Multi-agent Reinforcement Learning (1)

Prof. Jun Wang

Computer Science, UCL

Oct 15 2018, SJTU

Tables of Content

- Multi-agent RL: introduction and concepts
- Stochastic Games
  - Policy Iteration/Value Iteration (model based)
  - Equilibrium Learners (model free)
    - Nash-Q
    - Minimax-Q
    - Friend-Foe-Q
  - Best-Response Learners (model free)
    - JAL and Opponent Modelling
    - Iterated Gradient Ascent
    - Wolf-IGA

Tables of Content

Reinforcement Learning

Action a_t

Reward r_{t+1}

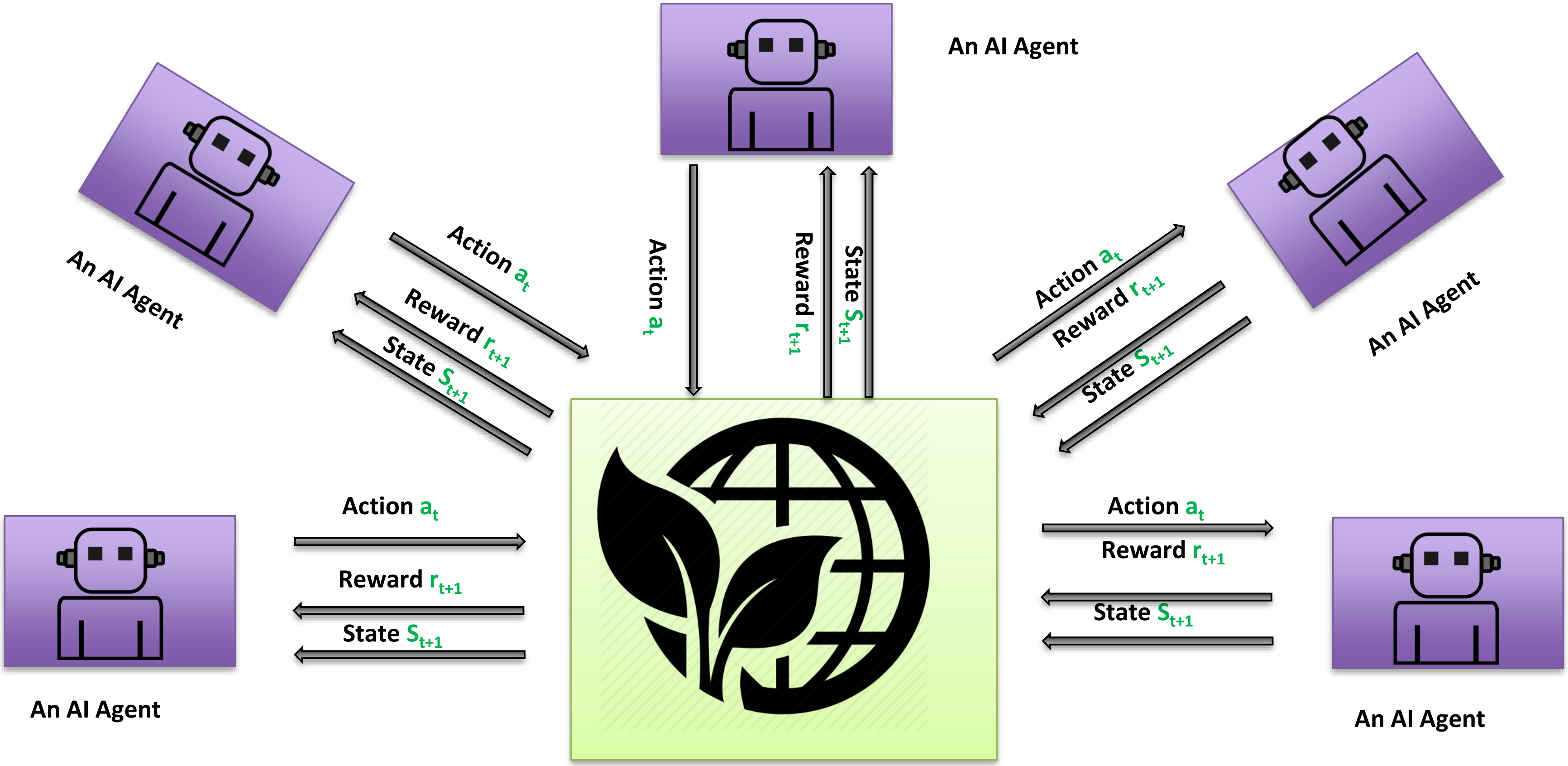
State S_{t+1}

An AI Agent

Environment

Optimal action policy a^* <--- Maximise $r_1 + r_2 + \ldots + r_t + \ldots$

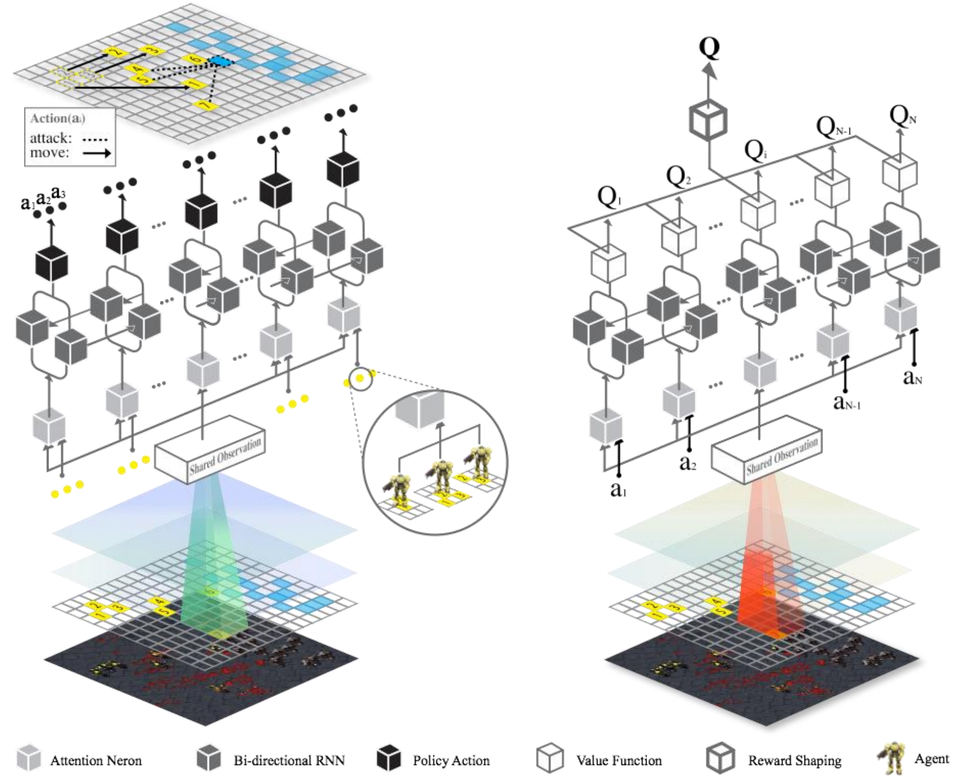
Multi-agent Reinforcement Learning



Environment

A set of autonomous agents that share a common environment

MARL Application: AI Plays Multiplayers Online Games



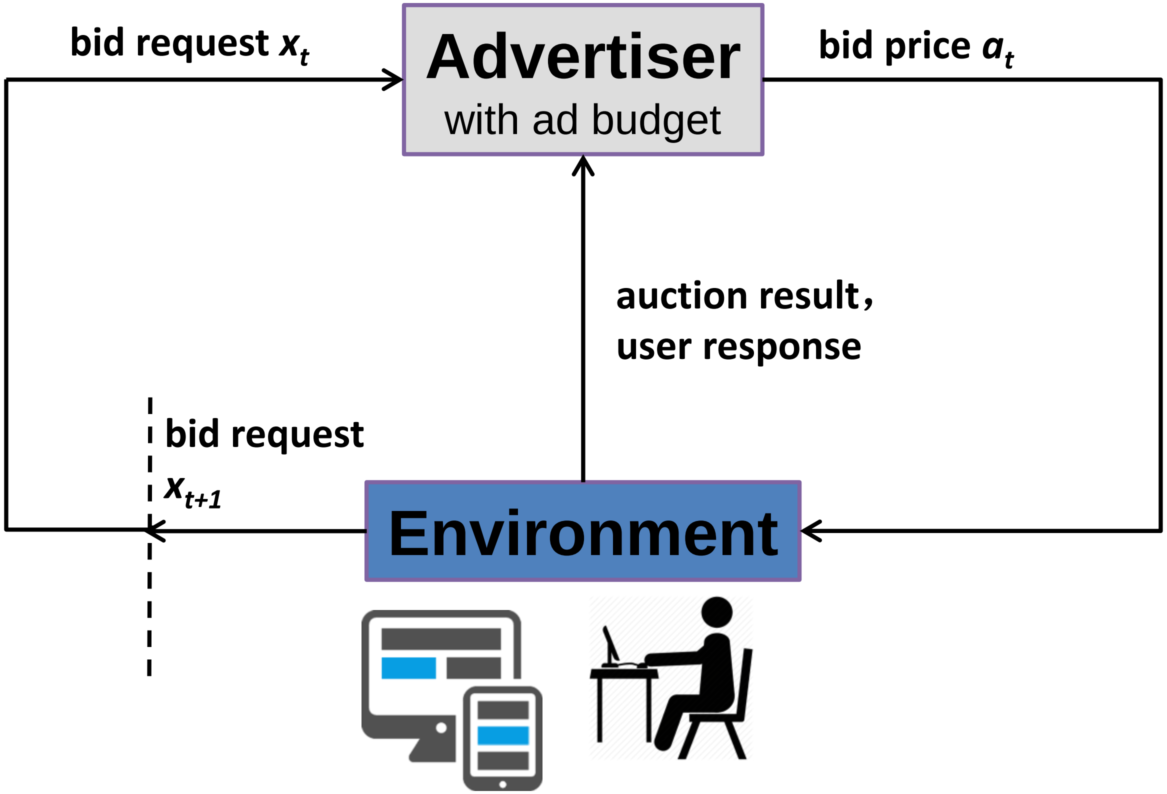
(a) Multiagent policy networks with grouping (b) Multiagent Q networks with reward shaping

https://www.youtube.com/watch?v=kW2q15MNFug

http://v.youku.com/v_show/id_XMjcyMTE0MDkwNA%3D%3D.html

Peng P, Yuan Q, Wen Y, Yang Y, Tang Z, Long H, Wang J. Multiagent Bidirectionally-Coordinated Nets for Learning to Play StarCraft Combat Games. NIPS17 Emergent Communication Workshop. (**StarCraft AI beating Facebook methods**)

MARL Application: Bidding Machine in Online Advertising



bid request x_t

Advertiser

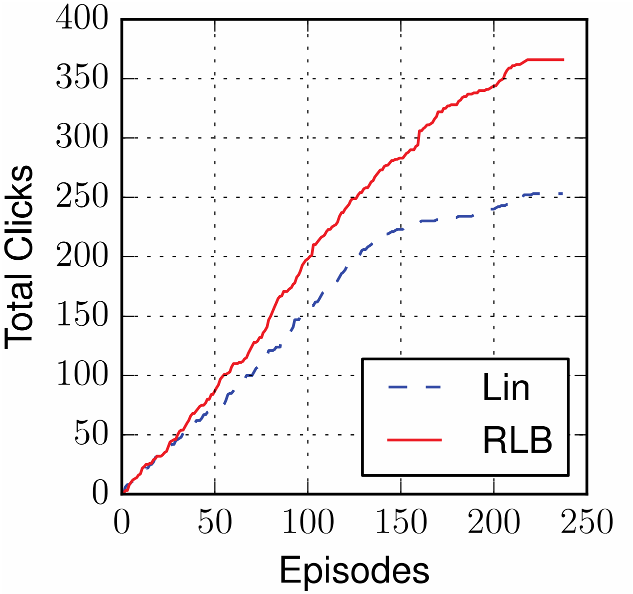
with ad budget

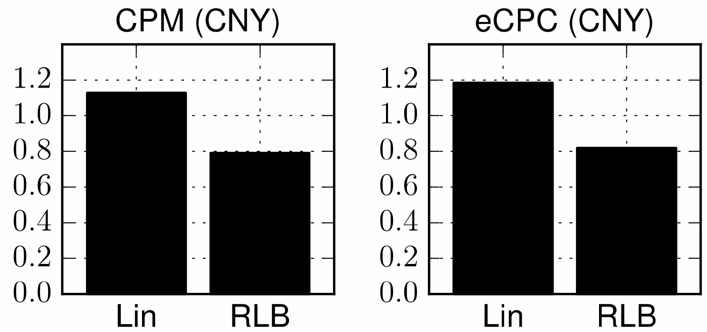
bid price a_t

auction result, user response

bid request x_{t+1}

Environment

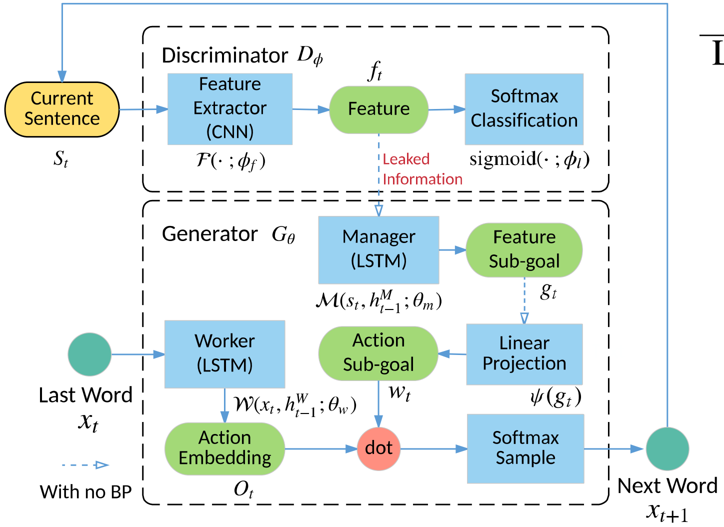


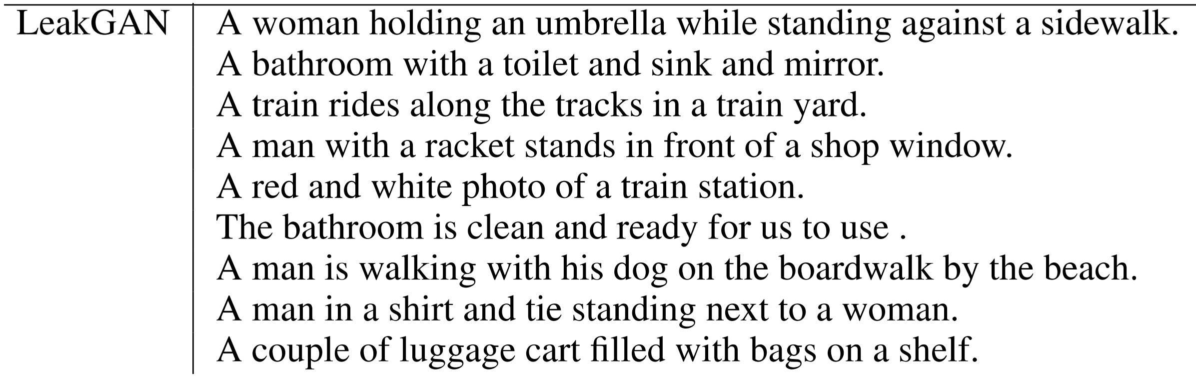


The goal is to maximise the user responses on displayed ads

Cai, H., K. Ren, W. Zhag, K. Malialis, and J. Wang. "Real-Time Bidding by Reinforcement Learning in Display Advertising." In *The Tenth ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 2017.

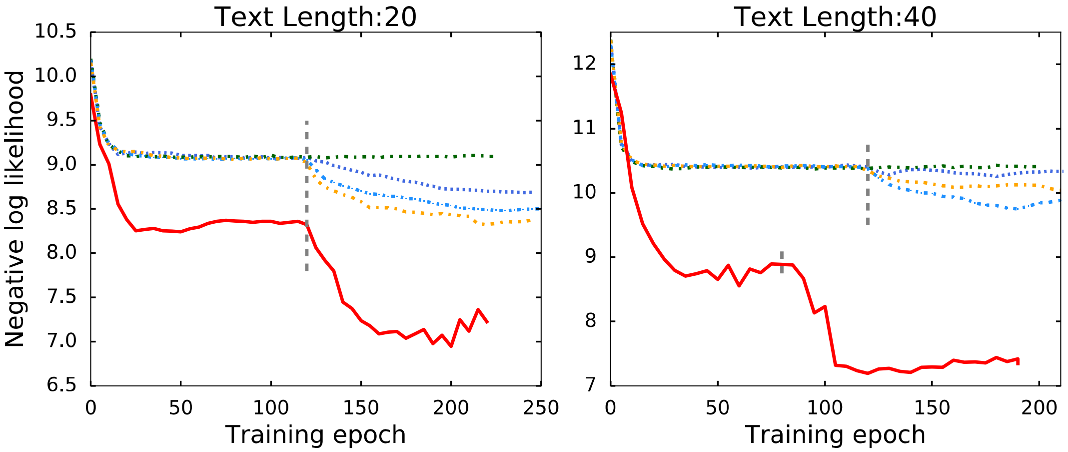
MARL Application: Text Generation





LeakGAN	A woman holding an umbrella while standing against a sidewalk.
	A bathroom with a toilet and sink and mirror.
	A train rides along the tracks in a train yard.
	A man with a racket stands in front of a shop window.
	A red and white photo of a train station.
	The bathroom is clean and ready for us to use .
	A man is walking with his dog on the boardwalk by the beach.
	A man in a shirt and tie standing next to a woman.
	A couple of luggage cart filled with bags on a shelf.

- The generator is responsible to generate the next word, and the discriminator adversarially judges the generated sentence
- The discriminator reveals its internal state to guide the generator more informatively and frequently.

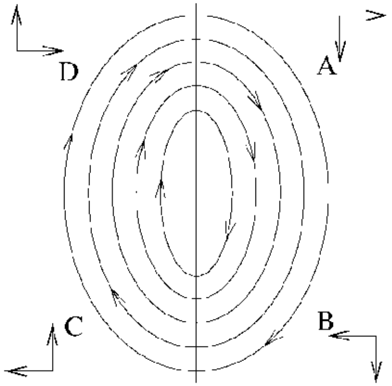


Long Text Generation via Adversarial Training with Leaked Information
Lianmin Zheng, Jiacheng Yang, Han Cai, Weinan Zhang, Jun Wang, and Yong Yu
arXiv:1709.08624v1, AAAI-2018

Difficulty in Multi-agent Learning(MAL)

- MAL is fundamentally difficult
  - since agents not only interact with the environment but also with each other
- If use single-agent Q learning by considering other agents as a part of the environment
  - Such a setting breaks the theoretical convergence guarantees and makes the learning unstable,

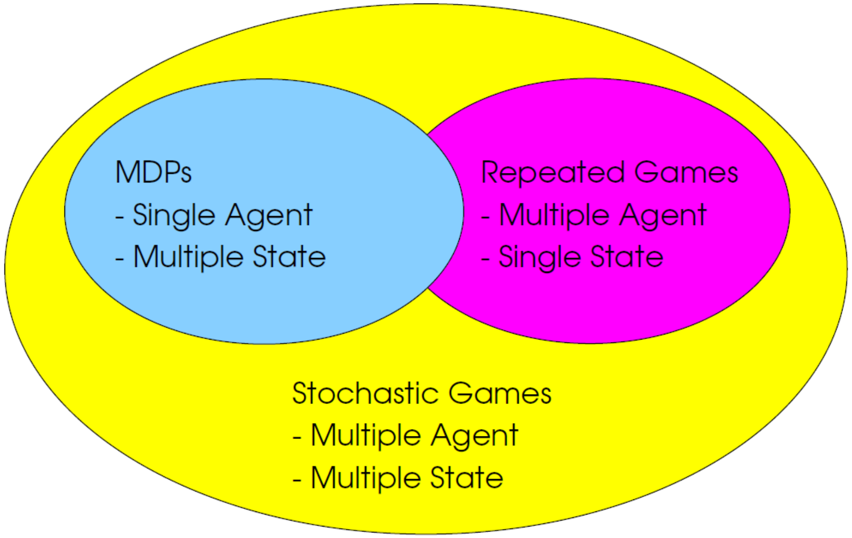
i.e., the changes in strategy of one agent would affect the strategies of other agents and vice versa



Sequential Decision Making

Includes:

- Markov decision processes
  - one decision maker
  - multiple states
- Repeated games
  - multiple decision makers
  - one state (e.g., one normal form game)
- Stochastic games (Markov games)
  - multiple decision makers
  - multiple states (e.g., multiple normal form games)



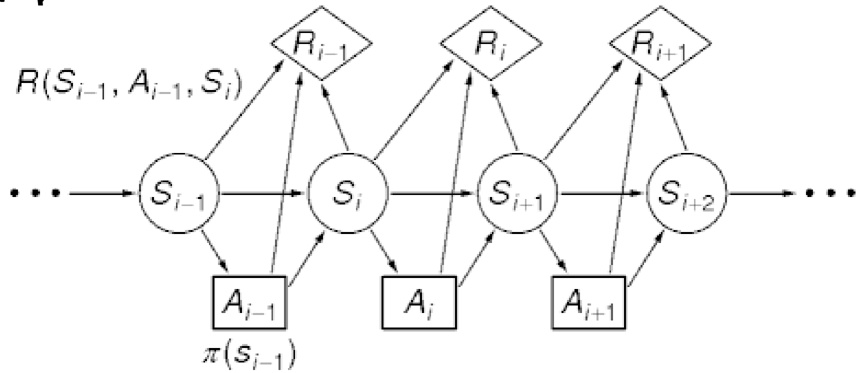
MDPs
- Single Agent
- Multiple State

Repeated Games
- Multiple Agent
- Single State

Stochastic Games
- Multiple Agent
- Multiple State

Recall: Markov Decision Processes

- MDP is a single-agent, multiple state framework
- A *Markov decision process* (MDP) is a tuple, (S, A, T , R),
  - where S is the set of States,
  - A is the set of actions,
  - T is a transition function S ×A×S →[0,1],
    - (The transition function defines a probability distribution over next states as a function of the current state and the agent's action), and
  - R is a reward function S × A → R.
    - (The reward function defines the reward received when selecting an action from the given state)
- Solving MDPs consists of finding a policy, π : S → A, mapping states to actions so as to maximize discounted future reward with discount factor γ

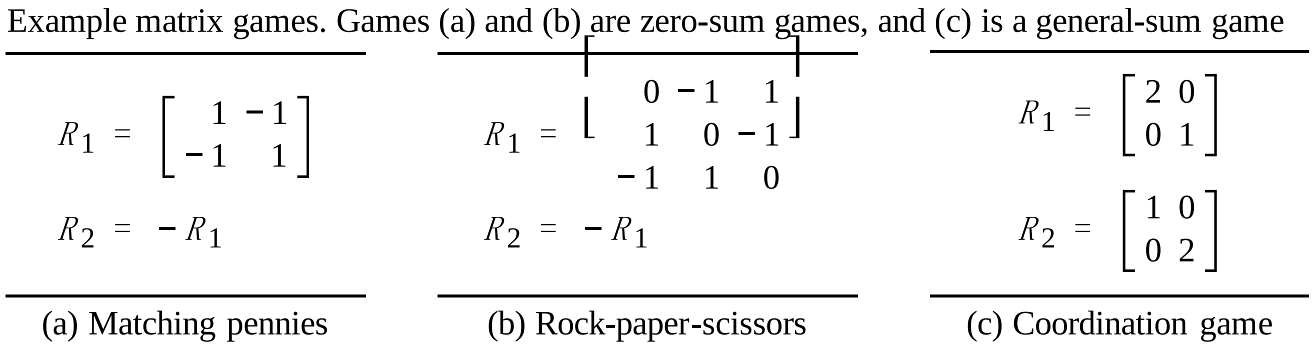


Recall: Matrix Games

- Matrix games are a multi-agent (player), single state framework
- A *matrix game* or normal-form *game* is a tuple $(n, A_{1 \ldots n}, R_{1 \ldots n})$, where
  - n is the number of players,
  - A_i is the set of actions available to player i
  - (and A is the joint action space $A_1 \times \cdots \times A_n$), and
  - R_i is player i's payoff function A → R.
- The players select actions from their available set and

receive a payoff that depends on *all* the players' actions.

