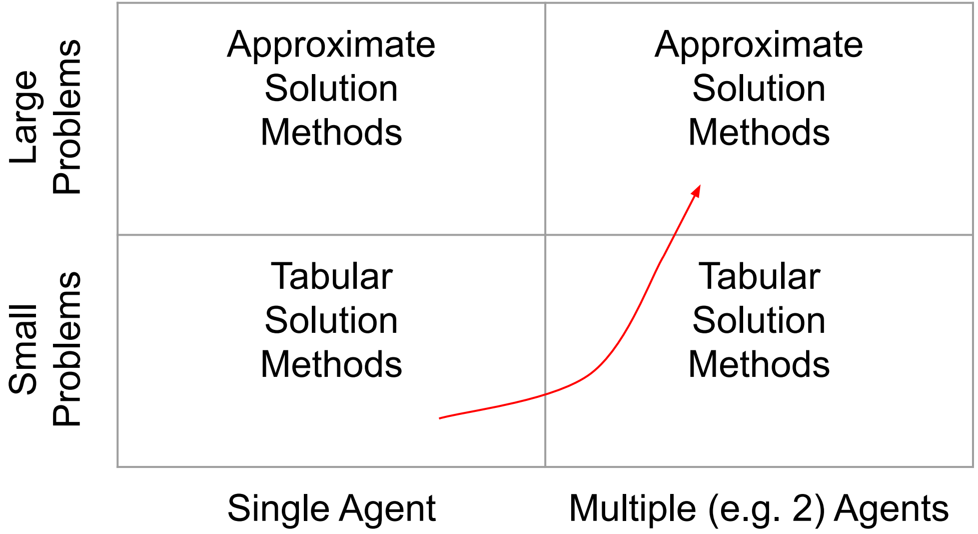
Motivations: Research in Multiagent RL

Multiagent Deep RL era ('16 - now)

	Single Agent	Multiple (e.g. 2) Agents
Large Problems	Approximate Solution Methods →	Approximate Solution Methods
Small Problems	Tabular Solution Methods	Tabular Solution Methods

Motivations: Research in Multiagent RL

Talk focus



Motivations: Research in Multiagent RL

My 10-year mission

	Single Agent	Multiple (e.g. 2) Agents
Large Problems	Approximate Solution Methods	Approximate Solution Methods
Small Problems	Tabular Solution Methods	Tabular Solution Methods

Biscuits vs Cookies

Brief note on Terminology

Games community	Reinforcement learning community
Player	Agent
Strategy	Policy
Best Response	Greedy Policy
Utility	Reward
State	(Information) State
Move	Action

DeepMind

2 Intro to MARL

Section Plan

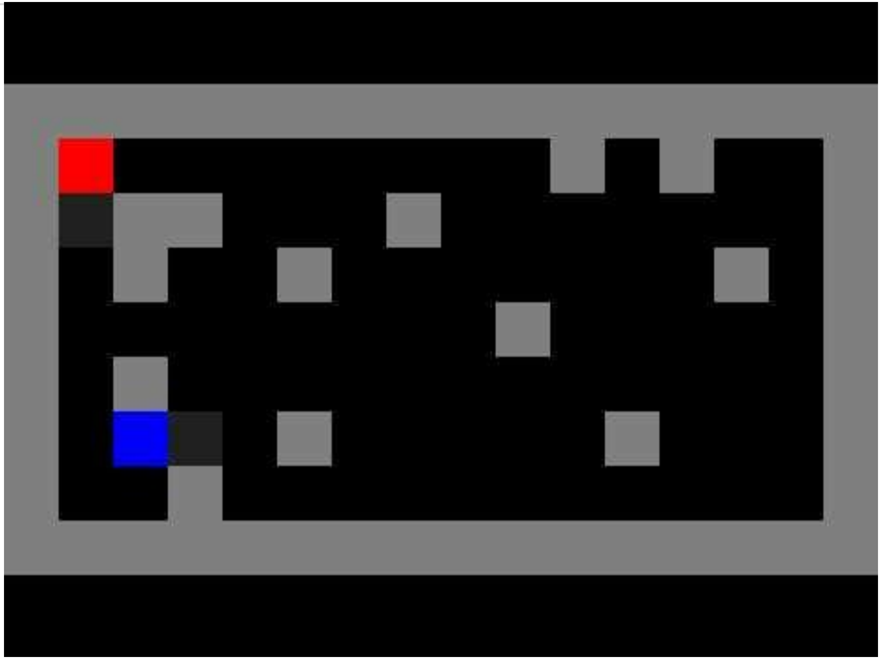
a. What is Multiagent Reinforcement Learning (MARL)?
b. Foundations & Background
c. Basic Formalisms & Algorithms
d. (Quick intro to) Advanced Topic
e. General MARL wrap-up

DeepMind

2a Intro to MARL

Multiagent Reinforcement Learning

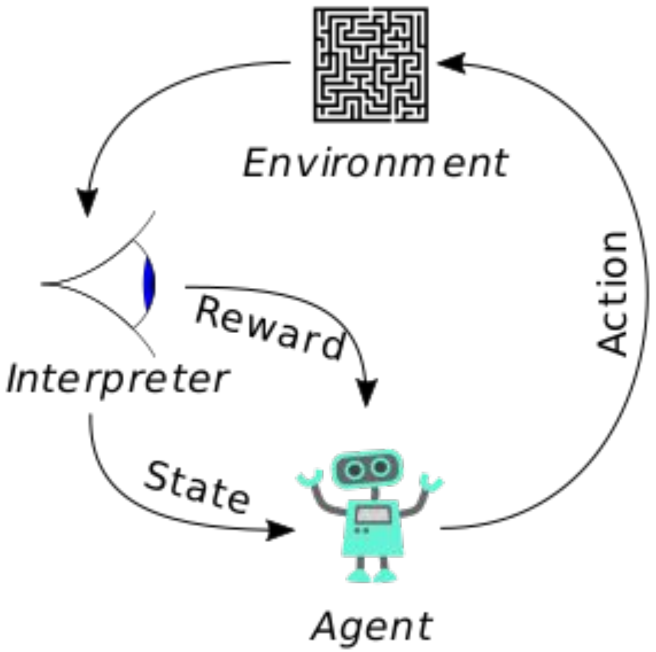
pommerman.com



Laser Tag

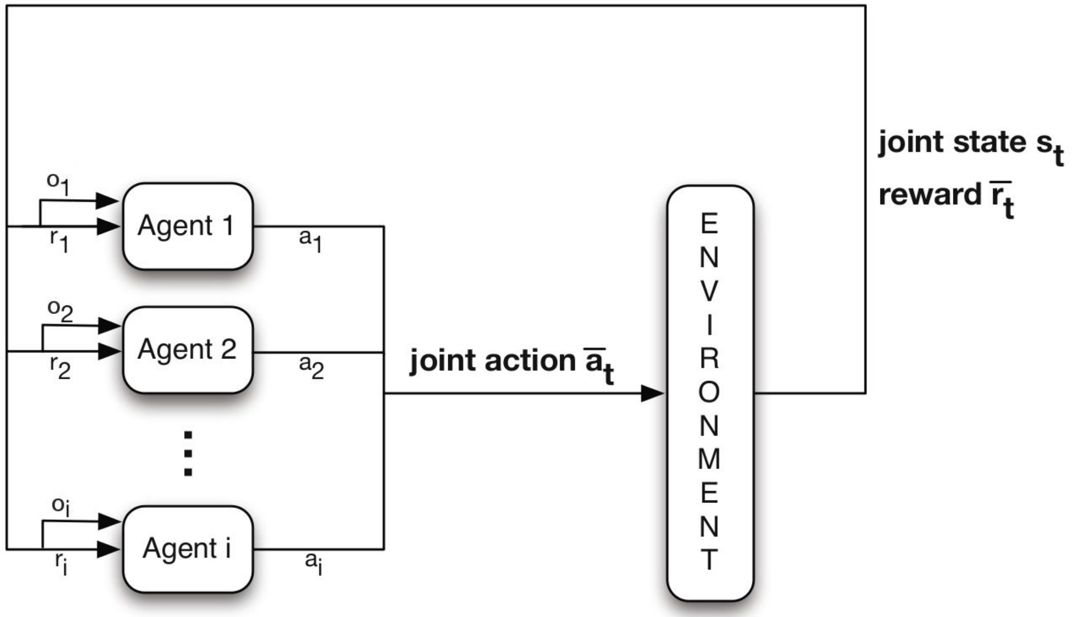
Multiagent Reinforcement Learning

Traditional (Single-Agent) RL



Source: Wikipedia

Multiagent Reinforcement Learning



Source: Nowe, Vrancx & De Hauwere 2012

Important Historical Note

If multi-agent learning is the answer,
what is the question?

Yoav Shoham, Rob Powers, and Trond Grenager
Stanford University
{shoham,powers,grenager}@cs.stanford.edu

February 15, 2006

Artificial Intelligence, Volume 171, Issue 7

Artificial Intelligence, Volume 171, Issue 7

Agendas for multi-agent learning

Geoffrey J. Gordon

Pages 392-401

Multiagent learning is not the answer. It is the question

Peter Stone

Pages 402-405

What evolutionary game theory tells us about multiagent learning

Karl Tuyls, Simon Parsons

Pages 406-416

Artificial Intelligence, Volume 171, Issue 7

Multi-agent learning and the descriptive value of simple models

Ido Erev, Alvin E. Roth

Pages 423-428

The possible and the impossible in multi-agent learning

H. Peyton Young

Pages 429-433

No regrets about no-regret

Yu-Han Chang

Pages 434-439

Artificial Intelligence, Volume 171, Issue 7

A hierarchy of prescriptive goals for multiagent learning

Martin Zinkevich, Amy Greenwald, Michael L. Littman

Pages 440-447

Learning equilibrium as a generalization of learning to optimize

Dov Monderer, Moshe Tennenholtz

Pages 448-452

Some Specific Axes of MARL

Centralized:

- One brain / algorithm deployed across many agents

Decentralized:

- All agents learn individually
- Communication limitations defined by environment

Some Specific Axes of MARL

Prescriptive:

- Suggests how agents *should* behave

Descriptive:

- Forecast how agent *will* behave

Some Specific Axes of MARL

Cooperative: Agents cooperate to achieve a goal

Competitive: Agents compete against each other

Neither: Agents maximize their utility which may require cooperating and/or competing

Our Focus

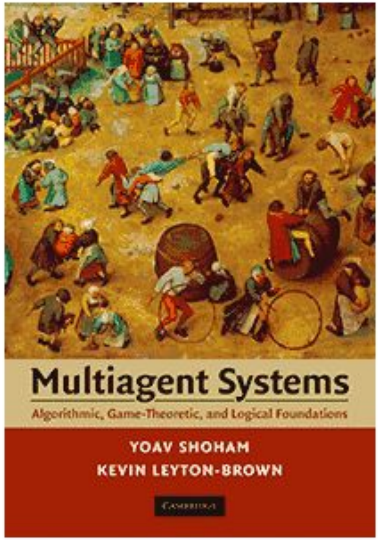
1. Centralized training for decentralized execution (very common)
2. Mostly prescriptive
3. Mostly competitive; sprinkle of cooperative and neither

DeepMind

2b Foundations & Background

Shoham & Leyton-Brown '09

Main Page Table of Contents Instructional Resources Errata eBook Download new!



Multiagent Systems
Algorithmic, Game-Theoretic, and Logical Foundations

Yoav Shoham
Stanford University
Kevin Leyton-Brown
University of British Columbia

Cambridge University Press, 2009
Order online: amazon.com

masfoundations.org

Foundations of (MA)RL

	Single Agent	Multiple (e.g. 2) Agents
Large Problems	Reinforcement Learning	Multiagent Reinforcement Learning
Small Problems	Approximate Dynamic Programming	Game Theory

Foundations of Multiagent RL

	Single Agent	Multiple (e.g. 2) Agents
Large Problems	Reinforcement Learning	Multiagent Reinforcement Learning
Small Problems	Approximate Dynamic Programming	**Game Theory**

Normal-form “One-Shot” Games

- Set of **players** $i \in \mathcal{N} = \{1, 2, \cdots, n\}$

Normal-form "One-Shot" Games

- Set of **players** $i \in \mathcal{N} = \{1, 2, \cdots, n\}$
- Each player has set of **actions** $\mathcal{A}_i \in \{a_1, a_2, \ldots\}$

Normal-form "One-Shot" Games

- Set of **players** $i \in \mathcal{N} = \{1, 2, \cdots, n\}$
- Each player has set of **actions** $\mathcal{A}_i \in \{a_1, a_2, \ldots\}$
- Set of **joint** actions $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$

Normal-form "One-Shot" Games

- Set of **players** $i \in \mathcal{N} = \{1, 2, \cdots, n\}$
- Each player has set of **actions** $\mathcal{A}_i \in \{a_1, a_2, \ldots\}$
- Set of **joint** actions $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$
- A **utility** function $u : \mathcal{N} \times \mathcal{A} \to U \subseteq \Re$

Example: (Bi-)Matrix Games (n = 2)

column player

		A	B
row player	a	0 , 0	1 , -1
	b	-1 , 1	0, 0

Example: (Bi-)Matrix Games (n = 2)

column player

actions

	A	B
a	0 , 0	1 , -1
b	-1 , 1	0, 0

row player

Example: (Bi-)Matrix Games (n = 2)

column player

row player		A	B
	a	0 , 0	1, -1
	b	-1 , 1	0, 0

utility to row player

Example: (Bi-)Matrix Games (n = 2)

column player

utility to column player

row player

	A	B
a	0 , 0	1 , -1
b	-1 , 1	0, 0

utility to row player

Example: (Bi-)Matrix Games (n = 2)

column player

row player

		A	B
row player	a	0 , 0	1 , -1
	b	-1 , 1	0, 0

utility to column player

utility to row player

for joint action (a,B)

Normal-form "One-Shot" Games

- Set of **players** $i \in \mathcal{N} = \{1, 2, \cdots, n\}$
- Each player has set of **actions** $\mathcal{A}_i \in \{a_1, a_2, \ldots\}$
- Set of **joint** actions $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$
- A **utility** function $u : \mathcal{N} \times \mathcal{A} \rightarrow U \subseteq \Re$

Each player: $\pi_i \in \Delta(\mathcal{A}_i)$, maximize $\mathbb{E}_{a \sim \pi}[u_i(a)]$

Normal-form "One-Shot" Games

- Set of **players** $i \in \mathcal{N} = \{1, 2, \cdots, n\}$
- Each player has set of **actions** $\mathcal{A}_i \in \{a_1, a_2, \ldots\}$
- Set of **joint** actions $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_n$
- A **utility** function $u : \mathcal{N} \times \mathcal{A} \rightarrow U \subseteq \Re$

Each player: $\pi_i \in \Delta(\mathcal{A}_i), \text{ maximize } \mathbb{E}_{a \sim \pi}[u_i(a)]$

Problem! This is a *joint* policy

Best Response

Suppose we are player i and we fix policies of other players

Best Response

Suppose we are player i and we fix policies of other players ($-i = \mathcal{N} - \{i\}$)

Best Response

Suppose we are player i and we fix policies of other players ($-i = \mathcal{N} - \{i\}$)

$$\pi_i \in \Delta(\mathcal{A}_i), \ \text{maximize} \ \mathbb{E}_{a \sim \pi}[u_i(a)]$$

Best Response

Suppose we are player i and we fix policies of other players ($-i = \mathcal{N} - \{i\}$)

$$\pi_i \in \Delta(\mathcal{A}_i), \ \ \text{maximize} \ \mathbb{E}_{a \sim \pi}[u_i(a)]$$

$$\pi_i \in BR(\pi_{-i}) \Leftrightarrow u_i(\pi_i, \pi_{-i}) = \max_{\pi_i'} \mathbb{E}_{a \sim (\pi_i', \pi_{-i})}[u_i(a)]$$

Best Response

Suppose we are player i and we fix policies of other players ($-i = \mathcal{N} - \{i\}$)

$$\pi_i \in \Delta(\mathcal{A}_i), \ \text{maximize}\ \mathbb{E}_{a\sim\pi}[u_i(a)]$$

$$\pi_i \in BR(\pi_{-i}) \Leftrightarrow u_i(\pi_i, \pi_{-i}) = \max_{\pi_i'} \mathbb{E}_{a\sim(\pi_i', \pi_{-i})}[u_i(a)]$$

π_i is a **best response** to π_{-i}

Solving a Matrix Game

column player

		A	B
row player	a	0 , 0	1 , -1
	b	-1 , 1	0, 0

Solving a Matrix Game

column player

		A	B
row player	a	0 , 0	1 , -1
	b	-1 , 1	0, 0

Let's start here

Solving a Matrix Game

column player

		A	B
row player	a	0 , 0	1 , -1
	b	-1 , 1	0, 0

Both players have *incentive to deviate*
(assuming the opponent stays fixed)

Solving a Matrix Game

column player

		A	B
row player	a	0 , 0	1 , -1
	b	-1 , 1	0, 0

Solving a Matrix Game

column player

row player

	A	B
a	0 , 0	1 , -1
b	-1 , 1	0, 0

Solving a Matrix Game

column player

row player

	A	B
a	0 , 0	1 , -1
b	-1 , 1	0, 0

(a,A) is a *fixed point* of this process

Solving a Matrix Game

column player

		A	B
row player	a	0 , 0	1 , -1
	b	-1 , 1	0, 0

(a,A) is a *fixed point* of this process

$$\pi_i \in \Delta(\mathcal{A}_i), \text{ maximize } \mathbb{E}_{a\sim\pi}[u_i(a)]$$

Let's Try Another….

column player

row player		A	B
	a	1 , -1	-1 , 1
	b	-1 , 1	1, -1

Let’s Try Another….

column player

row player

	A	B
a	1 , -1	-1 , 1
b	-1 , 1	1, -1

Nash equilibrium

A Nash equilibrium is a **joint policy** π such that no player has incentive to deviate *unilaterally*.

Nash equilibrium: A Solution Concept

A Nash equilibrium is a **joint policy** π such that no player has incentive to deviate *unilaterally*.

$$\forall i \in \mathcal{N}, \pi_i \in BR(\pi_{-i})$$

Some Facts

- Nash equilibrium always exists in finite games
- Computing a Nash eq. is PPAD-Complete
  - One solution is to focus on tractable subproblems
  - Another is to compute approximations
- Assumes players are (unbounded) rational
- Assumes knowledge:
  - Utility / value functions
  - Rationality assumption is common knowledge

Two-Player Zero-Sum Games

Matching Pennies: $u_1(\cdot) = -u_2(\cdot)$

column player

row player

	A	B
a	1 , -1	-1 , 1
b	-1 , 1	1, -1

Two-Player Zero-Sum Games

Matching Pennies: $u_1(\cdot) = -u_2(\cdot)$

$$\max V$$

column player

		A	B
row player	a	**1** , -1	**-1** , 1
	b	**-1** , 1	**1**, -1

Two-Player Zero-Sum Games

Matching Pennies: $u_1(\cdot) = -u_2(\cdot)$

$$\max V$$

$$\pi(a) - \pi(b) \geq V \quad \text{(vs. A)}$$

column player

row player

	A	B
a	1 , -1	-1 , 1
b	-1 , 1	1, -1

Two-Player Zero-Sum Games

Matching Pennies: $u_1(\cdot) = -u_2(\cdot)$

column player

row player

	A	B
a	**1** , -1	**-1** , 1
b	**-1** , 1	**1**, -1

$$\max V$$

$$\pi(a) - \pi(b) \geq V \quad \text{(vs. A)}$$

$$-\pi(a) + \pi(b) \geq V \quad \text{(vs. B)}$$

Two-Player Zero-Sum Games

Matching Pennies: $u_1(\cdot) = -u_2(\cdot)$

column player

row player

	A	B
a	1 , -1	-1 , 1
b	-1 , 1	1, -1

$$\max V$$

$$\pi(a) - \pi(b) \geq V \quad \text{(vs. A)}$$
$$-\pi(a) + \pi(b) \geq V \quad \text{(vs. B)}$$

$$\pi(a) + \pi(b) = 1$$

$$0 \leq \pi(a), \pi(b) \leq 1$$

Best Response Condition

For any (possibly stochastic) joint policy π_{-i} ,

There exists a **deterministic** best response:

$$\pi_i^b \in BR(\pi_{-i})$$

Best Response Condition

For any (possibly stochastic) joint policy π_{-i} ,

There exists a **deterministic** best response:

$$\pi_i^b \in BR(\pi_{-i})$$

Proof: Assume otherwise. The values of each deterministic policy (action) must be the same, by def. of BR. Then we can put full weight on any of them.

Two-Player Zero-Sum Games

Matching Pennies: $u_1(\cdot) = -u_2(\cdot)$

column player

row player

	A	B
a	**1** , -1	**-1** , 1
b	**-1** , 1	**1**, -1

$$\max V$$

$$\pi(a) - \pi(b) \geq V \quad \text{(vs. A)}$$
$$-\pi(a) + \pi(b) \geq V \quad \text{(vs. B)}$$

$$\pi(a) + \pi(b) = 1$$

$$0 \leq \pi(a), \pi(b) \leq 1$$

This is a Linear Program!

- Solvable in polynomial time (!)
  - Easy to apply off-the-shelf solvers
- Will find one solution
- Matching Pennies: $\pi(a) = \pi(b) = \frac{1}{2}, V = 0$

Minimax

John von Neumann 1928

Max-min: P1 looks for a π_1 such that

$$v_1 = \max_{\pi_1} \min_{\pi_2} u_1(\pi_1, \pi_2)$$

Min-max: P1 looks for a π_1 such that

$$v_1 = \min_{\pi_2} \max_{\pi_1} u_1(\pi_1, \pi_2)$$

In two-player, zero-sum these are the same!

---> The Minimax Theorem

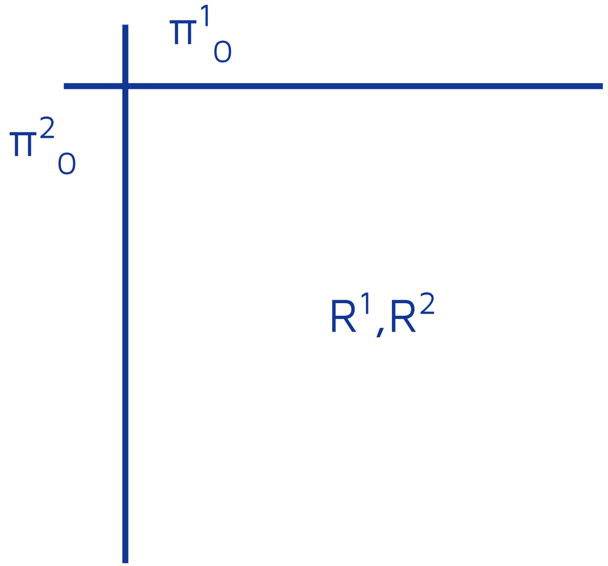
Consequences of Minimax

The optima $\pi^* = (\pi_1^*, \pi_2^*)$

- These exist! (They sometimes might be stochastic.)
- Called a minimax-optimal joint policy. Also, a Nash equilibrium.
- They are **interchangeable:**
- $\forall \pi^*, \pi^{*\prime} \Rightarrow (\pi_1^*, \pi_2^{*\prime}), (\pi_1^{*\prime}, \pi_2^*)$ also minimax-optimal
- Each policy is a best response to the other.

Normal Form Games: Algorithms

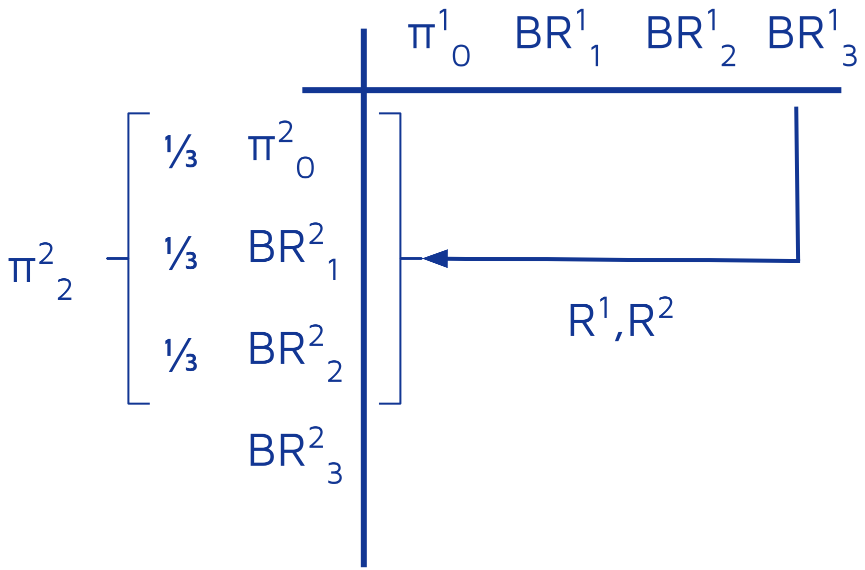
- Fictitious Play:



- Start with an arbitrary policy per player (π^1_0,π^2_0),

Normal Form Games: Algorithms

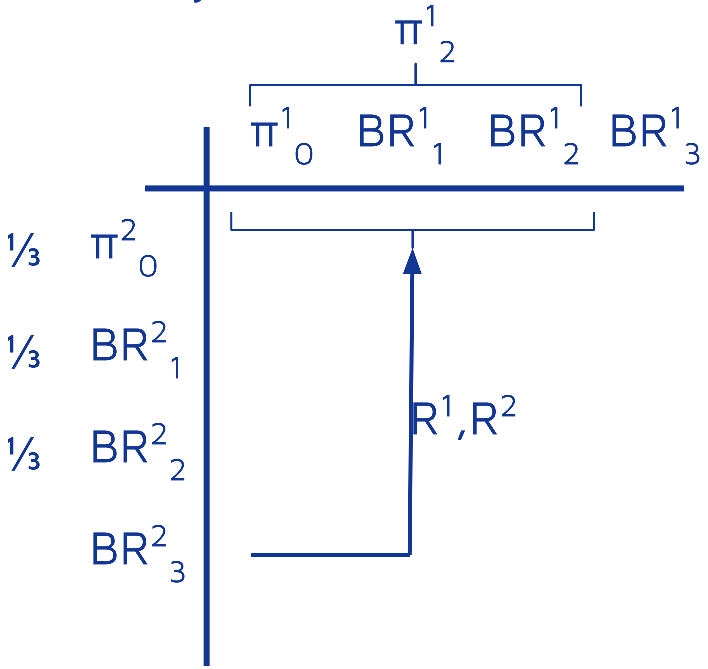
- Fictitious Play:



- Start with an arbitrary policy per player (π^1_0,π^2_0),
  - Then, play best response against a uniform distribution over the past policy of the opponent (BR^1_n,BR^2_n).

Normal Form Games: Algorithms

- Fictitious Play:



- Start with an arbitrary policy per player (π^1_0,π^2_0),
  - Then, play best response against a uniform distribution over the past policy of the opponent (BR^1_n,BR^2_n).

Normal Form Games: Algorithms

- Fictitious Play:
- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)

	R
R	0

Normal Form Games: Algorithms

- Fictitious Play:

	R	P
R	0	1
P	-1	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - (½, ½, 0), (½, ½, 0)

Normal Form Games: Algorithms

- Fictitious Play:

	R	P	P
R	0	1	1
P	-1	0	0
P	-1	0	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - (½, ½, 0), (½, ½, 0)
- Iteration 2:
  - BR^1_2, BR^2_2 = P, P
  - (⅓, ⅔, 0), (⅓, ⅔, 0)

Normal Form Games: Algorithms

- Fictitious Play:

	R	P	P	S
R	0	1	1	-1
P	-1	0	0	1
P	-1	0	0	1
S	1	-1	-1	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - (½, ½, 0), (½, ½, 0)
- Iteration 2:
  - BR^1_2, BR^2_2 = P, P
  - (⅓, ⅔, 0), (⅓, ⅔, 0)
- Iteration 3:
  - BR^1_3, BR^2_3 = S, S
  - (¼,½,¼), (¼,½,¼)

Normal Form Games: Algorithms

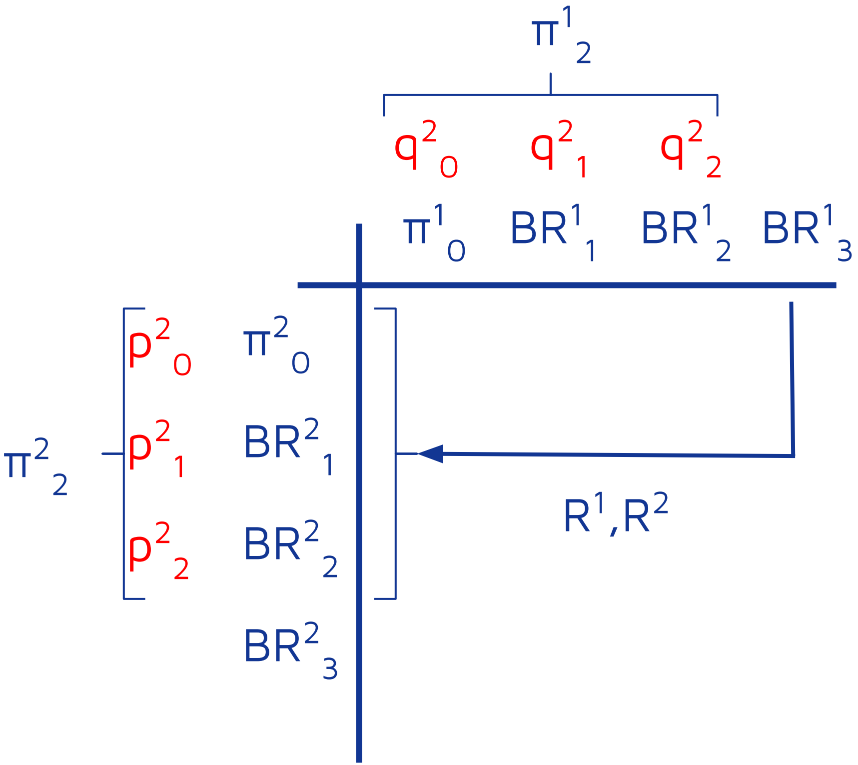
- Fictitious Play:

	R	P	P	S	S
R	0	1	1	-1	-1
P	-1	0	0	1	1
P	-1	0	0	1	1
S	1	-1	-1	0	0
S	1	-1	-1	0	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - (½, ½, 0), (½, ½, 0)
- Iteration 2:
  - BR^1_2, BR^2_2 = P, P
  - (⅓, ⅔, 0), (⅓, ⅔, 0)
- Iteration 3:
  - BR^1_3, BR^2_3 = S, S
  - (¼,½,¼), (¼,½,¼)

Normal Form Games: Algorithms

- double oracle [HB McMahan 2003]:



- Start with an arbitrary policy per player (π^1_0,π^2_0),
  - Compute (p^n,q^n) by solving the game at iteration n
  - Then, best response against (p^n,q^n) and get a new best response (BR^1_n,BR^1_n).

