TÉCNICO LISBOA

Multiagent Learning

Reference: www.marl-book.com
Chapter 6 – 6.1, 6.2, 6.3

AASMA 2024

Manuel Lopes
manuel.lopes@tecnico.ulisboa.pt

Multiagent Learning Applications

Applications [edit]

Multi-agent reinforcement learning has been applied to a variety of use cases in science and industry:

- Broadband cellular networks such as 5G[32]
- Content caching[32]
- Packet routing[32]
- Computer vision[33]
- Network security[32]
- Transmit power control[32]
- Computation offloading[32]
- Language evolution research[34]
- Global health[35]
- Integrated circuit design[36]
- Internet of Things[32]
- Microgrid energy management[37]
- Multi-camera control[38]
- Autonomous vehicles[39]
- Sports analytics[40]
- Traffic control[41] (Ramp metering[42])
- Unmanned aerial vehicles[43][32]
- Wildlife conservation[44]

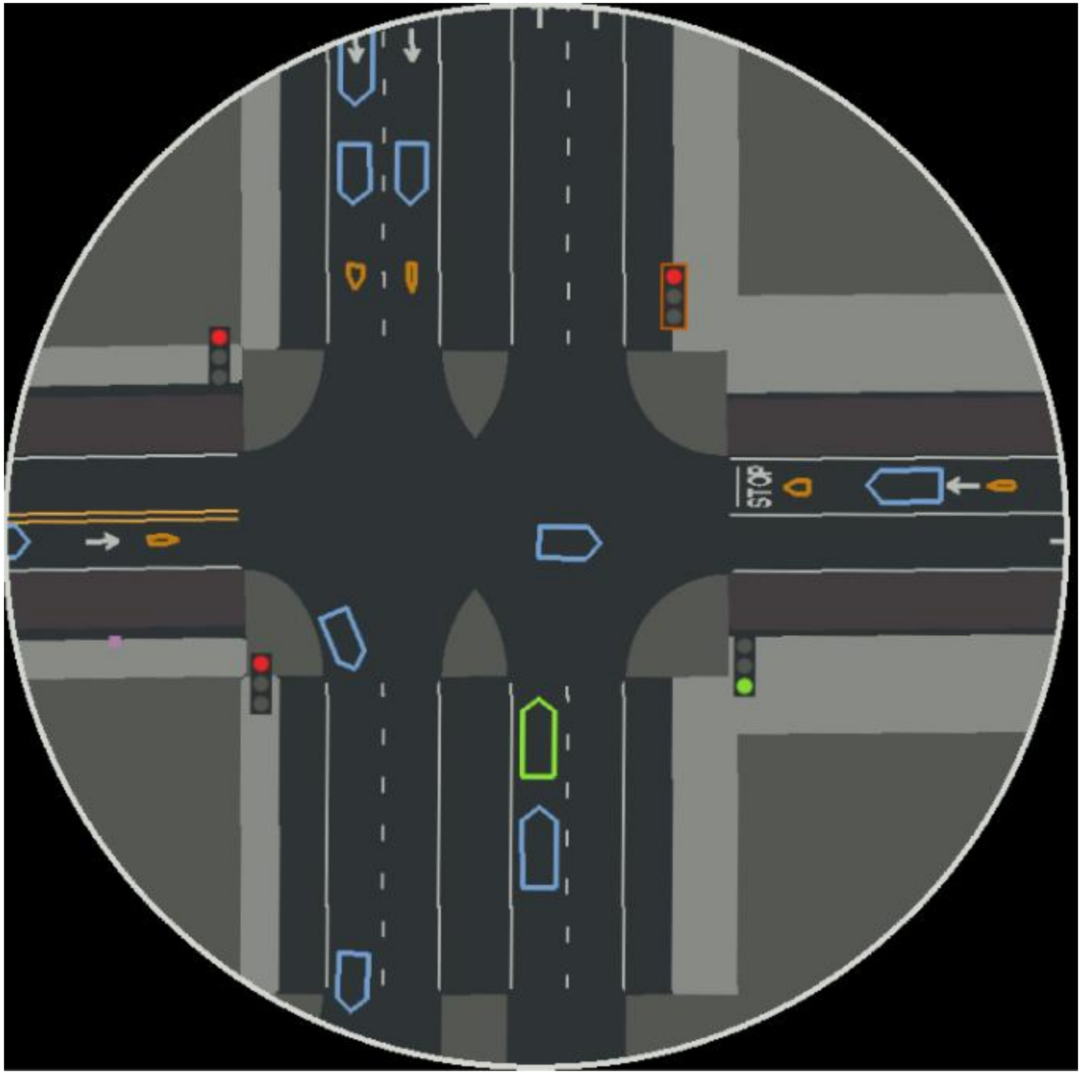
https://en.wikipedia.org/wiki/Multi-agent_reinforcement_learning