- These are often called matrix games, since the R_i functions can be written as n-dimensional matrices



Example matrix games. Games (a) and (b) are zero-sum games, and (c) is a general-sum game

$$R_1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \qquad R_2 = -R_1$$

(a) Matching pennies

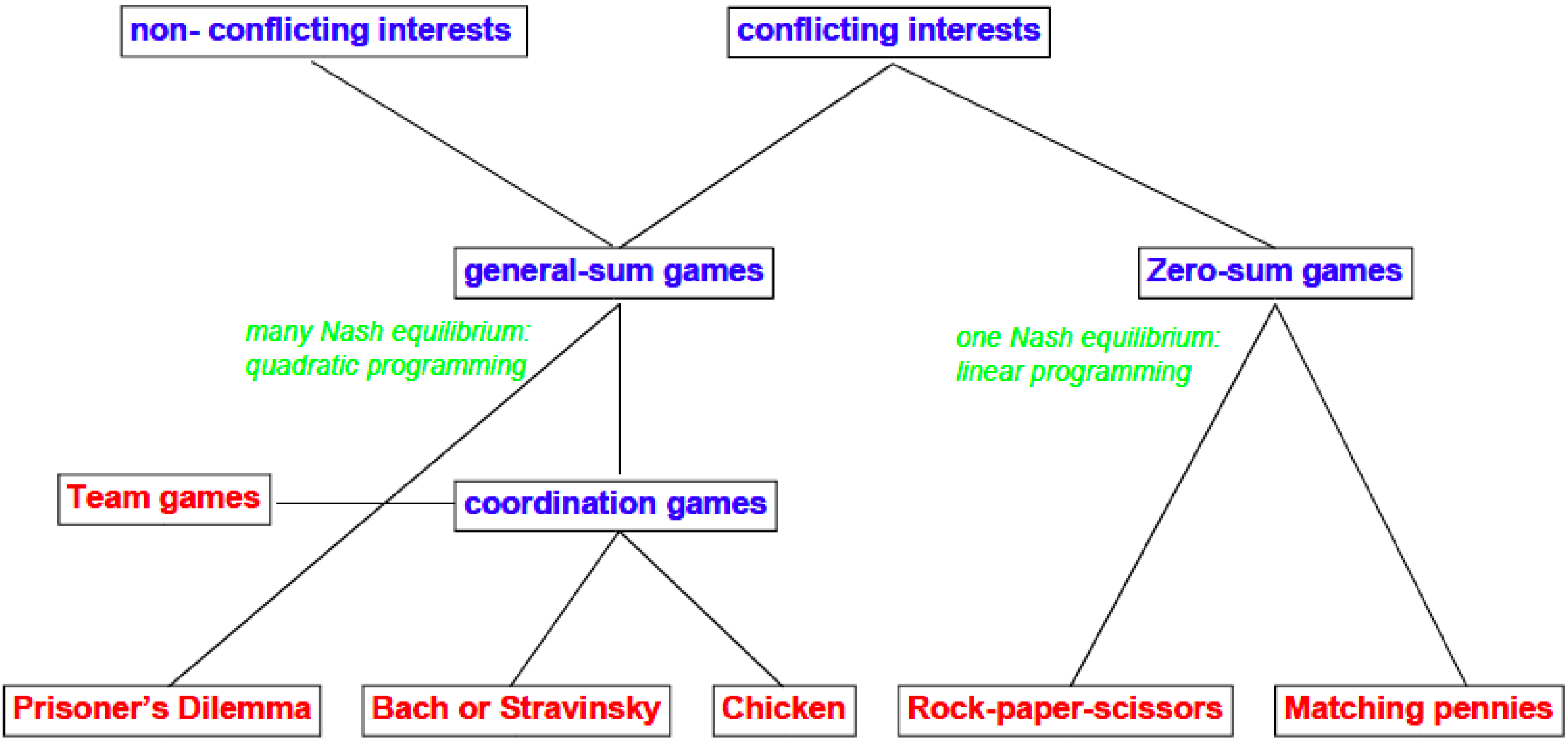
$$R_1 = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix} \qquad R_2 = -R_1$$

(b) Rock-paper-scissors

$$R_1 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix} \qquad R_2 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

(c) Coordination game

Recall: Matrix Games



non- conflicting interests

conflicting interests

general-sum games

Zero-sum games

many Nash equilibrium: quadratic programming

one Nash equilibrium: linear programming

Team games

coordination games

Prisoner's Dilemma

Bach or Stravinsky

Chicken

Rock-paper-scissors

Matching pennies

Exercise : 2x2 zero-sum game general solution

- Consider a general 2 × 2 zero-sum game matrix

$$A = \begin{pmatrix} a & b \\ d & c \end{pmatrix}$$

where a, b, c, d are the rewards for player 1 (row player). The reward matrix for col player is –**A**.

- find the value of the game and at least one optimal strategy for each player?
  - *Step 1, test whether there is pure strategy*
  - *Step 2, if not, solve by find equalizing strategy* (check previous slides on learning NE)

Game Theory, Second Edition, 2014 Thomas S. Ferguson Mathematics Department, UCLA

Exercise : 2x2 zero-sum game general solution

- Consider a general 2 × 2 zero-sum game matrix

$$A = \begin{pmatrix} a & b \\ d & c \end{pmatrix}$$

where a, b, c, d are the rewards for player 1 (col player)

- Step 1: in the situation there is no pure strategy:
  - If $a \geq b$, then $b < c$, as otherwise b is a pure strategy.
  - Since $b < c$, we must have $c > d$, as otherwise c is a pure strategy. Continuing thus, $d<a$ and $a>b$.
  - In other words, if $a \geq b$, then $a>b<c>d<a$. By symmetry, if $a \leq b$, then $a<b>c<d>a$.
- So the condition for no pure strategy:
  - either $a > b$, $b < c$, $c > d$ and $d < a$, or
  - $a < b$, $b > c$, $c<d$ and $d>a$.

Exercise : 2x2 zero-sum game general solution

- Consider a general 2 × 2 zero-sum game matrix

$$A = \begin{pmatrix} a & b \\ d & c \end{pmatrix}$$

where a, b, c, d are the rewards for player 1 (col player)

- Step 2: optimal strategies and value of the game
  - Player 1 chooses the first row with probability p (i.e. uses the mixed strategy (p, 1 – p)), we have

$$ap + d(1 - p) = bp + c(1 - p).$$

  - which gives $p = \frac{c - d}{(a - b) + (c - d)}.$
  - Player 1’s average return (the value of the game) using this strategy $v = ap + d(1 - p) = \frac{ac - bd}{a - b + c - d}.$

Exercise : 2x2 zero-sum game general solution

Example 1

$$A = \begin{pmatrix} -2 & 3 \\ 3 & -4 \end{pmatrix}$$

$$p = \frac{-4-3}{-2-3-4-3} = 7/12$$

$$q = \text{same}$$

$$v = \frac{8-9}{-2-3-4-3} = 1/12$$

Example 2

$$A = \begin{pmatrix} 0 & -10 \\ 1 & 2 \end{pmatrix}$$

$$p = \frac{2-1}{0+10+2-1} = 1/11$$

$$q = \frac{2+10}{0+10+2-1} \boxed{= 12/11.}$$

- But q must be between zero and one. What happened?
  - The trouble is we “forgot to test this matrix for a pure strategy, so of course it has one”.
  - The lower left corner is pure strategy. So p = 0 and q = 1 are optimal strategies, and the value is v = 1.

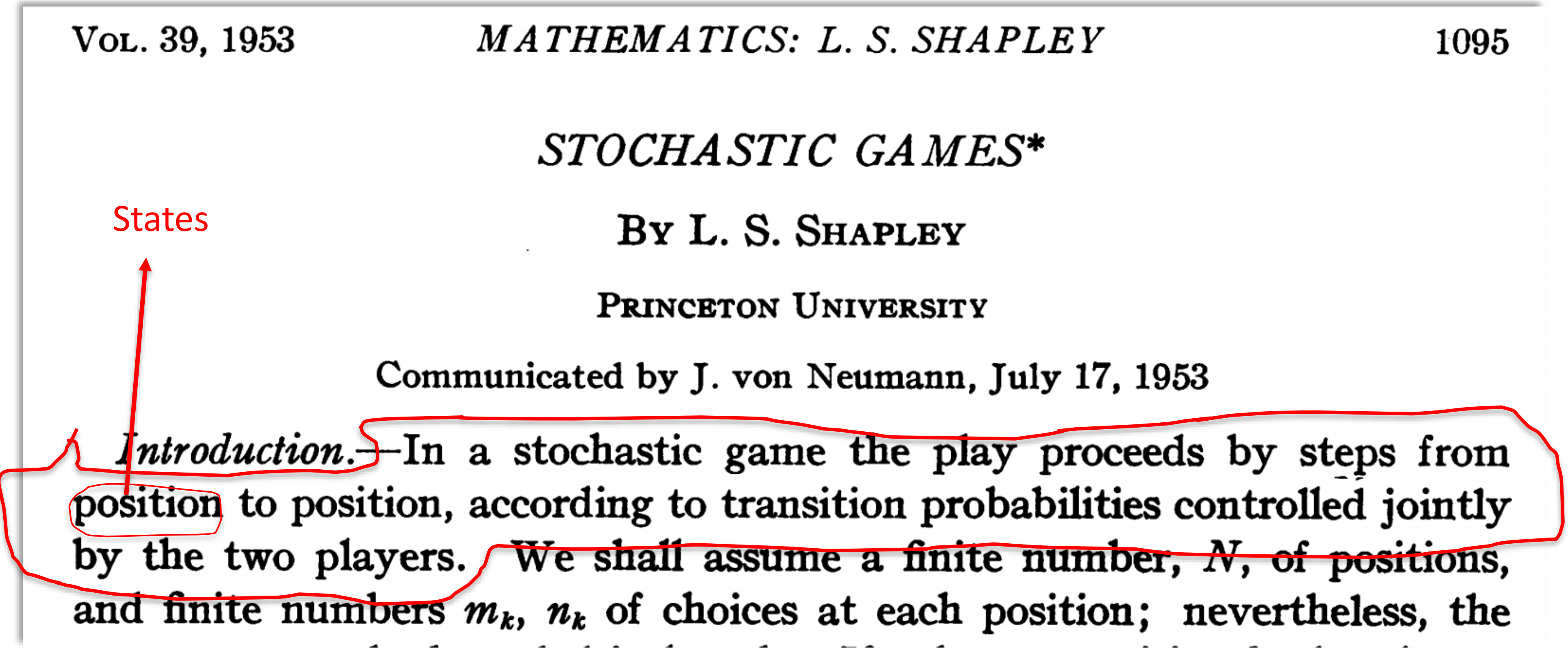
Recall: Repeated Games

- In a (typical) repeated game,
  - players play a normal-form game (aka. the stage game),
  - then they see what happened (and get the reward),
  - then they play again,
  - etc.
- Can be repeated finitely or infinitely many times
- Multiple agents, but still single stage

Tables of Content

- Multi-agent RL: introduction and concepts
- Stochastic Games
  - Policy Iteration/Value Iteration (model based)
  - Equilibrium Learners (model free)
    - Nash-Q
    - Minimax-Q
    - Friend-Foe-Q
  - Best-Response Learners (model free)
    - JAL and Opponent Modelling
    - Iterated Gradient Ascent
    - Wolf-IGA

The Origin of Stochastic Games



*STOCHASTIC GAMES**

By L. S. Shapley

Princeton University

Communicated by J. von Neumann, July 17, 1953

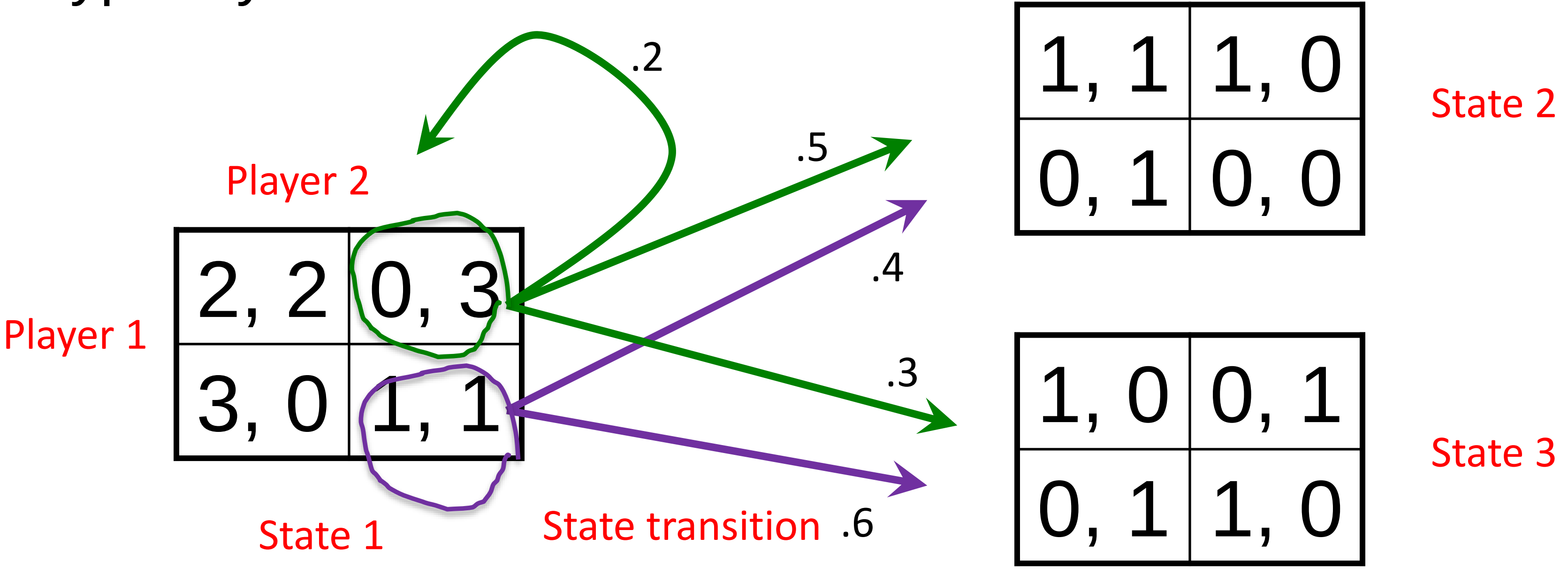
States

Introduction.—In a stochastic game the play proceeds by steps from position to position, according to transition probabilities controlled jointly by the two players. We shall assume a finite number, N, of positions, and finite numbers m_k, n_k of choices at each position; nevertheless, the

Shapley, Lloyd S. "Stochastic games." *Proceedings of the national academy of sciences* 39.10 (1953): 1095-1100.

Stochastic Games

- A stochastic game has multiple states and multiple agents
  - Each state corresponds to a normal-form game
  - After a round, the game randomly transitions to another state
  - Transition probabilities depend on state and joint actions taken by all agents
- Typically rewards are discounted over time



Player 2

Player 1

2, 2	0, 3
3, 0	1, 1

State 1

State transition

.2

.5

.4

.3

.6

State 2

1, 1	1, 0
0, 1	0, 0

State 3

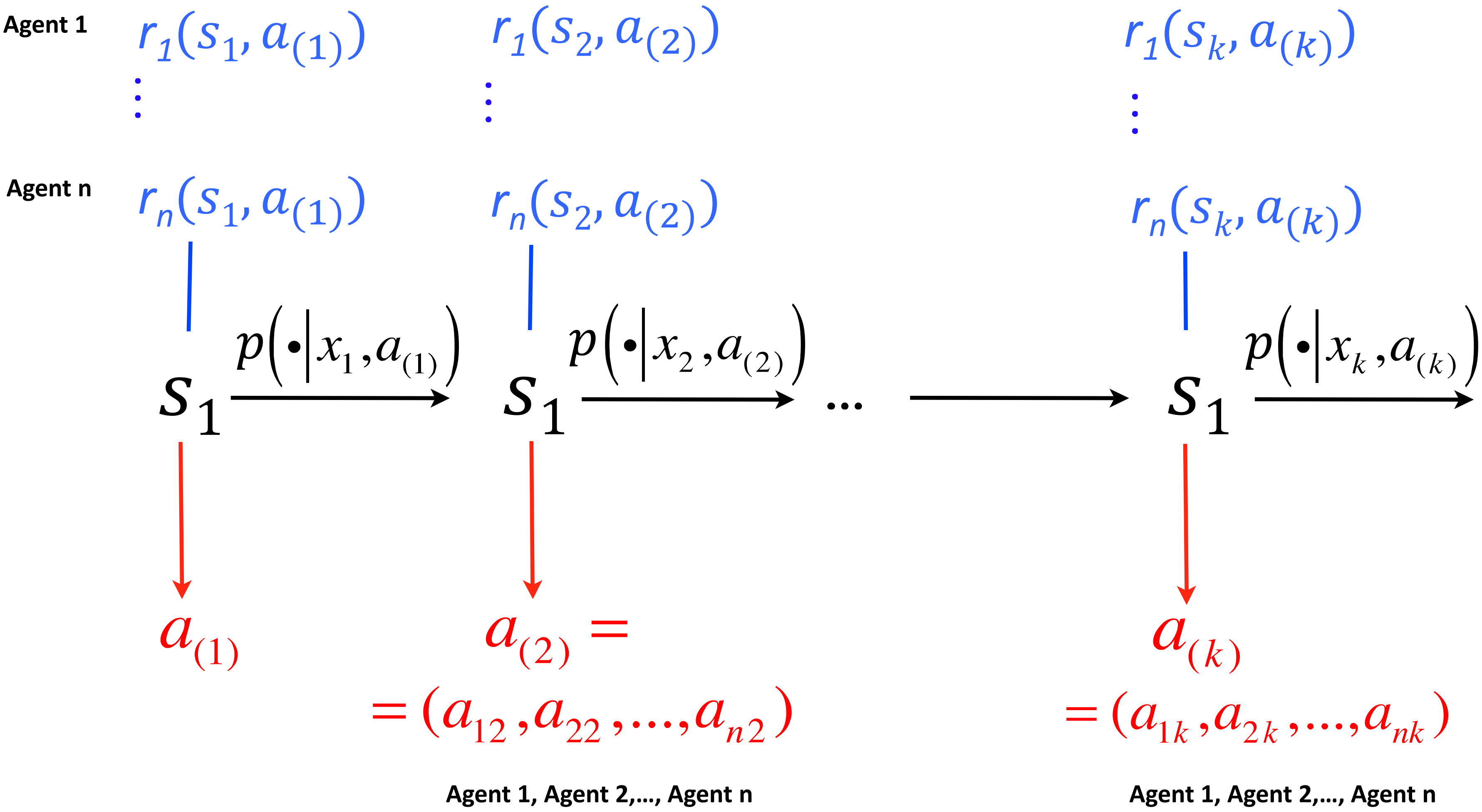
1, 0	0, 1
0, 1	1, 0

- 1-state stochastic game = (infinitely) repeated game
- 1-agent stochastic game = Markov Decision Process (MDP)

Definition of Stochastic Games

- Defined by a tuple (n, S, $A_{1\ldots n}$, T, $R_{1\ldots n}$), where
  - – n is the number of players,
  - – S is the set of states,
  - – A_i is the set of actions available to player i
    - (and A is the joint action space $A_1 \times \cdots \times A_n$),
  - – T is the transition function $S \times A \times S \rightarrow [0,1]$, and
  - – R_i is the reward function for the ith agent $S \times A \rightarrow R$.
- Different with MDP:
  - – there are multiple players selecting actions and
  - – the next state and rewards depend on the joint actions
  - – Each player has its own separate reward function.