Normal Form Games: Algorithms

- double oracle:
- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)

	R
R	0

Normal Form Games: Algorithms

- double oracle:

	R	P
R	0	1
P	-1	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - Solve the game : (0, 1, 0), (0, 1, 0)

Normal Form Games: Algorithms

- double oracle:

	R	P	S
R	0	1	-1
P	-1	0	1
S	1	-1	0

- Start with (R, P, S)= (1, 0, 0), (1, 0, 0)
- Iteration 1:
  - BR^1_1, BR^2_1 = P, P
  - Solve the game : (0, 1, 0), (0, 1, 0)
- Iteration 2:
  - BR^1_2, BR^2_2 = S, S
  - (⅓, ⅓, ⅓), (⅓, ⅓, ⅓)

Cooperative Games

$$u_i(\cdot) = u_j(\cdot)$$

column player

row player

	A	B	C
a	1, 1	0, 0	0, 0
b	0, 0	2, 2	0, 0
c	0, 0	0, 0	5, 5

Cooperative Games

$$u_i(\cdot) = u_j(\cdot)$$

column player

row player

	A	B	C
a	1, 1	0, 0	0, 0
b	0, 0	2, 2	0, 0
c	0, 0	0, 0	5, 5

These are all Nash equilibria!

General-Sum Games

No constraints on utilities!

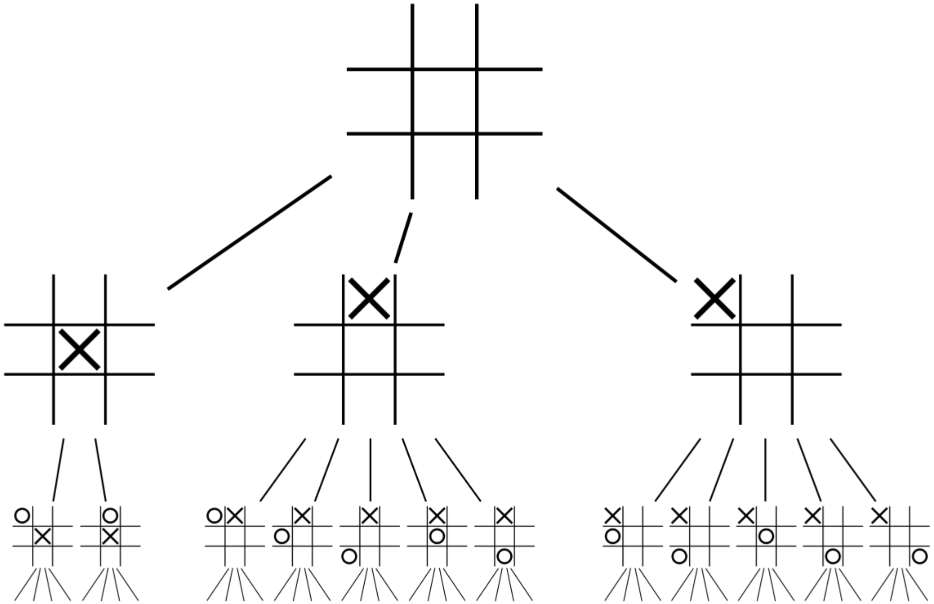
column player

		A	B
row player	a	3, 2	0, 0
	b	0, 0	2, 3

Sequential Setting: Extensive-Form Games

What about sequential games…?

Perfect Information Games



(Finite) Perfect Information Games: Model

- Start with an *episodic* MDP
- Add a **player identity** function:

$$\tau(s) \in \mathcal{N} \cup \{s\}$$

Simultaneous move node (many players play simultaneously)

- Define rewards *per player*:

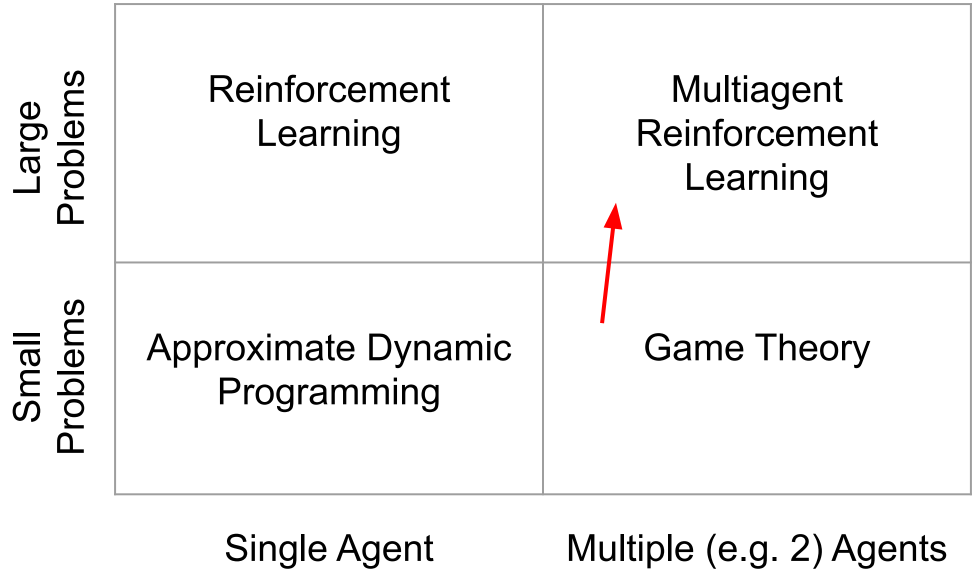
$$r_i(s, a, s') \text{ for } i \in \mathcal{N}$$

- (Similarly for returns: $G_{t,i}$ is the return *to* player i from s_t)

DeepMind

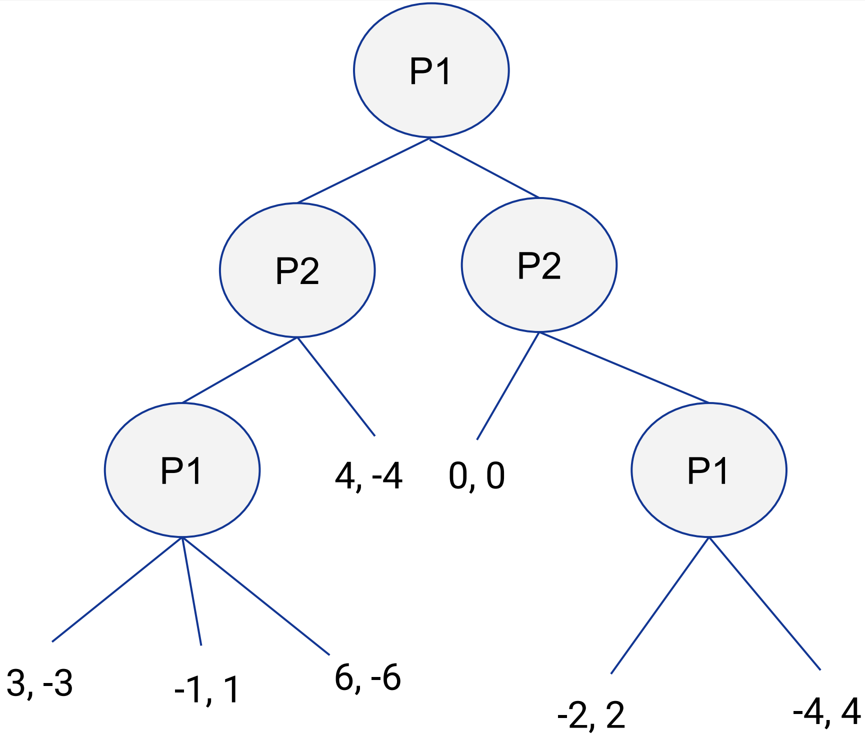
2c Basic Formalisms & Algorithms

Foundations of RL



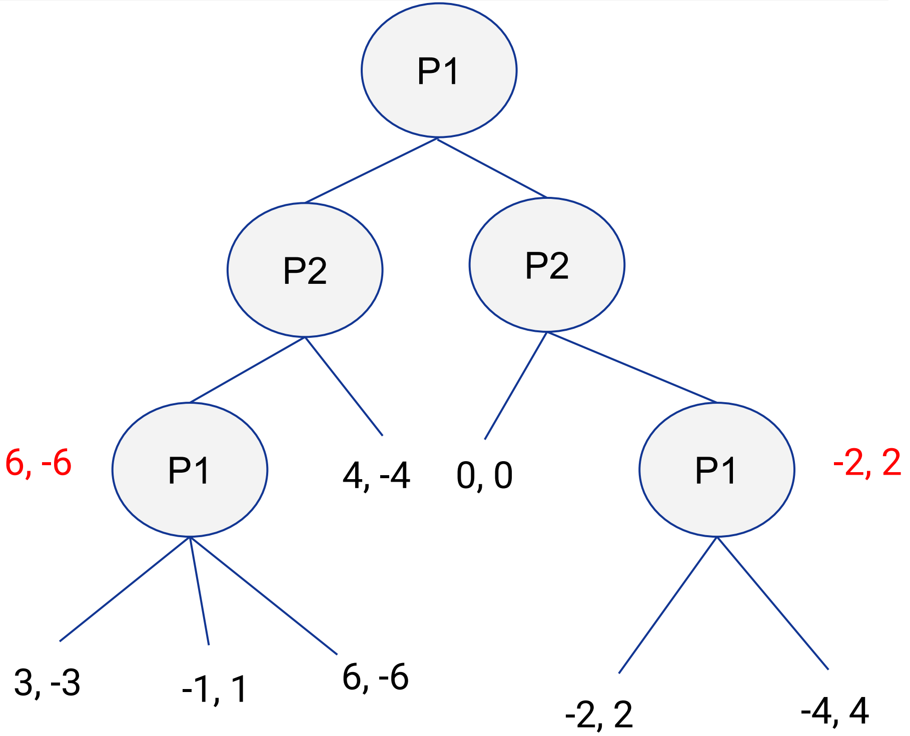
Backward Induction

Solving a *turn-taking* perfect information game



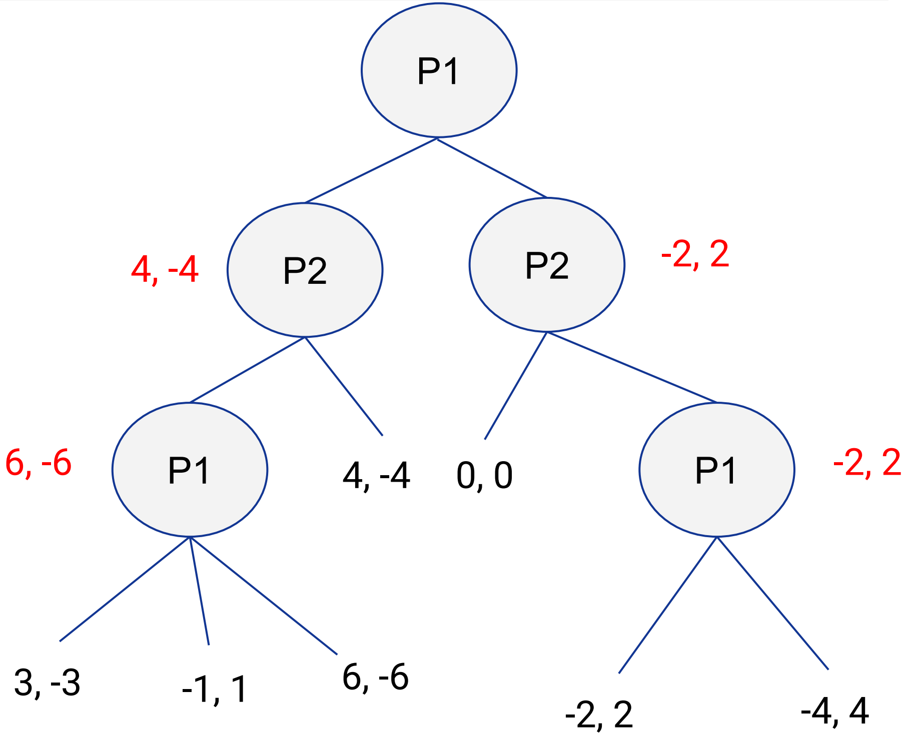
Backward Induction

Solving a *turn-taking* perfect information game



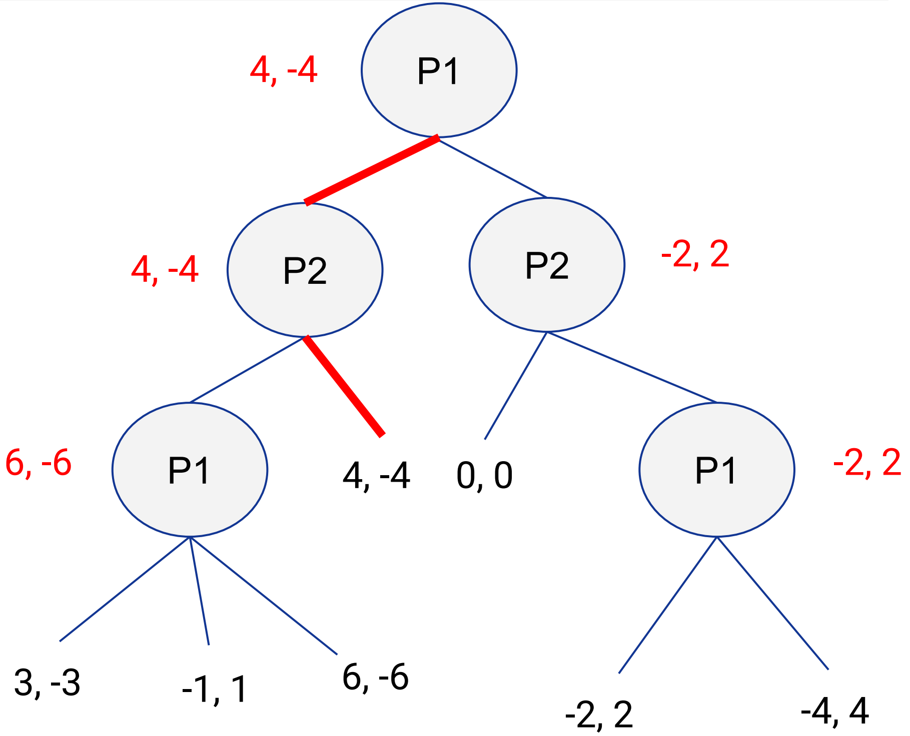
Backward Induction

Solving a *turn-taking* perfect information game



Backward Induction

Solving a *turn-taking* perfect information game



Intro to RL: Tabular Approximate Dyn. Prog.

Value iteration

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

Turn-Taking 2P Zero-sum Perfect Info. Games

- Player to play at s: $\tau(s)$
- Reward to player i: r_i
- Subset of legal actions $\textsc{LegalActions}(s)$
- Often assume episodic and $\gamma = 1$

Values of a state **to player i**: $V_i(s)$

Identities:

$$\forall s, a, s' : r_1 = -r_2, \quad V_1(s) = -V_2(s)$$

2P Zero-Sum Perfect Info. Value Iteration

Value iteration

Initialize array V_i arbitrarily (e.g., $V_i(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$: *Let $i = t(s)$*
    $v \leftarrow V_i(s)$
    $V_i(s) \leftarrow \max_a \sum_{s',r_i} p(s', r_i | s, a)\left[r_i + \gamma V_i(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V_i(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, $\pi \approx \pi_*$, such that *$i = t(s)$*
  $\pi(s) = \arg\max_a \sum_{s',r_i} p(s', r_i | s, a)\left[r_i + \gamma V_i(s')\right]$

Minimax

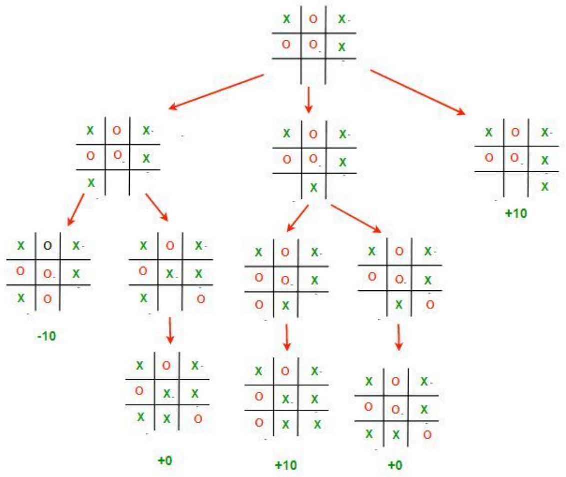
A.K.A. Alpha-Beta, Backward Induction, Retrograde Analysis, etc…

Start from search state s,

Compute a depth-limited approximation:

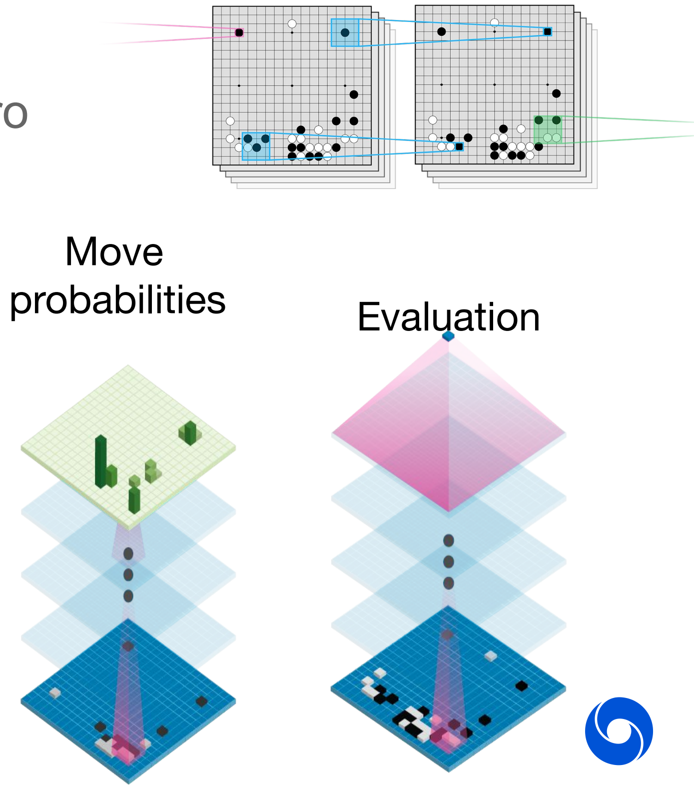
$$V_{i,d}(s) = \begin{cases} u_i(s) & \text{if } s \text{ is terminal,} \\ h_i(s) & \text{if } d = 0, \\ \sum_{s'} p(s,a,s') V_{i,d-1}(s') & \text{otherwise.} \end{cases}$$

---> Minimax Search



Two-Player Zero-Sum Policy Iteration

- Analogous to adaptation of value iteration
- Foundation of AlphaGo, AlphaGo Zero, AlphaZero
    - Better policy improvement via MCTS
    - Deep network func. approximation
        - Policy prior cuts down *breadth*
        - Value network cuts the *depth*



2P Zero-Sum Games with Simultaneous Moves

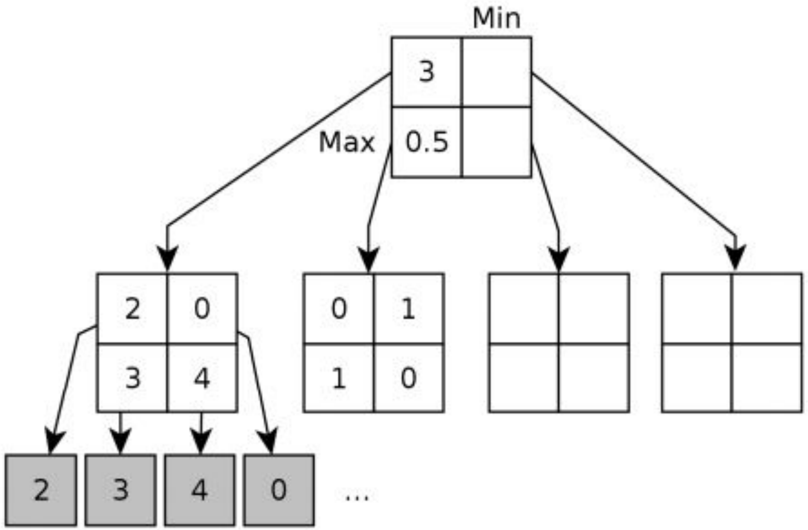


Image from Bozansky et al. 2016

Markov Games

"Markov Soccer"

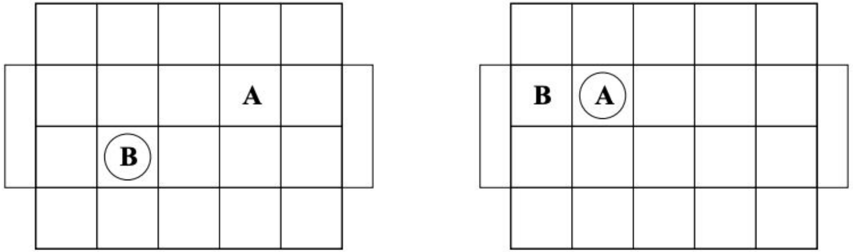
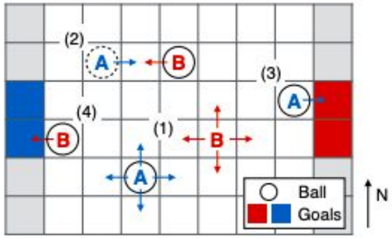


Figure 2: An initial board (left) and a situation requiring a probabilistic choice for A (right).



	Defensive	Offensive
w/ ball	Avoid opponent	Advance to goal
w/o ball	Defend goal	Intercept the ball

Figure 3. Left: Illustration of the soccer game. *Right:* Strategies of the hand-crafted rule-based agent.

Littman '94

He et al. '16

Also: Lagoudakis & Parr '02, Uther & Veloso '03, Collins '07

Value Iteration for Zero-Sum Markov Games

Value iteration

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow$ ~~$\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$~~ $\min_{\pi_2(s)} \max_{\pi_1(s)} \mathbb{E}_{a \sim \pi(s), s'}[r_1(s,a,s') + \gamma V_1(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a ~~deterministic~~ policy, $\pi \approx \pi_*$, ~~such that~~ computed above
  ~~$\pi(s) = \arg\max_a \sum_{s',r} p(s',r|s,a)[r + \gamma V(s')]$~~

First MARL Algorithm: Minimax-Q (Littman '94)

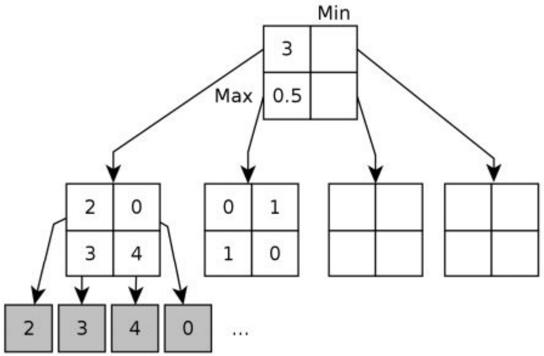
1. Start with arbitrary joint value functions $q(s, a, o)$

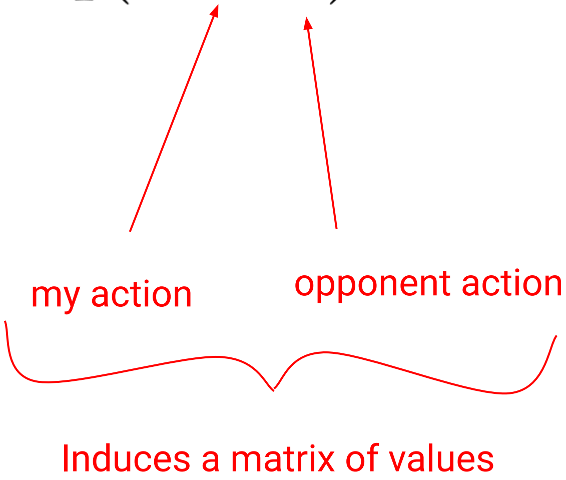
my action → a

opponent action → o

First MARL Algorithm: Minimax-Q (Littman '94)

1. Start with arbitrary joint value functions $q(s, a, o)$





First MARL Algorithm: Minimax-Q (Littman '94)

1. Start with arbitrary joint value functions $q(s, a, o)$
2. Define policy π as in value iteration (by solving an LP)

First MARL Algorithm: Minimax-Q (Littman '94)

1. Start with arbitrary joint value functions $q(s, a, o)$
2. Define policy π as in value iteration (by solving an LP)
3. Generate trajectories of tuple (s, a, o, s') using behavior policy $\pi' = \epsilon \textsc{Unif}(\mathcal{A}) + (1 - \epsilon)\pi$

First MARL Algorithm: Minimax-Q (Littman '94)

1. Start with arbitrary joint value functions $q(s, a, o)$
2. Define policy π as in value iteration (by solving an LP)
3. Generate trajectories of tuple (s, a, o, s') using behavior policy $\pi' = \epsilon \textsc{Unif}(\mathcal{A}) + (1 - \epsilon)\pi$
4. Update $q(s, a, o) = (1 - \alpha)q(s, a, o) + \alpha(r(s, a, o, s') + \gamma v(s'))$

First Era of MARL

Follow-ups to Minimax Q:

- Friend-or-Foe Q-Learning (Littman ‘01)
- Correlated Q-learning (Greenwald & Hall ‘03)
- Nash Q-learning (Hu & Wellman ‘03)
- Coco-Q (Sodomka et al. ‘13)

Function approximation:

- LSPI for Markov Games (Lagoudakis & Parr ‘02)

First Era of MARL

Nash Convergence of Gradient Dynamics in General-Sum Games

Satinder Singh
AT&T Labs
Florham Park, NJ 07932
baveja@research.att.com

Michael Kearns
AT&T Labs
Florham Park, NJ 07932
mkearns@research.att.com

Yishay Mansour
Tel Aviv University
Tel Aviv, Israel
mansour@math.tau.ac.il

Singh, Kearns & Mansour '03, Infinitesimal Gradient Ascent (IGA)

First Era of MARL

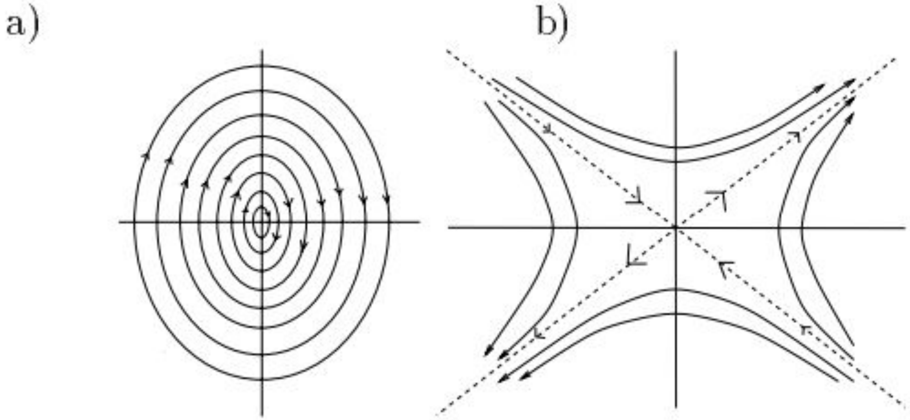


Figure 1: The general form of the dynamics: a) when U has imaginary eigenvalues and b) when U has real eigen-values.

Image from Singh, Kearns, & Mansour '03

Formalize optimization as a dynamical system:

policy gradients

Analyze using well-established techniques

First Era of MARL

→ Evolutionary Game Theory: **replicator dynamics**

$$\dot{\pi}_t(a) = \pi_t(a)\left[u(a, \boldsymbol{\pi}_t) - \bar{u}(\boldsymbol{\pi}_t)\right]$$

time derivative

First Era of MARL

→ Evolutionary Game Theory: **replicator dynamics**

$$\dot{\pi}_t(a) = \pi_t(a)\left[u(a, \boldsymbol{\pi}_t) - \bar{u}(\boldsymbol{\pi}_t)\right]$$

time derivative

utility of action a against
the joint policy / population
of other players

First Era of MARL

→ Evolutionary Game Theory: **replicator dynamics**

$$\dot{\pi}_t(a) = \pi_t(a)\left[u(a, \boldsymbol{\pi}_t) - \bar{u}(\boldsymbol{\pi}_t)\right]$$

time derivative

utility of action a against the joint policy / population of other players

Expected / average utility of the joint policy / population

First Era of MARL

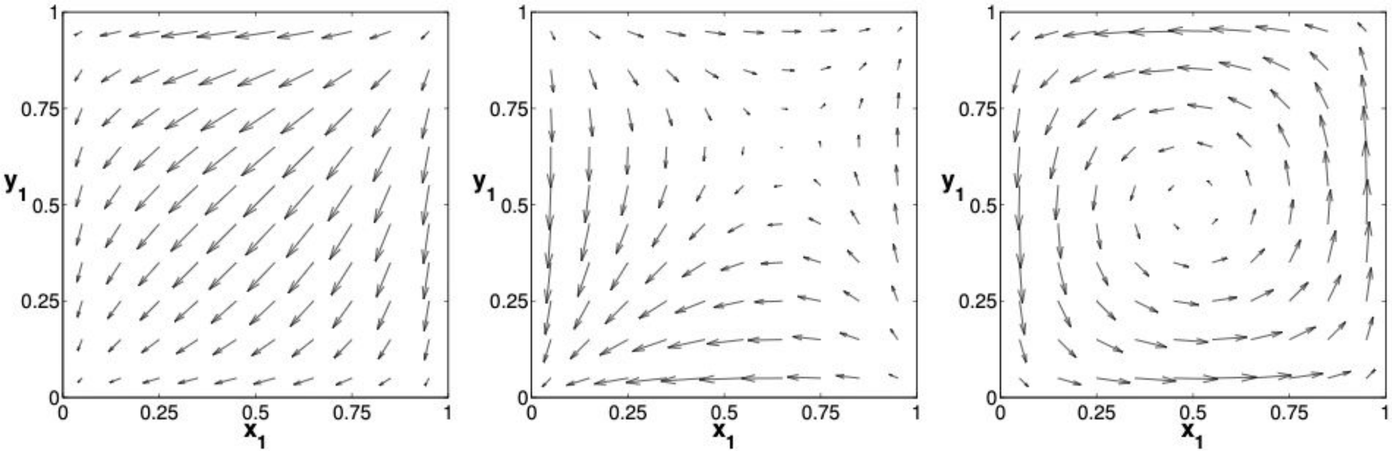


Figure 4: The replicator dynamics, plotted in the unit simplex, for the prisoner's dilemma (left), the stag hunt (center), and matching pennies (right).

[Bloembergen et al. 2015]

First Era of MARL

WoLF: Win or Learn Fast. (Bowling & Veloso ‘01).

IGA is **rational** but not **convergent**!

- *Rational*: opponents converge to a fixed joint policy

  → learning agent converges to a best response of joint policy
- *Convergent*: learner necessarily converges to a fixed policy

Use specific *variable learning rate* to ensure convergence (in 2x2 games)

First Era of MARL

Follow-ups to policy gradient and replicator dynamics:

- WoLF-IGA, WoLF-PHC
- WoLF-GIGA (Bowling ‘05)
- Weighted Policy Learner (Abdallah & Lesser ‘08)
- Infinitesimal Q-learning (Wunder et al. ‘10)
- Frequency-Adjusted Q-Learning (Kaisers et al. ‘10, Bloembergen et al. ‘11)
- Policy Gradient Ascent with Policy Prediction (Zhang & Lesser ‘10)
- Evolutionary Dynamics of Multiagent Learning (Bloembergen et al. ‘15)

So……

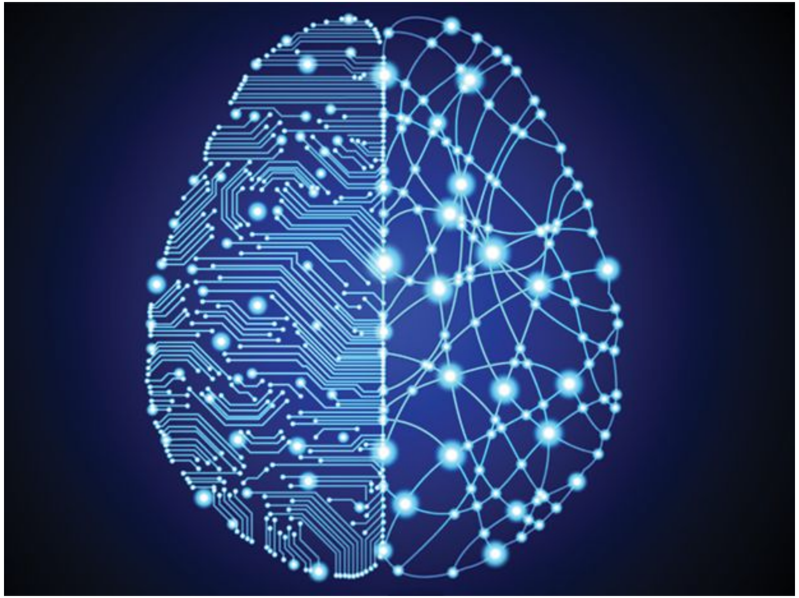
Why call it “the first era”?