Learning to Play: The Multi-Agent Reinforcement Learning in MalmÖ (MARLÖ) Competition

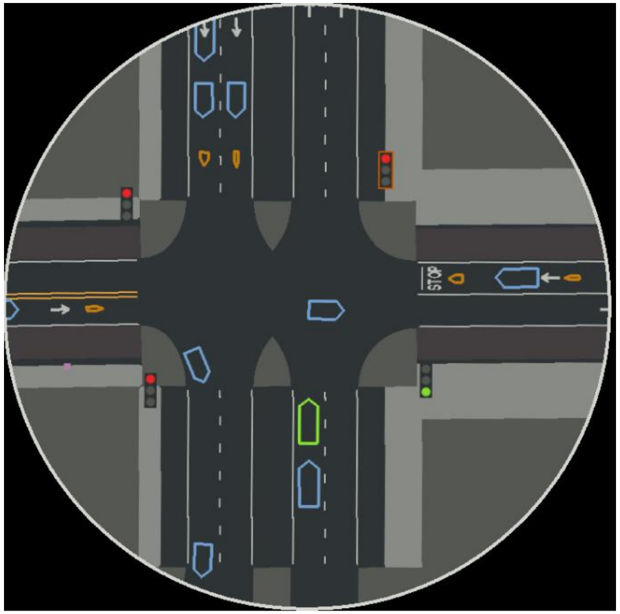
Learning to Play: The Multi-Agent Reinforcement Learning in MalmÖ (MARLÖ) Competition