Evolution of Stochastic Games



Agent 1 $r_1(s_1, a_{(1)})$ $r_1(s_2, a_{(2)})$ $r_1(s_k, a_{(k)})$

$\vdots$ $\vdots$ $\vdots$

Agent n $r_n(s_1, a_{(1)})$ $r_n(s_2, a_{(2)})$ $r_n(s_k, a_{(k)})$

$$s_1 \xrightarrow{p\left(\bullet \middle| x_1, a_{(1)}\right)} s_1 \xrightarrow{p\left(\bullet \middle| x_2, a_{(2)}\right)} \cdots \longrightarrow s_1 \xrightarrow{p\left(\bullet \middle| x_k, a_{(k)}\right)}$$

$a_{(1)}$

$a_{(2)} =$

$= (a_{12}, a_{22}, \dots, a_{n2})$

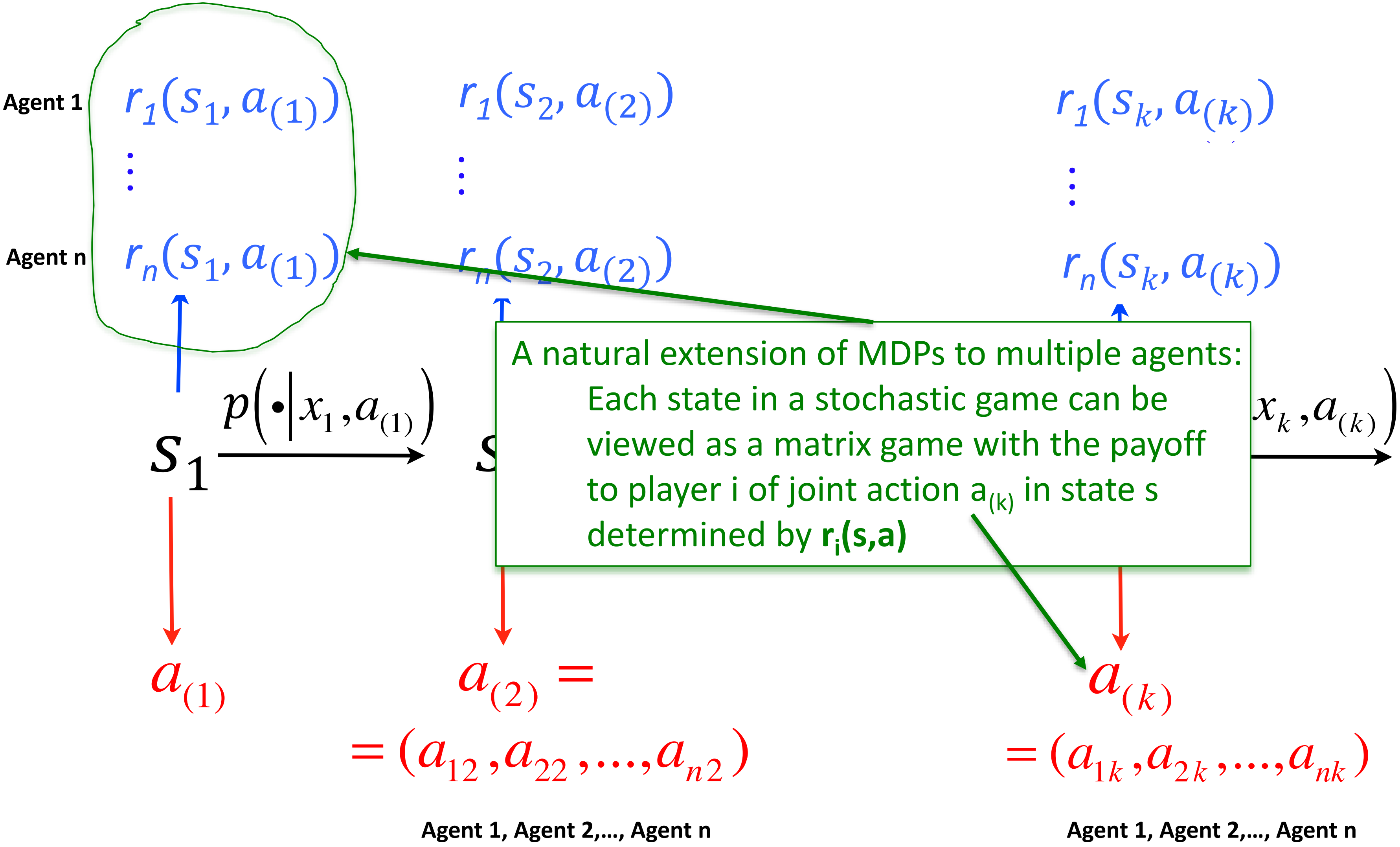
Agent 1, Agent 2,…, Agent n

$a_{(k)}$

$= (a_{1k}, a_{2k}, \dots, a_{nk})$

Agent 1, Agent 2,…, Agent n

Evolution of Stochastic Games



Agent 1 $r_1(s_1, a_{(1)})$ $r_1(s_2, a_{(2)})$ $r_1(s_k, a_{(k)})$

⋮ ⋮ ⋮

Agent n $r_n(s_1, a_{(1)})$ $r_n(s_2, a_{(2)})$ $r_n(s_k, a_{(k)})$

$s_1 \xrightarrow{p\left(\bullet \middle| x_1, a_{(1)}\right)}$ s $x_k, a_{(k)}\Big)$ $\longrightarrow$

$a_{(1)}$

$a_{(2)} =$

$= (a_{12}, a_{22}, \dots, a_{n2})$

Agent 1, Agent 2,…, Agent n

$a_{(k)}$

$= (a_{1k}, a_{2k}, \dots, a_{nk})$

Agent 1, Agent 2,…, Agent n

A natural extension of MDPs to multiple agents:

Each state in a stochastic game can be viewed as a matrix game with the payoff to player i of joint action $a_{(k)}$ in state s determined by $\mathbf{r_i(s,a)}$

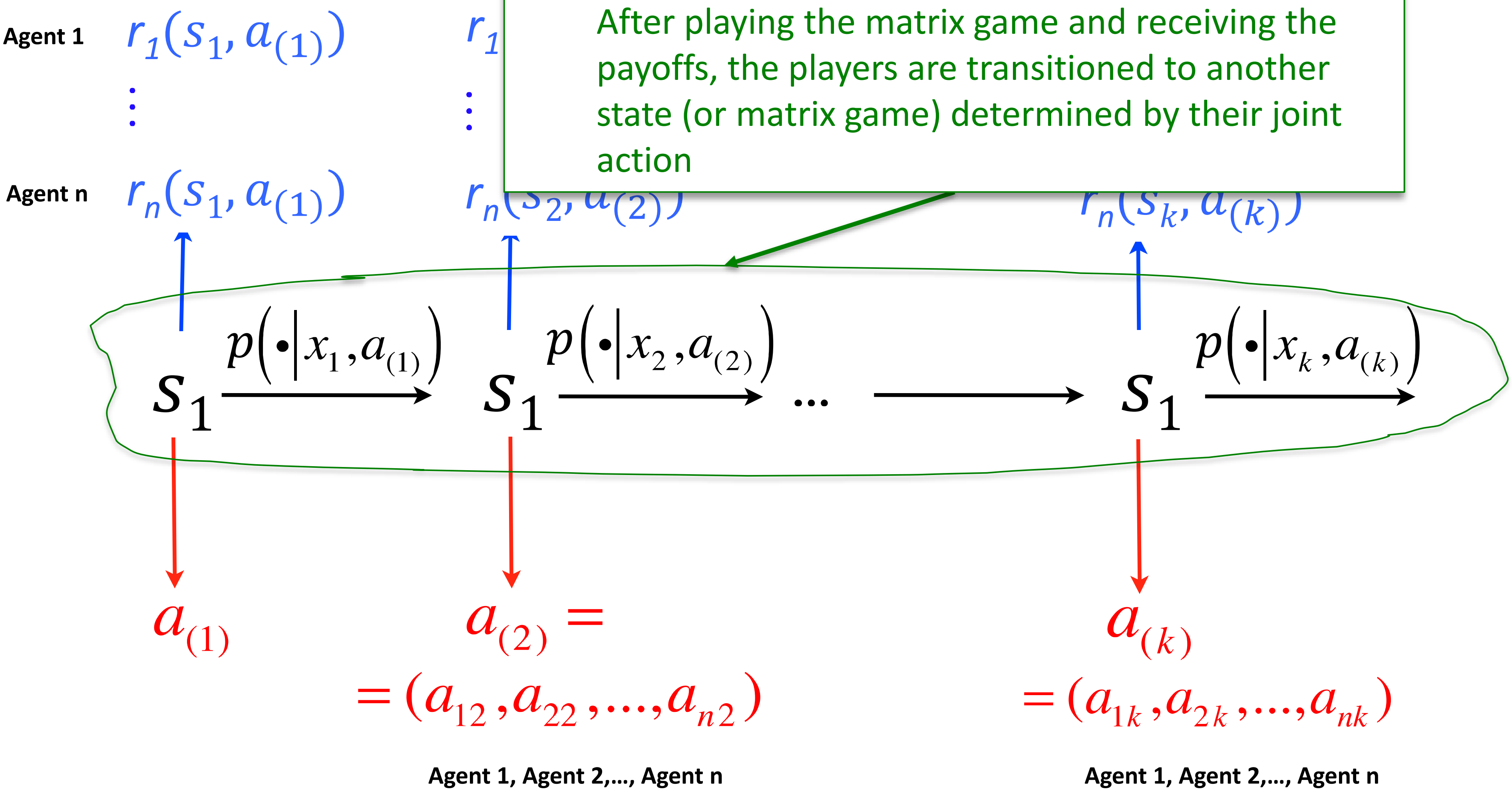
Stochastic Games vs. MDP

- If all but one player in a stochastic game play a fixed, then the problem for the remaining agent reverts back to an MDP.
  - This is because fixing the other agents' policies, even if stochastic, makes the transitions Markovian, depending only on the remaining player's actions.

Evolution of Stochastic Games

An extension of matrix games to multiple states:

After playing the matrix game and receiving the payoffs, the players are transitioned to another state (or matrix game) determined by their joint action



Agent 1 $r_1(s_1, a_{(1)})$ r_1

$\vdots$ $\vdots$

Agent n $r_n(s_1, a_{(1)})$ $r_n(s_2, a_{(2)})$ $r_n(s_k, a_{(k)})$

$$s_1 \xrightarrow{p\left(\bullet\middle|x_1, a_{(1)}\right)} s_1 \xrightarrow{p\left(\bullet\middle|x_2, a_{(2)}\right)} \cdots \longrightarrow s_1 \xrightarrow{p\left(\bullet\middle|x_k, a_{(k)}\right)}$$

$a_{(1)}$

$a_{(2)} =$

$= (a_{12}, a_{22}, \ldots, a_{n2})$

Agent 1, Agent 2,…, Agent n

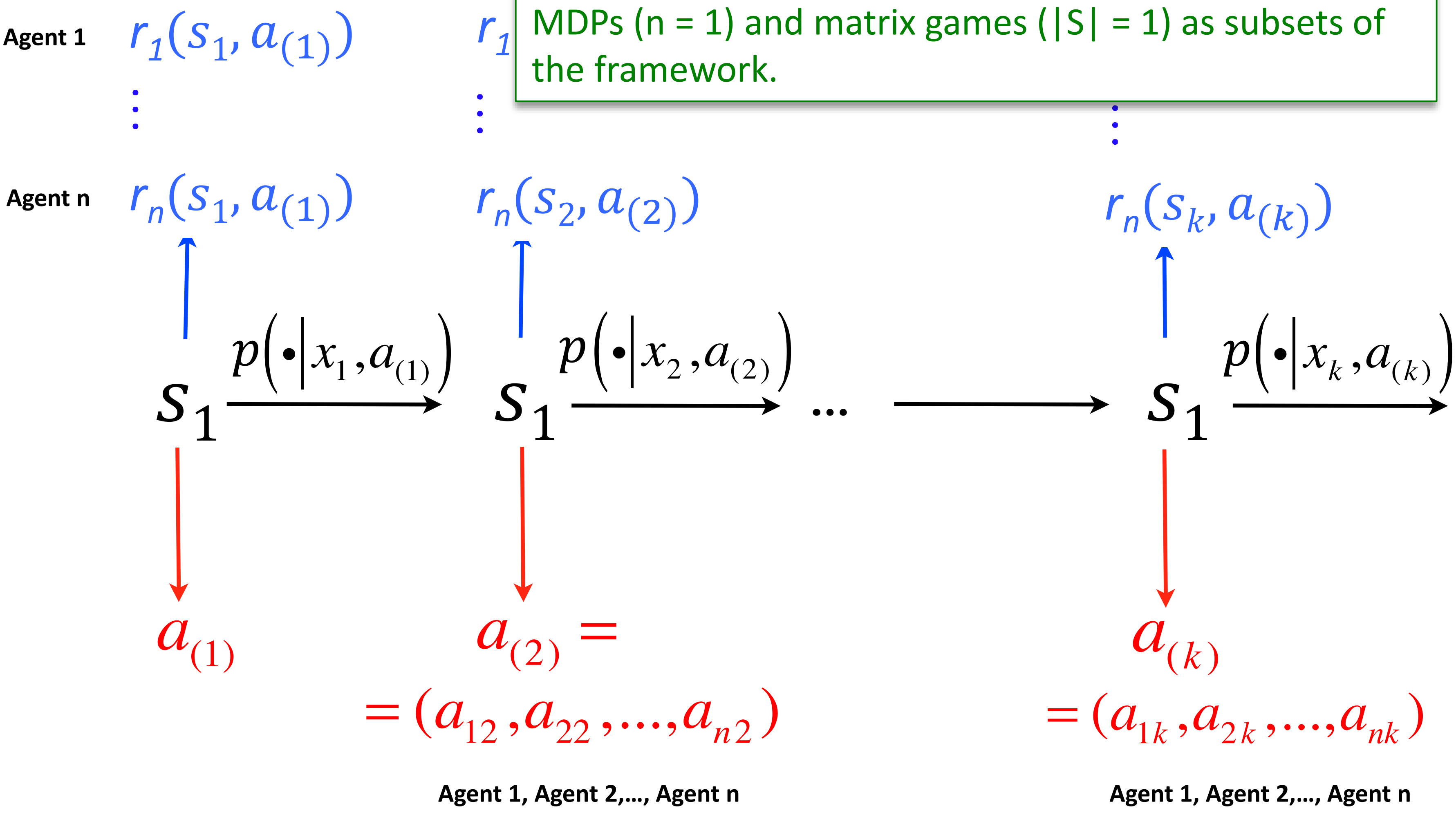
$a_{(k)}$

$= (a_{1k}, a_{2k}, \ldots, a_{nk})$

Agent 1, Agent 2,…, Agent n

Evolution of Stochastic Games

We can see that stochastic games then contain both MDPs (n = 1) and matrix games (|S| = 1) as subsets of the framework.



Agent 1 $r_1(s_1, a_{(1)})$ r_1

$\vdots$

Agent n $r_n(s_1, a_{(1)})$ $r_n(s_2, a_{(2)})$ $r_n(s_k, a_{(k)})$

$$s_1 \xrightarrow{p\left(\bullet \middle| x_1, a_{(1)}\right)} s_1 \xrightarrow{p\left(\bullet \middle| x_2, a_{(2)}\right)} \cdots \longrightarrow s_1 \xrightarrow{p\left(\bullet \middle| x_k, a_{(k)}\right)}$$

$a_{(1)}$

$a_{(2)} =$

$= (a_{12}, a_{22}, \ldots, a_{n2})$

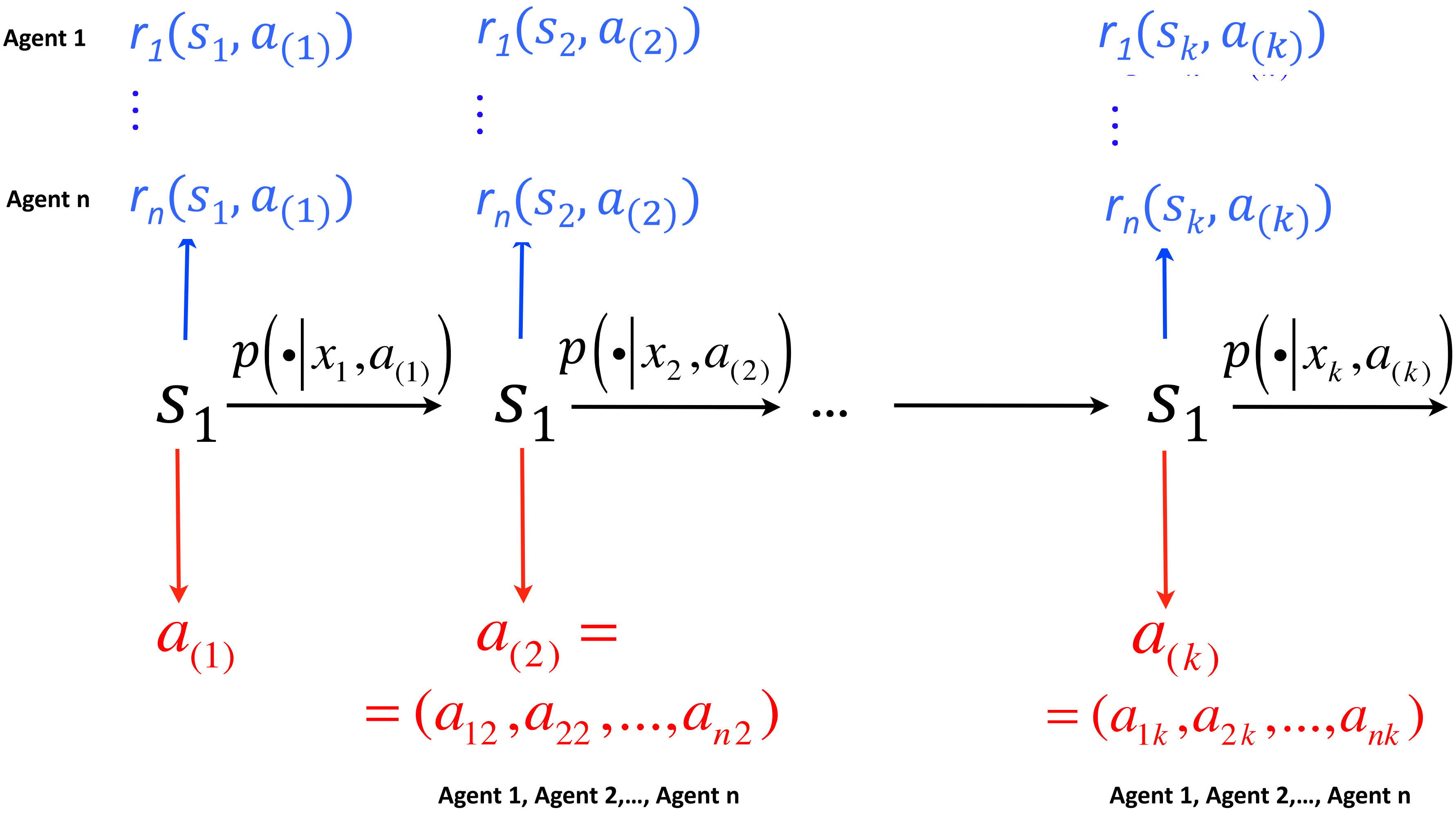
Agent 1, Agent 2,…, Agent n

$a_{(k)}$

$= (a_{1k}, a_{2k}, \ldots, a_{nk})$

Agent 1, Agent 2,…, Agent n

Evolution of Stochastic Games



Agent 1 $r_1(s_1, a_{(1)})$ $r_1(s_2, a_{(2)})$ $r_1(s_k, a_{(k)})$

$\vdots$ $\vdots$ $\vdots$

Agent n $r_n(s_1, a_{(1)})$ $r_n(s_2, a_{(2)})$ $r_n(s_k, a_{(k)})$

$$s_1 \xrightarrow{p\left(\bullet \middle| x_1, a_{(1)}\right)} s_1 \xrightarrow{p\left(\bullet \middle| x_2, a_{(2)}\right)} \cdots \longrightarrow s_1 \xrightarrow{p\left(\bullet \middle| x_k, a_{(k)}\right)}$$

$a_{(1)}$

$a_{(2)} =$

$= (a_{12}, a_{22}, \ldots, a_{n2})$

Agent 1, Agent 2,…, Agent n

$a_{(k)}$

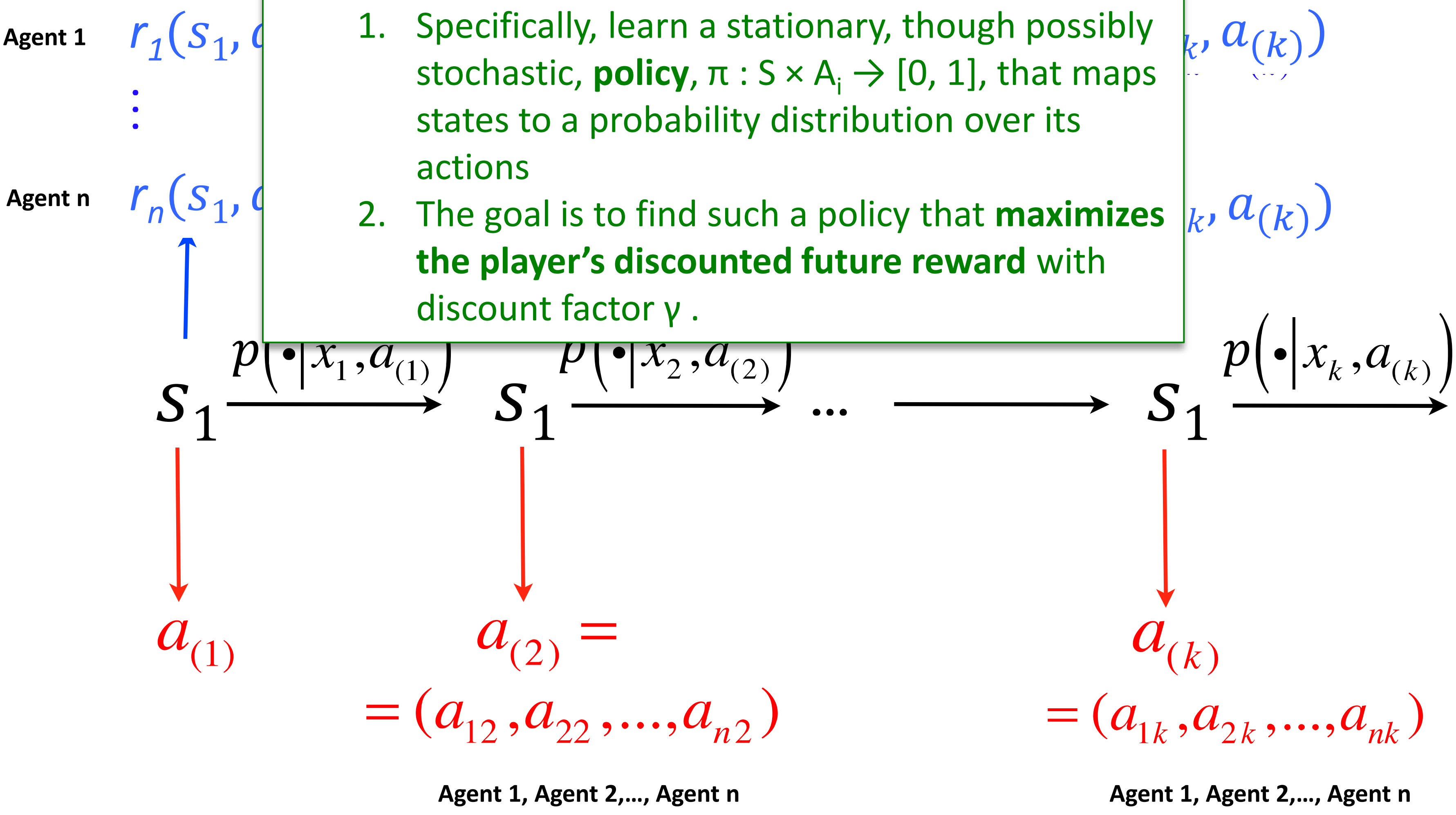
$= (a_{1k}, a_{2k}, \ldots, a_{nk})$

Agent 1, Agent 2,…, Agent n

Evolution of Stochastic Games

The goal is to determine a course of action for **a player**:

1. Specifically, learn a stationary, though possibly stochastic, **policy**, π : S × A_i → [0, 1], that maps states to a probability distribution over its actions
2. The goal is to find such a policy that **maximizes the player's discounted future reward** with discount factor γ .