So……

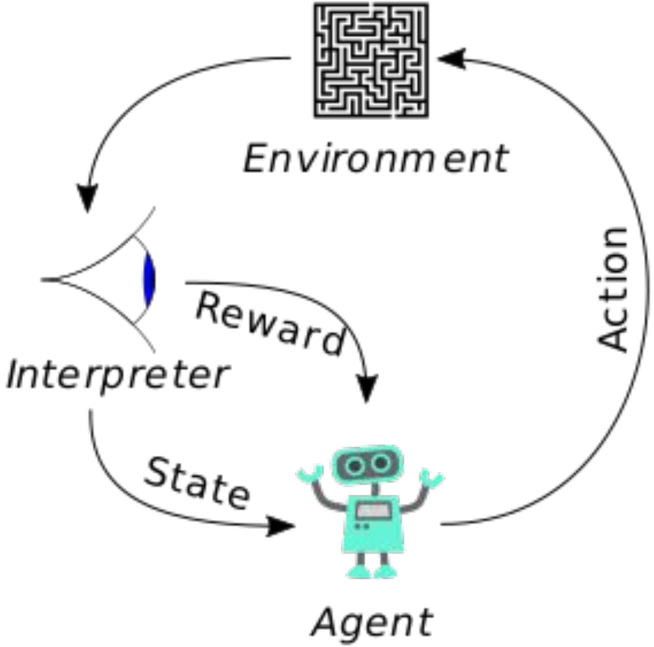
Why call it "the first era"?

Scalability was a major problem.

Second Era: Deep Learning meets Multiagent RL



Source: spectrum.ieee.org

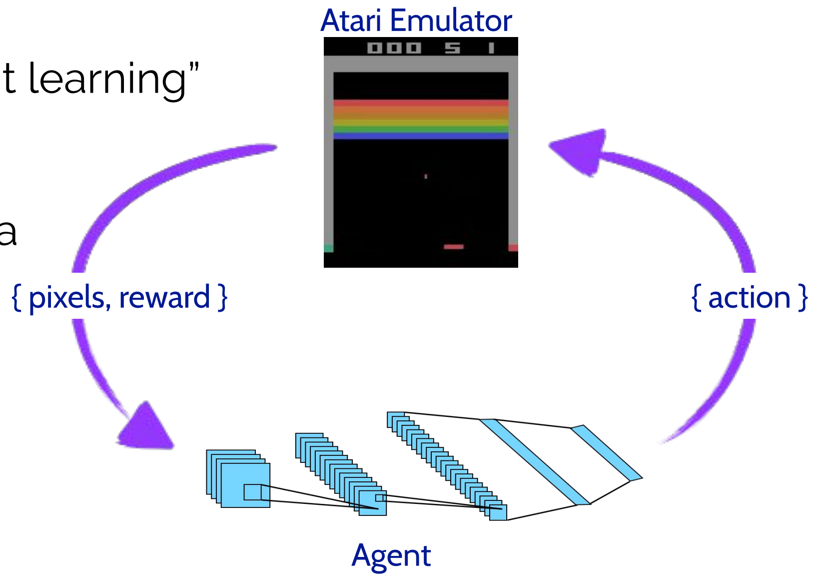


Source: wikipedia.org

Deep Q-Networks (DQN) Mnih et al. 2015

"Human-level control through deep reinforcement learning"

- Represent the action value (Q) function using a convolutional neural network.
- Train using end-to-end Q-learning.
- Can we do this in a stable way?

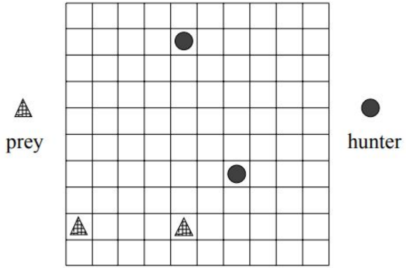


Independent Q-Learning Approaches

Independent Q-learning [Tan, 1993]

$$Q(x,a) \leftarrow Q(x,a) + \beta(r + \gamma V(y) - Q(x,a))$$

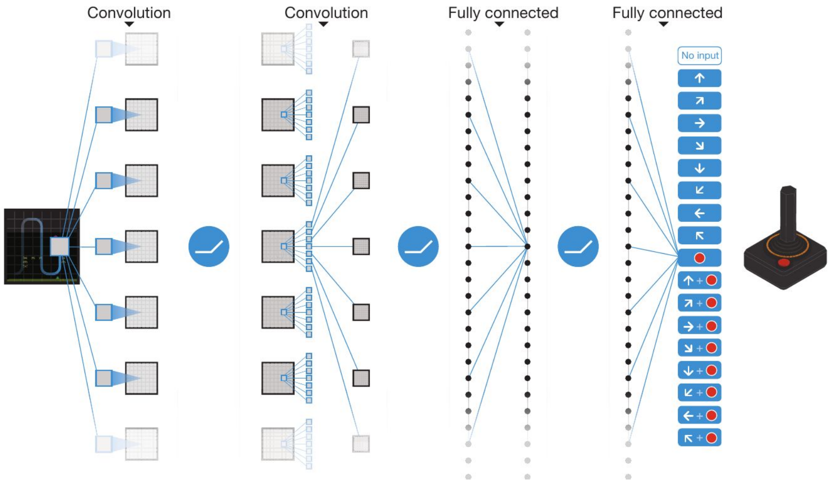
$$V(x) = \max_{b \in actions} Q(x,b)$$

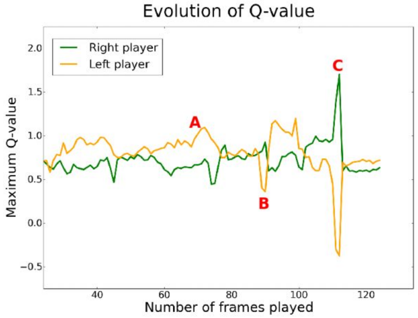


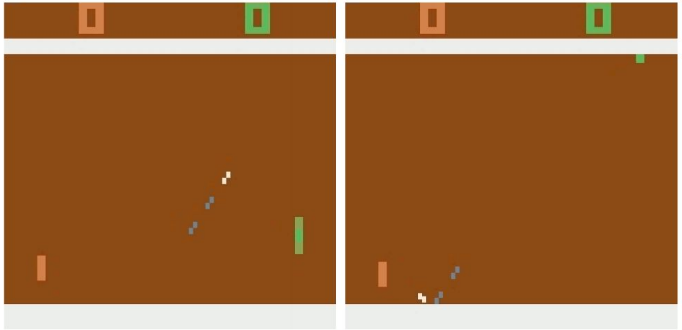
N-of-prey/N-of-hunters	1/1	1/2
Random hunters	123.08	56.47
Learning hunters	25.32	12.21

Table 1: Average Number of Steps to Capture a Prey

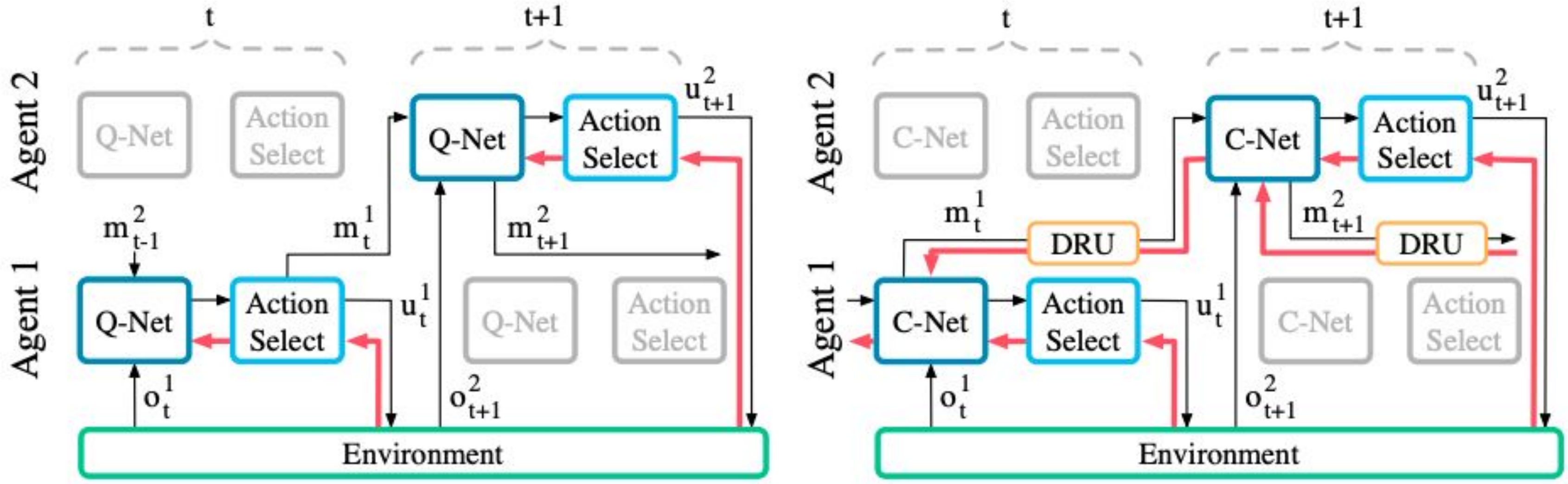
Independent Deep Q-Networks [Tampuu et al., 2015]







Learning to Communicate

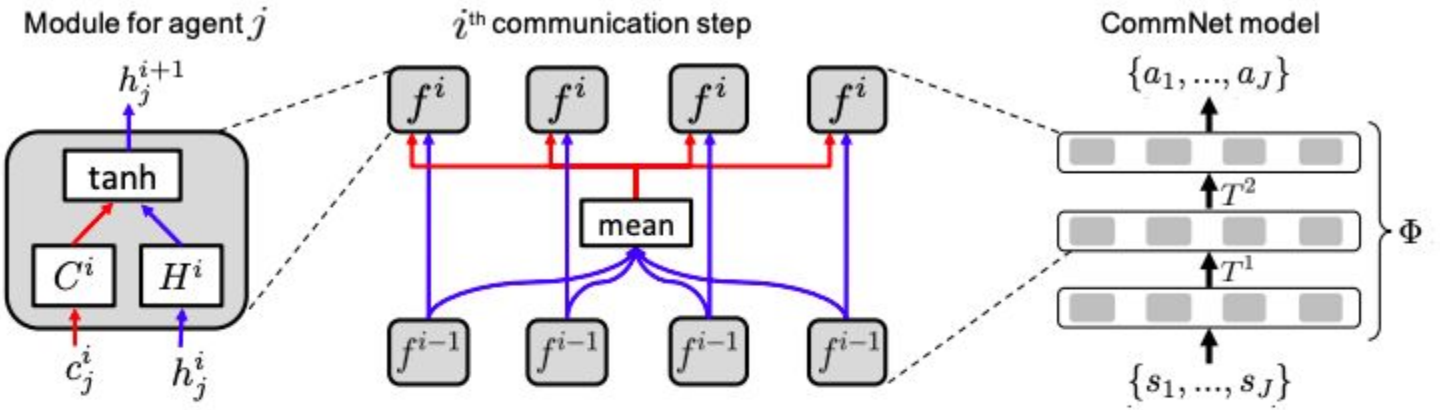


(a) RIAL - RL based communication

(b) DIAL - Differentiable communication

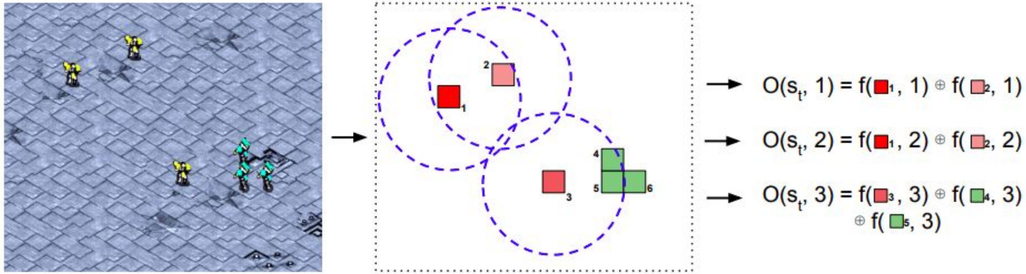
Foerster et al. ‘16

Learning to Communicate



Sukhbaatar et al. '16

Cooperative Multiagent Tasks

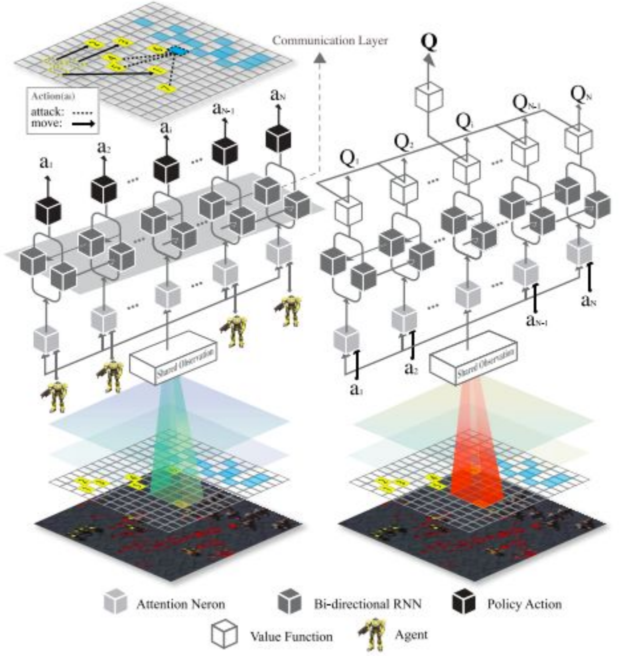


Foerster et al. ‘18

Episodic Exploration for Deep Deterministic Policies:
An Application to StarCraft Micromanagement Tasks

Nicolas Usunier*, Gabriel Synnaeve*, Zeming Lin, Soumith Chintala
Facebook AI Research
usunier,gab,zlin,soumith@fb.com

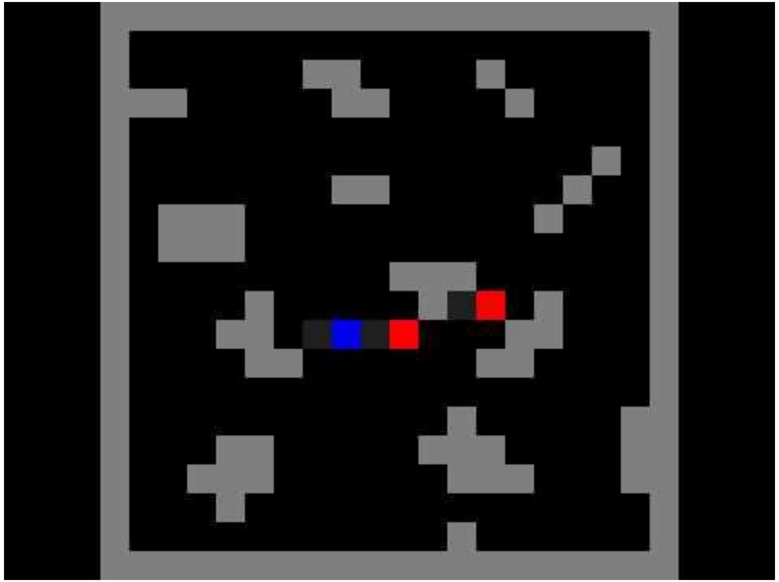
November 29, 2016



(a) Multiagent policy networks (b) Multiagent Q networks

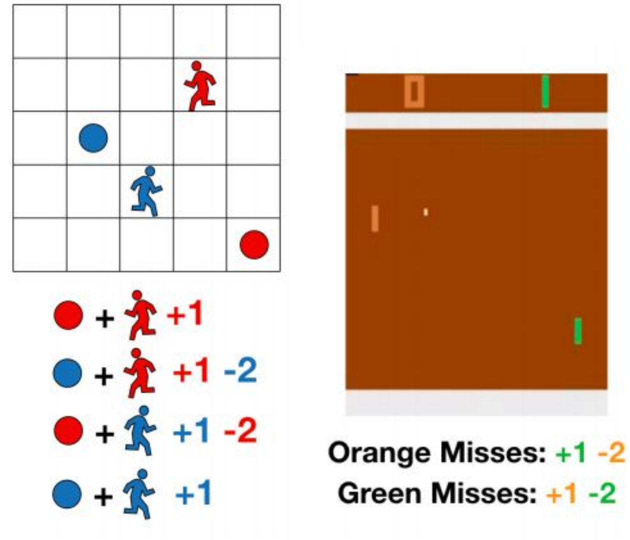
BIC-Net (Peng et al.’17)

Sequential Social Dilemmas



https://www.youtube.com/watch?v=0kaIqz6AvwE

Leibo et al. ‘17



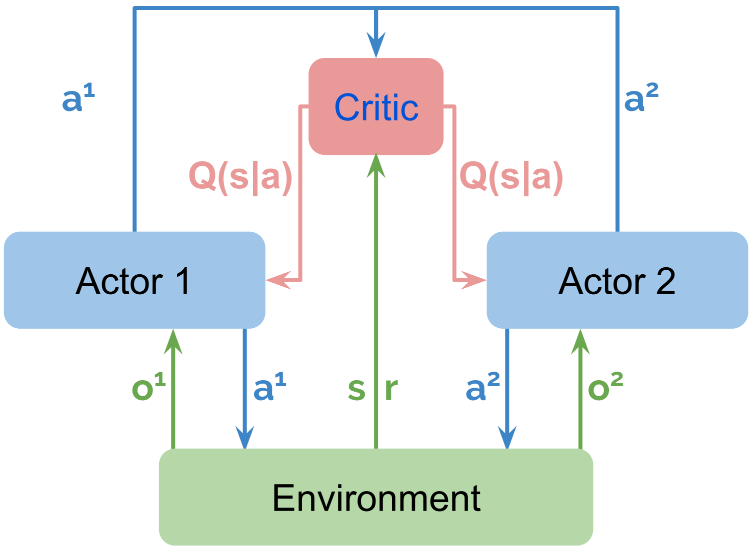
(a) Coins

(b) PPD

Lerer & Peyskavich ‘18

Centralized Critic Decentralized Actor Approaches

- **Idea:** reduce nonstationarity & credit assignment issues using a central critic
- **Examples:** MADDPG [Lowe et al., 2017] & COMA [Foerster et al., 2017]
- Apply to both cooperative and competitive games



Centralized critic trained to minimize loss:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{\mathbf{x},a,r,\mathbf{x}'}[(Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, \ldots, a_N) - y)^2],$$

$$y = r_i + \gamma Q_i^{\boldsymbol{\pi}'}(\mathbf{x}', a_1', \ldots, a_N')\big|_{a_j' = \boldsymbol{\pi}_j'(o_j)}$$

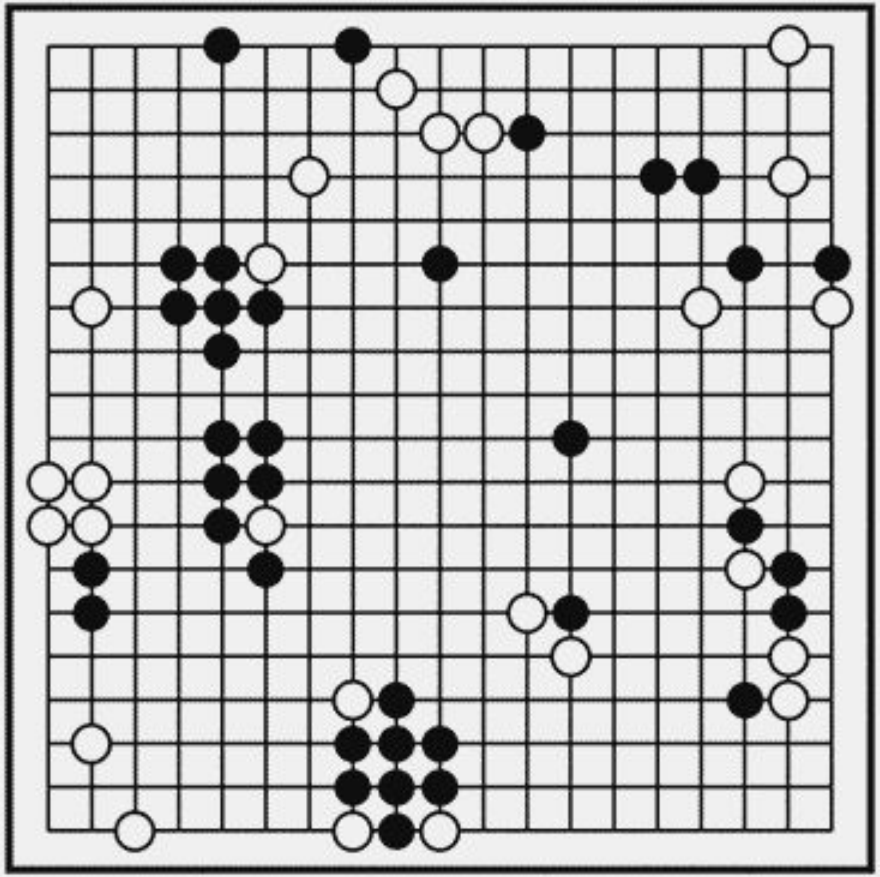
Decentralized actors trained via policy gradient:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s \sim p^{\boldsymbol{\mu}}, a_i \sim \boldsymbol{\pi}_i}[\nabla_{\theta_i} \log \boldsymbol{\pi}_i(a_i|o_i) Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, \ldots, a_N)]$$

Actor ($\nabla_{\theta_i} \log \boldsymbol{\pi}_i(a_i|o_i)$)

Critic ($Q_i^{\boldsymbol{\pi}}(\mathbf{x}, a_1, \ldots, a_N)$)

AlphaGo



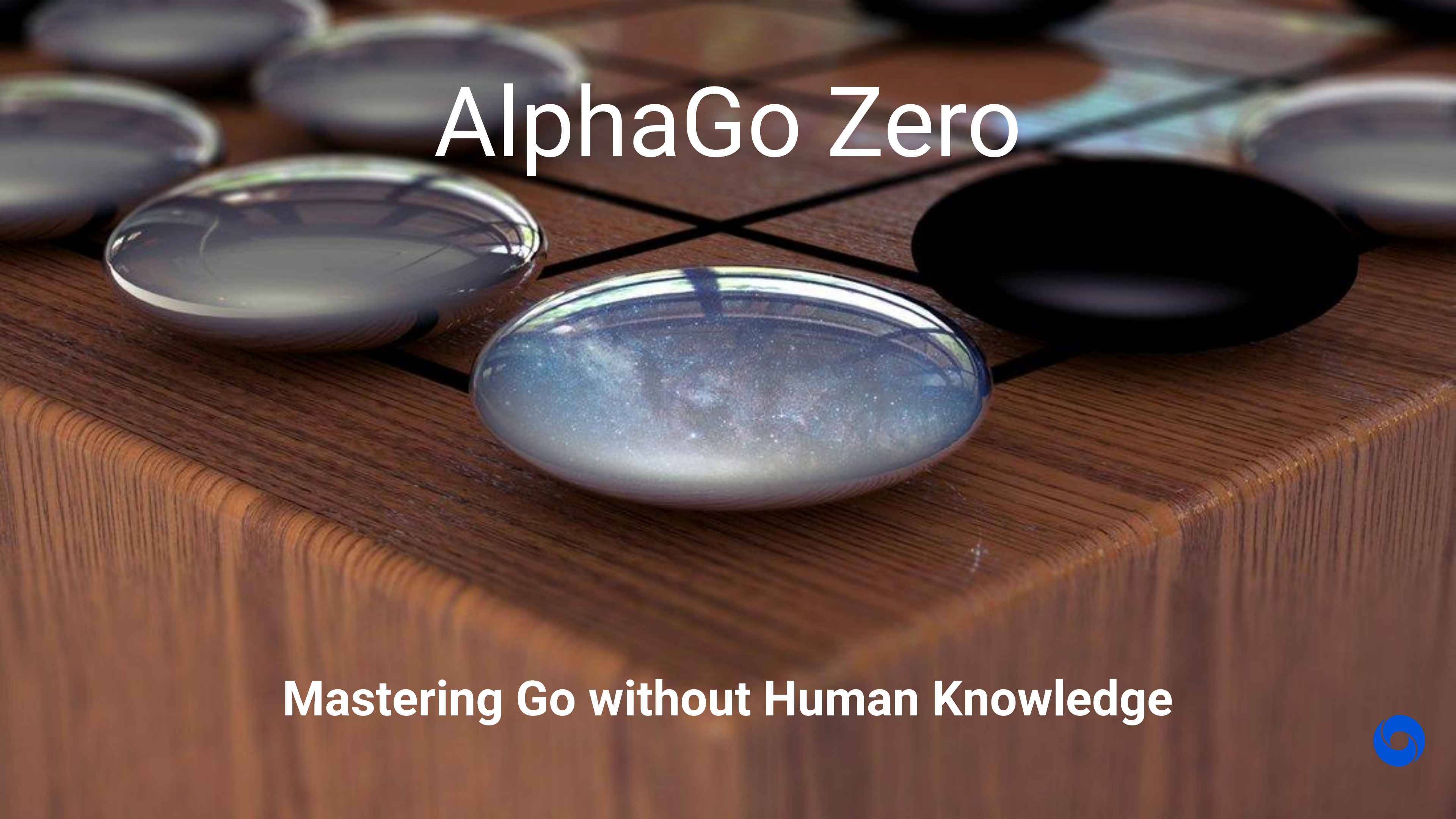
AlphaGo vs. Lee Sedol

Lee Sedol (9p): winner of 18 world titles

Match was played in Seoul, March 2016

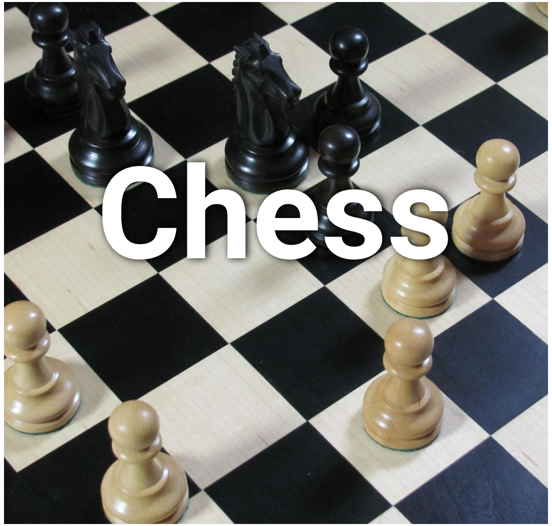
AlphaGo won the match 4-1

AlphaGo Zero

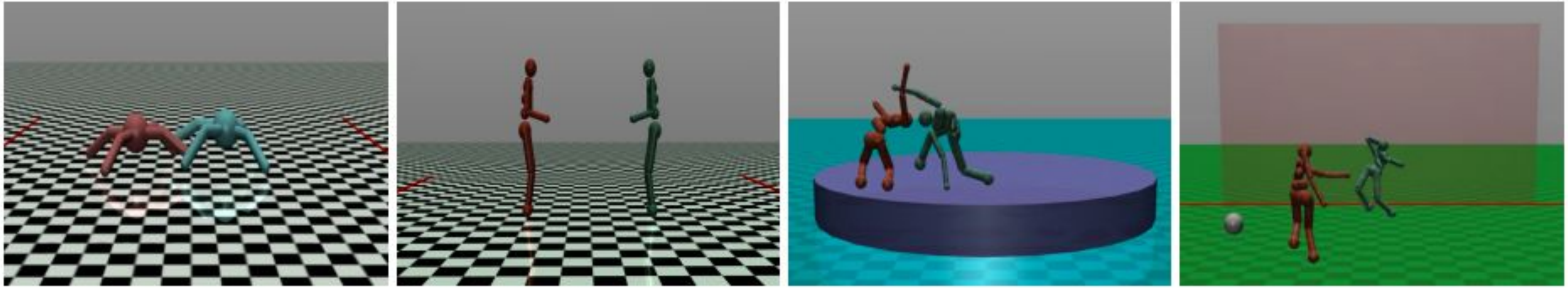


Mastering Go without Human Knowledge

AlphaZero: One Algorithm, Three Games

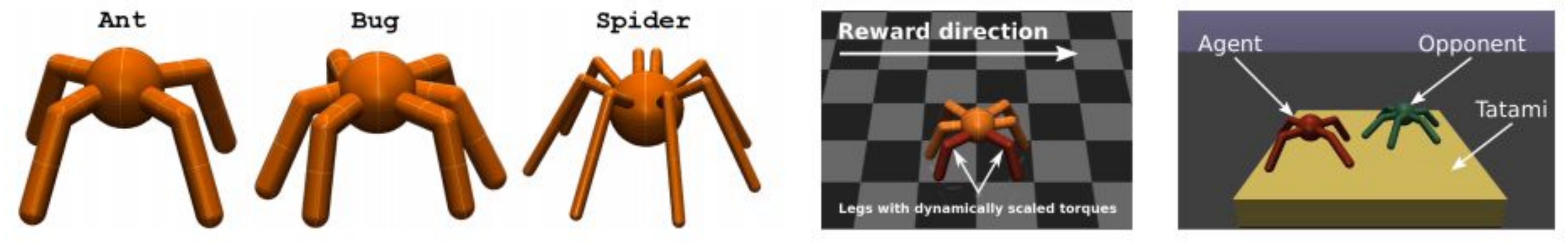


3D Worlds



Bansal et al. '18

Meta-Learning in RoboSumo



[Al-Shedivat et al. ‘17](#)