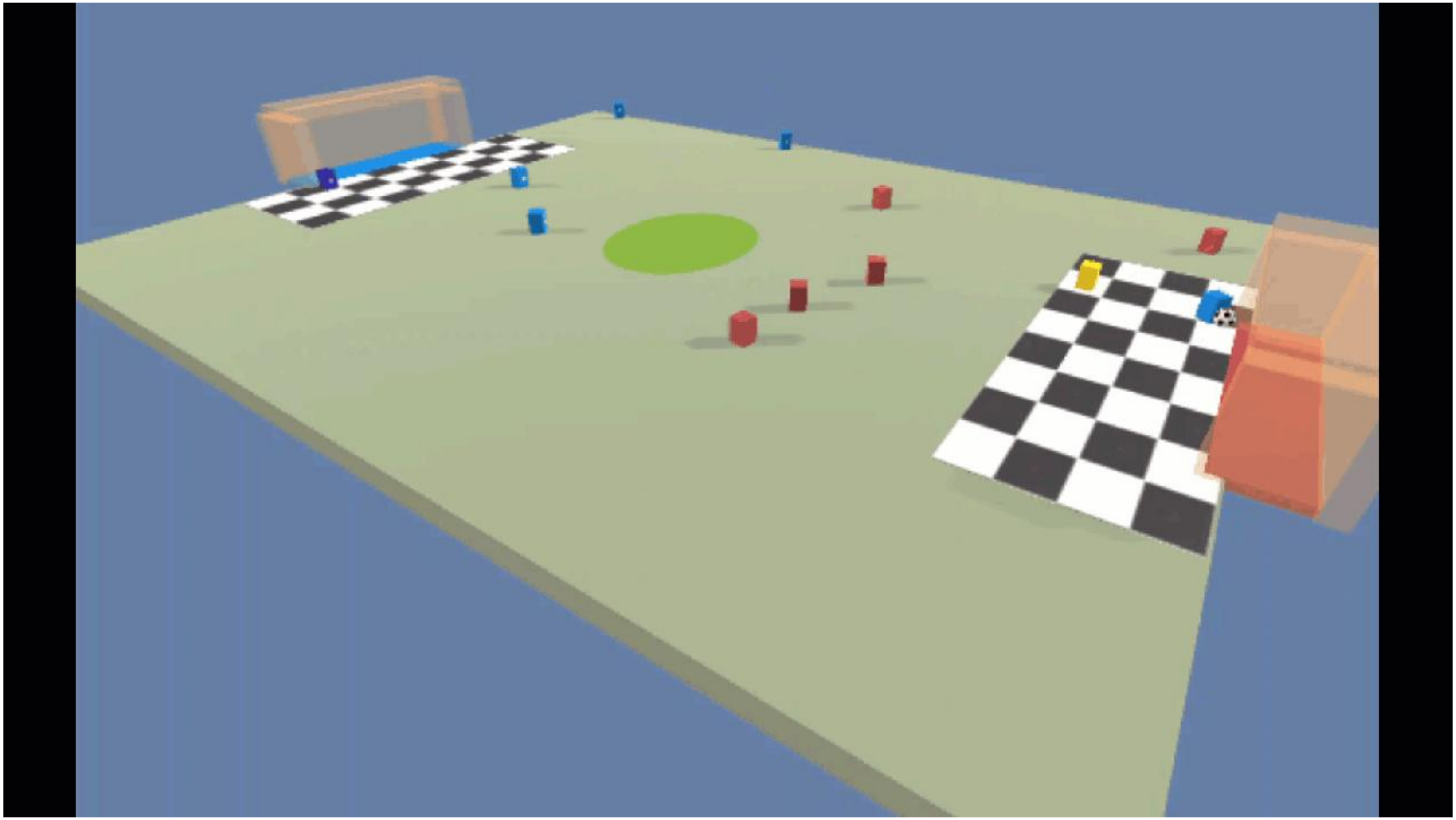
- Define the states
- Define the actions
- Define the reward
- Define the transitions

https://irobotics.aalto.fi/category/deep-multi-agent-reinforcement-learning-for-decision-making-in-autonomous-driving-systems/



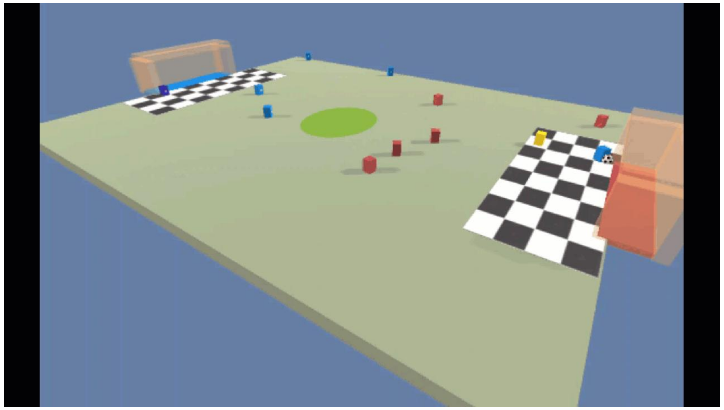
- Define the states
- Define the actions
- Define the reward
- Define the transitions