Agent 1 $r_1(s_1, a$ … $_k, a_{(k)})$

⋮

Agent n $r_n(s_1, a$ … $_k, a_{(k)})$

$$s_1 \xrightarrow{p(\bullet|x_1, a_{(1)})} s_1 \xrightarrow{p(\bullet|x_2, a_{(2)})} \cdots \longrightarrow s_1 \xrightarrow{p(\bullet|x_k, a_{(k)})}$$

$a_{(1)}$

$a_{(2)} = $

$= (a_{12}, a_{22}, \ldots, a_{n2})$

Agent 1, Agent 2,…, Agent n

$a_{(k)}$

$= (a_{1k}, a_{2k}, \ldots, a_{nk})$

Agent 1, Agent 2,…, Agent n

Example: Pollution Tax Model

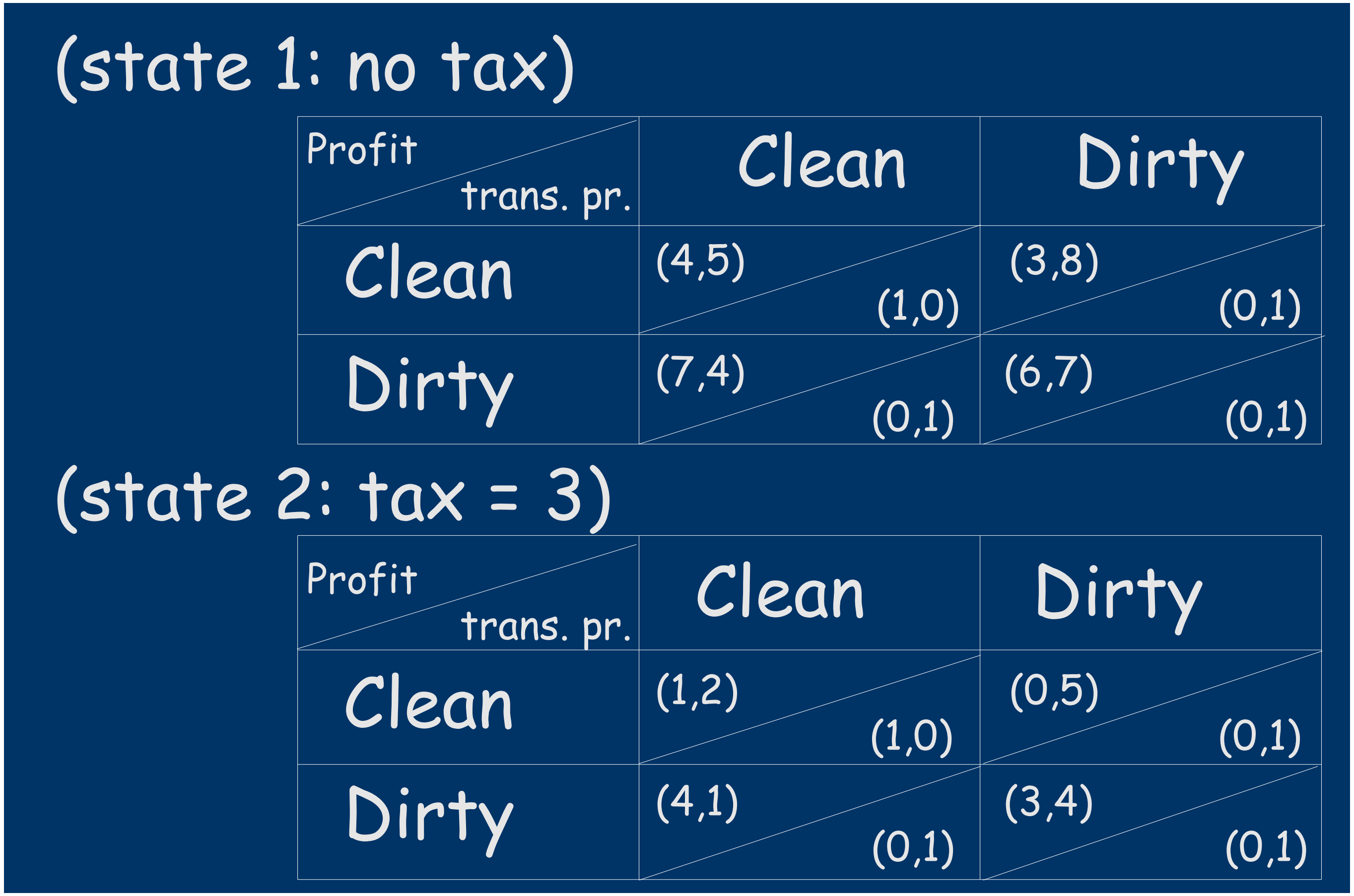
- Two firms contribute to the emission of certain pollutant.
- The government can detect only the combined emissions, and only if it is high.
- The Profit Matrix (no tax version):

Profit	Clean	Dirty
Clean	(4,5)	(3,8)
Dirty	(7,4)	(6,7)

- What is the Nash Equilibrium?

Example: Pollution Tax Model

- Suppose Gov. added Tax (Two-state Stochastic Game)



(state 1: no tax)

Profit trans. pr.	Clean	Dirty
Clean	(4,5) (1,0)	(3,8) (0,1)
Dirty	(7,4) (0,1)	(6,7) (0,1)

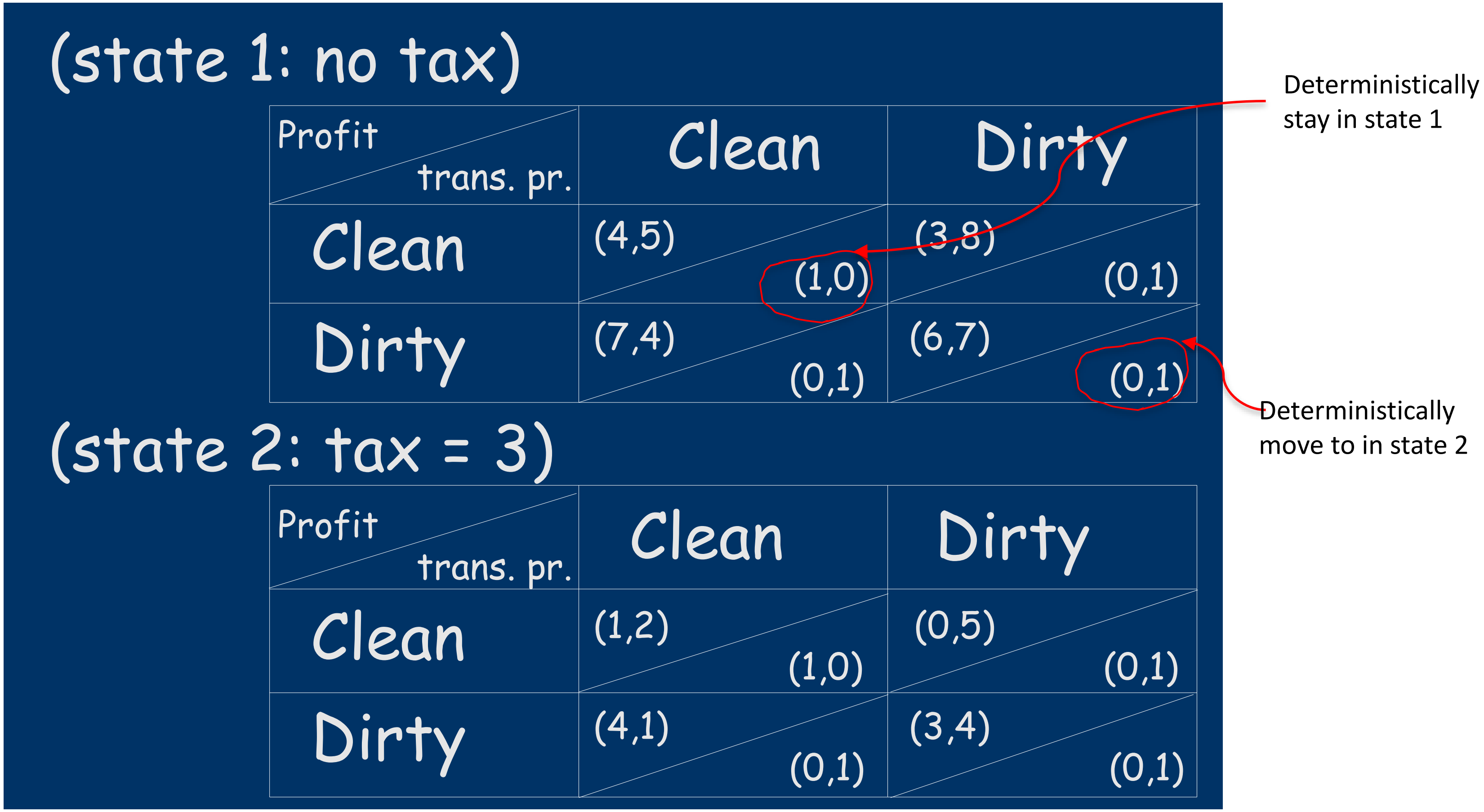
(state 2: tax = 3)

Profit trans. pr.	Clean	Dirty
Clean	(1,2) (1,0)	(0,5) (0,1)
Dirty	(4,1) (0,1)	(3,4) (0,1)

What is the Nash Equilibrium?

Example: Pollution Tax Model

- Suppose Gov. added Tax (Two-state Stochastic Game)



(state 1: no tax)

Profit trans. pr.	Clean	Dirty
Clean	(4,5) (1,0)	(3,8) (0,1)
Dirty	(7,4) (0,1)	(6,7) (0,1)

Deterministically stay in state 1

Deterministically move to in state 2

(state 2: tax = 3)

Profit trans. pr.	Clean	Dirty
Clean	(1,2) (1,0)	(0,5) (0,1)
Dirty	(4,1) (0,1)	(3,4) (0,1)

What is the Nash Equilibrium?

Example: the game of Dare

- Player 1, the leader, and Player 2, the challenger, simultaneously “pass” or “dare”.
  - If both pass, the payoff is zero (and the game is over).
  - If player 1 passes and player 2 dares, player 1 wins 1
  - If player 1 dares and player 2 passes, player 1 wins 3
  - If both dare, the basic game is played over with the roles of the players reversed
    - (the leader becomes the challenger and vice versa).
  - If the players keep daring forever, let the payoff be zero.

$$G = \begin{array}{c c} & \begin{array}{cc} \text{pass} & \text{dare} \end{array} \\ \begin{array}{c} \text{pass} \\ \text{dare} \end{array} & \begin{pmatrix} 0 & 1 \\ 3 & -G^T \end{pmatrix} \end{array}$$

where $-G^T$ represents the game with the roles of the players reversed. (Its matrix is the negative of the transpose of the matrix G.) The value of $-G^T$ is the negative of the value of G.

Game Theory, Second Edition, 2014 Thomas S. Ferguson Mathematics Department, UCLA

Example: the game of Dare

- If *v* represents the value of G, then v ≥ 0 because of the top row.
  - Therefore the matrix for G with $-G^T$ replaced by $-v$ does not have a pure strategy, and we have

$$v = \text{Val}\begin{pmatrix} 0 & 1 \\ 3 & -v \end{pmatrix} = \frac{3}{4+v}.$$

  - which gives $v^2 + 4v - 3 = 0$.
  - The only nonnegative solution is $v=\sqrt{7}-2$.
  - The optimal strategy for player 1 is$((5-\sqrt{7})/3,(\sqrt{7}-2)/3)$ and the optimal strategy for player 2 is $(3-\sqrt{7}, \sqrt{7}-2)$.

$$G = \begin{array}{c} \\ \text{pass} \\ \text{dare} \end{array} \begin{array}{c} \begin{array}{cc} \text{pass} & \text{dare} \end{array} \\ \begin{pmatrix} 0 & 1 \\ 3 & -G^T \end{pmatrix} \end{array}$$

where $-G^T$ represents the game with the roles of the players reversed. (Its matrix is the negative of the transpose of the matrix G.) The value of $-G^T$ is the negative of the value of G.

General solution of 2x2 zero-sum game

$$A = \begin{pmatrix} a & b \\ d & c \end{pmatrix}$$

$$v = ap + d(1-p) = \frac{ac - bd}{a - b + c - d}.$$

Exercise: Stochastic Movement Among Games

General solution of 2x2 zero-sum game

$$A = \begin{pmatrix} a & b \\ d & c \end{pmatrix}$$

$$v = ap + d(1 - p) = \frac{ac - bd}{a - b + c - d}.$$

- Suppose we allow the choice of the next game played to depend not only upon the pure strategy choices of the players, but also upon chance
- Let G_1 and G_2 be related as follows:

$$G_1 = \begin{pmatrix} \frac{1}{2}G_2 + \frac{1}{2}(0) & 1 \\ 2 & 0 \end{pmatrix} \qquad G_2 = \begin{pmatrix} \frac{2}{3}G_1 + \frac{1}{3}(-2) & 0 \\ 0 & -1 \end{pmatrix}$$

- The game must eventually end (with probability 1).
  - the players could not play forever even if they wanted to
  - when they choose the first row and first column forever, eventually the game would end with a payoff of 0 or −2

Game Theory, Second Edition, 2014 Thomas S. Ferguson Mathematics Department, UCLA

Exercise: Stochastic Movement Among Games

General solution of 2x2 zero-sum game

$$A = \begin{pmatrix} a & b \\ d & c \end{pmatrix}$$

$$v = ap + d(1-p) = \frac{ac - bd}{a - b + c - d}.$$

$$G_1 = \begin{pmatrix} \frac{1}{2}G_2 + \frac{1}{2}(0) & 1 \\ 2 & 0 \end{pmatrix} \qquad G_2 = \begin{pmatrix} \frac{2}{3}G_1 + \frac{1}{3}(-2) & 0 \\ 0 & -1 \end{pmatrix}$$

- To solve, let $v_i = \mathrm{Val}(G_i)$ for i = 1, 2.
- Then $0 \le v_1 \le 1$ and $-1 \le v_2 \le 0$, so neither game has a pure strategy. Hence,

$$v_1 = \mathrm{Val}\begin{pmatrix} \frac{1}{2}v_2 & 1 \\ 2 & 0 \end{pmatrix} = \frac{4}{6 - v_2} \qquad \text{and}$$

$$v_2 = \mathrm{Val}\begin{pmatrix} \frac{2}{3}v_1 - \frac{2}{3} & 0 \\ 0 & -1 \end{pmatrix} = -\frac{2(1 - v_1)}{5 - 2v_1}$$

Thus

$$v_1 = \frac{4}{6 + \frac{2(1 - v_1)}{5 - 2v_1}} = \frac{2(5 - 2v_1)}{16 - 7v_1}.$$

- This leads to the quadratic equation, $7v^2{}_1 - 20v_1 + 10 = 0$, with solution, $v_1 = (10 - \sqrt{30})/7$.
- Also $v_2 = -(2\sqrt{30} - 10)/5$

Tables of Content

Classification of Stochastic Games

- Zero-sum stochastic game: all of the states must define a zero-sum matrix game and
- Team stochastic game: all of the states must define team matrix games - their reward is the same for every joint action
- The one that do not fall in any of these categories are generally called general-sum stochastic games

State Value in Stochastic Games

- Similar to MDP, the state value of a SG is

$$
\begin{aligned}
& V_i^{\pi}(s) \\
&= \mathrm{E}_{\pi}\left\{\sum_{k=0}^{+\infty} \gamma^k r_i(t+k+1) \,\middle|\, s_t = s\right\} \\
&= \mathrm{E}_{\pi}\left\{r_i(t+1) + \gamma \sum_{k=0}^{+\infty} \gamma^k r_i(t+k+2) \,\middle|\, s_t = s\right\} \\
&= \sum_{a} \pi(a|s) \sum_{s'} p(s'|s,a)\left[r_i(s',a) + \gamma \mathrm{E}_{\pi}\left\{\sum_{k=0}^{+\infty} \gamma^k r_i(t+k+2) \,\middle|\, s_{t+1} = s'\right\}\right] \\
&= \sum_{a} \pi(a|s) \sum_{s'} p(s'|s,a)\left[r_i(s',a) + \gamma V_i^{\pi}(s')\right]
\end{aligned}
$$

π(s|a) is the probability of choosing joint action a in state s

- However, SG state values must defined for each agent and the expected value depends on the joint policy and not on the individual policies of the agents

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Vol. 1, No. 1). Cambridge: MIT press.

State Value in Stochastic Games

- Similar to MDP, the state value of a SG is

$$V_i^{\pi}(s) = \mathrm{E}_{\pi}\{\sum_{k=0}^{+\infty} \gamma^k r_i(t+k+1) \,|s_t = s\}$$

- The total expected payoff to either player is bounded by

$$V_i^{\pi}(s) \le \frac{M}{1-\gamma}, \qquad M \equiv \max_{i,s,a} |r_i(s,a)|.$$

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction* (Vol. 1, No. 1). Cambridge: MIT press.

Game at each state

- We use game $G^{(s)}$ at each state s

$$G^{(s)} = \left(r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} p(s'|s, a_i, a_{-i})\, G^{(s')} \right)$$

- In contrast to normal-form game, a payoff does not end the game.
- After a payoff is made, it is then decided at random whether the game ends with probability (1-γ) and,
- if not, which state should be played next.

Example:

$$G_1 = \begin{pmatrix} \frac{1}{2} G_2 + \frac{1}{2}(0) & 1 \\ 2 & 0 \end{pmatrix} \qquad G_2 = \begin{pmatrix} \frac{2}{3} G_1 + \frac{1}{3}(-2) & 0 \\ 0 & -1 \end{pmatrix}$$

Value Iterations in SG

- **Theorem (SG).** (Shapley (1952)) Each game $G^{(s)}$ has a value, V(s). These values are the unique solution of the set of equations,

$$V(s) \ = Val\left(r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} p(s'|s, a_i, a_{-i})\, V(s)\right) \qquad for\ \ s\ \in S$$

- Each player has a stationary optimal mixed strategy in state s with matrix

$$G^{(s)}(V) = \left(r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} p(s'|s, a_i, a_{-i})\, V(s)\right)$$

where V represents the values at different states, V = (V(s), . . .)

Shapley, Lloyd S. "Stochastic games." *Proceedings of the national academy of sciences* 39.10 (1953): 1095-1100.

Value Iterations in SG

- Shapley's value iterations

Shapley, Lloyd S. "Stochastic games." *Proceedings of the national academy of sciences* 39.10 (1953): 1095-1100.

1. Initialize V arbitrarily.
2. Repeat,
    (a) For each state, $s \in S$, compute the matrix,

$$G^{(s)}(V) = \left(r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} p(s' | s, a_i, a_{-i}) V(s) \right)$$

    (b) For each state, $s \in S$, update V,

$$V(s) \leftarrow Val(G^{(s)}(V))$$

- *Val operator* solves the matrix game and find the value of the game (in Nash Equilibrium), using e.g. *linear programing*

Compared to VI in MDP, the "Max " operator replaced by the "Val" operator

Bowling, Michael, and Manuela Veloso. An analysis of stochastic game theory for multiagent reinforcement learning. No. CMU-CS-00-165. Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 2000.

A Simple Example

Player 1 receives payoff 1 and with 3/5 chance to play the game again

- Consider the following 2x2 zero-sum stochastic game with just one state, call it G

$$G = \begin{pmatrix} 1 + (3/5)G & 3 + (1/5)G \\ 1 + (4/5)G & 2 + (2/5)G \end{pmatrix}$$

- From Player 2's viewpoint, column 1 is better than column 2 in terms of immediate payoff,
- but column 2 is more likely to end the game sooner than column 1, so that it should entail smaller future payoffs.
- Which column should he choose?

Game Theory, Second Edition, 2014 Thomas S. Ferguson Mathematics Department, UCLA

A Simple Example

General solution of 2x2 zero-sum game

$$A = \begin{pmatrix} a & b \\ d & c \end{pmatrix}$$

$$v = ap + d(1 - p) = \frac{ac - bd}{a - b + c - d}.$$

- Assume that all strategies are active (no pure strategy) - must check when we are finished to see if the assumption was correct. Then

$$v = \mathrm{Val}\begin{pmatrix} 1 + (3/5)v & 3 + (1/5)v \\ 1 + (4/5)v & 2 + (2/5)v \end{pmatrix}$$

$$= \frac{(1 + (4/5)v)(3 + (1/5)v) - (1 + (3/5)v)(2 + (2/5)v)}{1 + (4/5)v + 3 + (1/5)v - 1 - (3/5)v - 2 - (2/5)v} = 1 + v - (2/25)v^2$$

- This leads to $(2/25)v^2 = 1$ which gives two possible solutions v = ± $\sqrt{25/2}$.
- Since the value is obviously positive, we must use the plus sign. This is v = (5/2)√2 = 3.535. Thus the matrix above becomes

$$\begin{pmatrix} 1 + (3/2)\sqrt{2} & 3 + (1/2)\sqrt{2} \\ 1 + 2\sqrt{2} & 2 + \sqrt{2} \end{pmatrix}$$

  - The optimal strategy for Player 1 is p = (√2 − 1, 2 − √2) = (.414, .586), and
  - the optimal strategy for Player 2 is q = (1 − √2 /2, √2 /2) = (.293, .707).
- Since these are probability vectors, our assumption is correct and
- v = (5/2) √2 is the value of the stochastic game.

Game Theory, Second Edition, 2014 Thomas S. Ferguson Mathematics Department, UCLA

Value Iteration

- Shapley proves that $v_n(s)$ converges to the true value, v(s), of the stochastic game starting at s
  - First, the convergence is at an exponential rate: the maximum error goes down at least as fast as γ^n
  - Second, the maximum error at stage n + 1 is at most the maximum change from stage n to n + 1 multiplied by $\gamma/(1 - \gamma)$

Shapley, Lloyd S. "Stochastic games." *Proceedings of the national academy of sciences* 39.10 (1953): 1095-1100.

Exercise: Value Iteration

- Let us take an example of a 2x2 zero-sum stochastic game with two states. The corresponding games $G^{(1)}$ and $G^{(2)}$, are related as follows.

$$G^{(1)} = \begin{pmatrix} 4 + .3G^{(1)} & 0 + .4G^{(2)} \\ 1 + .4G^{(2)} & 3 + .5G^{(1)} \end{pmatrix} \qquad G^{(2)} = \begin{pmatrix} 0 + .5G^{(1)} & -5 \\ -4 & 1 + .5G^{(2)} \end{pmatrix}$$

e.g., in state 1, if choosing (row2,col1), then player 1 receives reward 1 and with 0.4 chance moves to state 2.

- What are the values of the game in states 1 and 2, respectively?
- What are the optimal strategies for players in states 1 and 2, respectively?

General solution of 2x2 zero-sum game

$$A = \begin{pmatrix} a & b \\ d & c \end{pmatrix}$$

$$v = ap + d(1 - p) = \frac{ac - bd}{a - b + c - d}.$$

VI update rule:

$$V(s) = Val\left(r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} p(s'|s, a_i, a_{-i})\, V(s) \right)$$

Exercise: Value Iteration

Using $\boldsymbol{v}_0 = (0,0)$ as the initial guess, we find $\boldsymbol{v}_1 = (2,-2)$, since

$$v_1(1) = \mathrm{Val}\begin{pmatrix} 4 & 0 \\ 1 & 3 \end{pmatrix} = 2 \qquad v_1(2) = \mathrm{Val}\begin{pmatrix} 0 & -5 \\ -4 & 1 \end{pmatrix} = -2.$$

The next iteration gives

$$v_2(1) = \mathrm{Val}\begin{pmatrix} 4.6 & -.8 \\ .2 & 4 \end{pmatrix} = 2.0174 \qquad v_2(2) = \mathrm{Val}\begin{pmatrix} 1 & -5 \\ -4 & 0 \end{pmatrix} = -2.$$

Continuing, we find

$$\begin{aligned} v_3(1) &= 2.0210 & v_3(2) &= -1.9983 \\ v_4(1) &= 2.0220 & v_4(2) &= -1.9977 \\ v_5(1) &= 2.0224 & v_5(2) &= -1.9974 \\ v_6(1) &= 2.0225 & v_6(2) &= -1.9974 \end{aligned}$$

The smallest stopping probability is .5, so the rate of convergence is at least $(.5)^n$ and the maximum error of $\boldsymbol{v}_6$ is at most .0002.