Emergent Coordination Through Competition

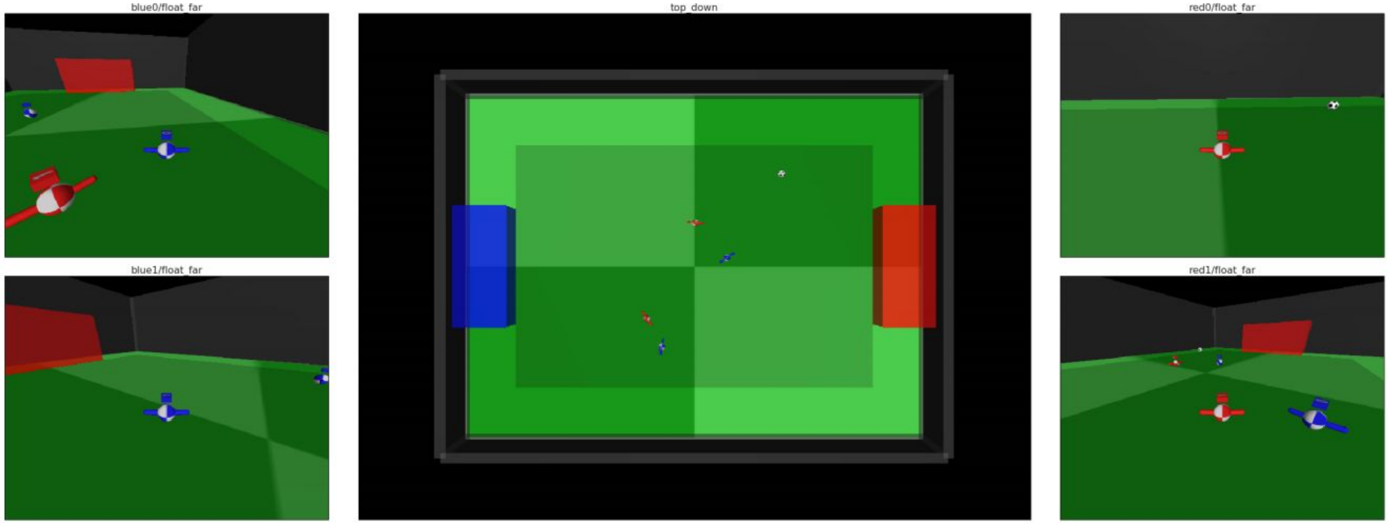
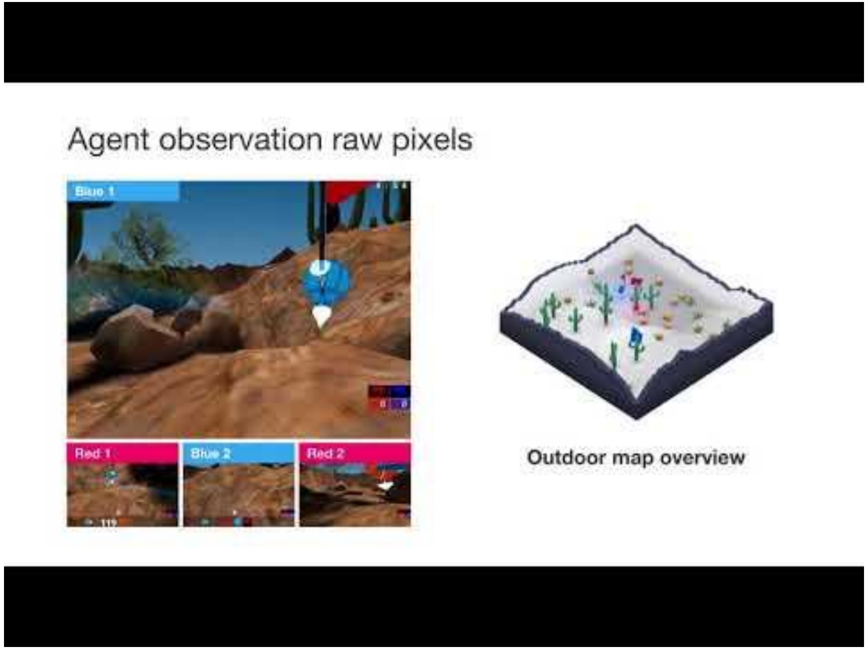
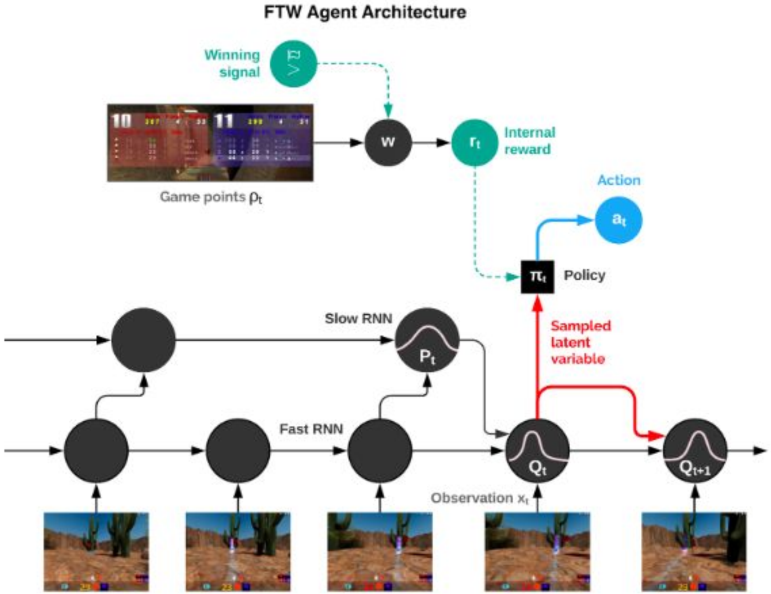


Figure 1: Top-down view with individual camera views of 2v2 multi-agent soccer environment.

Liu et al. ' 19 and http://git.io/dm_soccer

Capture-the-Flag (Jaderberg et al. ‘19)



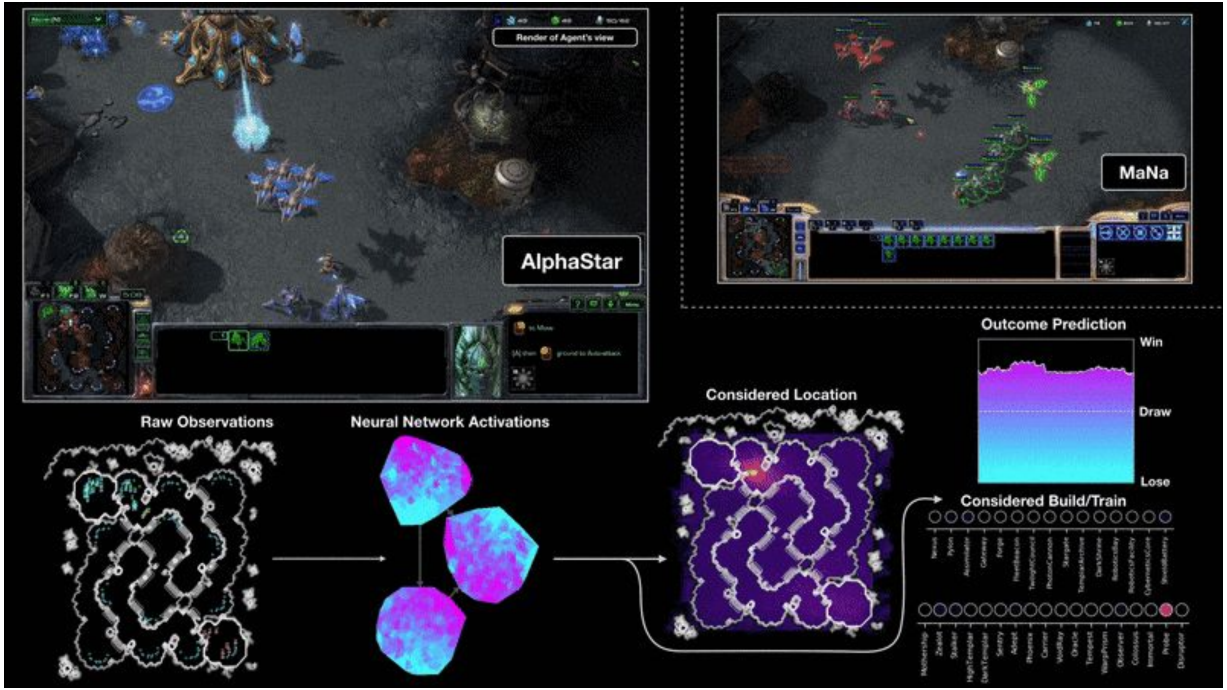


https://deepmind.com/blog/capture-the-flag-science/

Dota 2: OpenAI Five

https://openai.com/blog/openai-five-finals/

AlphaStar (Vinyals et al. ‘19)



https://deepmind.com/blog/article/AlphaStar-Grandmaster-level-in-StarCraft-II-using-multi-agent-reinforcement-learning

Deep Multiagent RL Survey

A Survey and Critique of Multiagent Deep Reinforcement Learning☆

Pablo Hernandez-Leal, Bilal Kartal and Matthew E. Taylor
{pablo.hernandez,bilal.kartal,matthew.taylor}@borealisai.com

Borealis AI
Edmonton, Canada

https://arxiv.org/abs/1810.05587

DeepMind

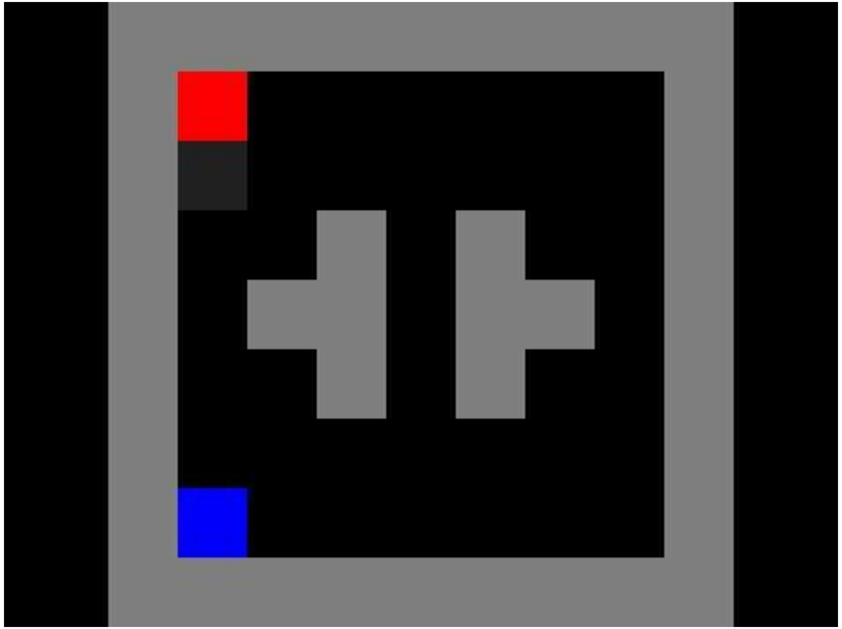
2d

Quick Sampler: Partial Observability

Foundations of Multiagent RL

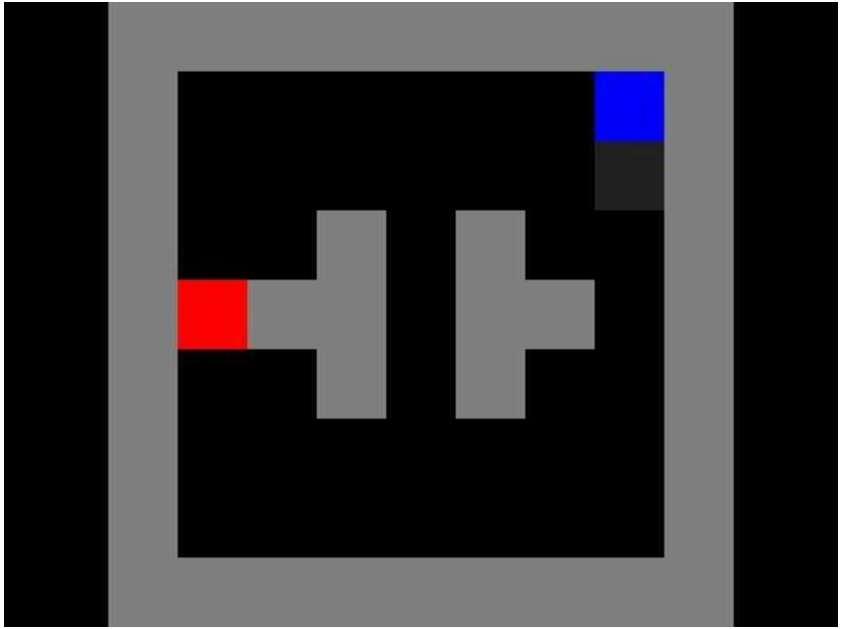
	Single Agent	Multiple (e.g. 2) Agents
Large Problems	Reinforcement Learning	**Multiagent Reinforcement Learning**
Small Problems	Approximate Dynamic Programming	Game Theory

Independent Deep Q-networks (See Lanctot et al. '17)



https://www.youtube.com/watch?v=8vXpdHuoQH8

Independent learners who learned together

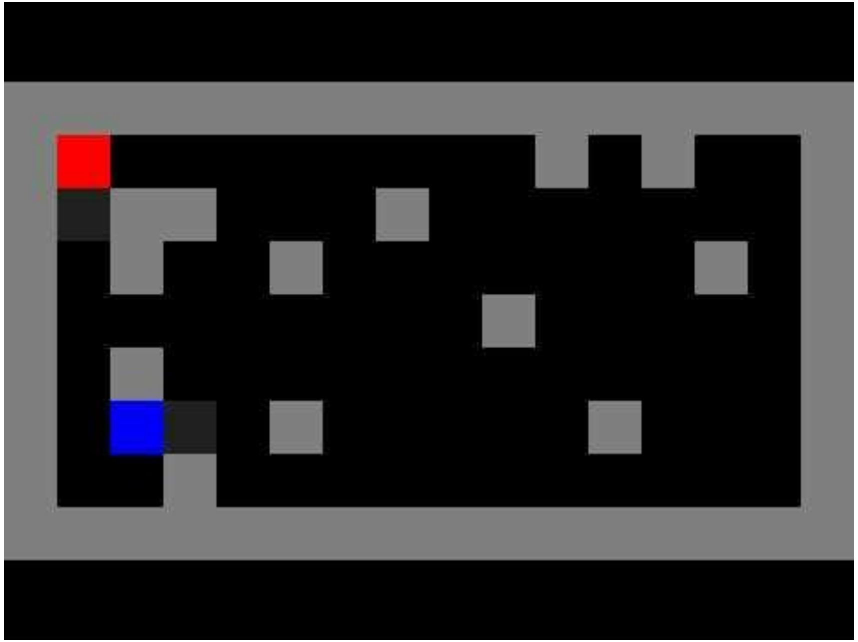


https://www.youtube.com/watch?v=jOjwOkCM_i8

Independent learners who learned using the same algorithm, same architecture, same hyperparameters, but different seed

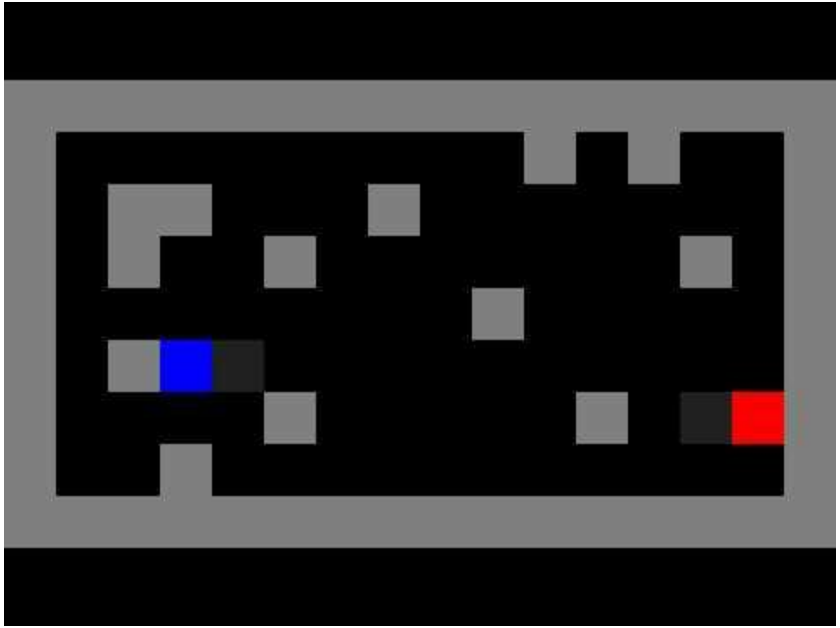
Independent Deep Q-networks

(See Lanctot et al. '17)



https://www.youtube.com/watch?v=Z5cpIG3GsLw

Independent learners who learned together



https://www.youtube.com/watch?v=zilU0hXvGK4

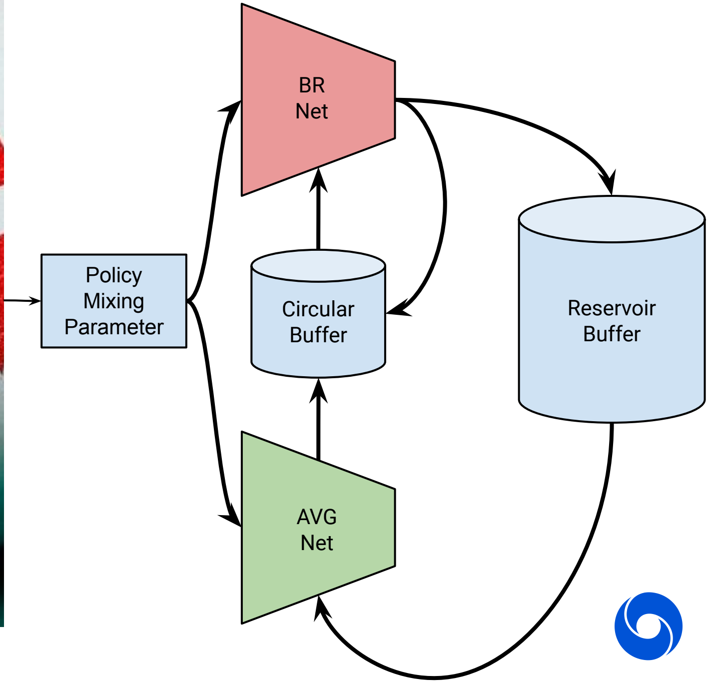
Independent learners who learned using the same algorithm, same architecture, same hyperparameters, but different seed

Fictitious Self-Play [Heinrich et al. '15, Heinrich & Silver 2016]

- **Idea:** Fictitious self-play (FSP) + reinforcement learning
- Update rule in sequential setting *equivalent* to standard fictitious play (matrix game)
- Approximate NE via two neural networks:

1. **Best response net (BR):**
    - Estimate a best response
    - Trained via RL

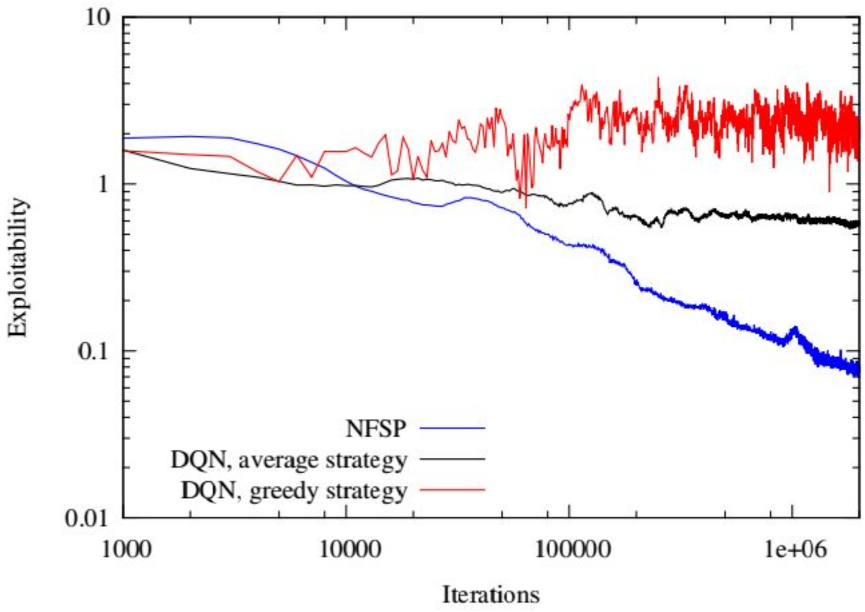
2. **Average policy net (AVG):**
    - Estimate the time-average policy
    - Trained via supervised learning



Neural Fictitious Self-Play [Heinrich & Silver 2016]

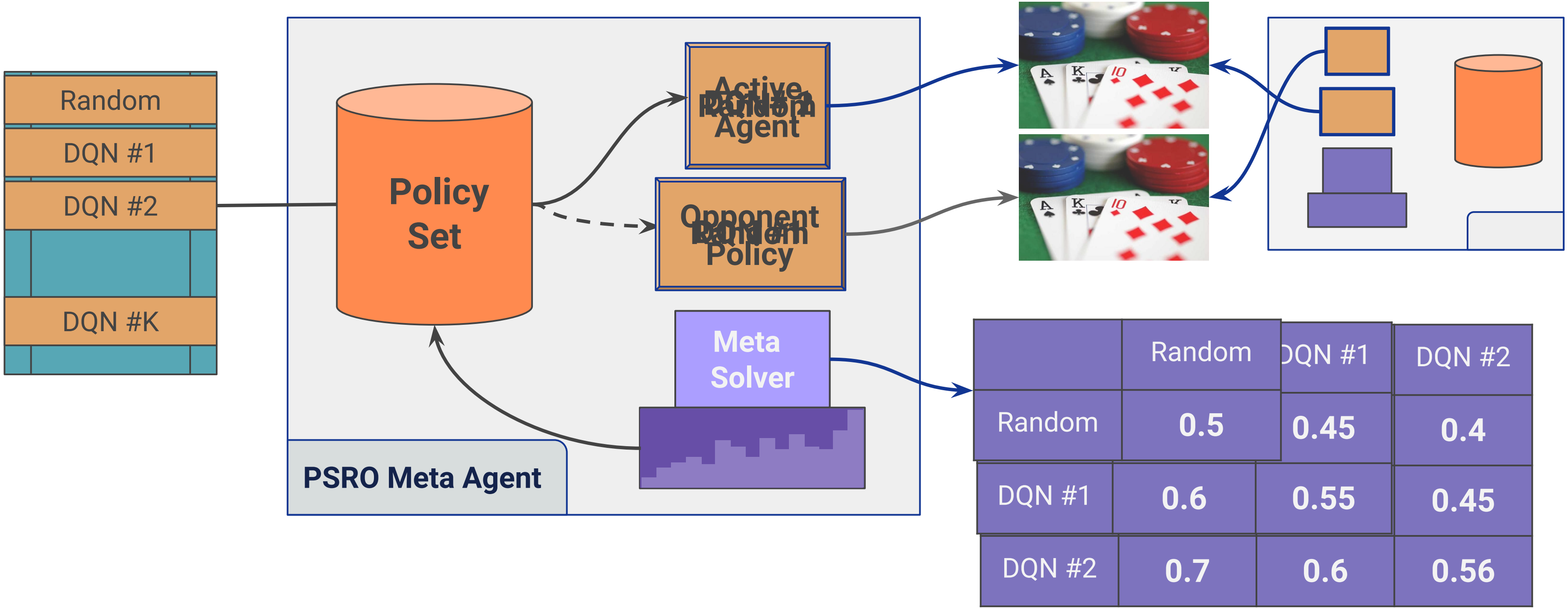
- Leduc Hold'em poker experiments:

"Closeness" to Nash



- 1st scalable end-to-end approach to learn **approximate Nash equilibria w/o prior domain knowledge**
    - Competitive with superhuman computer poker programs when it was released

Policy-Space Response Oracles (Lanctot et al. '17)



	Random	DQN #1	DQN #2
Random	0.5	0.45	0.4
DQN #1	0.6	0.55	0.45
DQN #2	0.7	0.6	0.56

Quantifying “Joint Policy Correlation”

In RL:

- Each player uses optimizes independently
- After many steps, joint policy (π_1 , π_2) co-learned for players 1 & 2

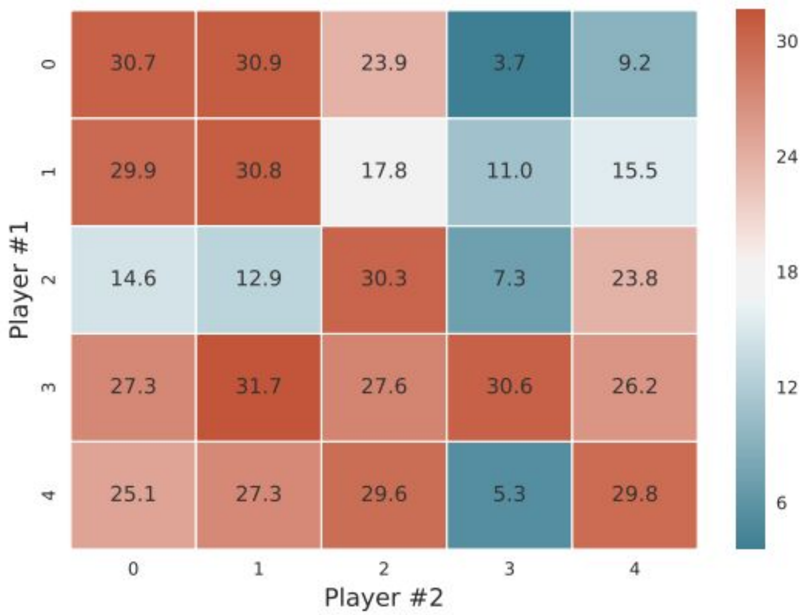
Computing **JPC:** start **5 separate instances of the *same experiment***, with

  - Same hyper-parameter values
  - Differ *only* by seed (!)

- Reload all 25 combinations and play π_1^i with π_2^j for instances i, j

Joint Policy Correlation in Independent RL

InRL in small2 (first) map



InRL in small4 map

JPC Results - Laser Tag

Game	Diag	Off Diag	Exp. Loss
LT small2	30.44	20.03	**34.2 %**
LT small3	23.06	9.06	**62.5 %**
LT small4	20.15	5.71	**71.7 %**
Gathering field	147.34	146.89	none
Pathfind merge	108.73	106.32	none

Exploitability Descent (Lockhart et al. '19)

Algorithm 2: Exploitability Descent (ED)

input : π^0 — initial joint policy

for $t \in \{1, 2, \cdots\}$ **do**

  for $i \in \{1, \cdots, n\}$ **do**

    Compute a best response $b_i^t(\pi_{-i}^{t-1})$

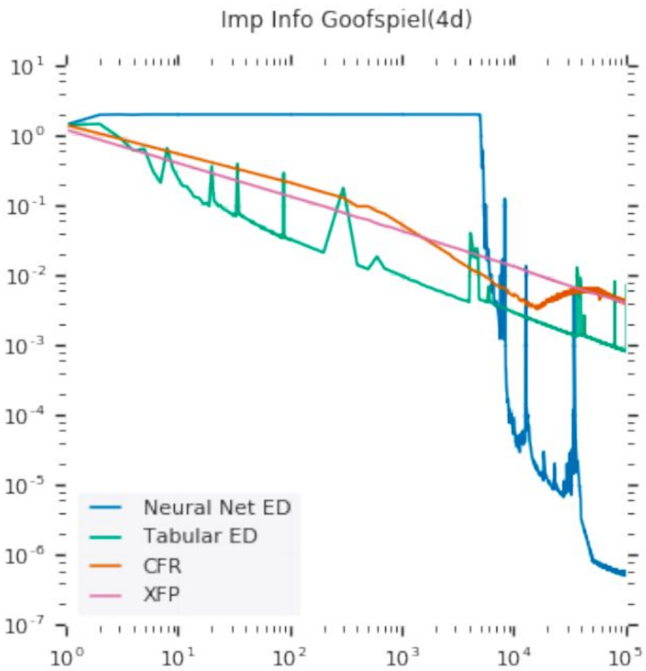
  for $i \in \{1, \cdots, n\}, s \in \mathcal{S}_i$ **do**

    Define $b_{-i}^t = \{b_j^t\}_{j \neq i}$

    Let $\mathbf{q}^b(s) = \textsc{ValuesVsBRs}(\pi_i^{t-1}(s), b_{-i}^t)$

    $\pi_i^t(s) = \textsc{GradAscent}(\pi_i^{t-1}(s), \alpha^t, \mathbf{q}^b(s))$

- A FP-like algorithm conv. *without averaging*!
- Amenable to function approximation



Counterfactual Regret Minimization (CFR)

Zinkevich et al. '08

- Algorithm to compute approx Nash eq. In 2P zero-sum games
- Hugely successful in Poker AI
- Size traditionally reduced apriori based on expert knowledge
- Key innovation: counterfactual values: $v_i^c(\pi, s, a)$ $\quad v_i^c(\pi, s)$

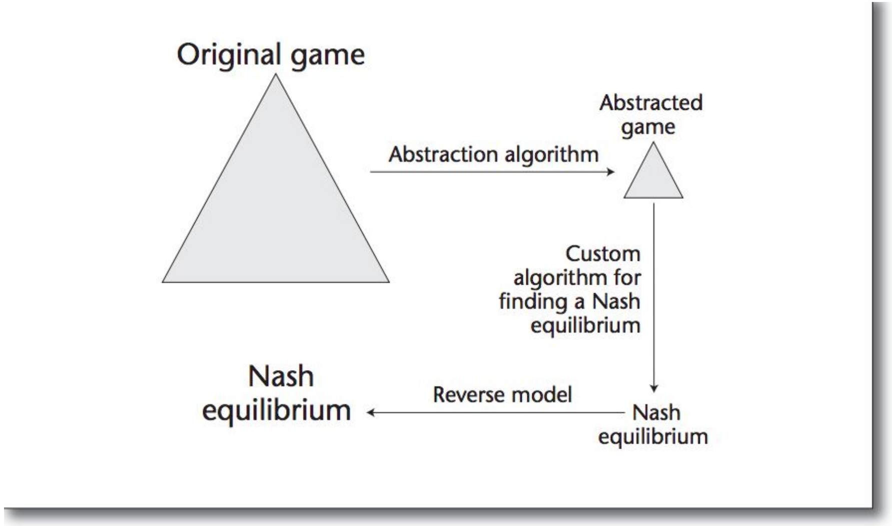


Figure 1. Current Paradigm for Solving Large Incomplete-Information Games.

Image form Sandholm '10

CFR is policy iteration!