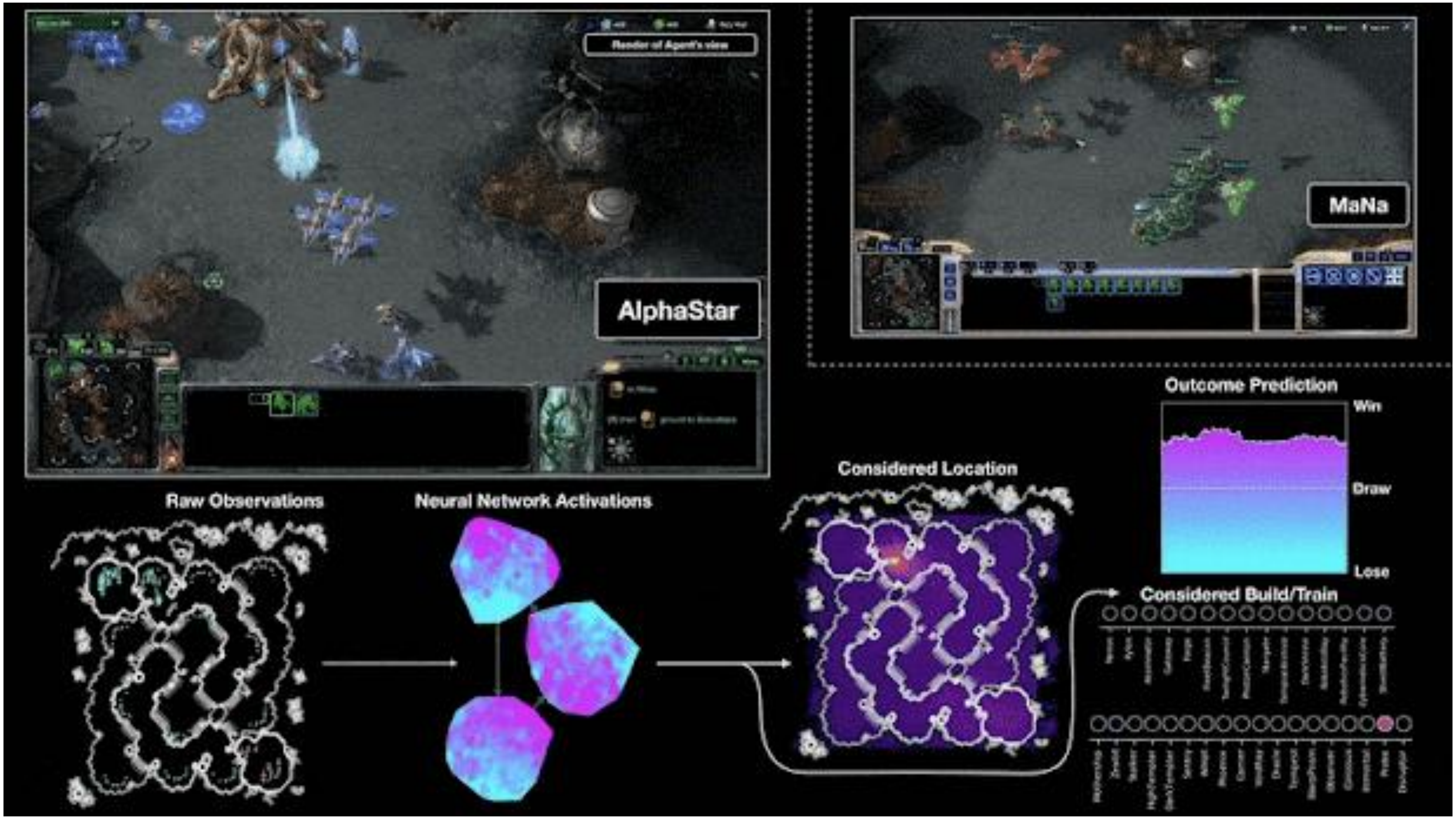


https://christinemcleavey.com/multi-agent-training-lessons-learned-from-training-3-separate-competing-networks/

- Define the states
- Define the actions
- Define the reward
- Define the transitions