General solution of 2x2 zero-sum game

$$A = \begin{pmatrix} a & b \\ d & c \end{pmatrix}$$

$$v = ap + d(1-p) = \frac{ac-bd}{a-b+c-d}.$$

VI update rule:

$$V(s) = Val\left(r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} p(s'|s, a_i, a_{-i})\, V(s) \right)$$

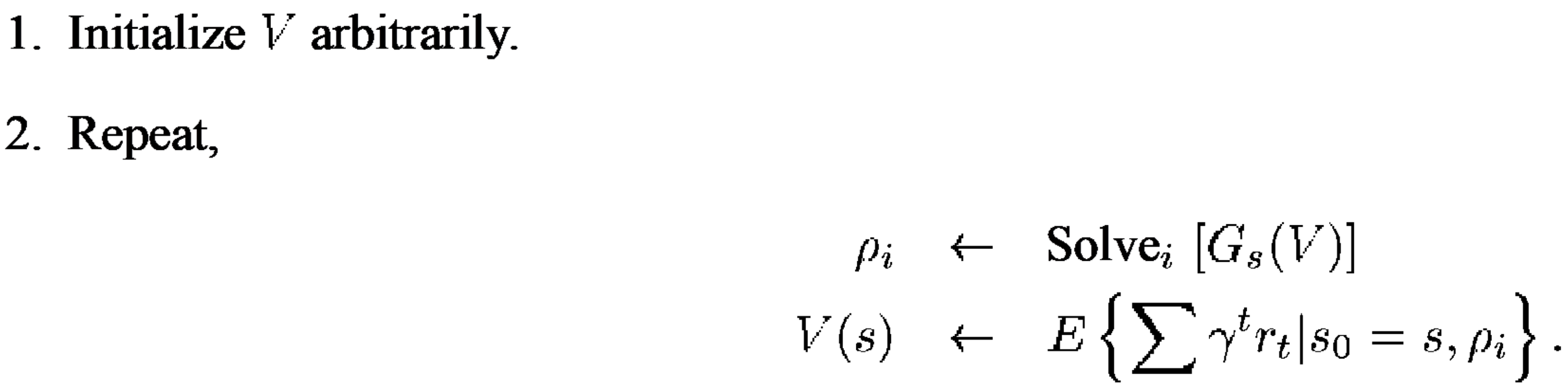
Game Theory, Second Edition, 2014 Thomas S. Ferguson Mathematics Department, UCLA

Exercise: Value Iteration

- The optimal strategies using v_6 (time step 6) are easily found.
  - For game $G^{(1)}$, the optimal strategies are
    - $p^{(1)} = (.4134, .5866)$ for Player 1 and
    - $q^{(1)} = (.5219, .4718)$ for Player 2
  - For game $G^{(2)}$, the optimal strategies are
    - $p^{(2)} = (.3996,.6004)$ for Player 1 and
    - $q^{(2)} = (.4995,.5005)$ for Player 2.

Policy Iterations (Pollatschek & Avi-Itzhak)

- Just as Shapley's algorithm is an extension of value iteration to SG, Pollatschek & Avi-Itzhak introduced an extension of policy iteration



```
1. Initialize $V$ arbitrarily.
2. Repeat,
       $\rho_i \leftarrow \textbf{Solve}_i\ [G_s(V)]$
       $V(s) \leftarrow E\left\{\sum \gamma^t r_t | s_0 = s, \rho_i\right\}.$
```

- Each player selects the equilibrium policy according to the current value function
- The value function is then updated based on the actual rewards of following these policies

Pollatschek, M. A., and B. Avi-Itzhak. "Algorithms for stochastic games with geometrical interpretation." *Management Science* 15.7 (1969): 399-415.

Fined-grained Definition of Strategies

- For agent *i*, a **deterministic** strategy specifies a choice of action for *i* at every stage of every possible history
- A **mixed strategy** is a probability distribution over deterministic strategies
- Several restricted classes of strategies:
  - As in dynamical games, a **behavioural strategy** is a mixed strategy in which the mixing take place at each history independently
  - A **Markov strategy** is a behavioural strategy such that for each time *t*, the distribution over actions depends only on the current state
    - But the distribution may be different at time t than at time $t' \neq t$
  - A **stationary strategy** is a Markov strategy in which the distribution over actions depends only on the current state (not on the time *t*)

Best-response Learners

- A best-response policy for player i is optimal with respect to some joint policy of the other players:

$$\pi_i \in BR_i(\pi_{-i})$$

where π_{-i} is the joint policy of other agents

- $\pi_i^* \in BRi(\pi_{-i})$if and only if:

$$\forall s \in S, V_i^{\langle \pi_i^*, \pi_{-i} \rangle} \geq V_i^{\langle \pi_i \ , \pi_{-i} \rangle}$$

Nash Equilibrium Learners

- A Nash equilibrium in SG is a collection of policies, one for each player,
  - so that all of this policies are best-response policies and
- no player can do better by changing its policy

$$\forall_{i=1\ldots n}, \pi_i^* \in BR_i(\pi_{-i}^*)$$

Tables of Content

- Multi-agent RL: introduction and concepts
- Stochastic Games
  - Policy Iteration/Value Iteration (model based)
  - Equilibrium Learners (model free)
    - Nash-Q
    - Minimax-Q
    - Friend-Foe-Q
  - Best-Response Learners (model free)
    - JAL and Opponent Modelling
    - Iterated Gradient Ascent
    - Wolf-IGA

Equilibrium Learners

- Equilibrium Learners aim to find policies which are Nash equilibria for the stochastic game
  - as it is hard to find such equilibria, they focus on a smaller class of problems, for example zero-sum games or two-person general-sum.
- The advantage of finding the Nash equilibrium is that the agent learns a lower bound for performance and,
  - in this situation, it becomes fairly independent of the policies being played by the other agents
  - it will get at least the amount of return which corresponds to the equilibrium

Q-value in Stochastic Games

- Similar to MDP, the Q value of a SG is

$$
\begin{aligned}
&Q_i^\pi(s,a)\\
&= \mathrm{E}_\pi\{\textstyle\sum_{k=0}^{+\infty}\gamma^k r_i(t+k+1)\,|s_t=s,a_t=a\}\\
&= \mathrm{E}_\pi\{r_i(t+1)+\gamma\textstyle\sum_{k=0}^{+\infty}\gamma^k r_i(t+k+2)\,|s_t=s,a_t=a\}\\
&= \sum_{s'} p(s'|s,a)[r_i(s',a)+\gamma\mathrm{E}_\pi\{\textstyle\sum_{k=0}^{+\infty}\gamma^k r_i(t+k+2)\,|s_{t+1}=s',a_t=a\}]\\
&= \sum_{s'} p(s'|s,a)[r_i(s',a)+\gamma V_i^\pi(s')]
\end{aligned}
$$

- π(s|a) is the probability of choosing joint action a in state s
- The individual Q-values also depend on the actions of all the players.

Equilibrium Learners

- Generally, a solution for an equilibrium learner would be a fixed point in $\pi^* = (\pi_i^*, \pi_{-i}^*)$ of the following system of equations:

$$\forall_{i=1\ldots n} \quad Q_i^*(s,a) = r_i(s,a) + \gamma \sum_{s'} p(s'|s,a) V_i^*(s')$$

  - where $V_i^*(s')$ represents the equilibrium value for agent i when the joint- policy being played is the Nash equilibrium π∗ and
  - is computed with respect to the Q-values.
  - This is similar to a Bellman optimality equation except for the way the state value function is computed.

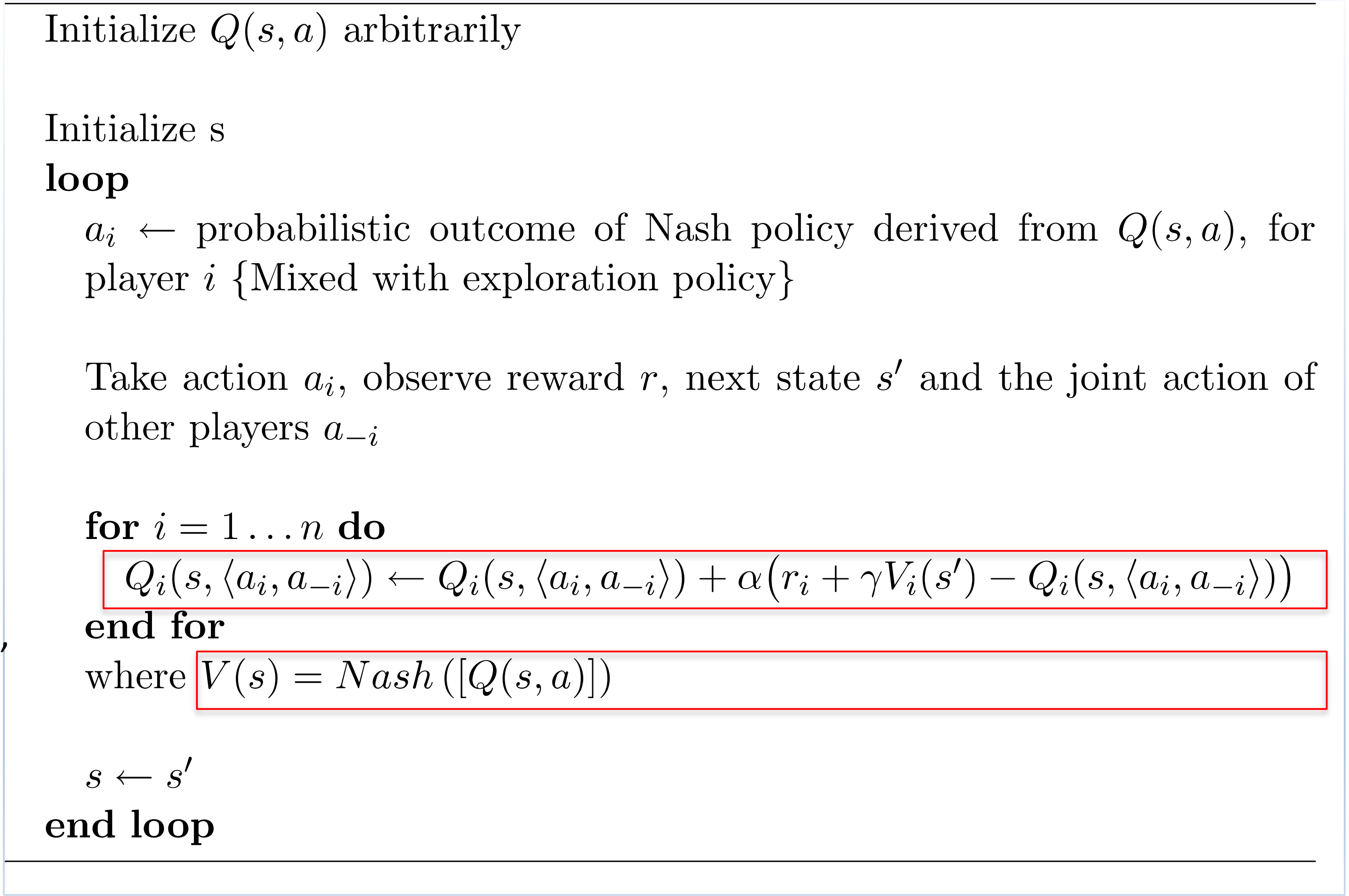
Nash-Q: general equilibrium learner

- Nash-Q addresses two-player general-sum games
  - quadratic programming is used for computing general-sum equilibrium
  - theoretical limitation on single equilibrium only

Hu, Junling, and Michael P. Wellman. "Nash Q-learning for general-sum stochastic games." *Journal of machine learning research* 4.Nov (2003): 1039-1069.

Nash Q: general equilibrium learner

- The Q-function could be estimated through a stochastic approximation procedure very similar to standard Q-learning:



```
Initialize $Q(s,a)$ arbitrarily

Initialize s
loop
    $a_i \leftarrow$ probabilistic outcome of Nash policy derived from $Q(s,a)$, for player $i$ {Mixed with exploration policy}

    Take action $a_i$, observe reward $r$, next state $s'$ and the joint action of other players $a_{-i}$

    for $i = 1 \ldots n$ do
        $\boxed{Q_i(s, \langle a_i, a_{-i} \rangle) \leftarrow Q_i(s, \langle a_i, a_{-i} \rangle) + \alpha \big( r_i + \gamma V_i(s') - Q_i(s, \langle a_i, a_{-i} \rangle) \big)}$
    end for
    where $\boxed{V(s) = Nash\left([Q(s,a)]\right)}$

    $s \leftarrow s'$
end loop
```

Instead of taking "max" as in Q learning

Hu, Junling, and Michael P. Wellman. "Nash Q-learning for general-sum stochastic games." *Journal of machine learning research* 4.Nov (2003): 1039-1069.

Minimax Q

- Minimax-Q is designed to work with zero-sum stochastic games
  - in zero-sum games there is only one equilibrium
  - it can be found using linear programming.

a: own actions,
o: opponent actions
PD(A): Prob. Distribution of Action

```
Initialize $Q(s, \langle a, o \rangle)$ and $\pi(s)$ arbitrarily

Initialize s
loop
    $a \leftarrow$ probabilistic outcome of $\pi(s)$ {Mixed with exploration policy}

    Take action $a$, observe reward $r$, next state $s'$ and opponent action $o$

    $Q(s, \langle a, o \rangle) \leftarrow Q(s, \langle a, o \rangle) + \alpha \big( r + \gamma V(s') - Q(s, \langle a, o \rangle) \big)$
    with $\boxed{V(s) = \max_{\pi' \in PD(A)} \min_{o' \in O} \sum_{a' \in A} \pi(s, a') \, Q(s, \langle a', o' \rangle)}$
    $\boxed{\pi(s) \rightarrow \arg \max_{\pi' \in PD(A)} \min_{o' \in O} \sum_{a' \in A} \pi(s, a') \, Q(s, \langle a', o' \rangle)}$

    $s \leftarrow s'$
end loop
```

Littman, Michael L. "Markov games as a framework for multi-agent reinforcement learning." *Machine Learning Proceedings 1994*. 1994. 157-163.

Friend-or-Foe-Q

- Extended Minimax-Q to solve a more general class of stochastic games.
  - In each state, the method is told whether the agent is playing with a Friend, and the Nash would be a coordination equilibria and a global optimum,
  - or against a Foe, with the game having an adversarial equilibrium in a saddle point.

Littman, Michael L. "Friend-or-foe Q-learning in general-sum games." *ICML*. Vol. 1. 2001.

Friend-or-Foe-Q

```
Initialize $Q(s, \langle a, o \rangle)$ and $\pi(s)$ arbitrarily

Initialize s
loop
    $a \leftarrow$ probabilistic outcome $\pi(s)$ {Mixed with exploration policy}

    Take action $a$, observe reward $r$, next state $s'$ and opponent action $o$

    $Q(s, \langle a, o \rangle) \leftarrow Q(s, \langle a, o \rangle) + \alpha\big(r + \gamma V(s') - Q(s, \langle a, o \rangle)\big)$
    where
    if Playing against foe then
        $V(s) = \max_{\pi' \in PD(A)} \min_{o' \in O} \sum_{a' \in A} \pi(s, a') \, Q(s, \langle a', o' \rangle)$                foe
        $\pi(s) \rightarrow \arg \max_{\pi' \in PD(A)} \min_{o' \in O} \sum_{a' \in A} \pi(s, a') \, Q(s, \langle a', o' \rangle)$
    else
        $V(s) = \max_{a' \in A, o' \in O} Q(s, \langle a', o' \rangle)$
        $\pi(s, a) = \begin{cases} 1 & a = \arg\max_{a' \in A} \{ \max_{o' \in O} Q(s, \langle a', o' \rangle) \} \\ 0 & \text{otherwise} \end{cases}$                friend
    end if

    $s \leftarrow s'$
end loop
```

Littman, Michael L. "Friend-or-foe Q-learning in general-sum games." *ICML*. Vol. 1. 2001.

Tables of Content

- Multi-agent RL: introduction and concepts
- Stochastic Games
  - Policy Iteration/Value Iteration (model based)
  - Equilibrium Learners (model free)
    - Nash-Q
    - Minimax-Q
    - Friend-Foe-Q
  - Best-Response Learners (model free)
    - JAL and Opponent Modelling
    - Iterated Gradient Ascent
    - Wolf-IGA

Desired Properties of Multi-agent Learners

- **Rationality**: If the other players' policies converge to stationary policies, then the learning algorithm will converge to a policy that is a best-response to the other players' policies
- **Convergence**: The learner will necessarily converge to a stationary policy.
  - **Definition** A learning algorithm for player i *converges* to a stationary policy π if and only if for any ε > 0 there exists a time T > 0 such that,

$$\forall t > T,\ a_i \in A_i,\ s \in S,\ P(s,t) > 0 \Rightarrow \left| P(a_i \mid s,t) - \pi(s,a_i) \right| < \varepsilon,$$

  - where P(s,t) is the probability that the game is in state s at time t, and $P(a_i|s,t)$ is the probability that the algorithm selects action a_i , given the game is in state s at time t

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

Desired properties of multi-agent learners

- **Rationality**: If the other players' policies converge to sta co other players' policies

Rationality -> whether it is a best response to others stationary policies

- **Convergence**: The learner will necessarily converge to a stationary policy.
  - **Definition** A learning algorithm for player i *converges* to a stationary policy π if and only if for any ε > 0 there exists a time T > 0 such that,

$$\forall t > T,\ a_i \in A_i,\ s \in S,\ P(s, t) > 0 \Rightarrow \left| P(a_i \mid s, t) - \pi(s, a_i) \right| < \varepsilon,$$

  - where P(s,t) is the probability that the game is in state s at time t, and $P(a_i|s,t)$ is the probability that the algorithm selects action a_i , given the game is in state s at time t

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

Desired properties of multi-agent learners

- **Rationality**: If the other players' policies converge to stationary policies, then the learning algorithm will converge to a policy that is a best-response to the other players' policies
- **Convergence**: The learner will necessarily converge to a stationary policy.

Definition A learning algorithm for player i *converges* to a

Convergence is usually conditioned on other players' learning algorithms, e.g., convergence with respect to rational players or self-play (all players using the same algos)

∀ ε,

time t, and $P(a_i|s,t)$ is the probability that the algorithm selects action a_i , given the game is in state s at time t

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

Desired properties of multi-agent learners

- **Rationality**: If the other players' policies converge to stationary policies, then the learning algorithm will converge to a policy that is a best-response to the other players' policies
- **Relationship to equilibria:** If all the players use **rational learning** algorithms **and** their policies **converge**, they must have converged to an equilibrium. Why?
  - **Definition** A learning algorithm for player i *converges* to a stationary policy π if and only if for any ε > 0 there exists a time T > 0 such that,

$$\forall t > T,\ a_i \in A_i,\ s \in S,\ P(s,t) > 0 \Rightarrow \left| P(a_i \mid s,t) - \pi(s,a_i) \right| < \varepsilon,$$

  - where P(s,t) is the probability that the game is in state s at time t, and $P(a_i|s,t)$ is the probability that the algorithm selects action a_i , given the game is in state s at time t

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

Learning against stationary policies

- When the policies of all but one of the agents are stationary, the stochastic game reduces to a MDP
- Why?

Learning against stationary policies

- When the policies of all but one of the agents are stationary, the stochastic game reduces to a MDP
- Why?
  - all the other agents' stationary policies are used to redefine the transition probabilities and reward structure for the equivalent MDP
- Suppose
  - agent i has learning policy π_i and
  - the joint policy of the other agents π_{-i} is fixed,
  - the parameters of the equivalent MDP are:
    - Transition prob:$p(s'|s, ai\ ) = \sum_{a_{-i}} \pi_{-i}(a_{-i}|s)p(s'|s, ai, a_{-i})$
    - Reward function:$r_i(s, a_i) = \sum_{a_{-i}} \pi_{-i}(a_{-i}|s)r_i(s, a_i, a_{-i})$

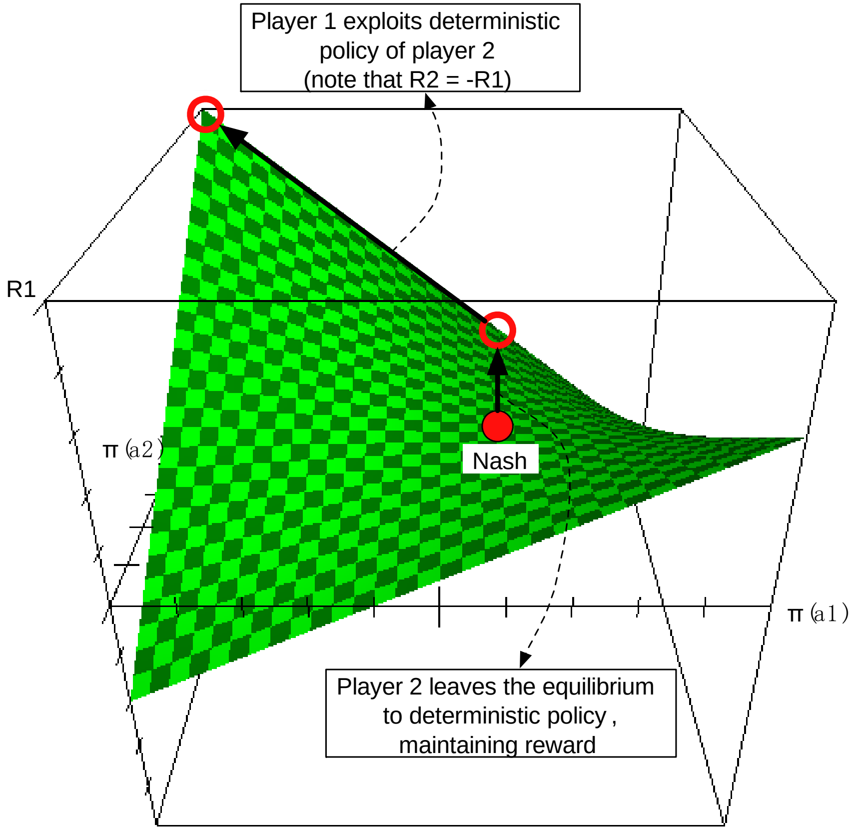
Independent learners vs. Joint Action Learners

- Independent learners (ILs) apply Q-learning in the classic sense, ignoring the existence of other agents:
  - Q learning: each agent uses $Q(S,a_i)$ independently
  - Q-learning does not play stochastic policies. This prevents Q-learners from being convergent in self-play
  - One may use a *soft* Q-learning (stochastic policies)
- Joint Action learners (JALs), instead, learn the value of their own actions in conjunction with those of other agents via integration of RL with equilibrium (or coordination) learning methods: $Q(S,a_i, a_{-i})$

Claus, Caroline, and Craig Boutilier. "The dynamics of reinforcement learning in cooperative multiagent systems." *AAAI/IAAI* 1998 (1998): 746-752.

Difficulty in Independent learners (ILs)

- assuming other agents are not learning is not very realistic
- if player 1 is playing the equilibrium strategy, the other may play a deterministic strategy and get the same reward
- However, once player 2 leaves the equilibrium, a learning player 1 can exploit that fact and play some policy which will lower the reward for player 2


Player 1 exploits deterministic policy of player 2 (note that R2 = -R1)

Player 2 leaves the equilibrium to deterministic policy , maintaining reward

Neto G. From single-agent to multi-agent reinforcement learning: Foundational concepts and methods[J]. Learning theory course, 2005.