- Policy evaluation is analogous
- Policy improvement: use regret minimization algorithms
  - Average strategies converge to Nash in self-play
- Convergence guarantees are on the *average policies*

DeepStack

[(Moravcik et al. ‘17)]

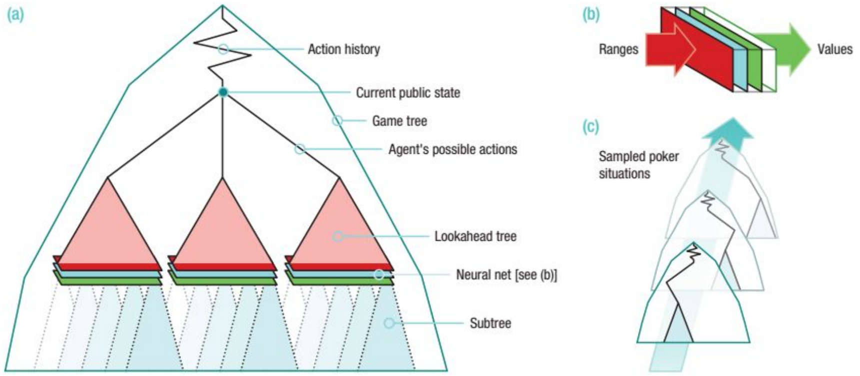
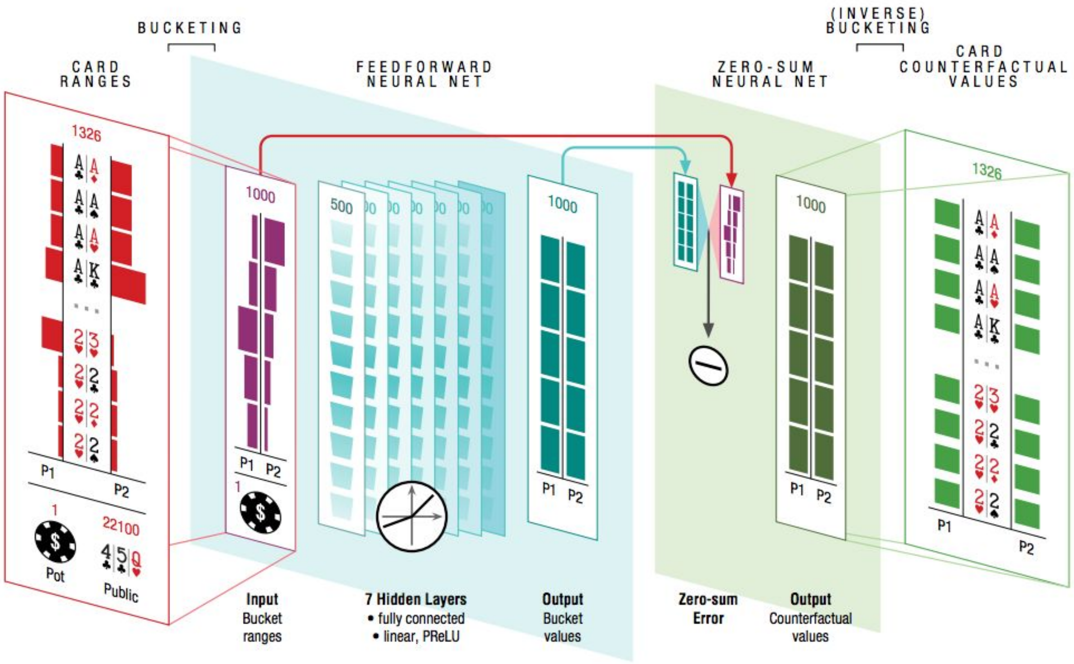


Figure 2: DeepStack overview. (a) DeepStack re-solves for its action at every public state it is to act, using a depth limited lookahead where subtree values are computed using a trained deep neural network (b) trained before play via randomly generated poker situations (c).

DeepStack

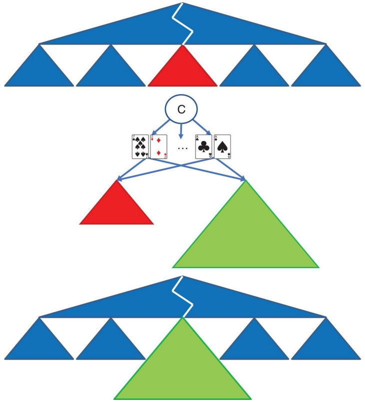
(Moravcik et al. '17)



Libratus [(Brown & Sandholm '18)]

RESEARCH ARTICLE

Superhuman AI for heads-up no-limit poker: Libratus beats top professionals



Policy Gradient Algorithms

Parameterized policy π_θ with parameters θ (e.g. a neural network)

Define a *score function* $J(\pi_\theta) = v_\pi(s_0) = \mathbb{E}_\pi[G_0]$

Main idea: do gradient ascent on J.

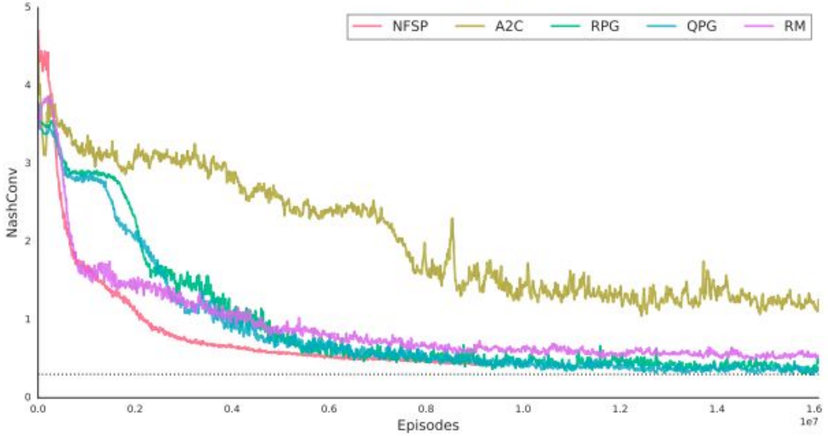
1. **REINFORCE** (Williams '92, see RL book ch. 13) + PG theorem:

   you can do this via estimates from sample trajectories.

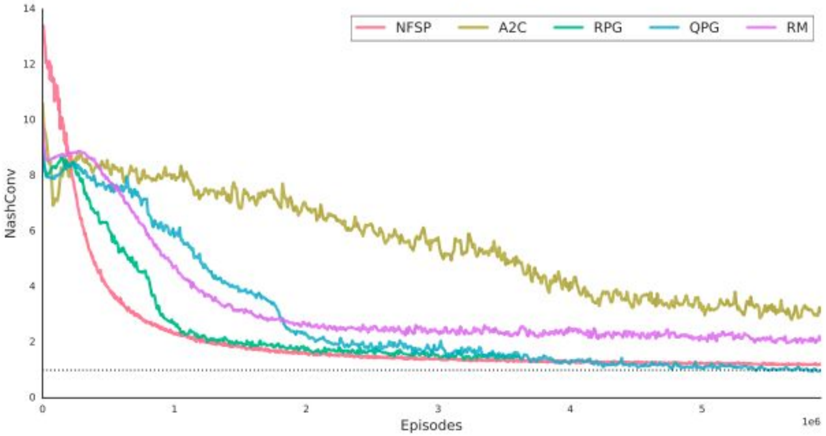
2. **Advantage Actor-Critic (A2C)** (Mnih et al '16): you can use deep networks to estimate the policy *and* baseline value v(s)

Regret Policy Gradients (Srinivasan et al. '18)

- Policy gradient is doing a form of CFR minimization!
- Several new policy gradient variants inspired connection to regret



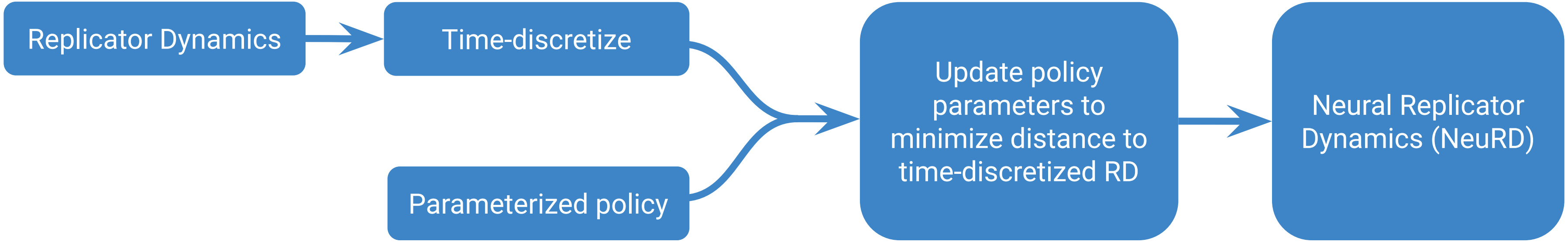
NASHCONV in 2-player Leduc



NASHCONV in 3-player Leduc

Hedging Policy Gradients (Previously "Neural Replicator Dynamics" / NeuRD)

[Omidshafiei, Hennes, Morrill et al. '19]

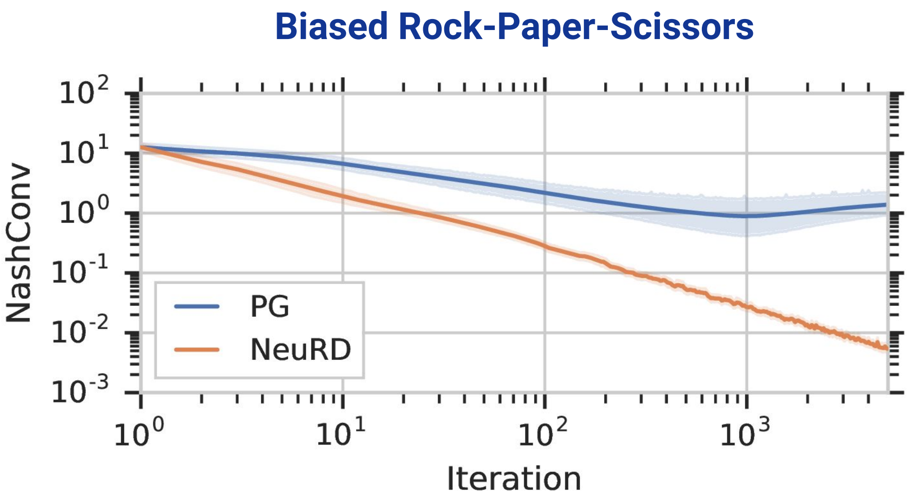


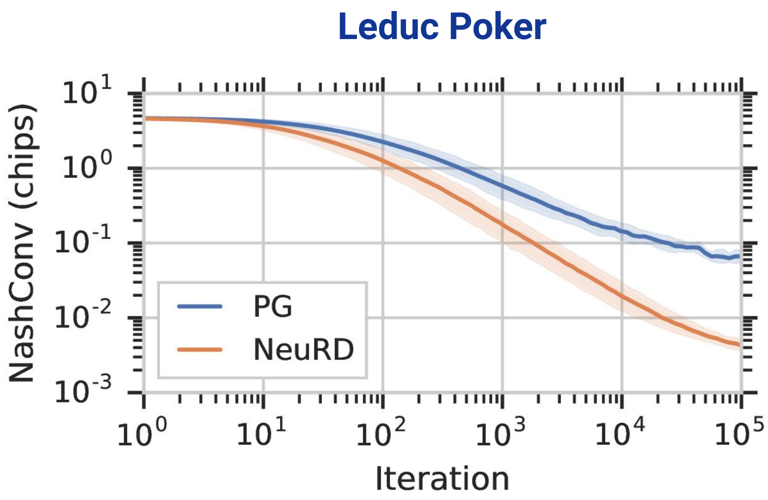
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t+1} + \eta \sum_{s,a} \nabla_{\boldsymbol{\theta}} y_{t-1}(s_t, a_t; \boldsymbol{\theta}) A(s_t, a_t; \boldsymbol{\theta}, \boldsymbol{w})$$

Logits, where policy is $\boldsymbol{\pi} = softmax(\boldsymbol{y})$

Advantage q(s,a)-v(s)

NeuRD: Results





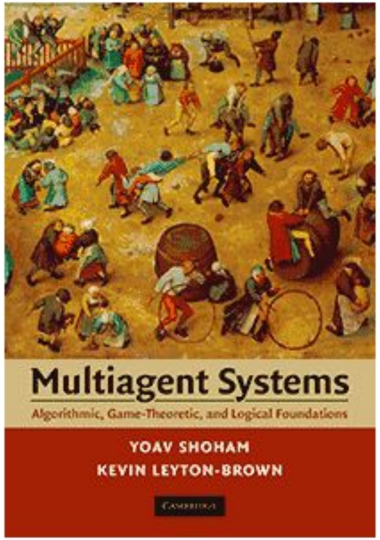
DeepMind

2e

General MARL Wrap-up

Shoham & Leyton-Brown '09

Main Page Table of Contents Instructional Resources Errata eBook Download *new!*



Multiagent Systems
Algorithmic, Game-Theoretic, and Logical Foundations

Yoav Shoham
Stanford University
Kevin Leyton-Brown
University of British Columbia

Cambridge University Press, 2009
Order online: amazon.com

masfoundations.org

Surveys and Food for Thought

- If multi-agent learning is the answer, what is the question?
    - Shoham et al. '06
    - Hernandez-Leal et al. '19
- A comprehensive survey of MARL (Busoniu et al. '08)
- Game Theory and Multiagent RL (Nowé et al. '12)
- Study of Learning in Multiagent Envs (Hernandez-Leal et al. '17)

The Hanabi Challenge

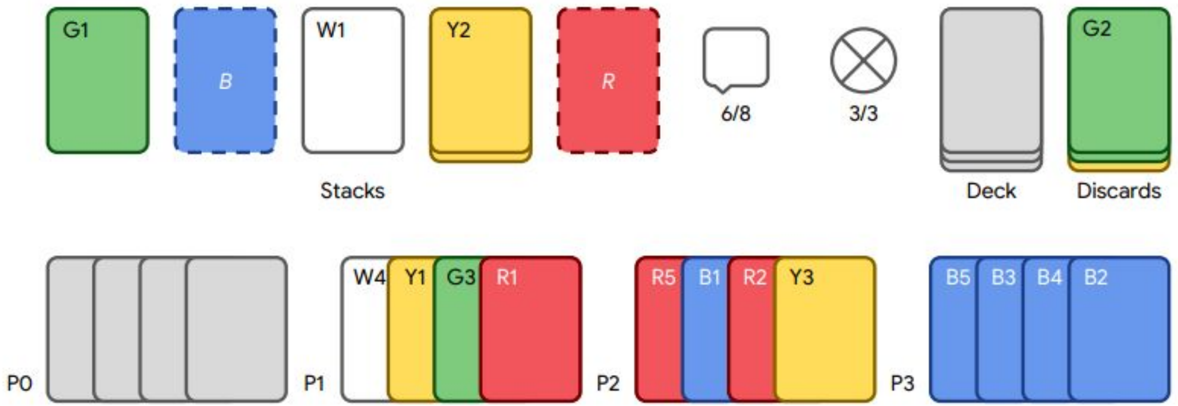


Figure 1: Example of a four player Hanabi game from the point of view of player 0. Player 1 acts after player 0 and so on.

[Bard et al. '19](#) Also Competition at IEEE Cog ([ieee-cog.org](#))

AAAI 2020 Workshop on RL in Games

AAAI19-RLG Summary:

- 39 accepted papers
  - 4 oral presentations
  - 35 posters
- 1 “Mini-Tutorial”
- 3 Invited Talks
- Panel & Discussion

http://aaai-rlg.mlanctot.info/

DeepMind

3 Adapting RL Algorithms to Zero-Sum Games

Plan: MARL in Zero-Sum Games

1. Worked out examples
    a. Adapting Q-learning
    b. Counterfactual Regret Minimization

2. Three important sub-topics:
    a. *Expected* values vs. *counterfactual* values
    b. Monte Carlo CFR: sample-based CFR
    c. Search in Imperfect Information games

DeepMind

3.1a Tabular Q-learning Exercise

Tabular Q-Learning Exercise

Please refer to handout.

- Either on your own or in small groups, try to answer **Q1**. [5 min]
- Then, now try to answer **Q2.** [5 min]

Let’s discuss the answers.

Tabular Q-learning Exercise

Suppose $\alpha = 0.1$, Q(s, a) = 0 for all s,a , and the following episodes are played by the agent(s):

- 0, 4, 8, 5, 2, 1, 7, 3
- 2, 1, 0, 4, 7, 5, 8, 6, 3

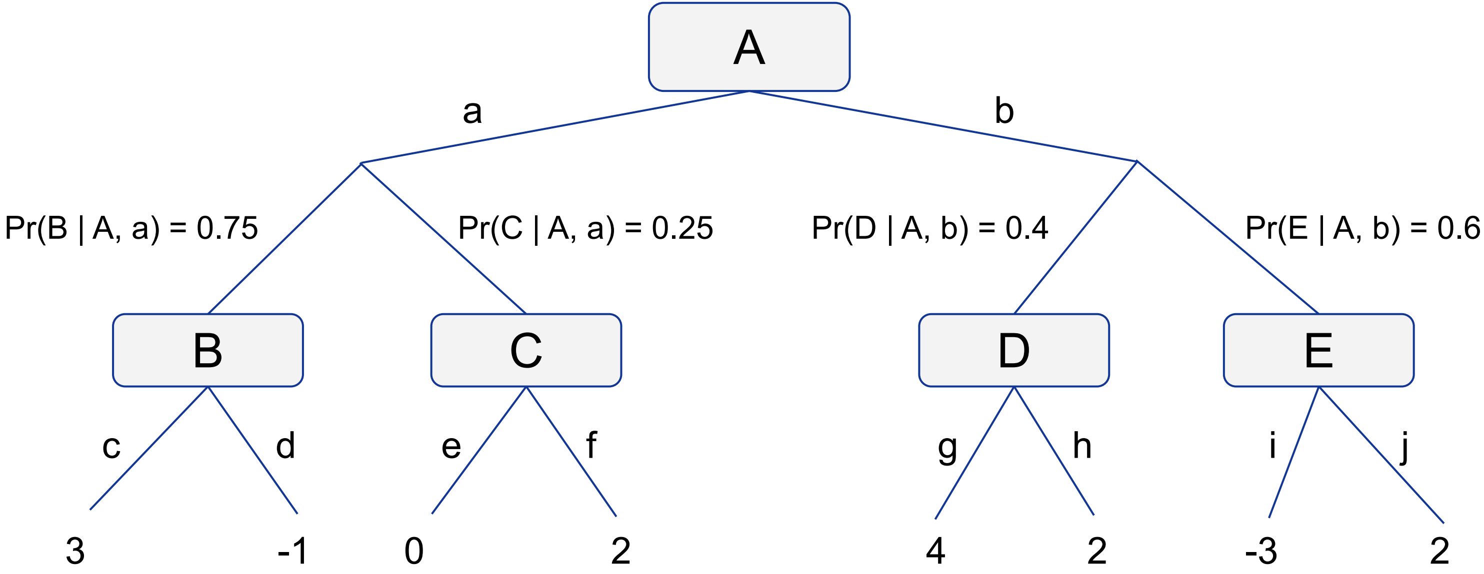
0	1	2
3	4	5
6	7	8

- Which state(s) have actions with non-zero Q-values?
- What are those action(s)?

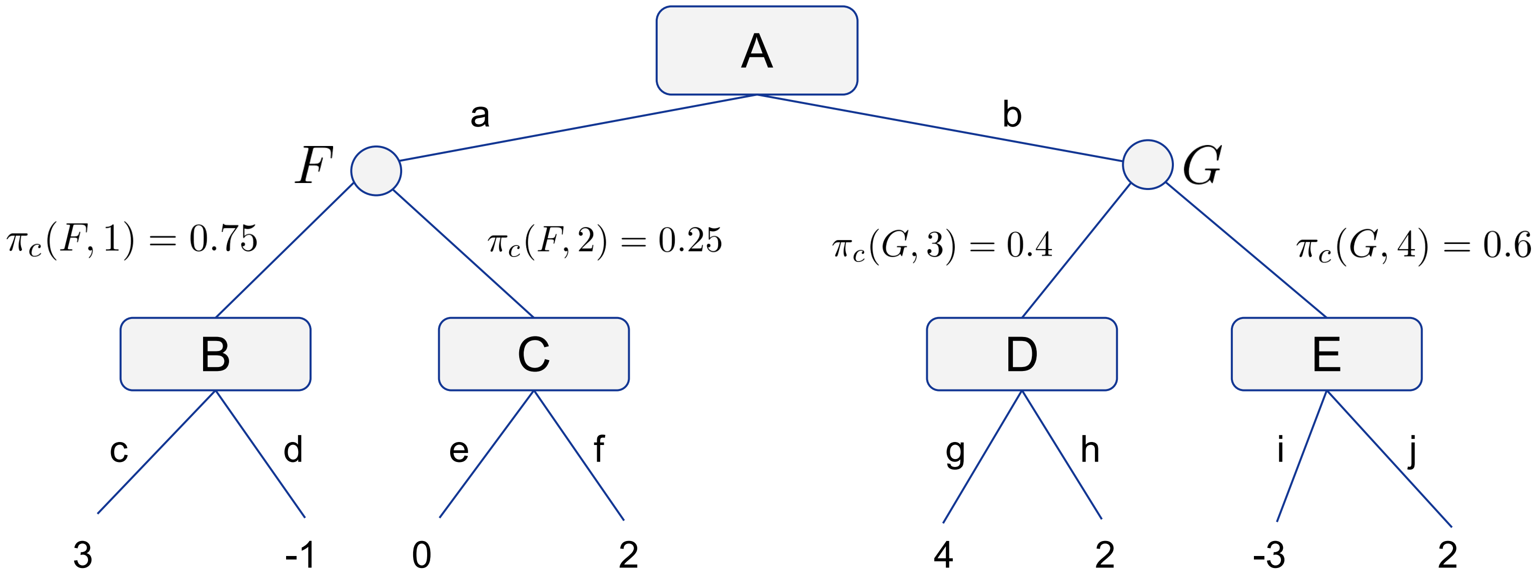
DeepMind

3.1b Counterfactual Regret Minimization Exercise

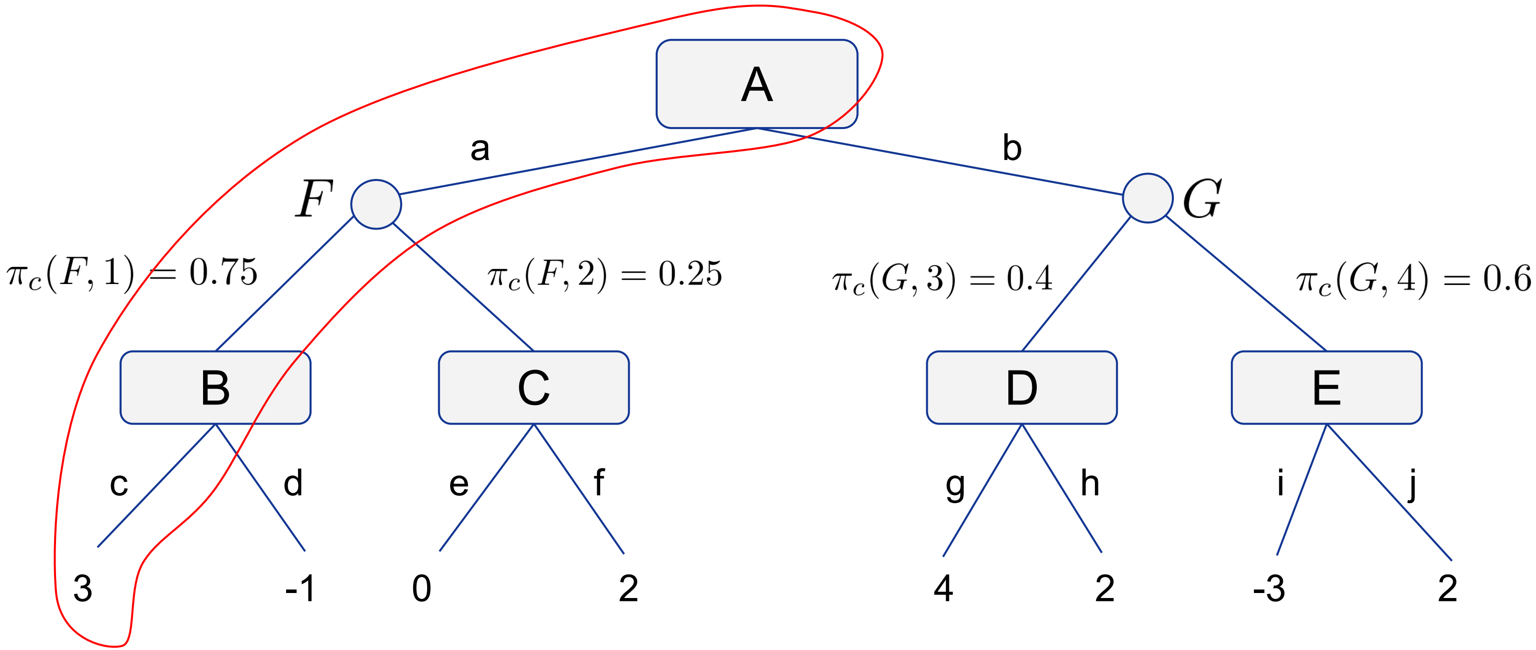
A simple MDP


A
a
b
Pr(B | A, a) = 0.75
Pr(C | A, a) = 0.25
Pr(D | A, b) = 0.4
Pr(E | A, b) = 0.6
B
C
D
E
c
d
e
f
g
h
i
j
3
-1
0
2
4
2
-3
2

A simple ~~MDP~~ Multiagent System

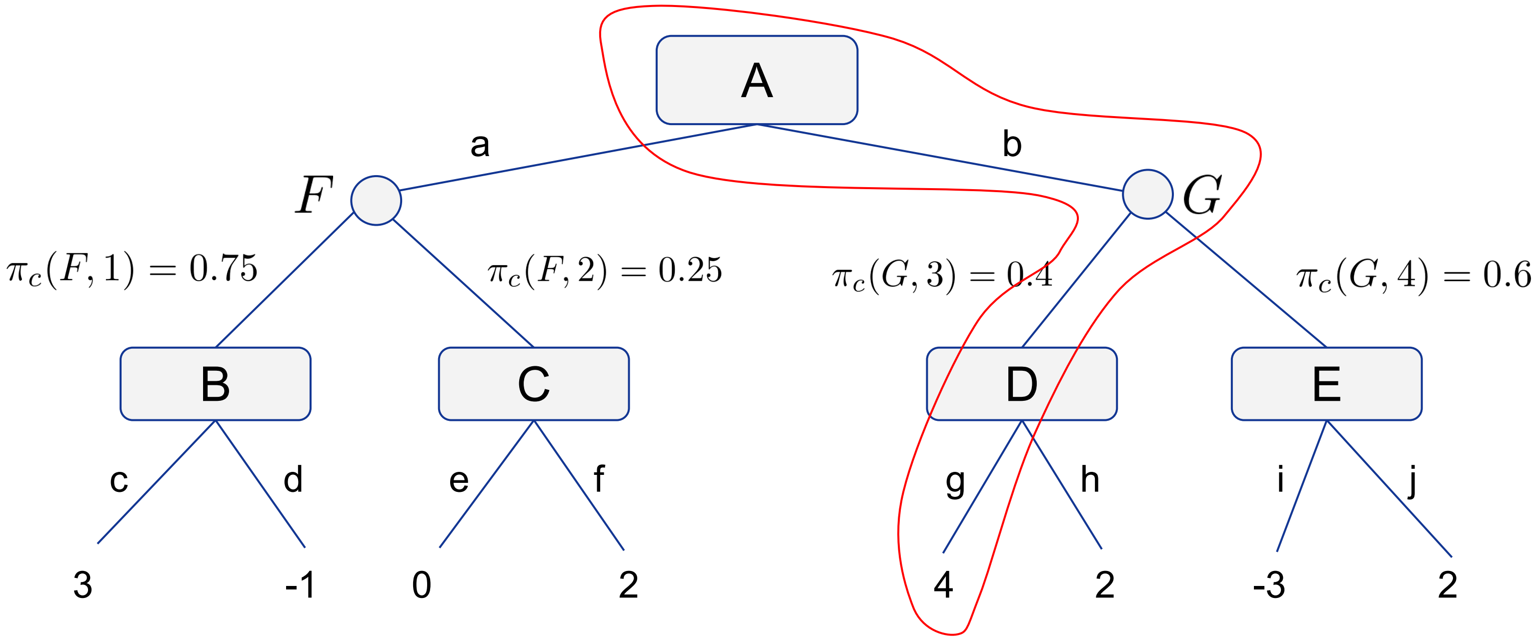


Terminal history A.K.A. Episode



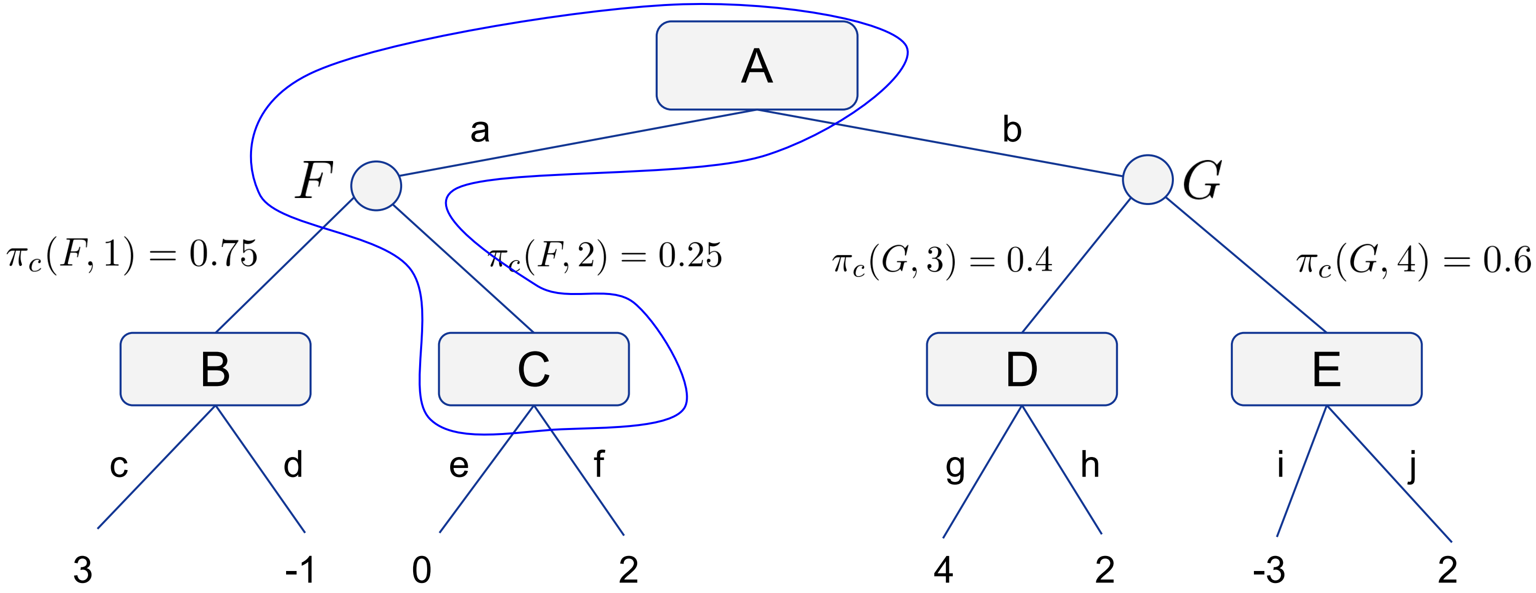
(A, a, F, 1, B, c) is a *terminal* history.

Terminal history A.K.A. Episode



(A, a, F, 1, B, c) is a *terminal* history. (A, b, G, 3, D, g) is a another terminal history.

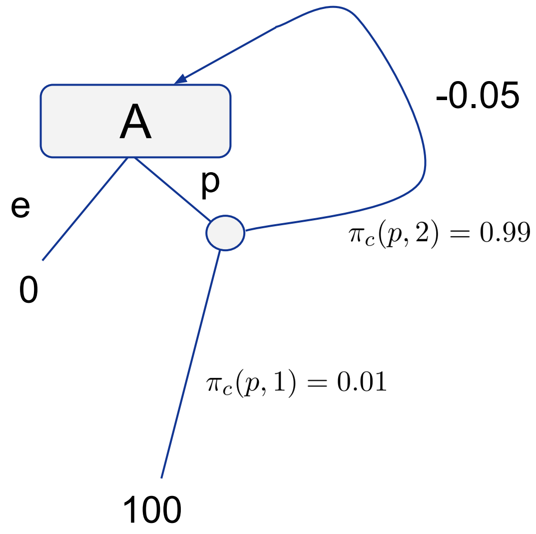
Prefix (non-terminal) Histories



(A, a, F, 2, C) is a history. It is a *prefix* of (A, a, F, 2, C, e) and (A, a, F, 2, C, f).