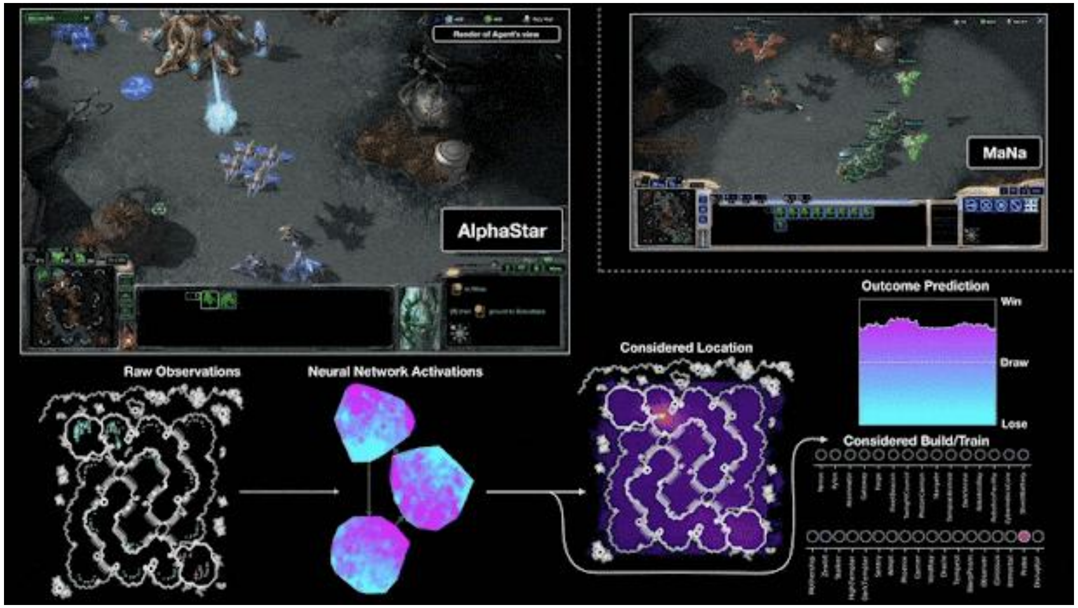
AlphaStar



https://deepmind.google/discover/blog/alphastar-grandmaster-level-in-starcraft-ii-using-multi-agent-reinforcement-learning/