Joint Action learners (JALs)

- Q-values are based on the joint-actions rather than just their own actions
  - relies on full observability of the state and of the other agents' actions
- However, as agents are not coordinated, there is no guarantee that the other players are at the same learning stage, or even if they are learning at all
- The Q-value can be updated on the basis of the observed actions

$$EV(a_i) = \sum_{a_{-i} \in A_{-i}} Q(\langle a_i, a_{-i} \rangle) \prod_{j \neq i} \widehat{\pi}_j(a_{-i}[j])$$

This is a stateless case, but multiple states cases can be done similarly

Claus, Caroline, and Craig Boutilier. "The dynamics of reinforcement learning in cooperative multiagent systems." *AAAI/IAAI* 1998 (1998): 746-752.

Opponent Modeling/Fictitious Play

- Learn explicit models of the other players, assuming that they are playing according to a stationary policy
- Like JALs, statistics of the number of visits to a state and the number of times an opponent chooses an action are maintained to obtain policy estimators for the other players.

```
Algorithm: Opponent Modeling Q-Learning for player $i$

(1) Initialize $Q$ arbitrarily, and $\forall s \in S, a_{-i} \in A_{-i}\ C(s, a_{-i}) \leftarrow 0$ and $n(s) \leftarrow 0$.
(2) Repeat,
    (a) From state $s$ select action $a_i$ that maximizes,
            $\sum_{a_{-i}} \frac{C(s, a_{-i})}{n(s)} Q(s, \langle a_i, a_{-i} \rangle)$
    (b) Observing other agents' actions $a_{-i}$, reward $r$, and next state $s'$,
            $Q(s, a) \leftarrow (1 - \alpha) Q(s, a) + \alpha(r + \gamma V(s'))$
            $C(s, a_{-i}) \leftarrow C(s, a_{-i}) + 1$
            $n(s) \leftarrow n(s) + 1$
        where,
            $a = (a_i, a_{-i})$
            $V(s) = \max_{a_i} \sum_{a_{-i}} \frac{C(s, a_{-i})}{n(s)} Q(s, \langle a_i, a_{-i} \rangle).$
```

To be more precise, opponent modeling regards the other agents as **one massive opponent** with the ability to play joint actions and maintains statistics over them

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

Opponent Modeling/Fictitious Play

- Learn explicit models of the other players, assuming that they are playing according to a stationary policy
- Like JALs, statistics of the number of visits to a state and the number of times an opponent chooses an action are maintained to obtain policy estimators for the other players.

```
Algorithm: Opponent Modeling Q-Learning for player $i$

(1) Initialize $Q$ arbitrarily, and $\forall s \in S, a_{-i} \in A_{-i}$ $C(s, a_{-i}) \leftarrow 0$ and $n(s) \leftarrow 0$.
(2) Repeat,
    (a) From state $s$ select action $a_i$ that maximizes,
            $n(s) \leftarrow n(s) + 1$

    where,

            $a = (a_i, a_{-i})$
            $V(s) = \max_{a_i} \sum_{a_{-i}} \frac{C(s, a_{-i})}{n(s)} Q\left(s, \langle a_i, a_{-i} \rangle\right).$
```

Essentially, JALs/Opponent Modeling/Fictitious Play are the same. However, we have three subtle variations:

1) The current opponent strategies are given **(Centralized solution)**
2) We don't know the current opponent strategies but maintain an estimation from observed actions for each opponent **(Distributed solution)**
3) We keep one estimation for all other agents **(Distributed solution)**

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

Discussions on JALs

- Like single-agent learners, opponent modelling is rational.
  - This is because eventually the player’s estimates of its opponent’s policy will converge to the true policy.
  - Since it finds best-response policies given its estimates, eventually it will converge to a best-response policy to the opponent’s true policy.
- Also, like single-agent learning it is not convergent.
  - The reason is identical: it only plays pure policies, and so cannot converge in games with only mixed equilibria.
- Even if the learners capable of playing stochastic policies, JALs still may not converge in self-play

Gradient ascent

- A simple two-player, two-action, general-sum repeated matrix games
  - the *row* player selects action i and
  - the *column* player selects action j
  - the row player receives a payoff r_{ij} and the column player receives the payoff c_{ij}
- α ∈ [0, 1] is a strategy for the row player, where α corresponds to the probability of selecting the first action and 1 – α is the probability the player selects the second action
- Similarly, β is a strategy for the column player
- The joint strategy (α, β) is a point constrained to the unit square

row player payoffs

$$R_r = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix},$$

col player payoffs

$$R_c = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}.$$

Vr(α,β) and Vc(α,β) are expected payoffs

$$\begin{aligned} V_r(\alpha,\beta) &= \alpha\beta r_{11} + \alpha(1-\beta)r_{12} + (1-\alpha)\beta r_{21} + (1-\alpha)(1-\beta)r_{22} \\ &= u\alpha\beta + \alpha(r_{12} - r_{22}) + \beta(r_{21} - r_{22}) + r_{22}, \\ V_c(\alpha,\beta) &= \alpha\beta c_{11} + \alpha(1-\beta)c_{12} + (1-\alpha)\beta c_{21} + (1-\alpha)(1-\beta)c_{22} \\ &= u'\alpha\beta + \alpha(c_{12} - c_{22}) + \beta(c_{21} - c_{22}) + c_{22}, \end{aligned}$$

where,

$$u = r_{11} - r_{12} - r_{21} + r_{22},$$

$$u' = c_{11} - c_{12} - c_{21} + c_{22}.$$

Singh, Satinder, Michael Kearns, and Yishay Mansour. "Nash convergence of gradient dynamics in general-sum games." *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000.

Gradient ascent

- By gradient ascent, a player can adjust its strategy after each iteration so as to increase its expected payoffs
  - the player can move their strategy in the direction of the current gradient with some step size, *η*

row player payoffs

$$R_r = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix},$$

col player payoffs

$$R_c = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}.$$

$$\alpha_{k+1} = \alpha_k + \eta \frac{\partial V_r(\alpha_k, \beta_k)}{\partial \alpha_k},$$

$$\beta_{k+1} = \beta_k + \eta \frac{\partial V_r(\alpha_k, \beta_k)}{\partial \beta_k}.$$

where,

$$\frac{\partial V_r(\alpha, \beta)}{\partial \alpha} = \beta u + (r_{12} - r_{22}),$$

$$\frac{\partial V_c(\alpha, \beta)}{\partial \beta} = \alpha u' + (c_{21} - c_{22}).$$

- This can be consider a simple JAL when the opponent strategy is given: $EV(a_i) = \sum_{a_{-i} \in A_{-i}} Q(\langle a_i, a_{-i} \rangle) \prod_{j \neq i} \widehat{\pi}_j(a_{-i}[j])$

Singh, Satinder, Michael Kearns, and Yishay Mansour. "Nash convergence of gradient dynamics in general-sum games." *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000.

Gradient ascent

- By gradient ascent, a player can adjust its strategy after each iteration so as to increase its expected payoffs
  - the player can move their strategy in the

row player payoffs

$$R_r = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix},$$

col player payoffs

What will happen **if both players are using gradient ascent to update their strategies?**
1) this algorithm is rational because fixing the other player’s strategy will eventually force the player to converge to the optimal pure strategy response
2) the algorithm is, however, not convergent

$$\alpha_{k+1} = \alpha_k + \eta \frac{}{\partial \alpha_k},$$

$$\beta_{k+1} = \beta_k + \eta \frac{\partial V_r(\alpha_k, \beta_k)}{\partial \beta_k}.$$

where,

$$\frac{}{\partial \alpha} = \beta u + (r_{12} - r_{22}),$$

$$\frac{\partial V_c(\alpha, \beta)}{\partial \beta} = \alpha u' + (c_{21} - c_{22}).$$

- This can be consider a simple JAL when the opponent strategy is given: $EV(a_i) = \sum_{a_{-i} \in A_{-i}} Q(\langle a_i, a_{-i} \rangle) \prod_{j \neq i} \widehat{\pi}_j(a_{-i}[j])$

Singh, Satinder, Michael Kearns, and Yishay Mansour. "Nash convergence of gradient dynamics in general-sum games." *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000.

Infinitesimal Gradient Ascent

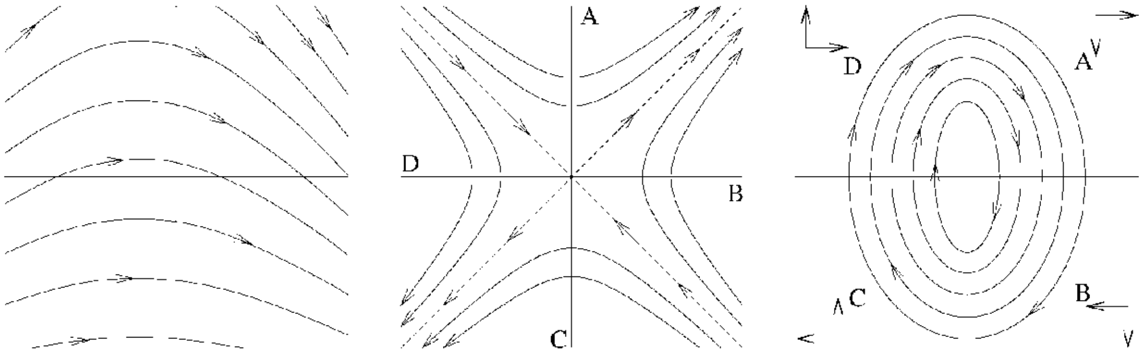
- Consider gradient ascent for the limiting case of infinitesimal step $\lim_{\eta \to 0}$ (IGA)
  - an algorithm with an appropriately decreasing step size will have the same properties as IGA

Theorem (IGA). *If both players follow Infinitesimal Gradient Ascent* (*IGA*), *where* η → 0, *then their strategies will converge to a Nash equilibrium OR the average payoffs over time will converge in the limit to the expected payoffs of a Nash equilibrium*

the dynamics of the strategy pair

$$\begin{bmatrix} \frac{\partial \alpha}{\partial t} \\ \frac{\partial \beta}{\partial t} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & u \\ u' & 0 \end{bmatrix}}_{U} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} (r_{12} - r_{22}) \\ (c_{21} - c_{22}) \end{bmatrix}.$$

$$(\alpha^*, \beta^*) = \left(\frac{(c_{22} - c_{21})}{u'}, \frac{(r_{22} - r_{12})}{u} \right).$$



(a) (b) (c)

converge to a point on the boundary NE *converge to a NE point* *center is inside the unit square. Not converge to a NE point*

the IGA dynamics. (a) U is not invertible. (b) U has real eigenvalues. (c) U has imaginary eigenvalues.

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

Singh, Satinder, Michael Kearns, and Yishay Mansour. "Nash convergence of gradient dynamics in general-sum games." *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000.

Infinitesimal Gradient Ascent

- Consider gradient ascent for the limiting case of infinitesimal step $\lim_{\eta \to 0}$ (IGA)
  - an algorithm with an appropriately decreasing step size will have the same properties as IGA

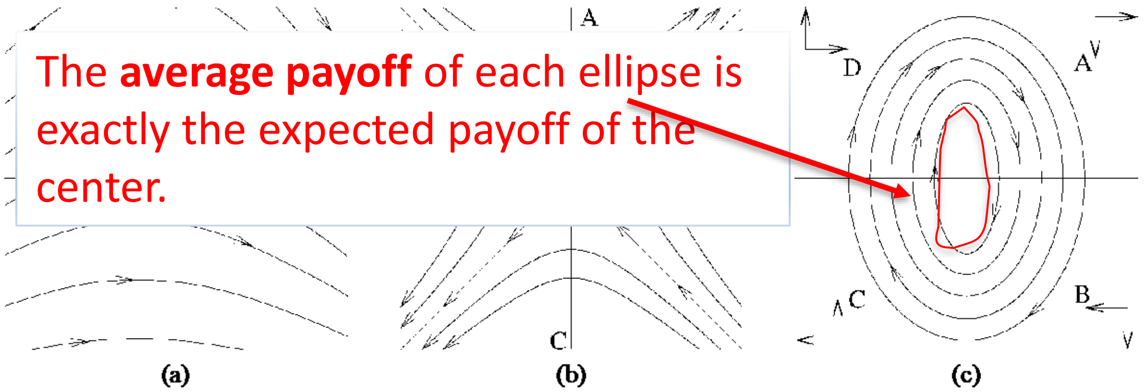
Theorem (IGA). *If both players follow Infinitesimal Gradient Ascent* (*IGA*), *where η → 0, then their strategies will converge to a Nash equilibrium OR the average payoffs over time will converge in the limit to the expected payoffs of a Nash equilibrium*

the dynamics of the strategy pair

$$\begin{bmatrix} \frac{\partial \alpha}{\partial t} \\ \frac{\partial \beta}{\partial t} \end{bmatrix} = \begin{bmatrix} 0 & u \\ u' & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} (r_{12} - r_{22}) \\ (c_{21} - c_{22}) \end{bmatrix}$$

U

$$(\alpha^*, \beta^*) = \left(\frac{(c_{22} - c_{21})}{u'}, \frac{(r_{22} - r_{12})}{u} \right).$$



The **average payoff** of each ellipse is exactly the expected payoff of the center.

converge to a point on the boundary NE *converge to a NE point* *center is inside the unit square. Not converge to a NE point*

the IGA dynamics. (a) U is not invertible. (b) U has real eigenvalues. (c) U has imaginary eigenvalues.

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

Singh, Satinder, Michael Kearns, and Yishay Mansour. "Nash convergence of gradient dynamics in general-sum games." *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2000.

Infinitesimal Gradient ascent

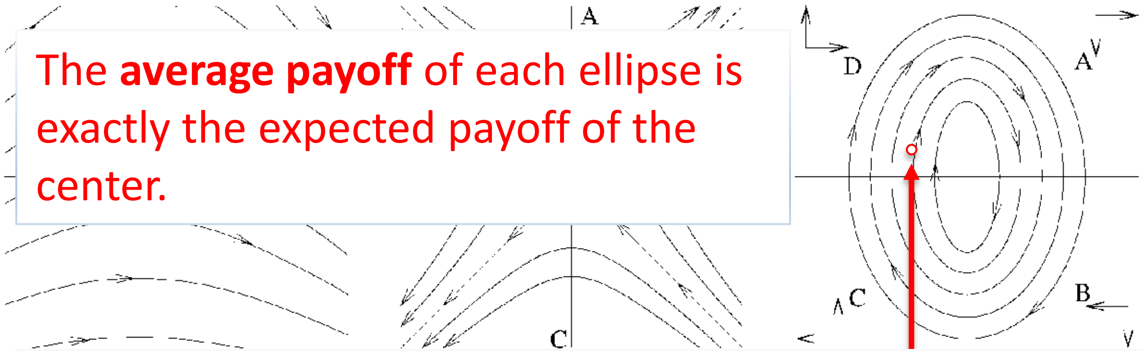
- Consider gradient ascent for the limiting case of infinitesimal step $\lim_{\eta \to 0}$ (IGA)
  - an algorithm with an appropriately decreasing step size will have the same properties as IGA

Theorem (IGA). *If both players follow Infinitesimal Gradient Ascent* (*IGA*), *where* η → 0, *then their strategies will converge to a Nash equilibrium OR the average payoffs over time will converge in the limit to the expected payoffs of a Nash equilibrium*

the dynamics of the strategy pair

$$\begin{bmatrix} \frac{\partial \alpha}{\partial t} \\ \frac{\partial \beta}{\partial t} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & u \\ u' & 0 \end{bmatrix}}_{U} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} (r_{12} - r_{22}) \\ (c_{21} - c_{22}) \end{bmatrix}.$$

$$(\alpha^*, \beta^*) = \left(\frac{(c_{22} - c_{21})}{u'}, \frac{(r_{22} - r_{12})}{u} \right).$$



The **average payoff** of each ellipse is exactly the expected payoff of the center.

However, at any moment in time **the expected payoff** of a player could be arbitrarily poor. 1) difficult to evaluate a learner, 2) difficult in temporal difference learning for multiple state stochastic games.

conv

the

Bo

Sir

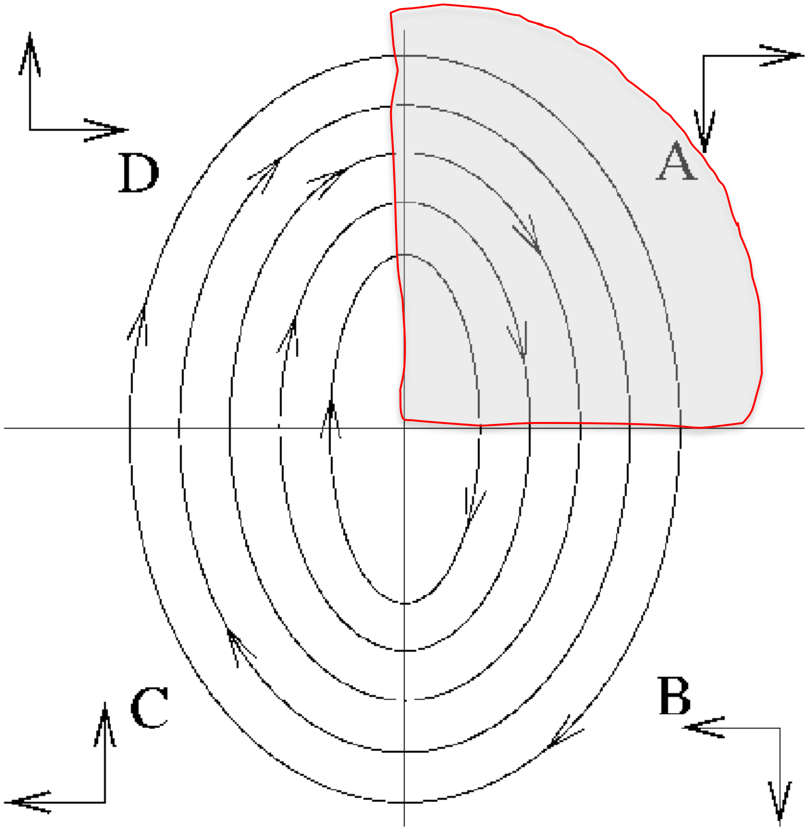
co

36.2 (2002): 215-250.

eedings of the Sixteenth

Why not converge?

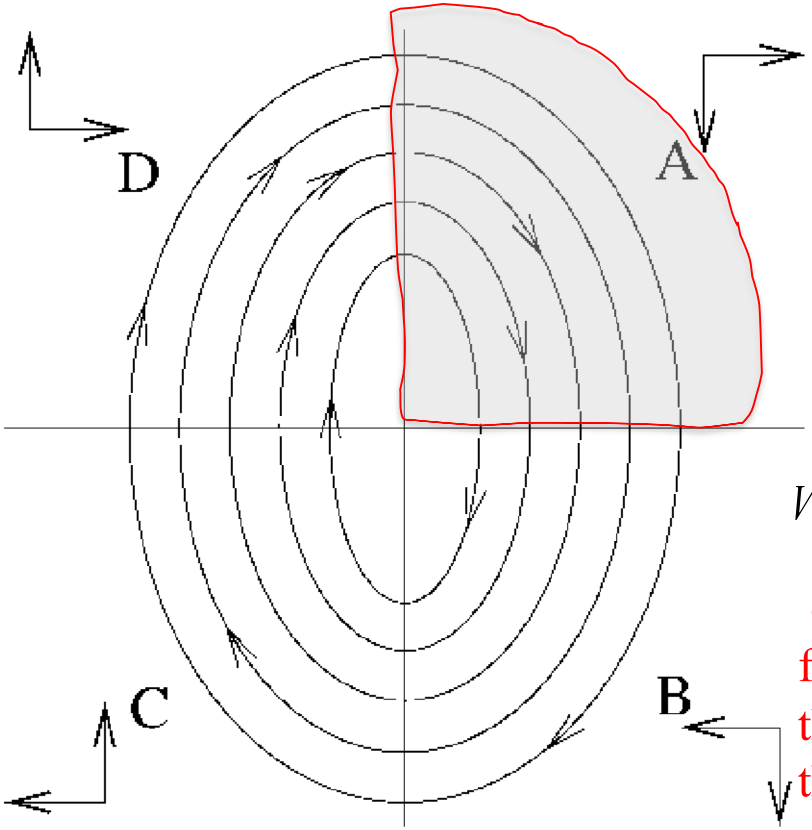
- In any region, if one player approaches to the center, the other moves away



The center is inside the unit square. not converge to a NE point

How we know someone moves away?

- **Lemma.** *The player is "winning" if and only if that player's strategy is moving away from the center.*



The player is "winning" when its current expected payoff is larger than the expected payoffs if it were to play its selected equilibrium.

$$V_r(\alpha, \beta) - V_r(\alpha^e, \beta) > 0.$$

α^e is the equilibrium strategy for the row player, and β^e is the equilibrium strategy for the column player.

The center is inside the unit square. not converge to a NE point

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

How we know someone moves away?

- **Lemma.** *The player is "winning" if and only if that player's strategy is moving away from the center.*

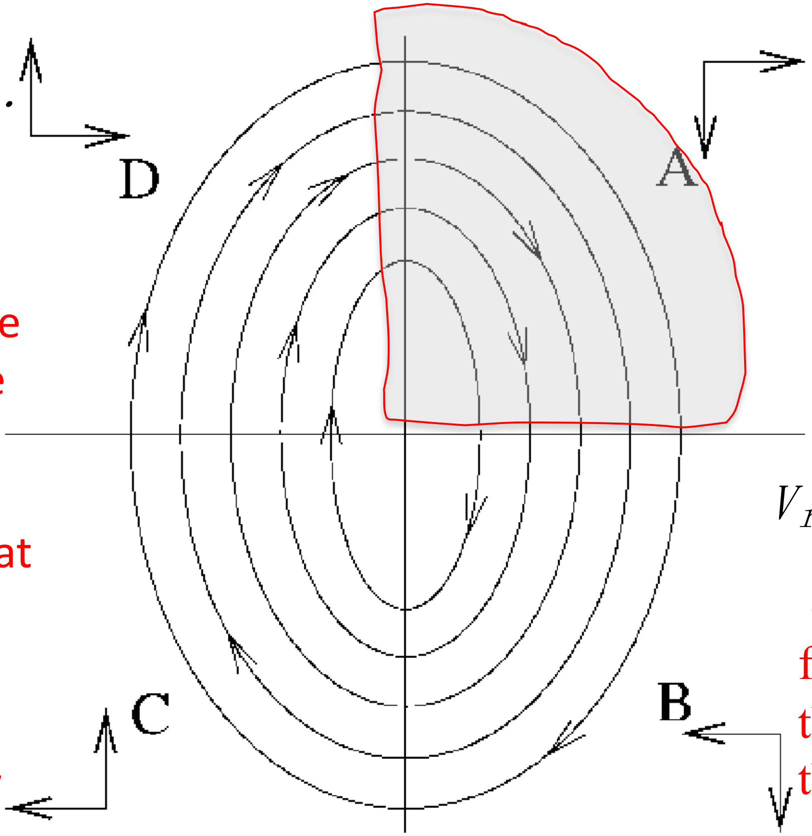
The winning condition

$$V_r(\alpha, \beta) - V_r(\alpha^e, \beta) > 0.$$

$$(\alpha - \alpha^*)\frac{\partial V_r(\alpha, \beta)}{\partial \alpha} > 0.$$

the above is true when the two left hand factors have the same sign.