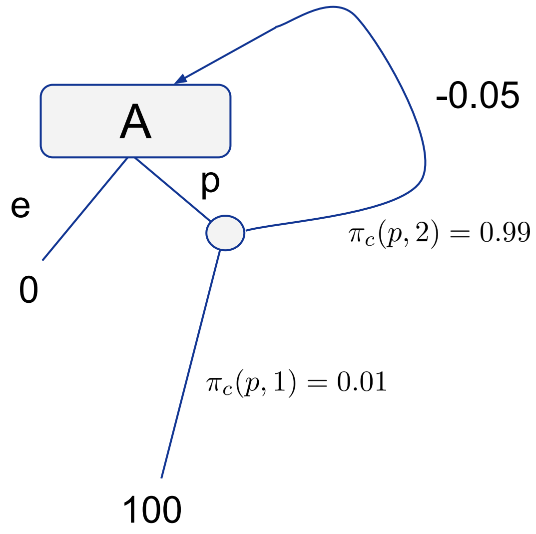
Perfect Recall of Actions and Observations

Another simple MDP:

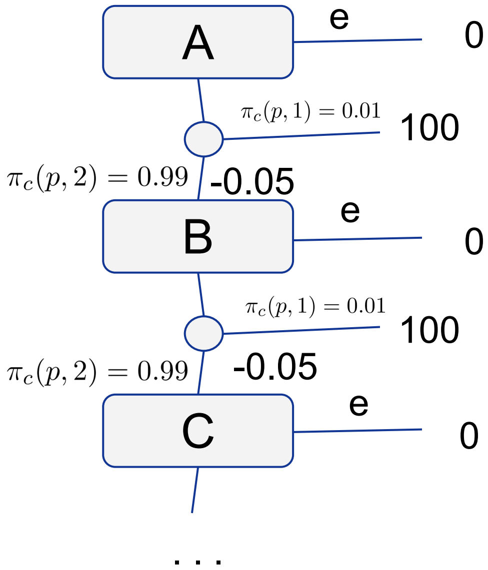


Perfect Recall of Actions and Observations

Another simple MDP:



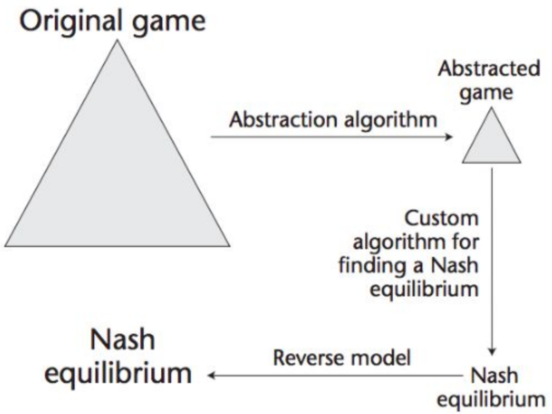
A different MDP:



Counterfactual Regret (CFR) Minimization

Zinkevich et al. '08

- Algorithm to compute an approx. Nash eq. in 2-player 0-sum games
- Hugely successful in computer Poker
- Size usually reduced apriori based on expert knowledge
- Key innovations:
    - Counterfactual values
    - CFR Theorem

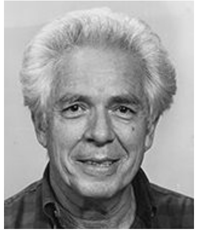


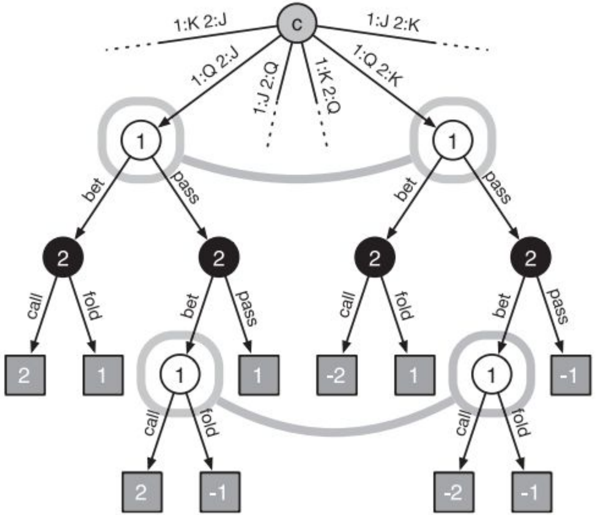
Source: Sandholm '10

Partially Observable Zero-Sum Games

Kuhn (simplified) poker

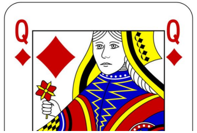
- Players start w/ 2 chips
- Each: ante 1 chip
- 3-card deck
- 2 actions: pass, bet
- Reward: money diff





Terminology

- An **information state,** s, corresponds to a *sequence of observations*
  - with respect to the player to play at s

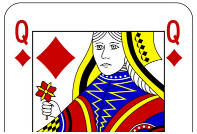
Ante: 1 chip per player, 

, P1 bets (raise), P2 bets (call)

Terminology

- An **information state,** s, corresponds to a *sequence of observations*
  - with respect to the player to play at s

private observation

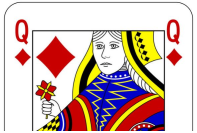
Ante: 1 chip per player, 

, P1 bets (raise), P2 bets (call)

Terminology

- An **information state,** s, corresponds to a *sequence of observations*
  - with respect to the player to play at s

private observation

Ante: 1 chip per player, 

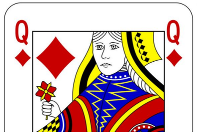
, P1 bets (raise), P2 bets (call)

Environment is in one of many **world states** $h \in s$

Terminology

- An **information state,** s, corresponds to a *sequence of observations*
  - with respect to the player to play at s

private observation

Ante: 1 chip per player,  , P1 bets (raise), P2 bets (call)

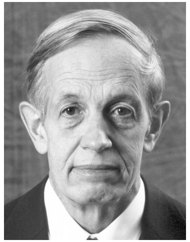
Environment is in one of many **world states** $h \in s$

full **history** of actions (including nature's!!)

Goal: (Approximate) Nash Equilibria and minimax

Minimax & Nash equilibrium

von Neumann 1928



Nash 1950

$$v_1 = \max_{\pi_1} \min_{\pi_2} u_1(\pi_1, \pi_2)$$

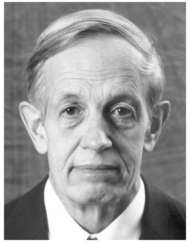
$$v_1 = \min_{\pi_2} \max_{\pi_1} u_1(\pi_1, \pi_2)$$

In 2P zero-sum, these are the same!

Goal: (Approximate) Nash Equilibria and minimax

Minimax & Nash equilibrium

von Neumann 1928



Nash 1950

$$v_1 = \max_{\pi_1} \min_{\pi_2} u_1(\pi_1, \pi_2)$$

$$v_1 = \min_{\pi_2} \max_{\pi_1} u_1(\pi_1, \pi_2)$$

In 2P zero-sum, these are the same!

2P Zero-sum Equilibria

The optima: $\pi^* = (\pi_1^*, \pi_2^*)$

- Exist! (May be stochastic.)
- Called **minimax-optimal** joint policy
  - A.K.A. Nash equilibrium
- They are interchangeable!
  - $\forall \pi^*, \pi^{*\prime} \Rightarrow (\pi_1^*, \pi_2^{*\prime}), (\pi_1^{*\prime}, \pi_2^*)$
- Each policy is a **best response** to the other

Counterfactual Minimization

CFR is policy iteration:

1. Evaluate policy to compute values
2. Improve the policy

Counterfactual Minimization

CFR is (special kind of) policy iteration:

1. Evaluate policy to compute counterfactual values: $q^c_{\pi,i}(s,a)$, $v^c_{\pi,i}(s)$
2. Improve the policy (using *state-local* regret minimization)
3. Compute an average joint policy $\bar{\pi} = (\bar{\pi}_1, \bar{\pi}_2)$

Counterfactual Minimization

CFR is (special kind of) policy iteration:

1. Evaluate policy to compute counterfactual values: $q^c_{\pi,i}(s,a)$, $v^c_{\pi,i}(s)$
2. Improve the policy (using *state-local* regret minimization)
3. Compute an average joint policy $\bar{\pi} = (\bar{\pi}_1, \bar{\pi}_2)$

CFR Theorem: $\bar{\pi}$ converges to an ϵ-Nash eq. with $\epsilon \leq O\left(\frac{1}{\sqrt{T}}\right)$

Counterfactual Minimization

CFR is (special kind of) policy iteration:

1. Evaluate policy to compute counterfactual values: $q^c_{\pi,i}(s,a)$, $v^c_{\pi,i}(s)$
2. Improve the policy (using *state-local* regret minimization)
3. Compute an average joint policy $\bar{\pi} = (\bar{\pi}_1, \bar{\pi}_2)$

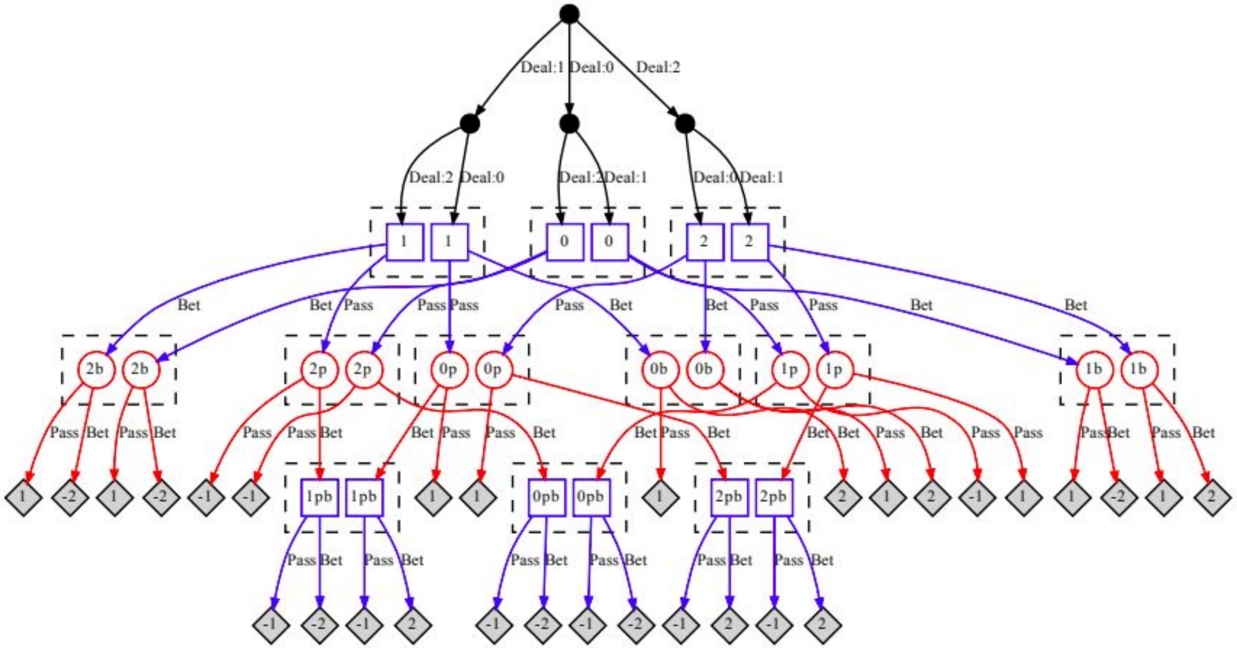
CFR Theorem: $\bar{\pi}$ converges to an ϵ-Nash eq. with $\epsilon \leq O\left(\frac{1}{\sqrt{T}}\right)$

neither player can gain more than ϵ by deviating

CFR Example

Kuhn poker:

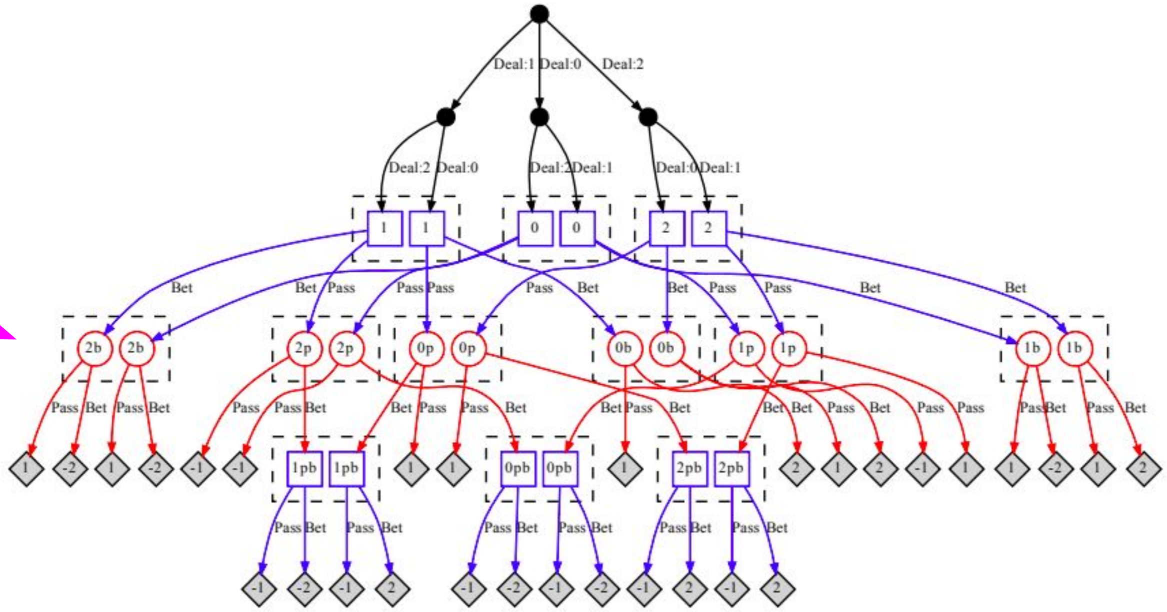
- Players: 2 chips
- 3-card deck
- Ante 1 chip
- Actions:
  - Pass
  - Bet
- Util = money diff
- Shown: util to p1



CFR Example

Uniform initial policies:

Let’s compute CFR values for state



Counterfactual values

$$q^c_{\pi,i}(s,a) = \sum_{h,z \in Z(s,a)} \eta^{\pi}_{-i}(h)\eta^{\pi}(h,z)u_i(z)$$

Terminal histories reachable from any h in s after taking action a

Counterfactual values

$$q^c_{\pi,i}(s,a) = \sum_{h,z \in Z(s,a)} \eta^{\pi}_{-i}(h)\eta^{\pi}(h,z)u_i(z)$$

Terminal histories reachable from any h in s after taking action a

Opponents' reach probabilities along h

Counterfactual values

$$q^{c}_{\pi,i}(s,a) = \sum_{h,z \in Z(s,a)} \eta^{\pi}_{-i}(h)\eta^{\pi}(h,z)u_i(z)$$

Terminal histories reachable from any h in s after taking action a

Opponents' reach probabilities along h

Both players' reach from h to z

Counterfactual values

$$q^c_{\pi,i}(s,a) = \sum_{h,z \in Z(s,a)} \eta^{\pi}_{-i}(h)\eta^{\pi}(h,z)u_i(z)$$

Terminal histories reachable from any h in s after taking action a

Opponents' reach probabilities along h

Both players' reach from h to z

Utility to player i of Terminal z

Counterfactual values

$$v^c_{\pi,i}(s) = \sum_{a \in A(s)} \pi(s,a) q^c_{\pi,i}(s,a)$$

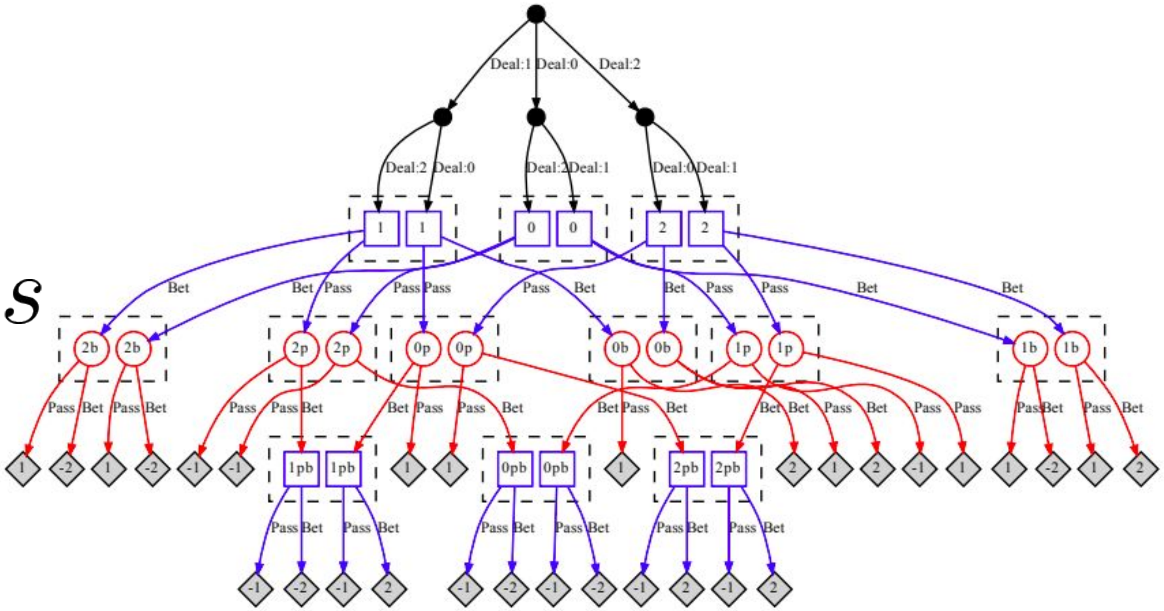
CFR Example

Two histories:

- h = 12b
- h′ = 02b

$$q^c_{\pi,2}(s,p) =$$

$$\frac{1}{12} \cdot \frac{1}{2} \cdot (-1)$$



CFR Example

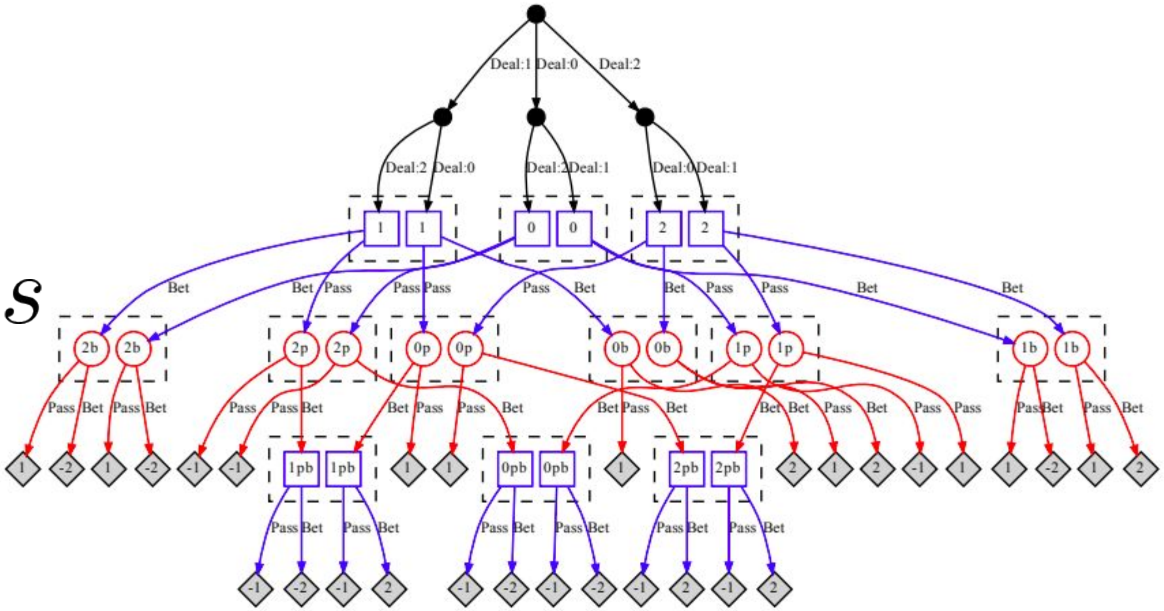
Two histories:

- h = 12b
- h′ = 02b

$$q^{c}_{\pi,2}(s,p) =$$

$$\frac{1}{12} \cdot \frac{1}{2} \cdot (-1)$$

$$+\frac{1}{12} \cdot \frac{1}{2} \cdot (-1)$$

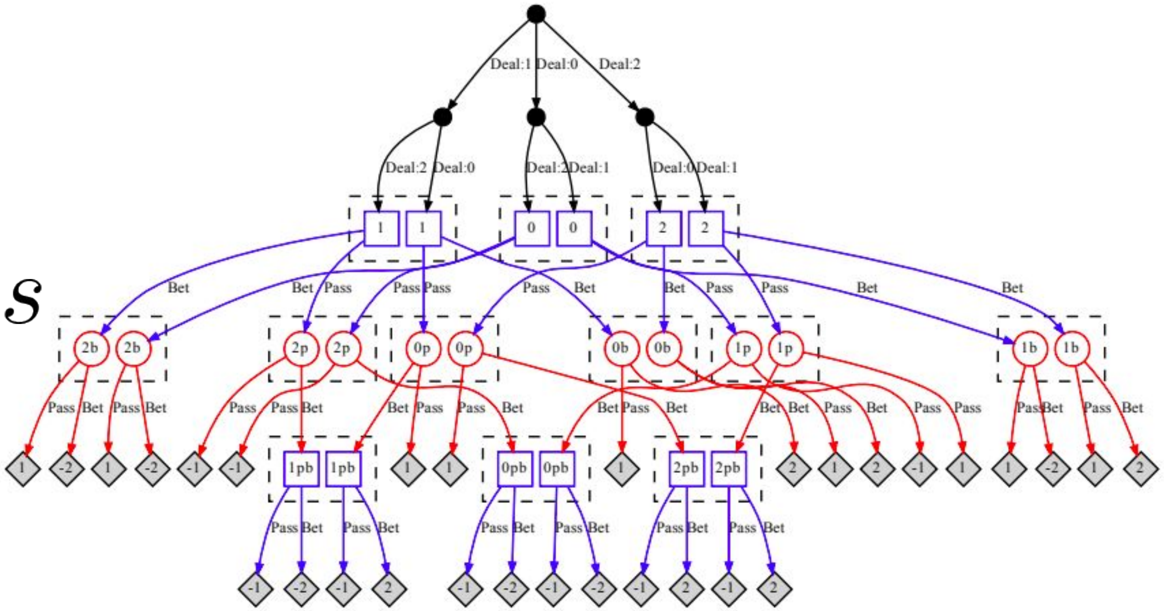


CFR Example

Two histories:

- h = 12b
- h′ = 02b

$$q^{c}_{\pi,2}(s,p) = -\frac{1}{12}$$



CFR Example

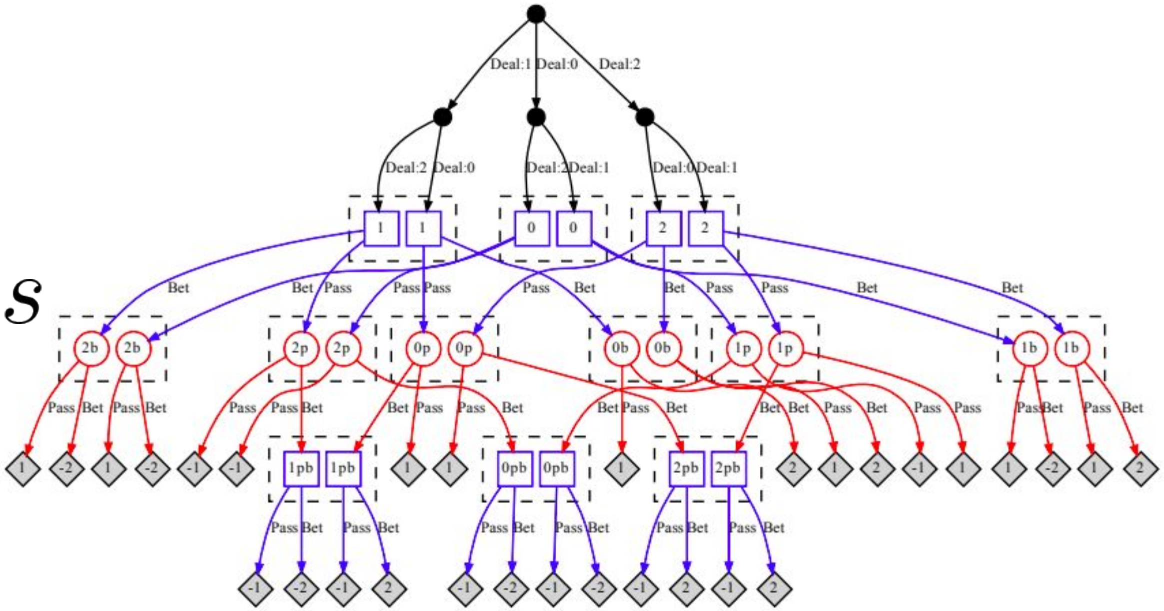
Two histories:

- h = 12b
- h' = 02b

$$q^c_{\pi,2}(s,p) = -\frac{1}{12}$$

$$q^c_{\pi,2}(s,b) =$$

$$\frac{1}{12} \cdot \frac{1}{2} \cdot 2$$



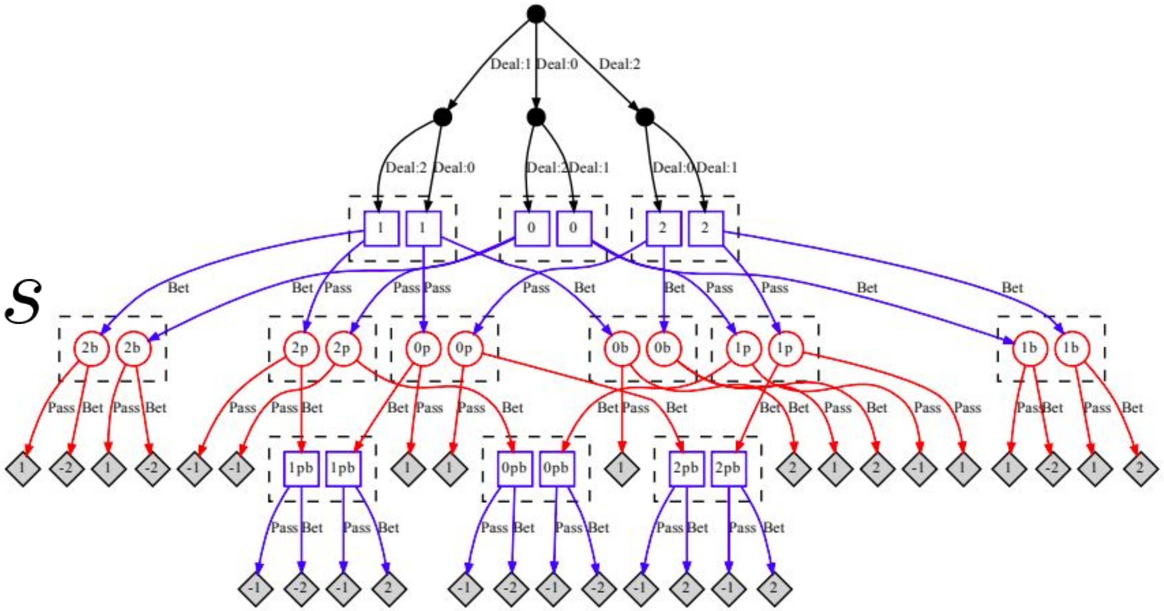
CFR Example

Two histories:

- h = 12b
- h′ = 02b

$$q^{c}_{\pi,2}(s,p) = -\frac{1}{12}$$

$$q^{c}_{\pi,2}(s,b) = \frac{1}{12}\cdot\frac{1}{2}\cdot 2 + \frac{1}{12}\cdot\frac{1}{2}\cdot 2$$



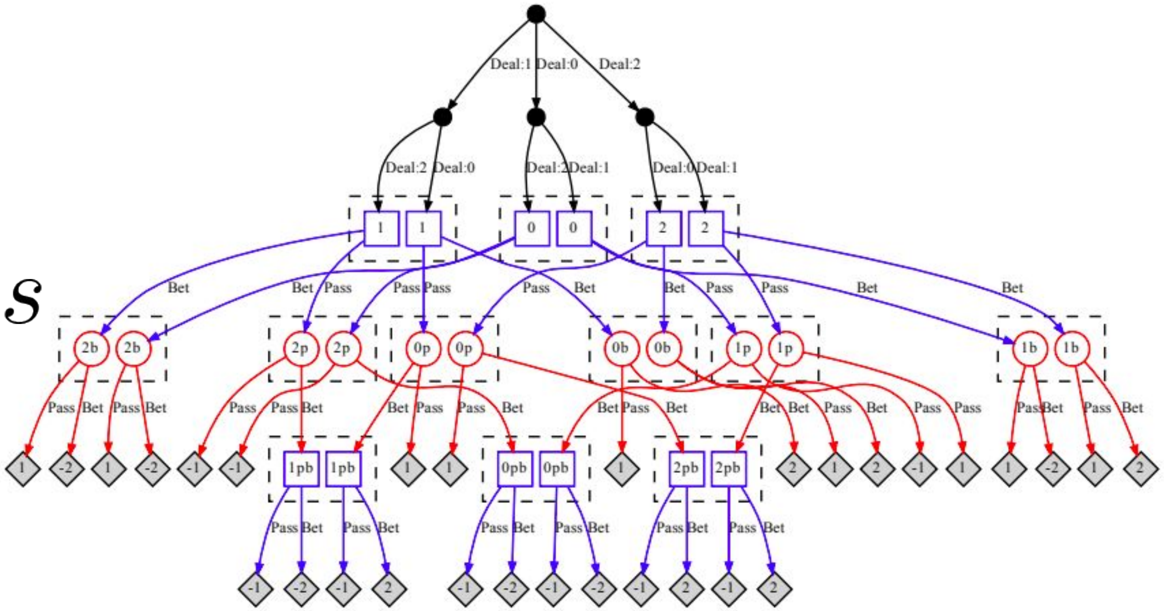
CFR Example

Two histories:

- h = 12b
- h′ = 02b

$$q^c_{\pi,2}(s,p) = -\frac{1}{12}$$

$$q^c_{\pi,2}(s,b) = \frac{1}{6}$$



CFR Example

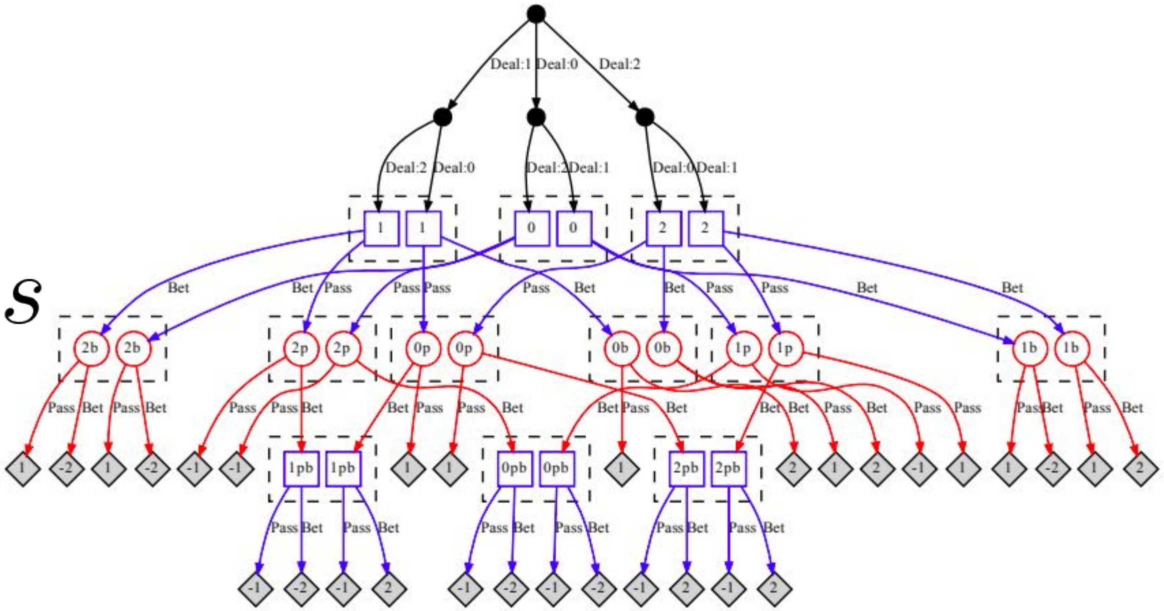
Two histories:

- h = 12b
- h′ = 02b

$$q^c_{\pi,2}(s,p) = -\frac{1}{12}$$

$$q^c_{\pi,2}(s,b) = \frac{1}{6}$$

$$v^c_{\pi,2}(s) = \frac{1}{2}\cdot\left(-\frac{1}{12}\right) + \frac{1}{2}\cdot\frac{1}{6}$$



CFR Example

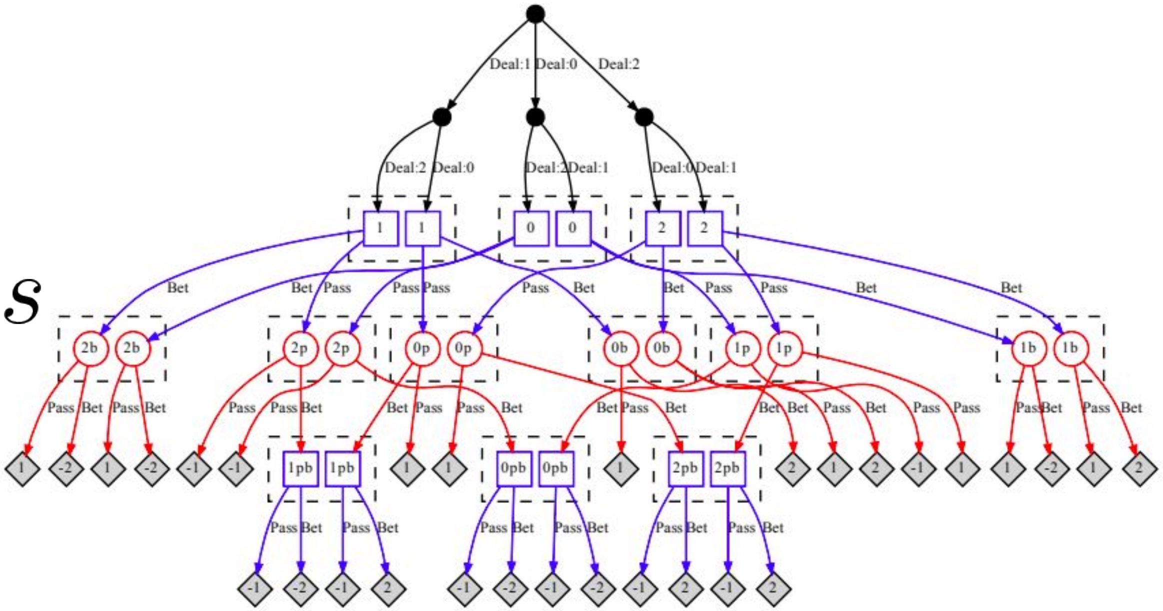
Two histories:

- h = 12b
- h′ = 02b

$$q^c_{\pi,2}(s,p) = -\frac{1}{12}$$

$$q^c_{\pi,2}(s,b) = \frac{1}{6}$$

$$v^c_{\pi,2}(s) = \frac{1}{24}$$



CFR Example

Two histories:

- h = 12b
- h′ = 02b

$$q^c_{\pi,2}(s,p) = -\frac{1}{12}$$

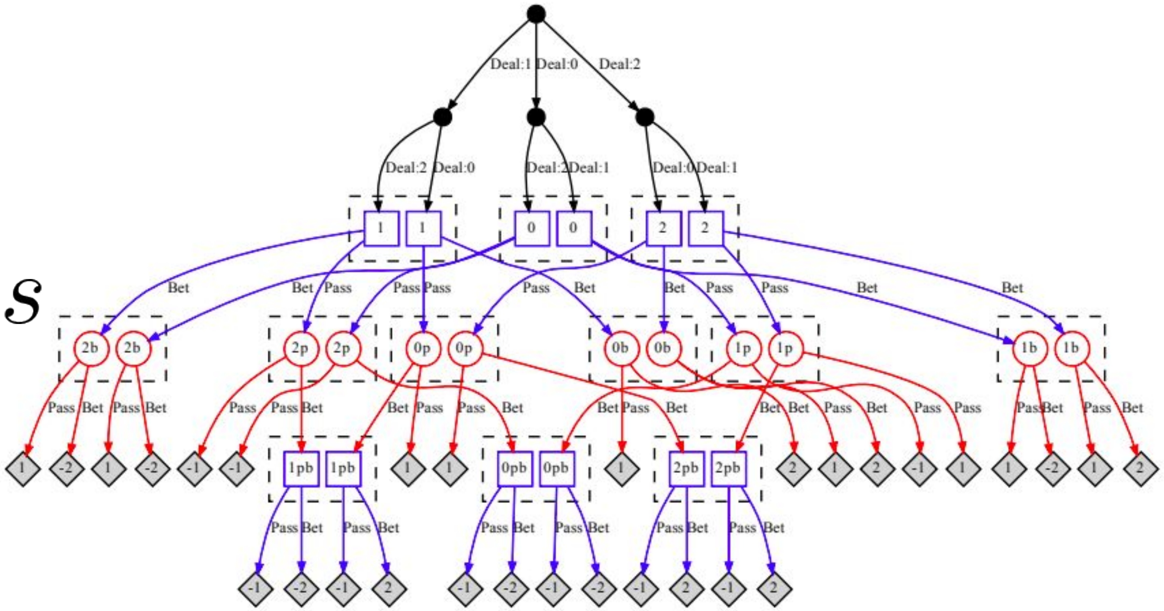
$$q^c_{\pi,2}(s,b) = \frac{1}{6}$$

$$v^c_{\pi,2}(s) = \frac{1}{24}$$

$$r(s,p) = q^c_{\pi,2}(s,p) - v^c_{\pi,2}(s) = -\frac{3}{24}$$

$$r(s,b) = \frac{3}{24}$$

→ Update policy: $\pi(s,p) = 0, \pi(s,b) = 1$



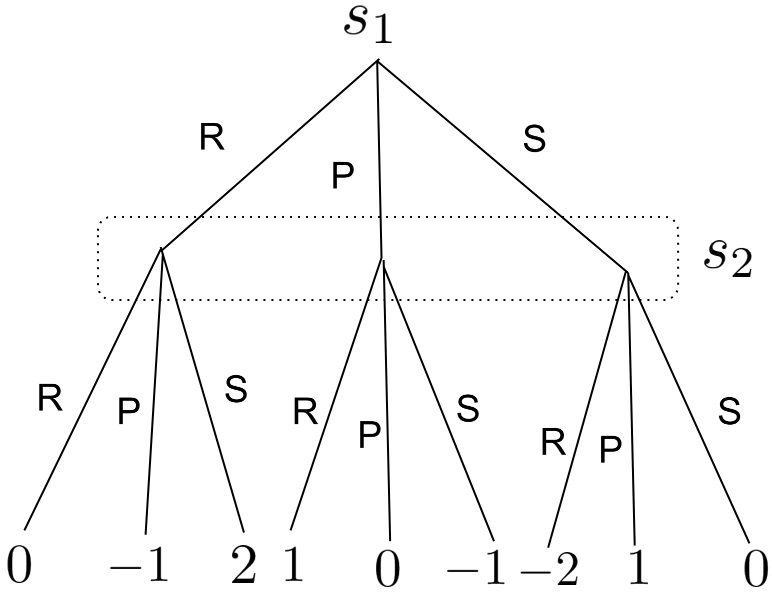
CFR Exercise

Biased Rock, Paper, Scissors: (utility for first player shown)

	R	P	S
R	0	-1	ν
P	1	0	-1
S	$-\nu$	1	0

Assume $\nu = 2$.

- What is the policy at both states after one iteration of CFR?
- By inspection: what action will have the largest regret for player 2 in the next iteration? How does this affect their policy?



DeepMind

3.2a Expected values vs. counterfactual values

Advantage vs. Regrets

A key notion in CFR is an **immediate regret**:

$$r(s, a) = q^c_{\pi,i}(s, a) - v^c_{\pi,i}(s)$$

Advantage vs. Regrets

A key notion in CFR is an **immediate regret**:

$$r(s, a) = q^c_{\pi, i}(s, a) - v^c_{\pi, i}(s)$$

counterfactual q-value

joint policy

return to player i

(player to play at s)

Advantage vs. Regrets

A key notion in CFR is an **immediate regret**:

$$r(s, a) = q^c_{\pi,i}(s, a) - v^c_{\pi,i}(s)$$

counterfactual q-value

joint policy

return to player i

(player to play at s)

$\rightarrow$ This is just a (counterfactual) advantage!

RL values vs. Counterfactual values

What................... is a q-value?

$$q_{\pi,i}(s, a)$$

RL values vs. Counterfactual values

What.................... is a q-value?

$$q_{\pi,i}(s, a)$$

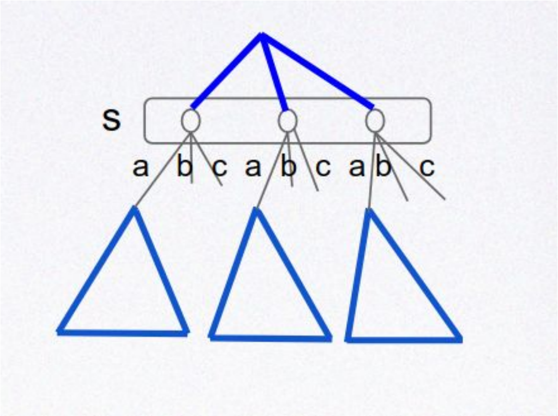
Exp. return playing from s **given:**

s reached, take a, then follow π

RL values vs. Counterfactual values

What.................... is a q-value?

$$q_{\pi,i}(s, a)$$



Exp. return playing from s **given:**

s reached, take a , then follow π

RL values vs. Counterfactual values

What........ is a counterfactual value?

$$q^{c}_{\pi,i}(s, a)$$

RL values vs. Counterfactual values

What........ is a counterfactual value?

$$q^{c}_{\pi,i}(s, a)$$

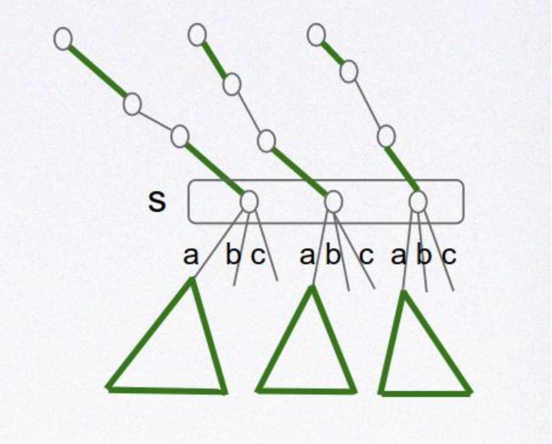
Portion of the exp. return to player i from start, **given:**

player i plays to get to s (others use π), then take a

RL values vs. Counterfactual values

What........ is a counterfactual value?

$$q^{c}_{\pi,i}(s, a)$$



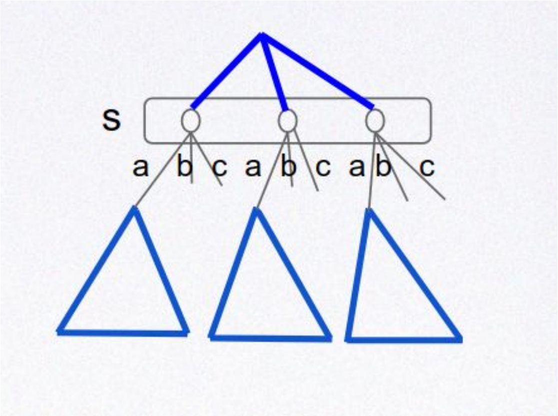
Portion of the exp. return to player i <u>from start</u>, **given:**

player i plays to get to s (others use π), then take a

RL values vs. Counterfactual values

What.................. is a q-value?

$$q_{\pi,i}(s, a)$$



$$q_{\pi,i}(s_t, a_t) = \mathbb{E}_{\rho\sim\pi}[G_t \mid S_t = s_t, A_t = a_t]$$

RL values vs. Counterfactual values

$$= \sum_{h,z \in \mathcal{Z}(s_t, a_t)} \Pr(h \mid s_t)\eta^{\pi}(ha, z)u_i(z)$$

All **terminal histories** z reachable from s, paired with their prefix histories ha, where h is in s

Reach probabilities: product of all policies' state-action probabilities along the portion of the history between ha and z

Return achieved over terminal history z

RL values vs. Counterfactual values

$$= \sum_{h,z \in \mathcal{Z}(s_t, a_t)} \frac{\Pr(s_t \mid h) \Pr(h)}{\Pr(s_t)} \eta^{\pi}(ha, z) u_i(z)$$

by Bayes' rule

RL values vs. Counterfactual values

$$= \sum_{h,z \in \mathcal{Z}(s_t,a_t)} \frac{\Pr(h)}{\Pr(s_t)} \eta^{\pi}(ha, z) u_i(z)$$

Since h is in s_t and unique to s_t

RL values vs. Counterfactual values

$$= \sum_{h,z \in \mathcal{Z}(s_t, a_t)} \frac{\eta^\pi(h)}{\sum_{h' \in s_t} \eta^\pi(h')} \eta^\pi(ha, z) u_i(z)$$

RL values vs. Counterfactual values

Only player i's reach probabilities

Player i's opponent's probabilities (*inc. chance*!)

$$\sum_{h,z \in \mathcal{Z}(s_t,a_t)} \frac{\eta_i^\pi(h)\eta_{-i}^\pi(h)}{\sum_{h' \in s_t} \eta_i^\pi(h')\eta_{-i}^\pi(h')} \eta^\pi(ha,z)u_i(z)$$

Similarly here

and here

RL values vs. Counterfactual values

$$= \sum_{h,z \in \mathcal{Z}(s_t, a_t)} \frac{\eta_i^\pi(h)\eta_{-i}^\pi(h)}{\eta_i^\pi(h) \sum_{h' \in s_t} \eta_{-i}^\pi(h')} \eta^\pi(ha, z) u_i(z)$$

Due to perfect recall!

RL values vs. Counterfactual values

$$= \sum_{h,z \in \mathcal{Z}(s_t, a_t)} \frac{\eta^{\pi}_{-i}(h)}{\sum_{h' \in s_t} \eta^{\pi}_{-i}(h')} \eta^{\pi}(ha, z) u_i(z)$$

RL values vs. Counterfactual values

$$= \sum_{h,z \in \mathcal{Z}(s_t,a_t)} \frac{\eta^{\pi}_{-i}(h)}{\sum_{h' \in s_t} \eta^{\pi}_{-i}(h')} \eta^{\pi}(ha, z) u_i(z)$$

This is a counterfactual value!

RL values vs. Counterfactual values

$$= \frac{1}{\sum_{h \in s_t} \eta^{\pi}_{-i}(h)} q^{c}_{\pi,i}(s_t, a_t)$$

$$= \frac{1}{\beta_{-i}(\pi, s)} q^{c}_{\pi,i}(s_t, a_t)$$

Q-based Policy Gradient

A.K.A. "all-actions" policy gradient

A.K.A. Mean Actor-Critic (Allen et al. '17)

$$\nabla_{\boldsymbol{\theta}}^{\mathrm{QPG}}(s) = \sum_a [\nabla_\theta \pi(s, a; \boldsymbol{\theta})] \left(q(s, a; \mathbf{w}) - \sum_b \pi(s, b; \boldsymbol{\theta}) q(s, b, \mathbf{w}) \right)$$

Regret-based Policy Gradient (Srinivasan et al. '18)

Instead of maximizing objective, **minimize regret**:

$$\nabla_{\boldsymbol{\theta}}^{\text{RPG}}(s) = -\sum_{a} \nabla_{\theta} \left(q(s,a;\mathbf{w}) - \sum_{b} \pi(s,b;\boldsymbol{\theta}) q(s,b;\mathbf{w}) \right)^{+}$$

Gradient **descent** (instead of ascent)

DeepMind

3.2b Monte Carlo Counterfactual Regret Minimization

Counterfactual Minimization

CFR is special kind of policy iteration:

1. Evaluate policy to compute counterfactual values: $q^c_{\pi,i}(s,a)$, $v^c_{\pi,i}(s)$
2. Improve the policy (using *state-local* regret minimization)
3. Compute an average joint policy $\bar{\pi} = (\bar{\pi}_1, \bar{\pi}_2)$

CFR Theorem: $\bar{\pi}$ converges to an ϵ-Nash eq. with $\epsilon \leq O\left(\frac{1}{\sqrt{T}}\right)$

Monte Carlo Counterfactual Minimization

MCCFR is sample-based CFR:

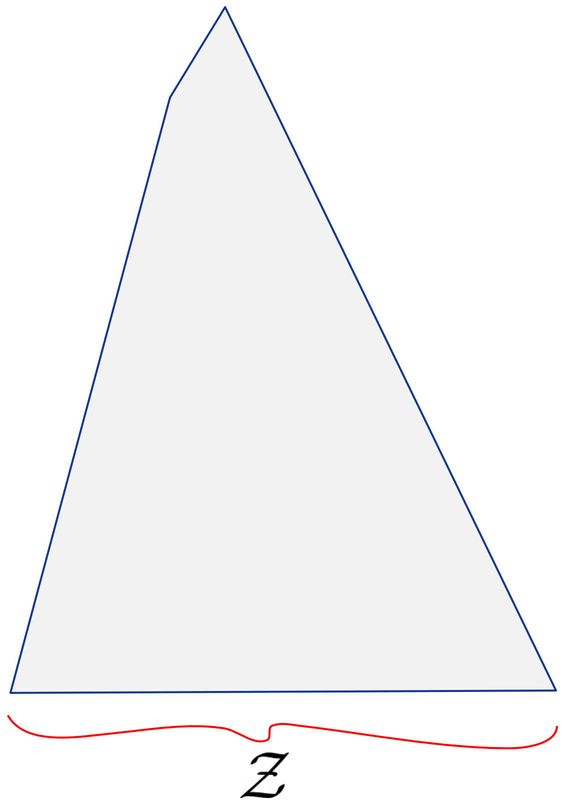
1. Evaluate estimated counterfactual values: $\hat{q}^c_{\pi,i}(s,a)$, $\hat{v}^c_{\pi,i}(s)$
2. Improve the policy (using *state-local* regret minimization)
3. Compute an estimated average joint policy $\hat{\bar{\pi}} = (\hat{\bar{\pi}}_1, \hat{\bar{\pi}}_2)$

MCCFR Theorem: with probability $1-p$, $\hat{\bar{\pi}}$ converges to an ϵ-Nash eq. with

$$\epsilon \leq O\left(\frac{1}{\delta\sqrt{pT}}\right)$$

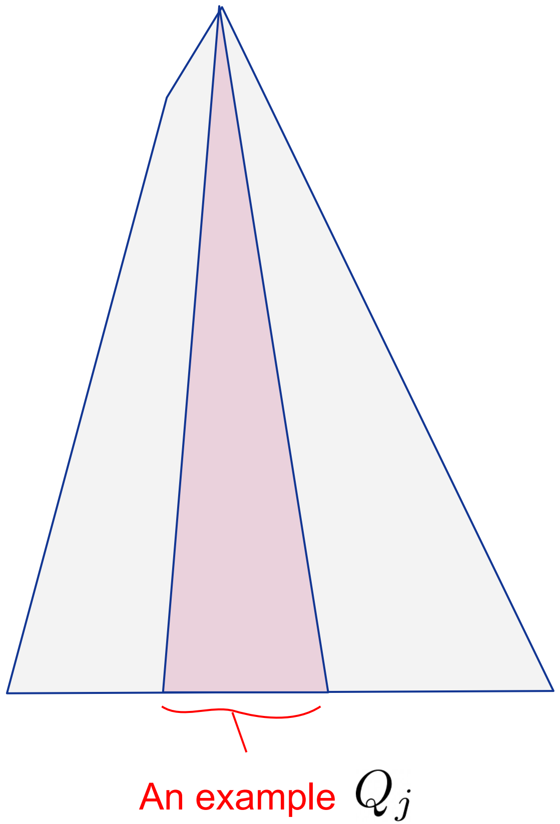
depends on sampling scheme and structure of game

MCCFR Overview



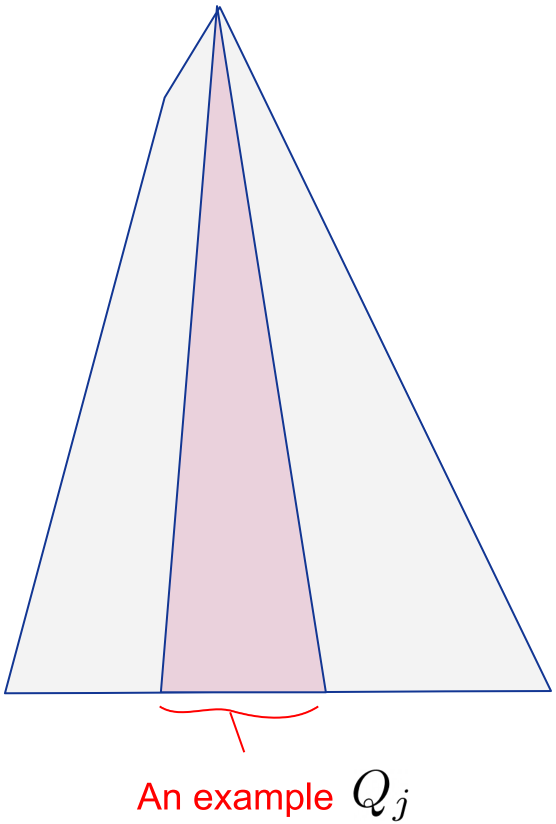
- All terminal histories: $\mathcal{Z}$

MCCFR Overview



- All terminal histories: $\mathcal{Z}$
- Define **blocks** $Q_j \in \mathcal{Q}$:
  - $Q_j \subseteq \mathcal{Z}$ for all j
  - $\cup_j Q_j = \mathcal{Z}$

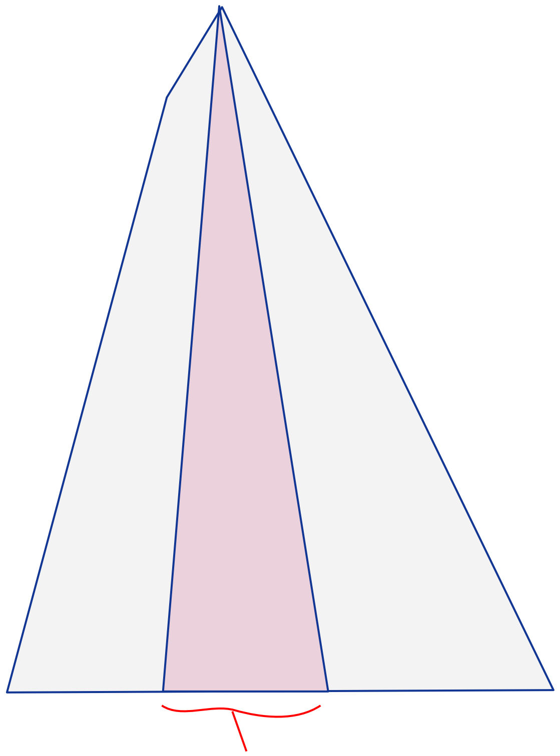
MCCFR Overview



- All terminal histories: $\mathcal{Z}$
- Define **blocks** $Q_j \in \mathcal{Q}$ **:**
    - $Q_j \subseteq \mathcal{Z}$ for all j
    - $\cup_j Q_j = \mathcal{Z}$
- **Sampled counterfactual values:**

$$\tilde{v}^c_{\pi,i}(s|j) \quad \tilde{q}^c_{\pi,i}(s,a|j)$$

MCCFR Overview



- All terminal histories: $\mathcal{Z}$
- Define **blocks** $Q_j \in \mathcal{Q}$:
  - $Q_j \subseteq \mathcal{Z}$ for all j
  - $\cup_j Q_j = \mathcal{Z}$
- **Sampled counterfactual values:**

$$\tilde{v}^c_{\pi,i}(s|j) \quad \tilde{q}^c_{\pi,i}(s,a|j)$$

- **Sampled counterfactual regret:**

$$\tilde{r}_{\pi,i}(s,a) = \tilde{q}^c_{\pi,i}(s,a|j) - \tilde{v}^c_{\pi,i}(s)$$

MCCFR Overview

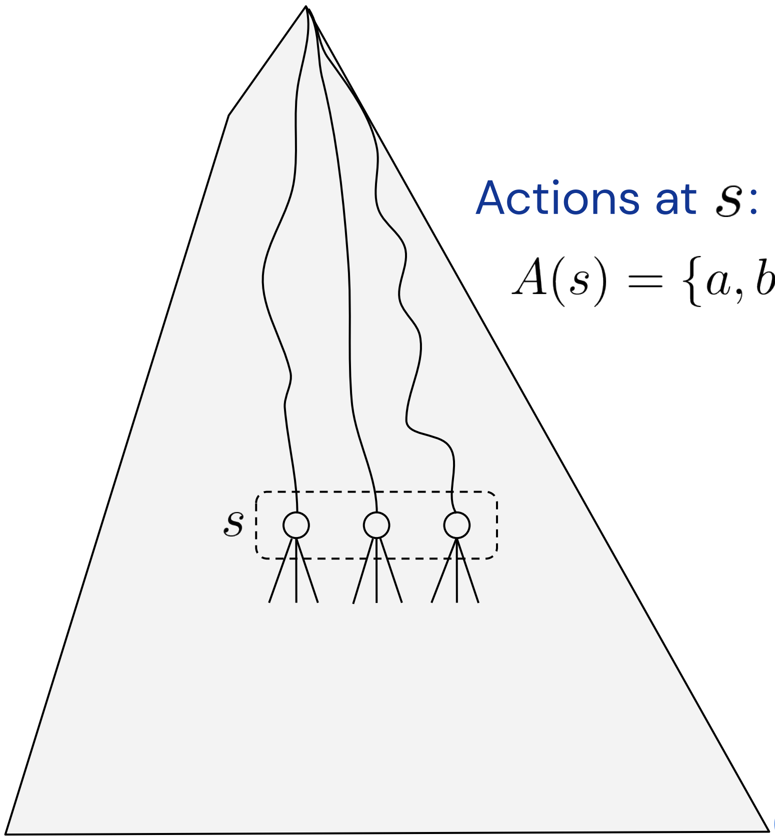
- Let $q_j = \Pr(Q_j)$

MCCFR Overview

- Let $q_j = \Pr(Q_j)$
- Let $q(z) = \sum_{j: z \in Q_j} q_j$

MCCFR Overview

- Let $q_j = \Pr(Q_j)$
- Let $q(z) = \sum_{j:z\in Q_j} q_j$

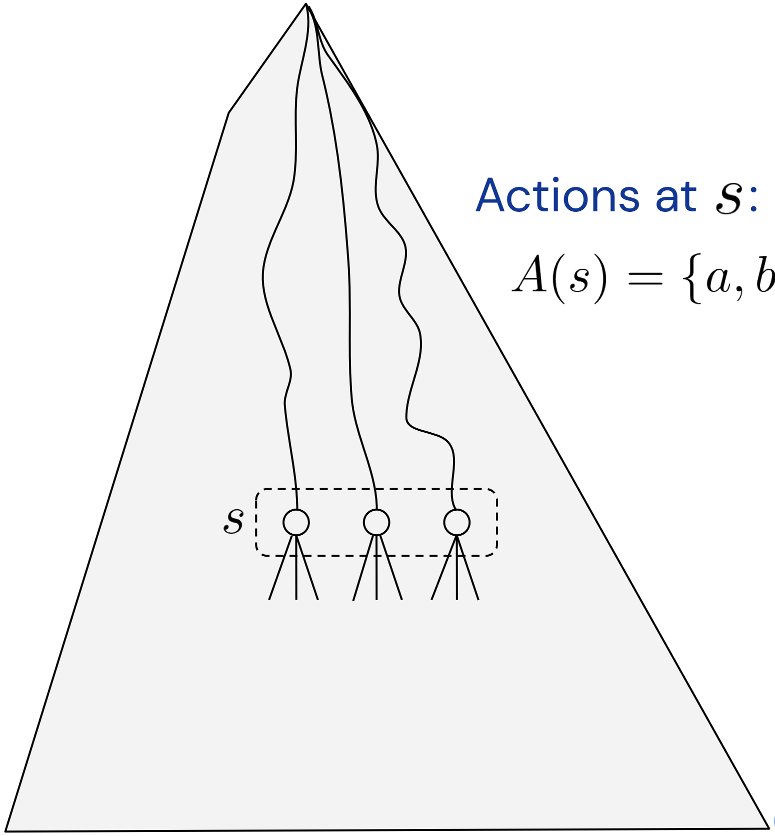


Actions at s:

$A(s) = \{a, b, c\}$

MCCFR Overview

- Let $q_j = \Pr(Q_j)$
- Let $q(z) = \sum_{j:z\in Q_j} q_j$
- Let $h \sqsubseteq z$ mean that h is a **prefix**

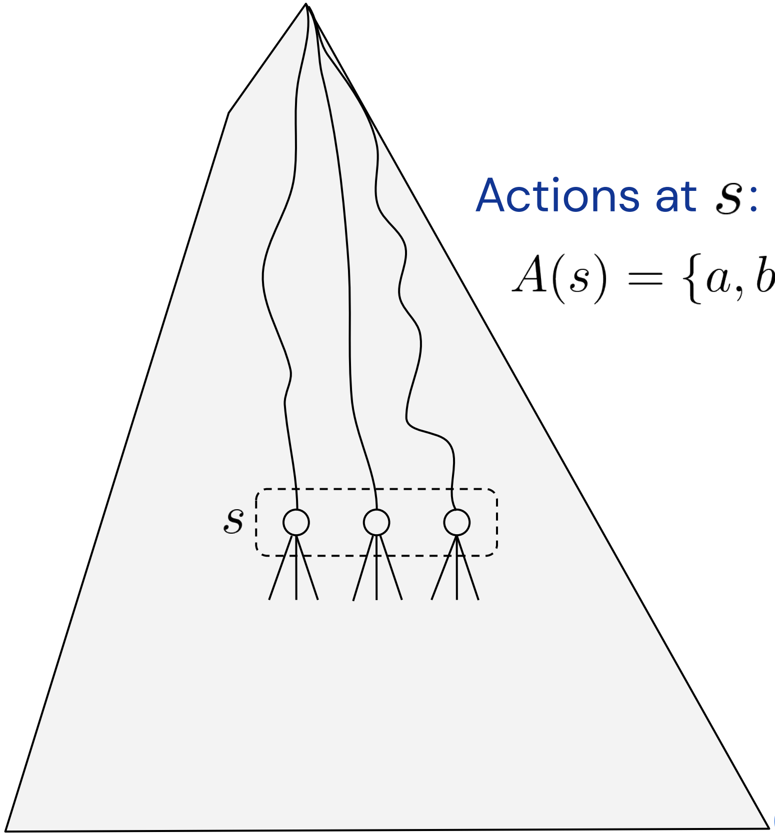


Actions at s:

$$A(s) = \{a, b, c\}$$

MCCFR Overview

- Let $q_j = \Pr(Q_j)$
- Let $q(z) = \sum_{j:z \in Q_j} q_j$
- Let $h \sqsubseteq z$ mean that h is a **prefix**
- Let $Z(s) = \{z \mid h \in s, h \sqsubseteq z\}$