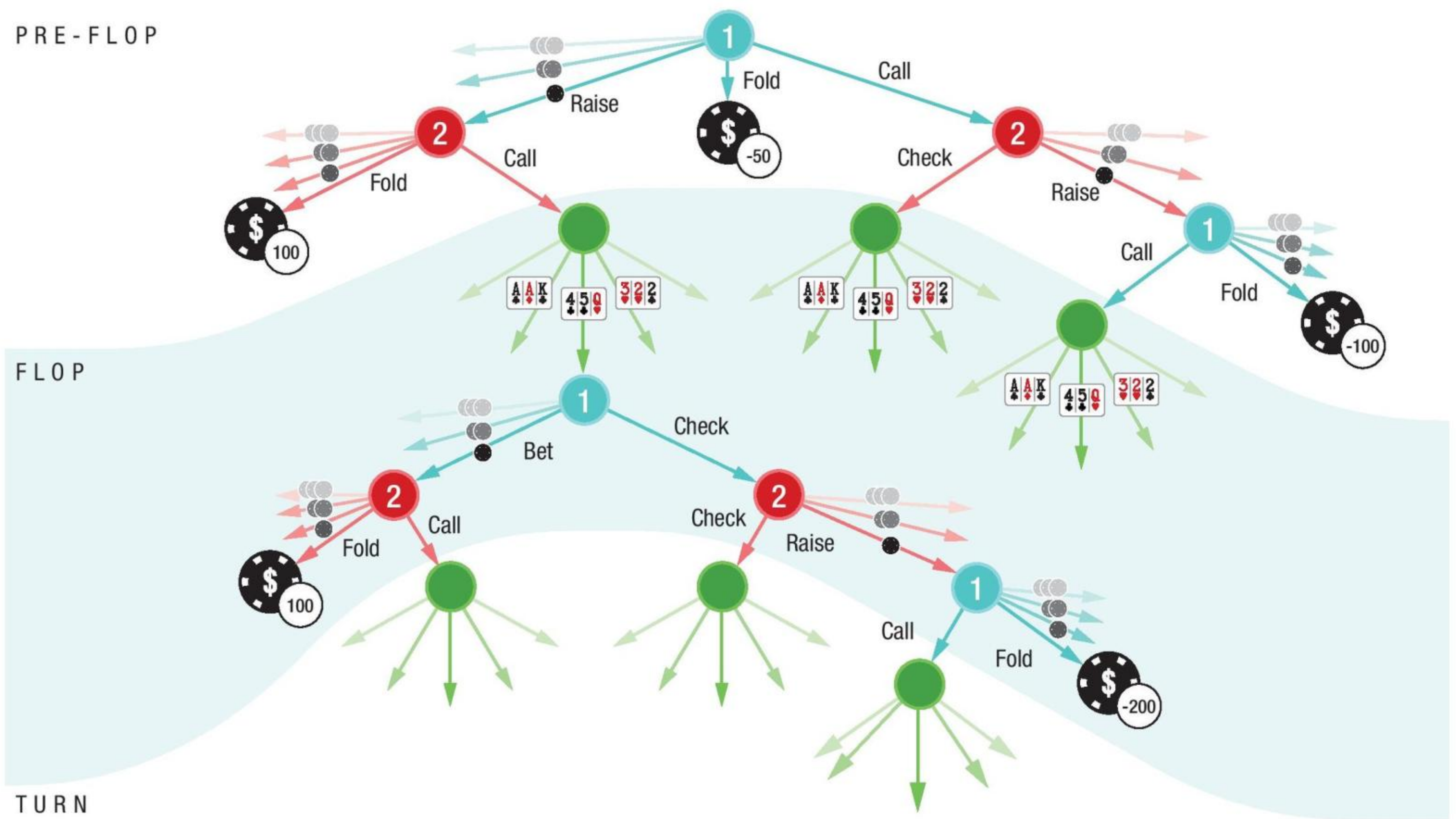
AlphaStar

- Define the states
- Define the actions
- Define the reward
- Define the transitions



https://deepmind.google/discover/blog/alphastar-grandmaster-level-in-starcraft-ii-using-multi-agent-reinforcement-learning/

https://www.deepstack.ai/



Outline

- **Stochastic games**
- Independent learning
- Learning in repeated games
- Multiagent learning
  - Joint-Action Learners
  - Agent-Modelling

Stochastic Games

- In repeated games, agents played the same stage game over and over again

Prisoner's Dilemma

	C	***D***
C	$3, 3$	$0, 5$
D	$5, 0$	$1, 1$

	C	***D***
C	$3, 3$	$0, 5$
D	$5, 0$	$1, 1$

	C	***D***
C	$3, 3$	$0, 5$
D	$5, 0$	$1, 1$

Stochastic Games

- What if agents **repeatedly played several normal-form games from a collection**

Stag Hunt

	A	B
X	3, 3	0, 2
Y	2, 0	1, 1

Prisoner's Dilemma

	A	B
X	3, 3	0, 5
Y	5, 0	1, 1

Battle of Sexes

	A	B
X	2, 1	0, 0
Y	0, 0	1, 2

And **the particular game played** at any given iteration **depends probabilistically on the previous game played and on the actions taken by all agents** in that game.

Stochastic Games

- **Definition (Stochastic game)**: *A stochastic game (also known as a Markov game) is a tuple* (Q, N, A, P, r), *where:*
  - Q *is a **finite set of games (states)**;*
  - N *is a **finite set of*** n ***agents**;*
  - $A = A_1 \times \cdots \times A_n$, *where* A_i *is a **finite set of actions** available to agent* i*;*
  - $P: Q \times A \times Q \longmapsto [0,1]$ *is **the transition probability function**;* $P(q, a, \hat{q})$ *is the probability of transitioning from state* q *to state* $\hat{q}$ *after action profile* a*; and*
  - $R = r_1, \dots, r_n$, *where* $r_i: Q \times A \longmapsto \mathbb{R}$ *is a **real-valued payoff function** for agent* i.