Therefore 1) strategy α is greater than the strategy at the center α∗ and it is increasing 2) it's smaller than the center and decreasing-> **moves away**



The player is "winning" when its current expected payoff is larger than the expected payoffs if it were to play its selected equilibrium.

$$V_r(\alpha, \beta) - V_r(\alpha^e, \beta) > 0.$$

α^e is the equilibrium strategy for the row player, and β^e is the equilibrium strategy for the column player.

The center is inside the unit square. not converge to a NE point

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

WoLF (Win or Learn Fast)

- So we can have a variable learning rate:

$$\alpha_{k+1} = \alpha_k + \eta \ell_k^r \frac{\partial V_r(\alpha_k, \beta_k)}{\partial \alpha},$$

$$\beta_{k+1} = \beta_k + \eta \ell_k^c \frac{\partial V_r(\alpha_k, \beta_k)}{\partial \beta},$$

where the variable learning rates are given,

$$\ell_k^{r,c} \in [\ell_{\min}, \ell_{\max}] > 0$$

- How to update the learning rate?
  - the WoLF (“Win or Learn Fast”) principle (learn quickly when losing, and cautiously when winning

$$\ell_k^r = \begin{cases} \ell_{\min} & \text{if } V_r(\alpha_k, \beta_k) > V_r(\alpha^e, \beta_k) \quad \text{WINNING}, \\ \ell_{\max} & \text{otherwise} \quad \text{LOSING}, \end{cases}$$

$$\ell_k^c = \begin{cases} \ell_{\min} & \text{if } V_c(\alpha_k, \beta_k) > V_c(\alpha_k, \beta^e) \quad \text{WINNING}, \\ \ell_{\max} & \text{otherwise} \quad \text{LOSING}. \end{cases}$$

α^e is the equilibrium strategy for the row player, and β^e is the equilibrium strategy for the column player.

- The intuition:
  - a learner should adapt quickly when it is doing more poorly than expected.
  - when it is doing better than expected, it should be cautious since the other players are likely to change their policy

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

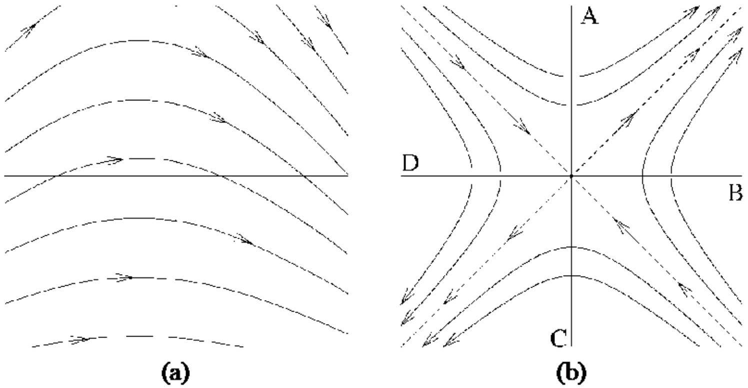
WoLF-IGA

- With IGA, we have the following dynamics

$$\begin{bmatrix} \frac{\partial\alpha}{\partial t} \\ \frac{\partial\beta}{\partial t} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & \ell^{r}(t)u \\ \ell^{c}(t)u' & 0 \end{bmatrix}}_{U(t)} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} \ell^{r}(t)(r_{12} - r_{22}) \\ \ell^{c}(t)(c_{21} - c_{22}) \end{bmatrix}.$$

- **Theorem (WoLF-IGA)** If in a two-person, two-action, iterated general-sum game, both players follow the WoLF-IGA algorithm (with $\ell_{\max} > \ell_{\min}$), then their strategies will converge to a Nash equilibrium.

Convergence is the same as before for
(a) U(t) is *invertible*
(b) *(b)* U(t) has has purely real eigenvalues



(a) *converge to a point on the boundary NE*

(b) *converge to a NE point*

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

WoLF-IGA

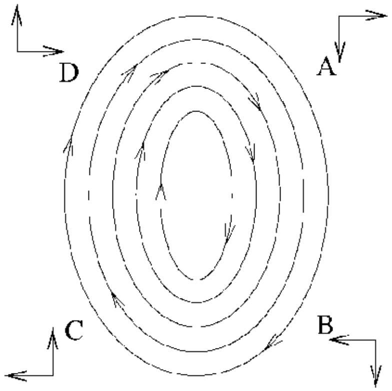
- With IGA, we have the following dynamics

$$\begin{bmatrix} \frac{\partial \alpha}{\partial t} \\ \frac{\partial \beta}{\partial t} \end{bmatrix} = \begin{bmatrix} 0 & \ell^{r}(t)u \\ \ell^{c}(t)u' & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} \ell^{r}(t)(r_{12} - r_{22}) \\ \ell^{c}(t)(c_{21} - c_{22}) \end{bmatrix}.$$

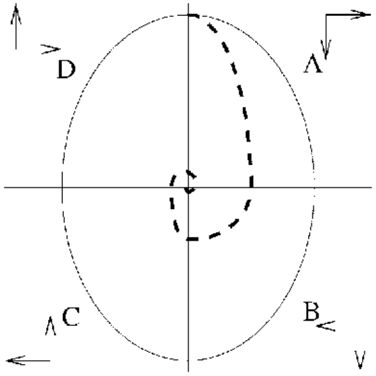
U(t)

- **Theorem (WoLF-IGA)** If in a two-person, two-action, iterated general-sum game, both players follow the WoLF-IGA algorithm (with $\ell_{\max} > \ell_{\min}$), then their strategies will converge to a Nash equilibrium.

But for U(t) has purely imaginary eigenvalues and *center is inside the unit square, it becomes converged!*



IGA



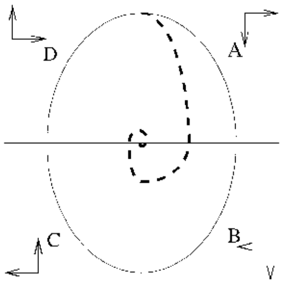
WoLF-IGA

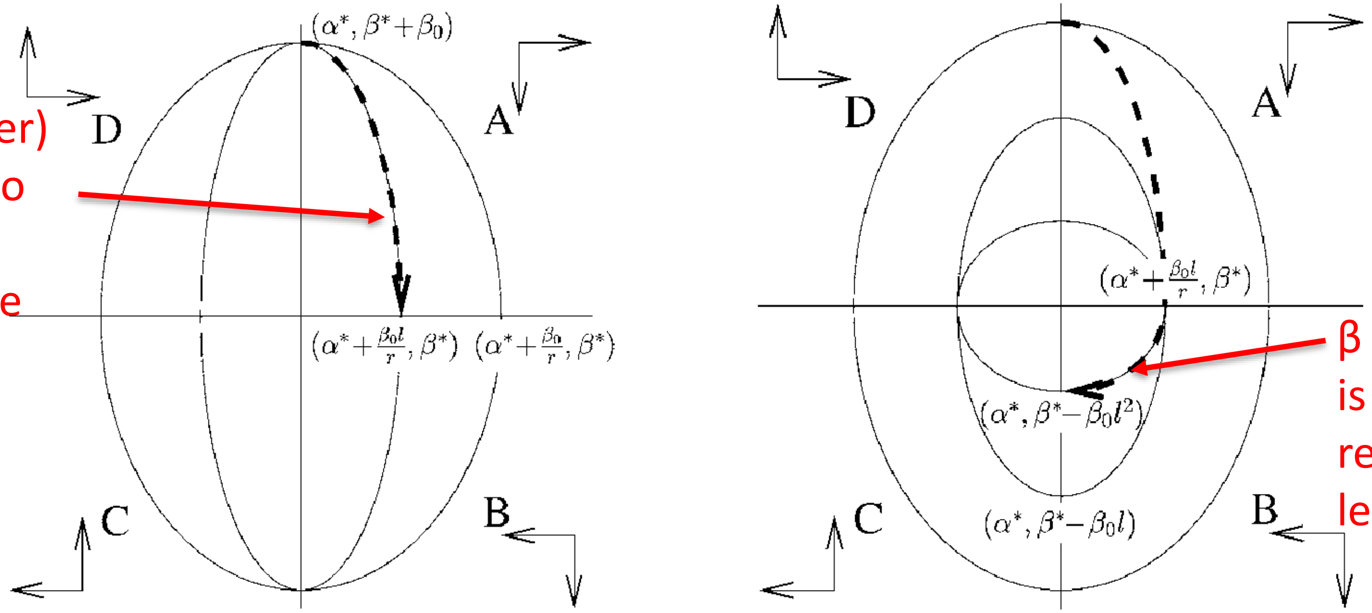
Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

WoLF-IGA

- **Lemma (WoLF-IGA)** For any initial strategy pair, $(\alpha^*, \beta^* + \beta_0)$ or $(\alpha* + \alpha_0, \beta^*)$, that is “sufficiently close” to the center, the strategy pair will converge to the center.

  “Sufficiently close” here means that the elliptical trajectory from this point defined when both players use 1 as their learning rate lies entirely within the unit square.





α (row player) is winning so reduce the learning rate

β (col player) is winning so reduce the learning rate

WoLF-IGA trajectory

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

WoLF-IGA

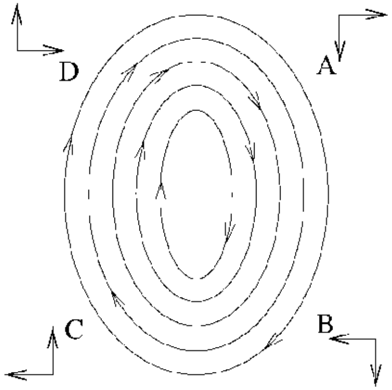
- With IGA, we have the following dynamics

$$\begin{bmatrix} \frac{\partial \alpha}{\partial t} \\ \frac{\partial \beta}{\partial t} \end{bmatrix} = \begin{bmatrix} 0 & \ell^r(t)u \\ \ell^c(t)u' & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} \ell^r(t)(r_{12} - r_{22}) \\ \ell^c(t)(c_{21} - c_{22}) \end{bmatrix}.$$

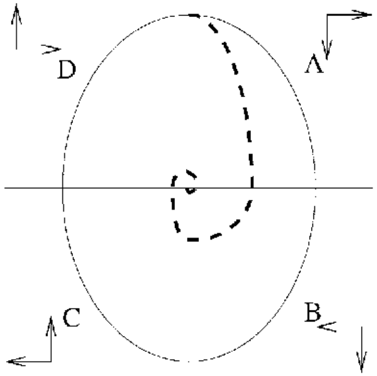
U(t)

- ***Theorem (WoLF-IGA)*** *If in a two-person, two-action, iterated general-sum game, both players follow the WoLF-IGA algorithm (with $\ell_{max} > \ell_{min}$), then their strategies will converge to a Nash equilibrium.*

convergence



IGA

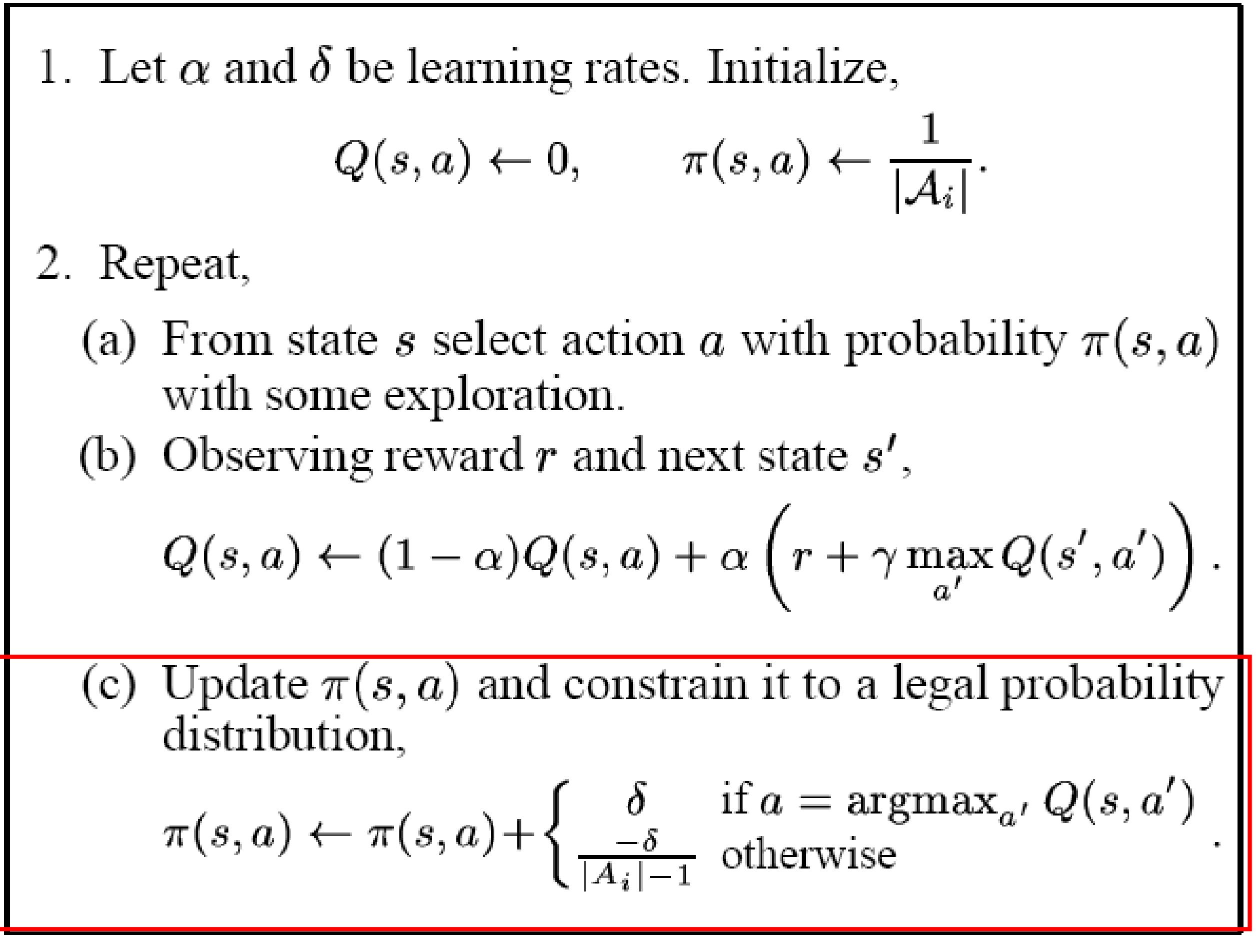


WoLF-IGA

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

Practical algorithm of *WoLF:* PHC Basis

Policy hill-climbing (PHC):
Simple Q-Learner that plays mixed strategies



```
1. Let $\alpha$ and $\delta$ be learning rates. Initialize,
       $Q(s,a) \leftarrow 0, \qquad \pi(s,a) \leftarrow \frac{1}{|\mathcal{A}_i|}.$
2. Repeat,
   (a) From state $s$ select action $a$ with probability $\pi(s,a)$ with some exploration.
   (b) Observing reward $r$ and next state $s'$,
       $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a')\right).$
   (c) Update $\pi(s,a)$ and constrain it to a legal probability distribution,
       $\pi(s,a) \leftarrow \pi(s,a) + \begin{cases} \delta & \text{if } a = \operatorname{argmax}_{a'} Q(s,a') \\ \frac{-\delta}{|A_i|-1} & \text{otherwise} \end{cases}.$
```

Updating a mixed strategy by giving more weight to the action that Q-learning believes is the best

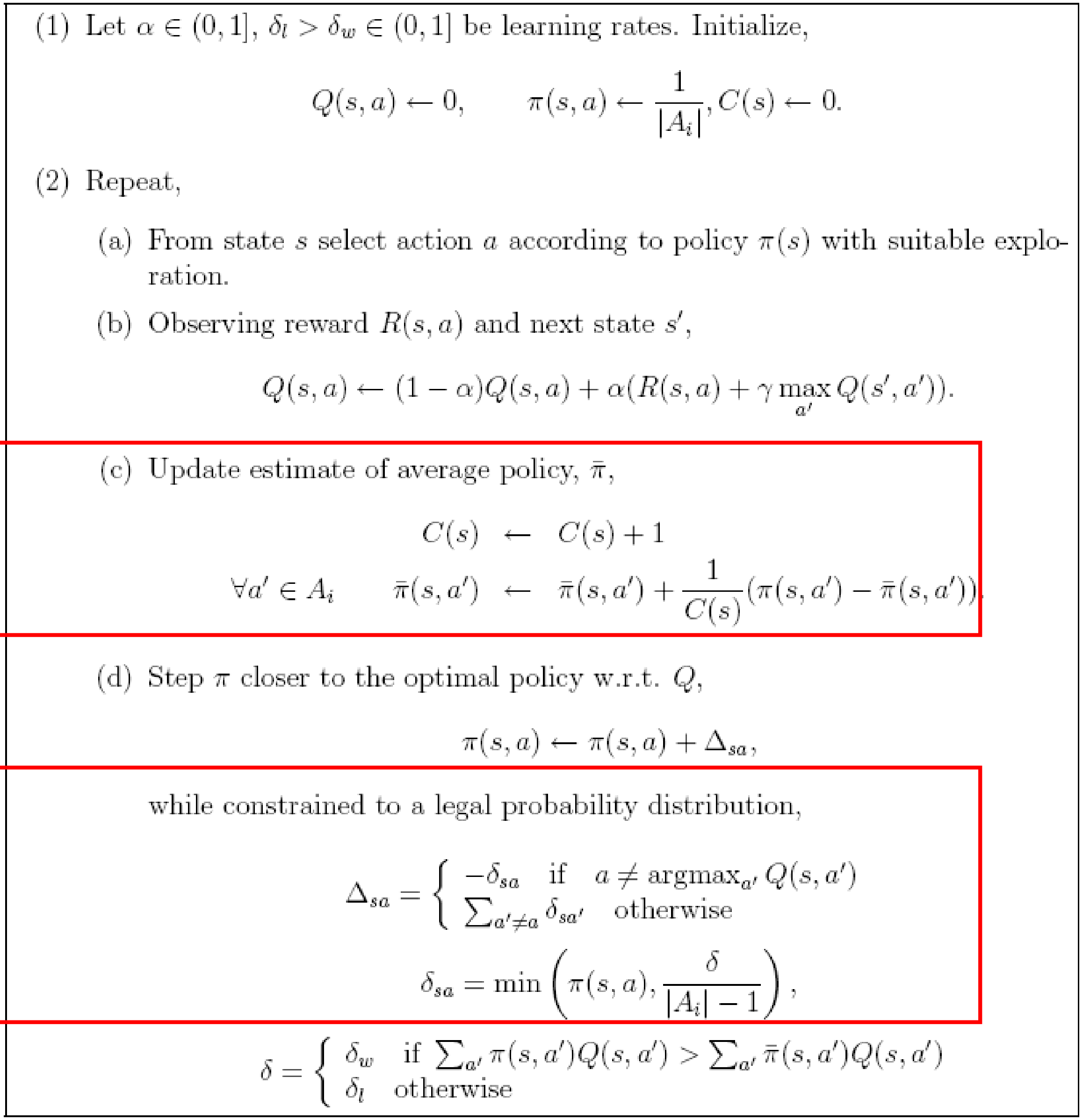
Problems:

- guarantees rationality against stationary opponents
- does not converge in self-play

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

Practical algorithm of *WoLF:* WoLF-HPC

- **Agents only need to see its own payoff**
- Converges for two player two action SG's in self-play



(1) Let $\alpha \in (0,1]$, $\delta_l > \delta_w \in (0,1]$ be learning rates. Initialize,

$$Q(s,a) \leftarrow 0, \qquad \pi(s,a) \leftarrow \frac{1}{|A_i|}, C(s) \leftarrow 0.$$

(2) Repeat,

(a) From state s select action a according to policy $\pi(s)$ with suitable exploration.

(b) Observing reward $R(s,a)$ and next state s',

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(R(s,a) + \gamma \max_{a'} Q(s',a')).$$

Maintaining average policy

Probability of playing action

(c) Update estimate of average policy, $\bar{\pi}$,

$$\begin{aligned} C(s) &\leftarrow C(s)+1 \\ \forall a' \in A_i \qquad \bar{\pi}(s,a') &\leftarrow \bar{\pi}(s,a') + \frac{1}{C(s)}(\pi(s,a') - \bar{\pi}(s,a')). \end{aligned}$$

(d) Step π closer to the optimal policy w.r.t. Q,

$$\pi(s,a) \leftarrow \pi(s,a) + \Delta_{sa},$$

Determination of "W" and "L":
by comparing the expected value of the current policy to that of the average policy

while constrained to a legal probability distribution,

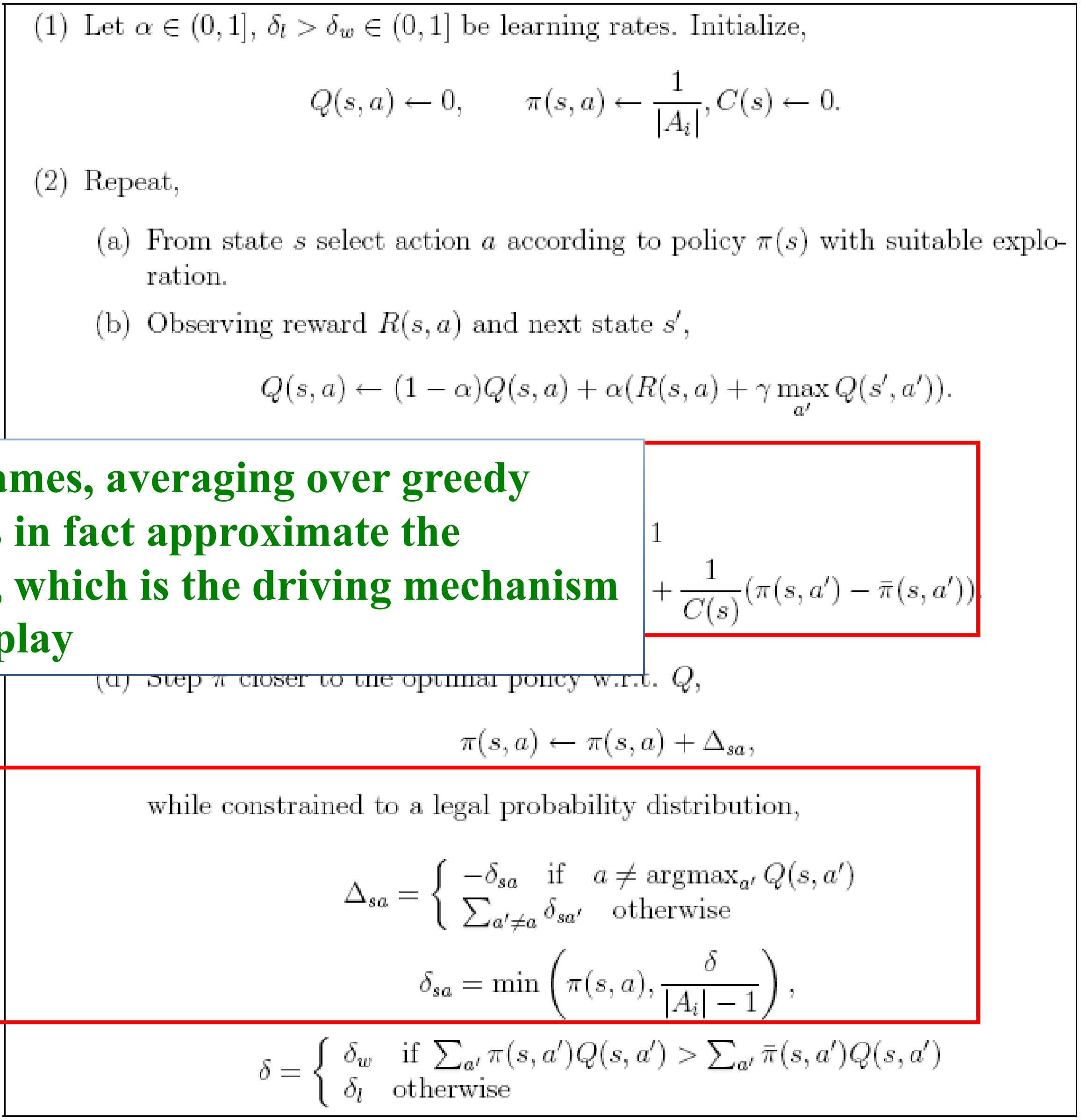
$$\Delta_{sa} = \begin{cases} -\delta_{sa} & \text{if } a \neq \operatorname{argmax}_{a'} Q(s,a') \\ \sum_{a' \neq a} \delta_{sa'} & \text{otherwise} \end{cases}$$

$$\delta_{sa} = \min\left(\pi(s,a), \frac{\delta}{|A_i|-1}\right),$$

$$\delta = \begin{cases} \delta_w & \text{if } \sum_{a'} \pi(s,a')Q(s,a') > \sum_{a'} \bar{\pi}(s,a')Q(s,a') \\ \delta_l & \text{otherwise} \end{cases}$$

Practical algorithm of *WoLF:* WoLF-HPC

- **Agent only need to see its own payoff**
- **Converges for two player two action SG's in self-play**



(1) Let $\alpha \in (0,1]$, $\delta_l > \delta_w \in (0,1]$ be learning rates. Initialize,

$$Q(s,a) \leftarrow 0, \qquad \pi(s,a) \leftarrow \frac{1}{|A_i|}, C(s) \leftarrow 0.$$

(2) Repeat,

(a) From state s select action a according to policy $\pi(s)$ with suitable exploration.