Actions at s:

$A(s) = \{a, b, c\}$

MCCFR Overview

- Let $q_j = \Pr(Q_j)$
- Let $q(z) = \sum_{j:z \in Q_j} q_j$
- Let $h \sqsubseteq z$ mean that h is a **prefix**
- Let $Z(s) = \{z \mid h \in s, h \sqsubseteq z\}$

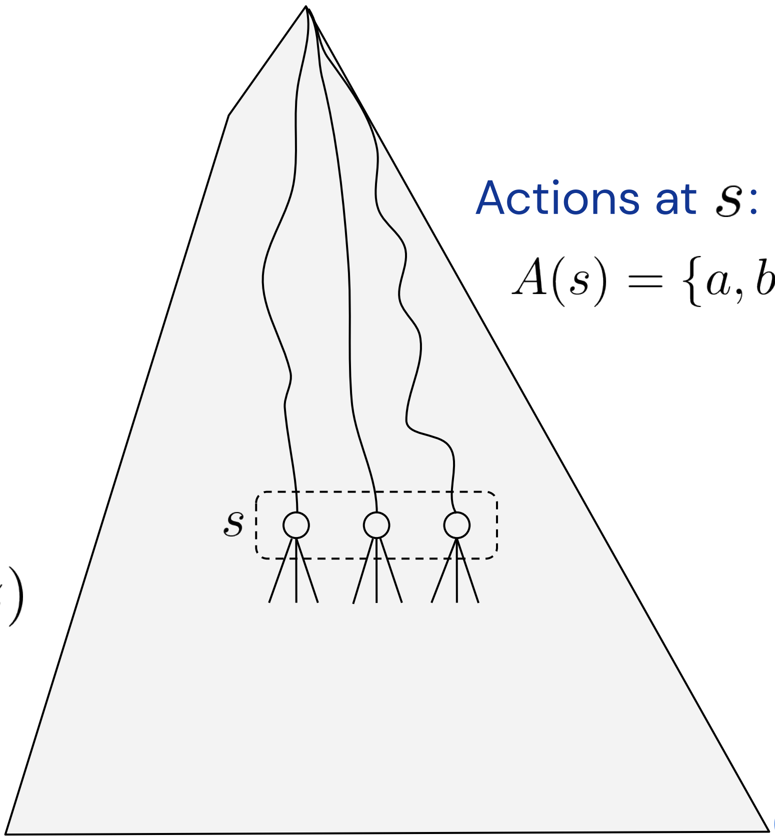
Sampled counterfactual value:

$$\tilde{v}^c_{\pi,i}(s|j) =$$

$$\sum_{h \in s, z \in Q_j \cap Z(s)} \frac{1}{q(z)} \eta^\pi_{-i}(h) \eta^\pi(h,z) u_i(z)$$

Reach probabilities

Utility to player i



Actions at s:

$$A(s) = \{a, b, c\}$$

MCCFR Overview

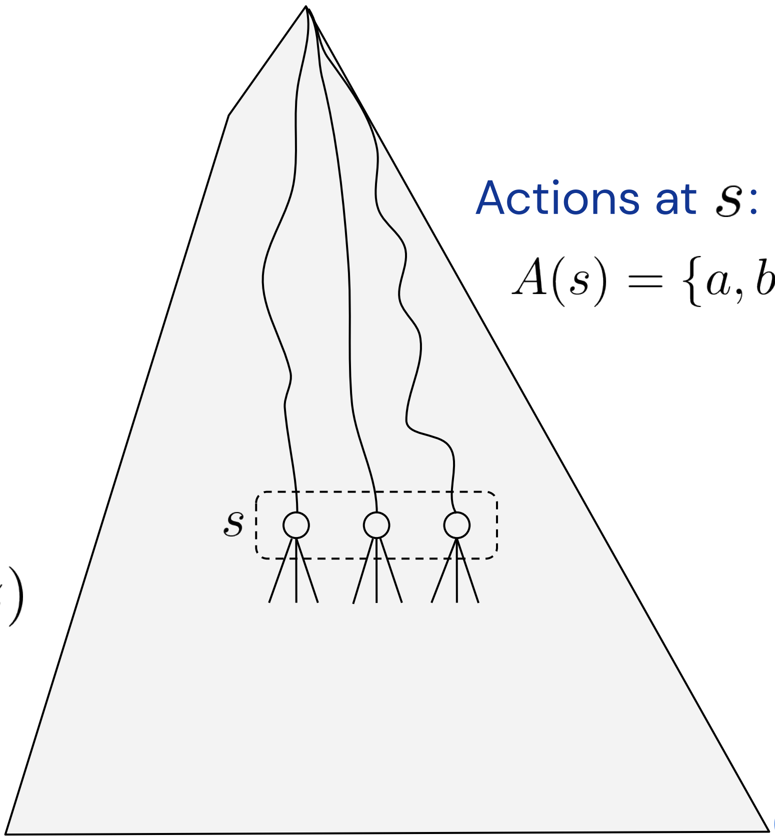
- Let $q_j = \Pr(Q_j)$
- Let $q(z) = \sum_{j : z \in Q_j} q_j$
- Let $h \sqsubseteq z$ mean that h is a **prefix**
- Let $Z(s) = \{z \mid h \in s, h \sqsubseteq z\}$

Sampled counterfactual value:

$$\tilde{v}^c_{\pi,i}(s|j) =$$

$$\sum_{h \in s, z \in Q_j \cap Z(s)} \frac{1}{q(z)} \eta^{\pi}_{-i}(h) \eta^{\pi}(h,z) u_i(z)$$

Importance sampling correction term



Actions at s:

$$A(s) = \{a, b, c\}$$

General MCCFR Lemma

$$\mathbb{E}[\tilde{v}^c_{\pi,i}(s|j)] = v^c_{\pi,i}(s)$$

General MCCFR Lemma

$$\mathbb{E}[\tilde{v}^{c}_{\pi,i}(s|j)] =$$

$$\sum_j q_j \tilde{v}_{\pi,i}(s|j)$$

General MCCFR Lemma

$$= \sum_j \sum_{h \in s, z \in Q_j \cap Z(s)} \frac{q_j}{q(z)} \eta^{\pi}_{-i}(h) \eta^{\pi}(h, z) u_i(z)$$

General MCCFR Lemma

$$= \sum_j \sum_{h \in s, z \in Q_j \cap Z(s)} \frac{q_j}{q(z)} \eta^{\pi}_{-i}(h) \eta^{\pi}(h, z) u_i(z)$$

$$= \sum_{z \in Z(s)} \frac{\sum_{j : z \in Q_j} q_j}{q(z)} \eta^{\pi}_{-i}(h) \eta^{\pi}(h, z) u_i(z)$$

General MCCFR Lemma

$$= \sum_{z \in Z(s)} \eta^{\pi}_{-i}(h)\eta^{\pi}(h, z)u_i(z)$$

General MCCFR Lemma

$$= \sum_{z \in Z(s)} \eta^{\pi}_{-i}(h)\eta^{\pi}(h, z)u_i(z)$$

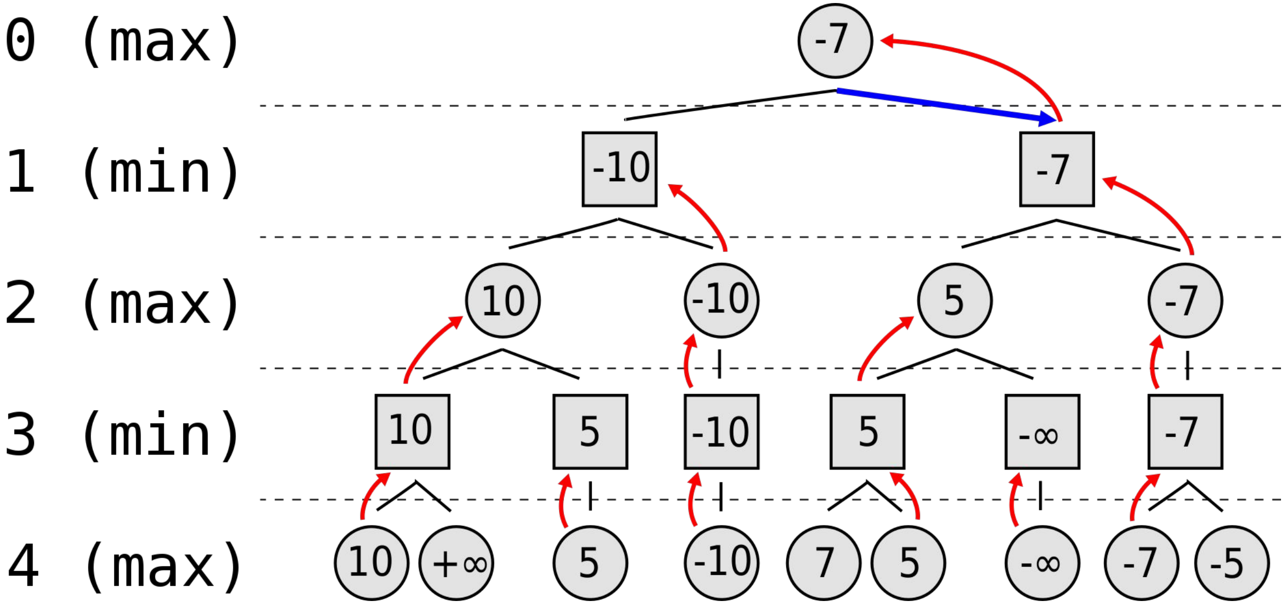
$$= v^{c}_{\pi,i}(s)$$

DeepMind

3.2c Search in Imperfect Information Games

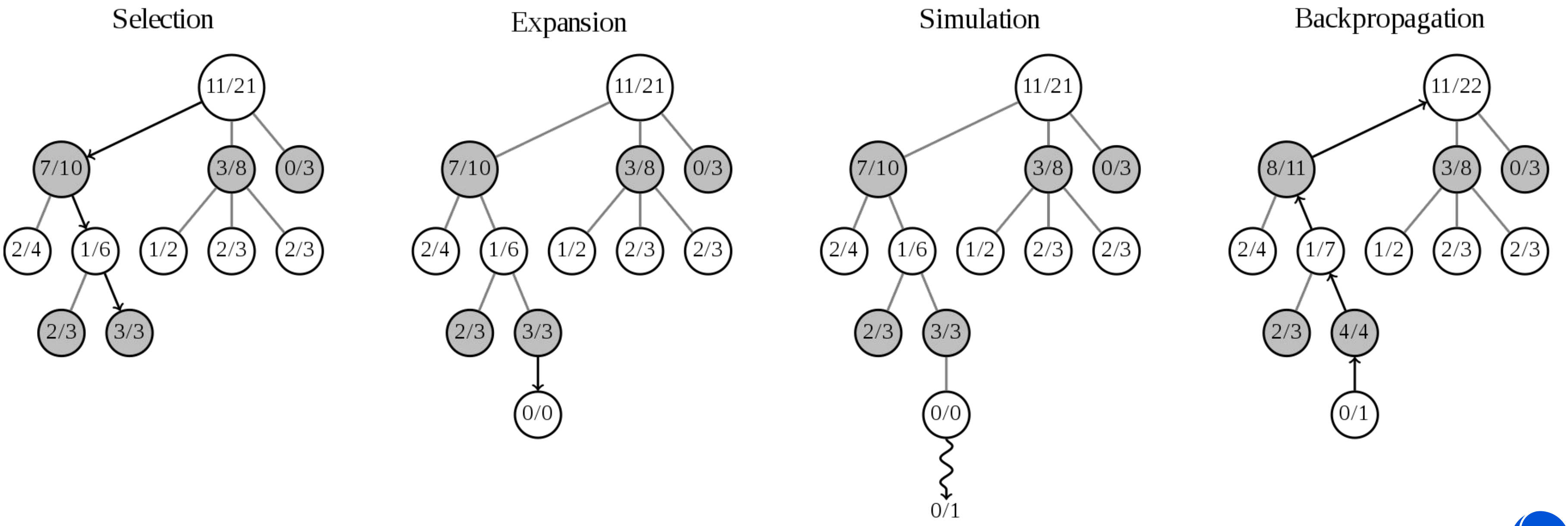
Search in Perfect Information Games

Classic Minimax game-tree search (von Neumann '28, Knuth & Moore '75)



Search in Perfect Information Games

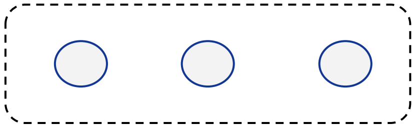
Monte Carlo Tree Search (MCTS) (Coulom '06, Kocsis & Szepesvari '06)



Search in Imperfect Information Games

One solution: Perfect Information (Monte Carlo / Minimax)

s



1. Repeat:
    a. Sample a world $h \sim D(s)$
    b. Recommandation = PerfInfoSearch(s)
2. Aggregate recommendations and choose a single action

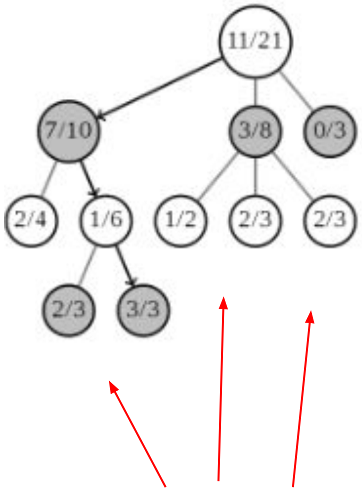
Two problems

- **Strategy fusion**: assumes one can use different strategies in different worlds— “*averaging over clairvoyance*” (Russell & Norvig)
- **Non-locality**: value of an information set is not expressable only from values of its subtrees

Fixing Strategy Fusion: Information Set MCTS

Aggregate MCTS statistics over information states!

1. Repeat:
   a. Sample a world $h \sim D(s)$
   b. Simulate using MCTS, storing store statististics at s s.t. $h \in s$
2. Return action with highest estimate



Nodes corresponds to information states, *not* worlds!

The Problem of Non-Locality (Lisy et al. '15)

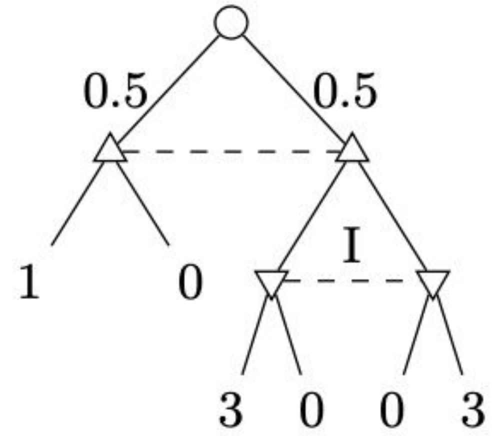


Figure 1: An extensive-form game demonstrating the problem of non-locality with maximizing △, minimizing ▽ and chance ○ players.

Subgame Decomposition

Solving Imperfect Information Games with Decomposition (Burch et al. '14)

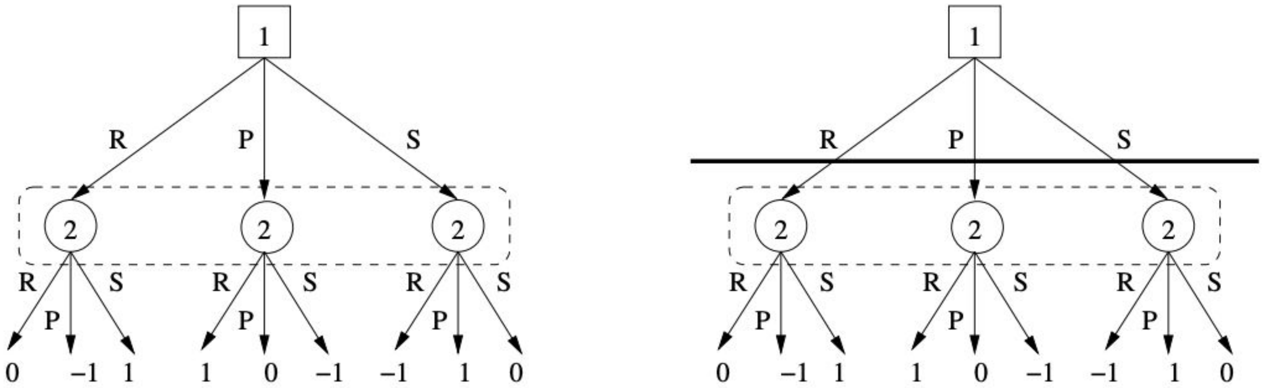


Figure 1: Left: rock-paper-scissors. Right: rock-paper-scissors split into trunk and one subgame.

Subgame Decomposition

"Solving Imperfect Information Games with Decomposition (Burch et al. '14)

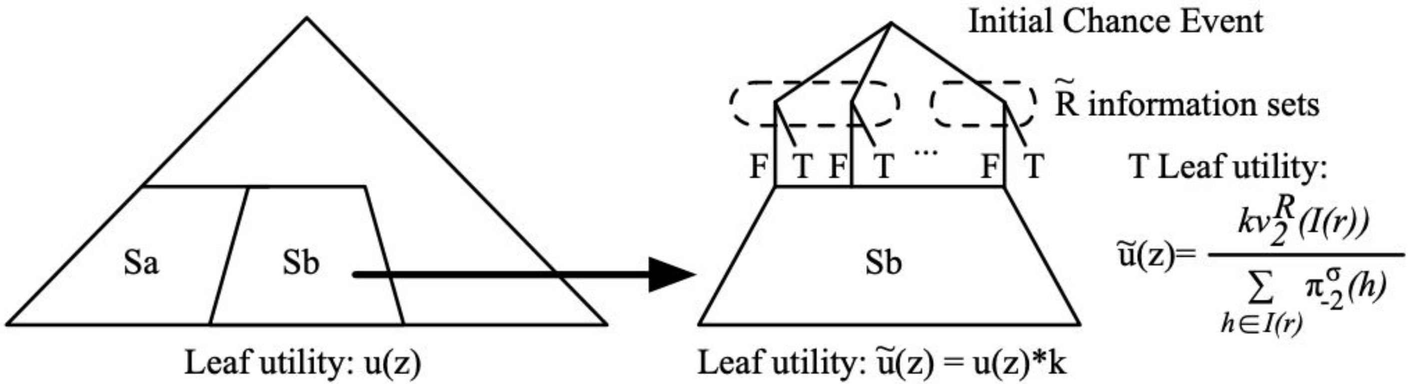


Figure 2: Construction of the Re-Solving Game

DeepMind

4 Practical Exercises with OpenSpiel

Plan

1. Intro + install and test OpenSpiel
2. Run the example
3. Experiments:
    a. Q-learning in Tic-Tac-Toe
    a. CFR in Kuhn Poker

OpenSpiel

- Open source framework for research in RL & Games
- C++, Python, and Swift impl's
- 25+ games
- 10+ algorithms

Released Aug' 19

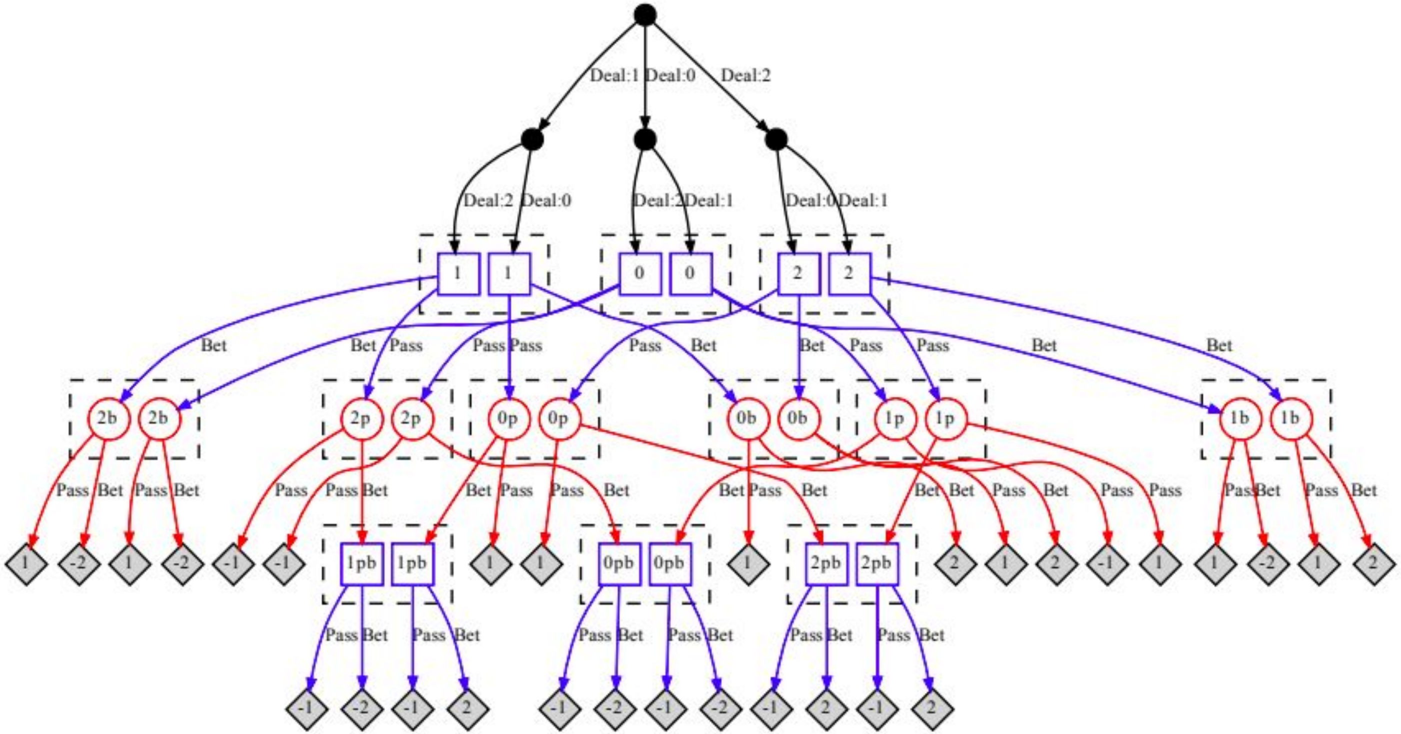
OpenSpiel

Supports:

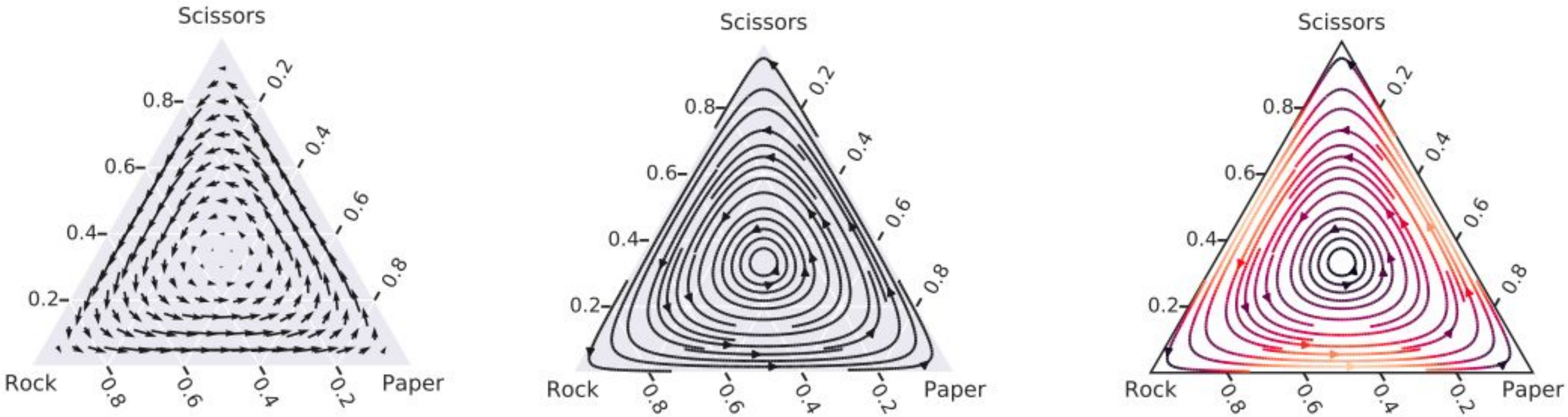
- n-player games
- Zero-sum, coop, general-sum
- Perfect / imperfect info
- Simultaneous-move games

Paper @ https://arxiv.org/abs/1908.09453

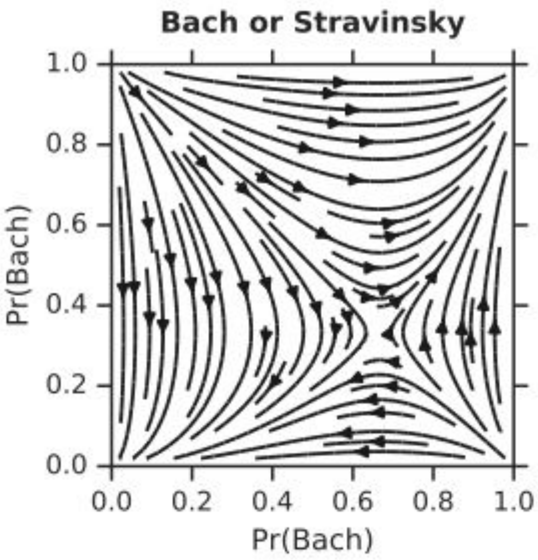
OpenSpiel: Example Viz (Kuhn Poker)

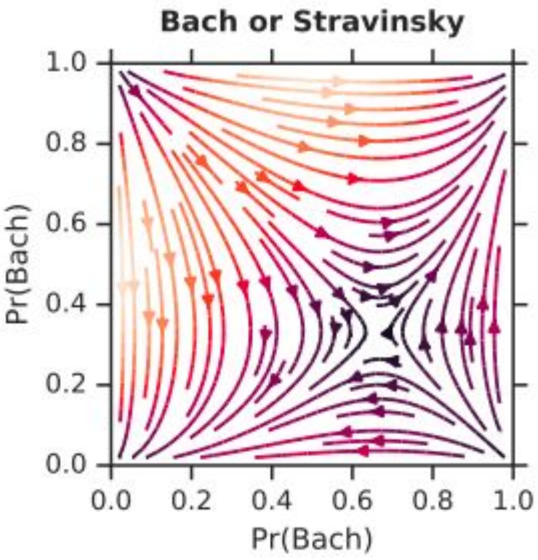


OpenSpiel: Example Viz (Replicator dynamics)



OpenSpiel: Example Viz (Replicator dynamics)





OpenSpiel: Design & Code

Design Philosophy

1. **Keep it simple.**
2. **Keep it light.**

Main structure:

- C++ core + Python API
- Swift port
- Go API (in the works)
- Games in C++
- Algs in C++ and Python
- Many examples / colabs

Example

```python
import random
import pyspiel
import numpy as np

game = pyspiel.load_game("kuhn_poker")
state = game.new_initial_state()
while not state.is_terminal():
  legal_actions = state.legal_actions()
  if state.is_chance_node():
    # Sample a chance event outcome.
    outcomes_with_probs = state.chance_outcomes()
    action_list, prob_list = zip(*outcomes_with_probs)
    action = np.random.choice(action_list, p=prob_list)
    state.apply_action(action)
  else:
    # The algorithm can pick an action based on an observation (fully observable
    # games) or an information state (information available for that player)
    # We arbitrarily select the first available action as an example.
    action = legal_actions[0]
    state.apply_action(action)
```

Install OpenSpiel

1. Full instructions on here https://github.com/deepmind/open_spiel
2. Fast install instruction on page 6 of https://arxiv.org/abs/1908.09453:

```
sudo apt-get install git cmake g++
git clone https://github.com/deepmind/open_spiel.git
cd open_spiel
./install.sh  # Install various dependencies (note: assumes Debian-based distro!)
pip3 install --upgrade -r requirements.txt  # Install Python dependencies
mkdir build
cd build
# Note: Python version installed should be >= Python_TARGET_VERSION specified here
CXX=g++ cmake -DPython_TARGET_VERSION=3.6 -DCMAKE_CXX_COMPILER=g++ ../open_spiel
make -j12   # The 12 here is the number of parallel processes used to build
ctest -j12  # Run the tests to verify that the installation succeeded
```

Run the Example

First, set the PYTHONPATH (can add this to .bashrc, .profile, or .bash_profile)

```
# For the Python modules in open_spiel.
export PYTHONPATH=$PYTHONPATH:/<path_to_open_spiel>
# For the Python bindings of Pyspiel
export PYTHONPATH=$PYTHONPATH:/<path_to_open_spiel>/build/python
```

Once built:

```
cd ..
python3 open_spiel/python/examples/example.py
```

Interact from Python directly


```
[lanctot@lanctot-macbookair2:open_spiel$
[lanctot@lanctot-macbookair2:open_spiel$ python3
Python 3.7.4 (default, Aug 27 2019, 23:45:03)
[Clang 10.0.1 (clang-1001.0.46.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
[>>> import pyspiel
[>>> game = pyspiel.load_game("tic_tac_toe")
[>>> state = game.new_initial_state()
[>>> print(state)
...
...
...
[>>> state.apply_action(4)
[>>> print(state)
...
.x.
...
[>>> print(state.legal_actions())
[0, 1, 2, 3, 5, 6, 7, 8]
[>>> print(state.is_terminal())
False
[>>> print(state.current_player())
1
>>> 
```

OpenSpiel Experiments

1. Run Q-learning in Tic-Tac-Toe for 100 episodes:
    a. Can you beat the agent?
    b. Try running it now for 100000 episodes? Is it harder to beat? If so, in what way?
2. Run CFR on Kuhn poker for 1 iteration:
    a. Print the current policy. What do you notice about the it? Can you explain?
    b. Print the average policy. What do you notice about the it? Can you explain?
    c. Now run for 1000 iterations. What does the average strategy look like? Can you explain its general form?
3. Now, try to run CFR on Tic-Tac-Toe. Any idea why it takes so long?

```
python3 open_spiel/python/examples/tic_tac_toe_qlearner.py --num_episodes=100
python3 open_spiel/python/examples/cfr_example.py --iterations=1
```

Hint for 2: add a `__str__` function to `python.policy.TabularPolicy`, which loops over `self.state_lookup`, then uses `action_probabilities` to get the policy for each info state

DeepMind

The end and thank you

Marc Lanctot
lanctot@google.com
mlanctot.info

06/11/2019