(b) Observing reward $R(s,a)$ and next state s',

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(R(s,a) + \gamma \max_{a'} Q(s',a')).$$

$$1$$

$$+ \frac{1}{C(s)}(\pi(s,a') - \bar{\pi}(s,a')).$$

(d) Step π closer to the optimal policy w.r.t. Q,

$$\pi(s,a) \leftarrow \pi(s,a) + \Delta_{sa},$$

while constrained to a legal probability distribution,

$$\Delta_{sa} = \begin{cases} -\delta_{sa} & \text{if } \quad a \neq \text{argmax}_{a'} Q(s,a') \\ \sum_{a' \neq a} \delta_{sa'} & \text{otherwise} \end{cases}$$

$$\delta_{sa} = \min\left(\pi(s,a), \frac{\delta}{|A_i| - 1}\right),$$

$$\delta = \begin{cases} \delta_w & \text{if } \sum_{a'} \pi(s,a')Q(s,a') > \sum_{a'} \bar{\pi}(s,a')Q(s,a') \\ \delta_l & \text{otherwise} \end{cases}$$

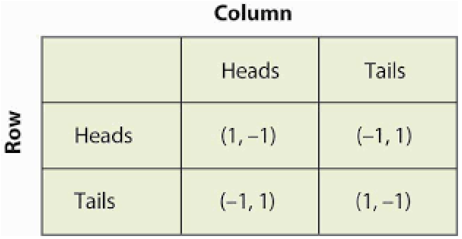
Maintaining average

Probability of playin

For many games, averaging over greedy policies does in fact approximate the equilibrium, which is the driving mechanism in fictitious play

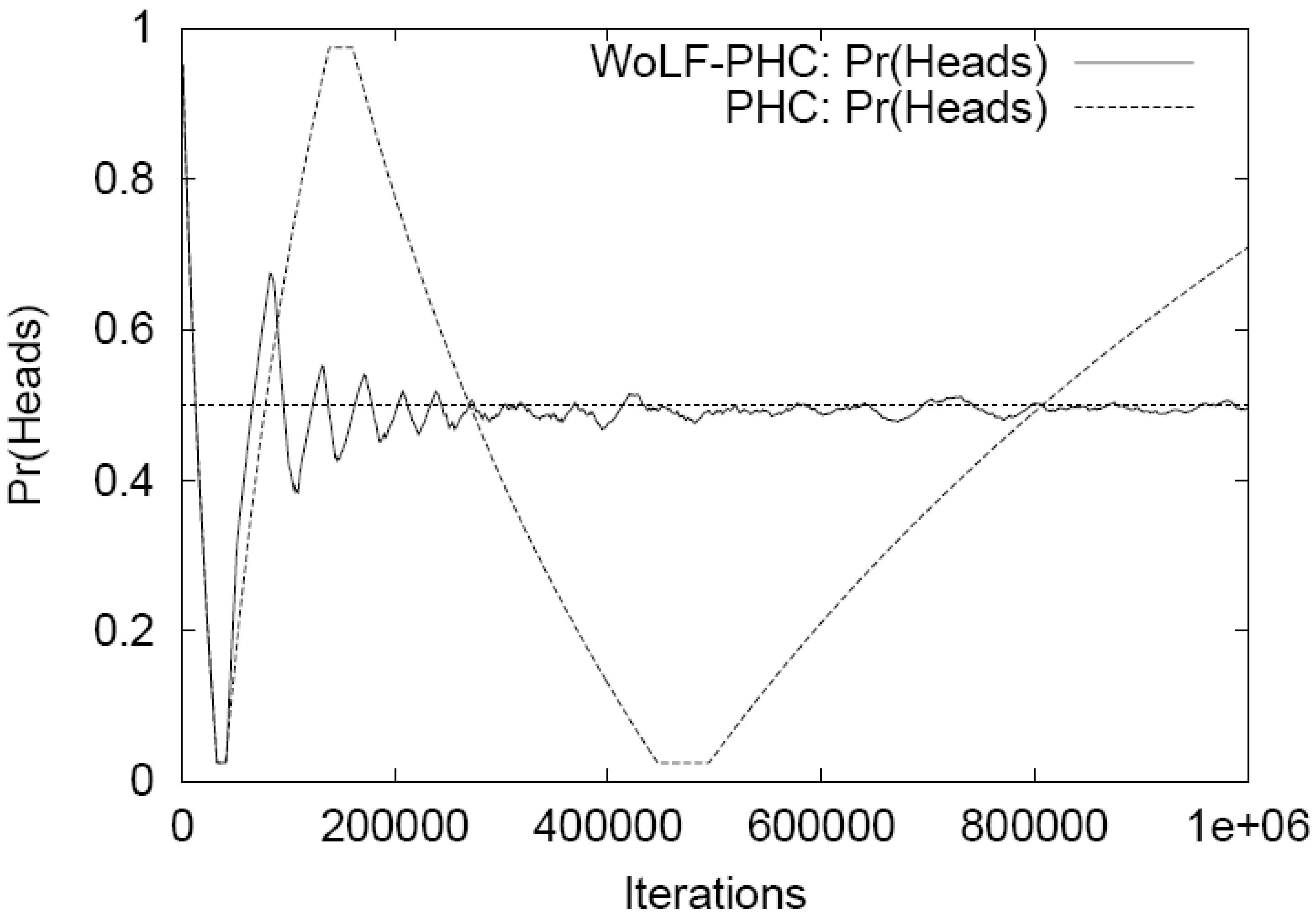
Determination of "W" and "L":
by comparing the expected value of the current policy to that of the average policy

Matching Pennies

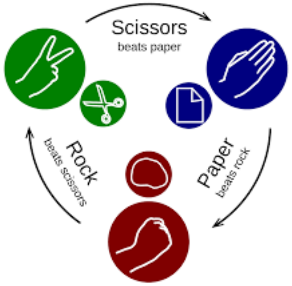


Column

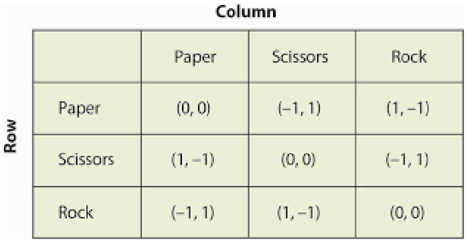
Row	Heads	Tails
Heads	(1, –1)	(–1, 1)
Tails	(–1, 1)	(1, –1)



Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.



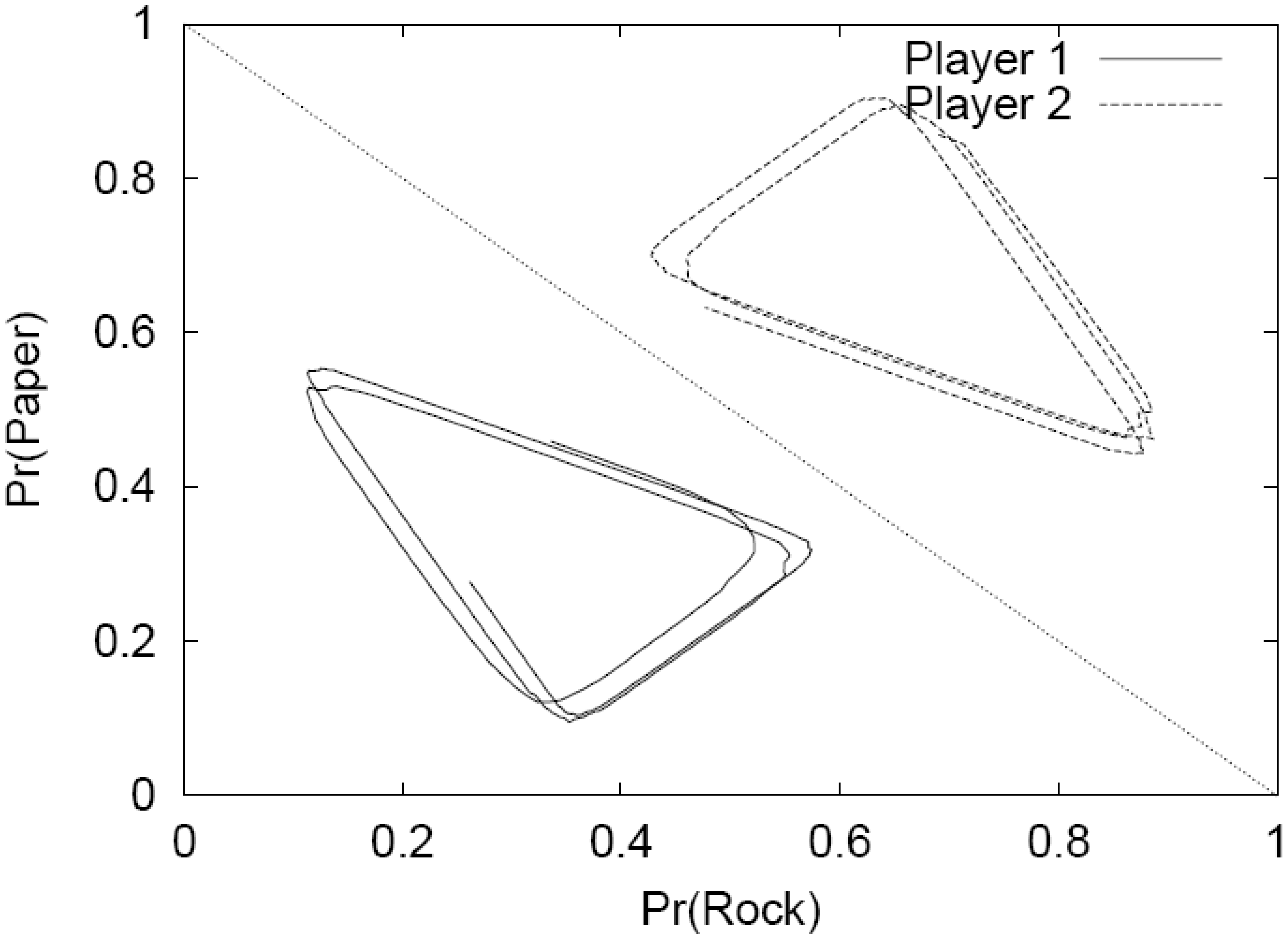
Rock-paper-scissors: PHC



Column

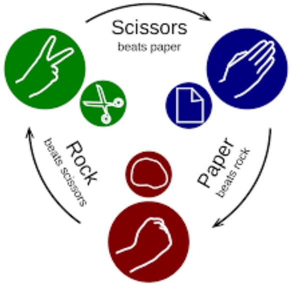
Row

	Paper	Scissors	Rock
Paper	(0, 0)	(−1, 1)	(1, −1)
Scissors	(1, −1)	(0, 0)	(−1, 1)
Rock	(−1, 1)	(1, −1)	(0, 0)

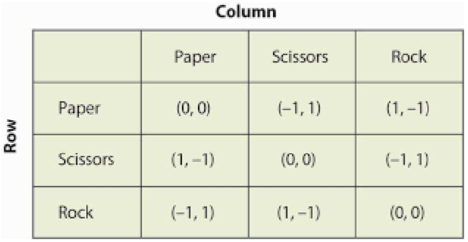


(a) Policy Hill-Climbing

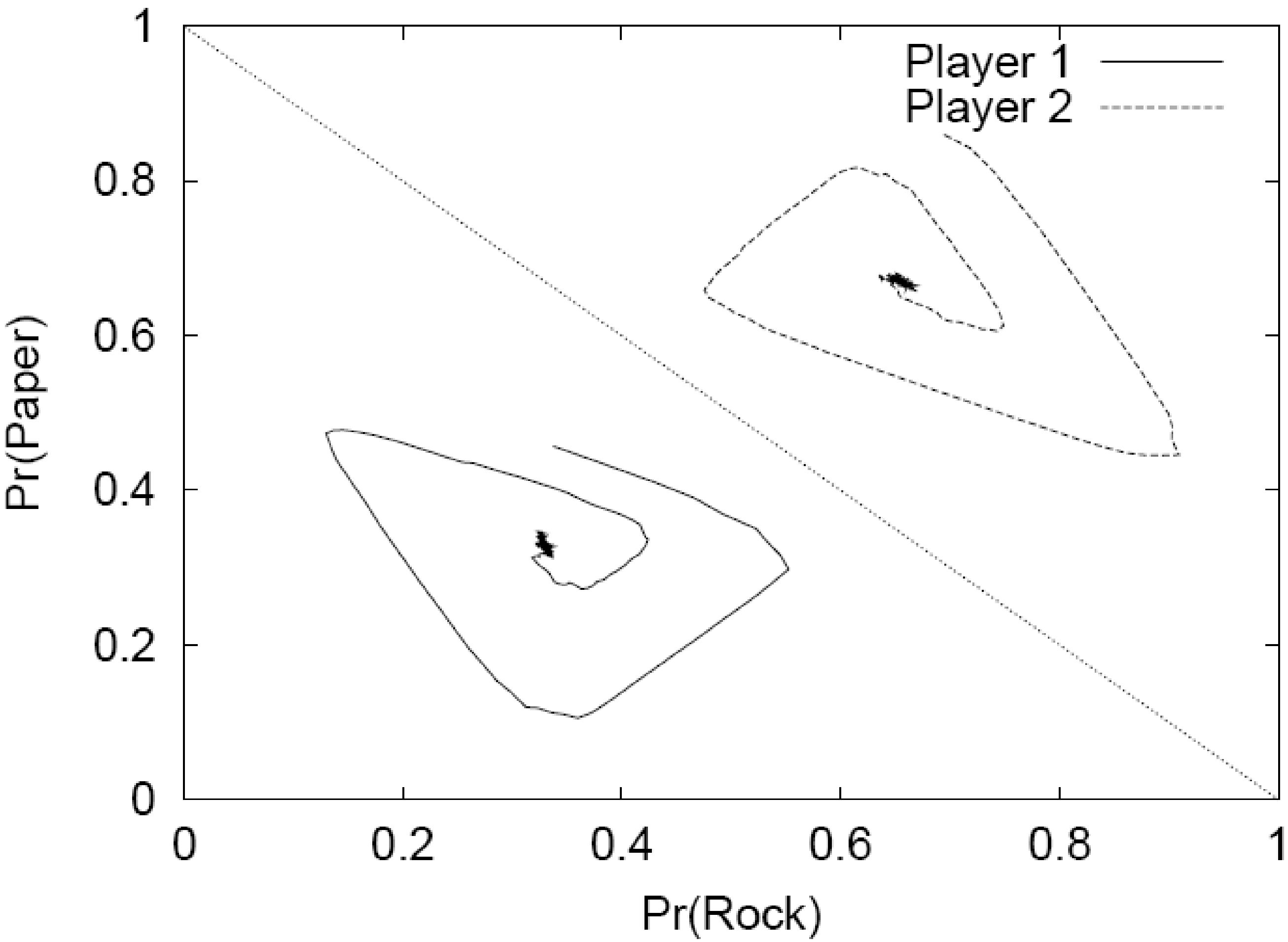
Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.



Rock-paper-scissors: WoLF PHC



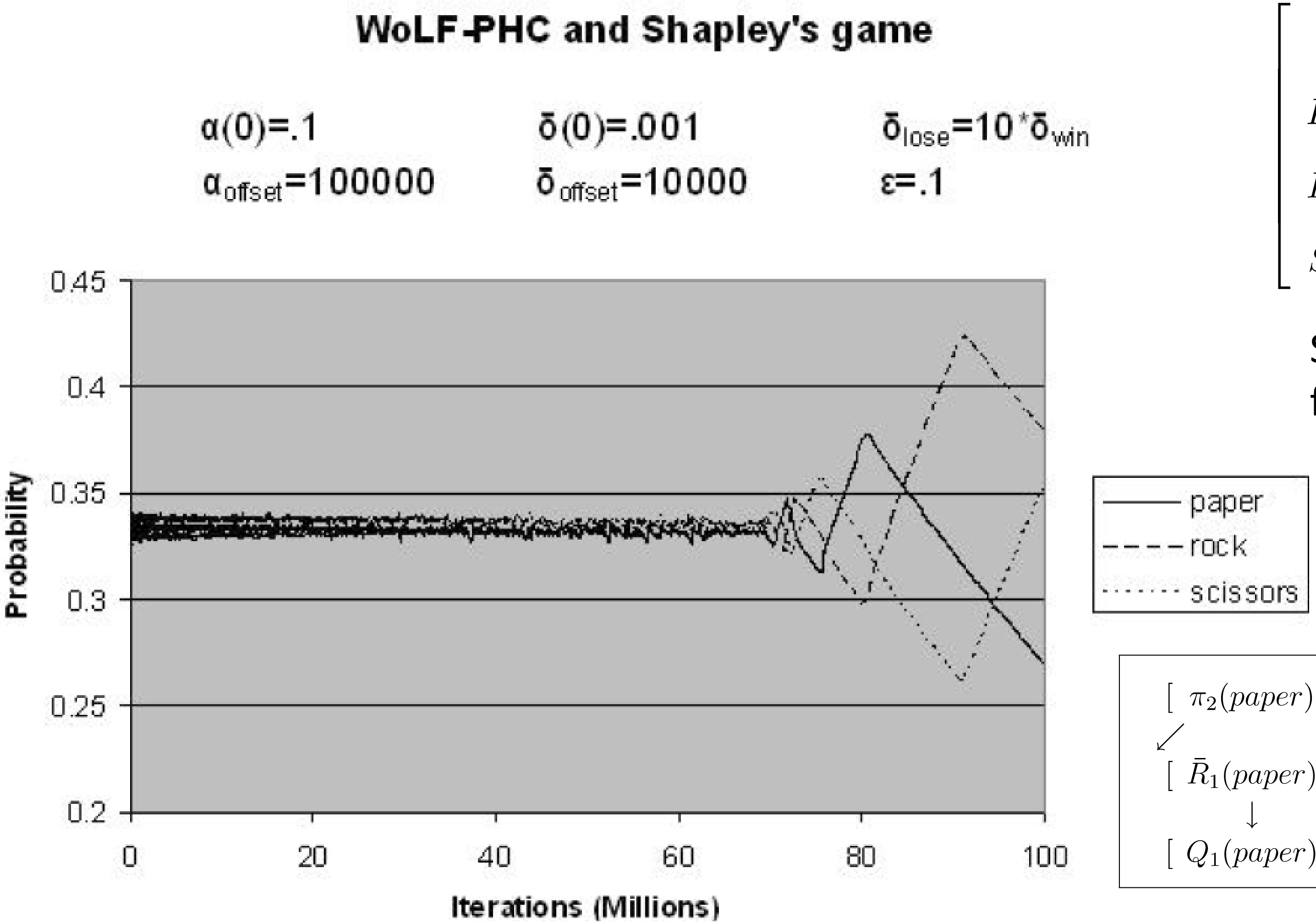
		Column		
		Paper	Scissors	Rock
Row	Paper	(0, 0)	(−1, 1)	(1, −1)
	Scissors	(1, −1)	(0, 0)	(−1, 1)
	Rock	(−1, 1)	(1, −1)	(0, 0)



(b) WoLF Policy Hill-Climbing

Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.

Limitation of WoLF PHC :Pseudo Convergence



$$\begin{bmatrix} & P & R & S \\ P & (0,0) & (1,0) & (0,1) \\ R & (0,1) & (0,0) & (1,0) \\ S & (1,0) & (0,1) & (0,0) \end{bmatrix}$$

Shapley's counter game for fictitious play

$$\begin{array}{ccccccccc} & [& \pi_2(paper) & , & \pi_2(rock) & , & \pi_2(scissors) &] & \\ \swarrow & & \swarrow & & \swarrow & & & & \swarrow \\ & [& \bar{R}_1(paper) & , & \bar{R}_1(rock) & , & \bar{R}_1(scissors) &] & \\ & & \downarrow & & \downarrow & & \downarrow & & \\ & [& Q_1(paper) & , & Q_1(rock) & , & Q_1(scissors) &] & \end{array}$$

The circular shift from one agent's policy to the other's average reward and Q-values

Cook, Philip R. "Limitations and extensions of the wolf-phc algorithm." (2007).

References

- Slides are based on
  - Game Theory, Second Edition, 2014 Thomas S. Ferguson Mathematics Department, UCLA
  - Bowling, Michael, and Manuela Veloso. "Multiagent learning using a variable learning rate." *Artificial Intelligence* 136.2 (2002): 215-250.
  - Neto G. From single-agent to multi-agent reinforcement learning: Foundational concepts and methods. Learning theory course, 2005.
  - Multiagent Reinforcement Learning (MARL) September 27, 2013 - ECML’13
  - Mul2-agent Reinforcement Learning， Subramanian Ramamoorthy， 2017
  - Game Theory and Multi-Agent Learning Mini-Tutorial， Enrique Munoz de Cote Politecnico di Milano