Stochastic Games

- A **stochastic game is generalization** of a:
  - **Repeated game**, where Q has **only one game** (one state)
  - **Markov decision process**, where N has **only one agent**

Learning in Markov Games

- Independent – cannot see others, other agents are considered part of the environment
- Centralized – even if there are several entities a central entity decides everything
- Joint-Action Learners – agents are aware of others, they will learn and decide what to do based on a given criteria, e.g. equilibria, worst-case, …
- Agent Modelling –agents are aware of others, they will learn and decide what to do based on a model of the others agents

Outline

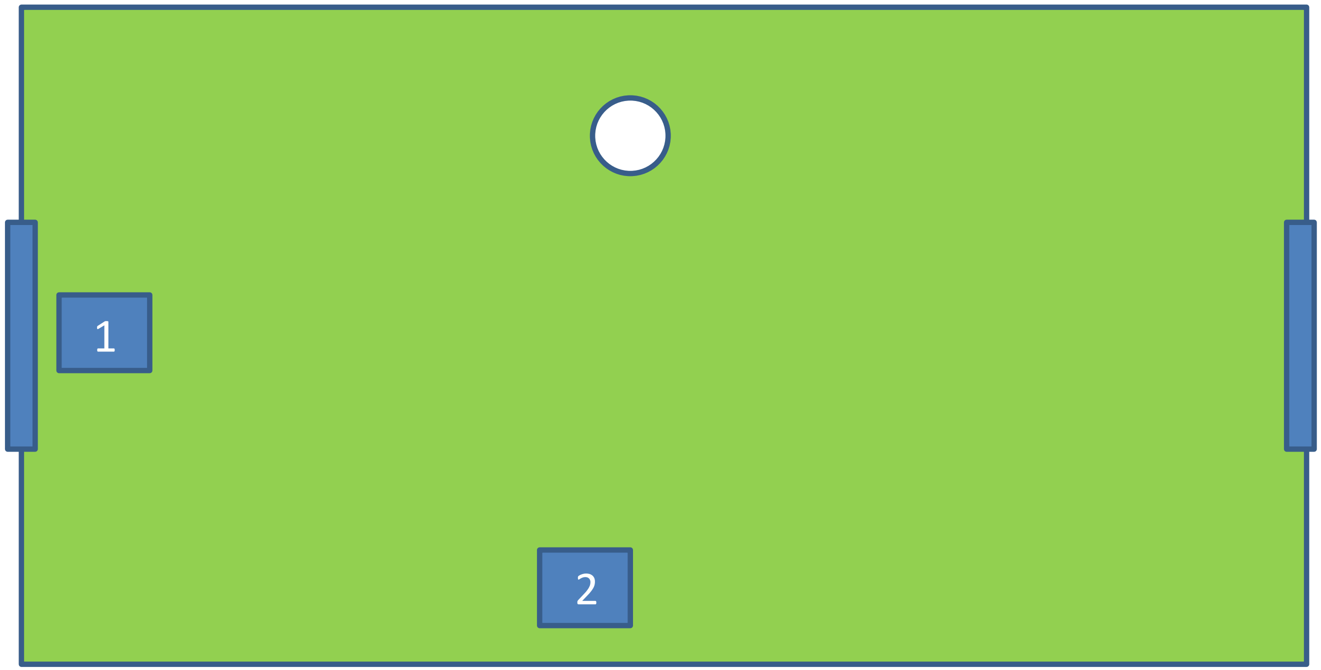
- Stochastic games
- **Independent learning**
- Learning in repeated games
- Multiagent learning
  - Joint-Action Learners
  - Agent-Modelling

Independent Learning

- In a Stochastic game, **each agent can learn separately**
  - Ignoring the presence of the other agents in the system
- More specifically, each agent can **treat the other agents as part of its environment**
  - Thus, not trying to model the other agents or predict their actions

Independent Learning

- For example, agent 1 and agent 2 can use Q-learning to learn their policies.
  - The state definition of agent 1 could include: position of agent 1, position of agent 2, position of the ball, etc.



Independent Learning

- However this approach is **inherently flawed**:
  - In an environment with multiple learning agents, the (hypothetical) **transition model** $p(s'|s, a_i)$ of agent i **may be changing continuously** due to the policy of the other agents (who are also learning)
  - the **convergence of Q-learning** relies on an underlying **transition model** that is **stationary**
    - stationary = does not change with time

Independent Learning

- Although independent **Q-learning cannot be justified theoretically**, the method has been **employed in practice with reported success**

  - Mataric. M. J. (1994). Reward Functions for Accelerated Learning. In *Proc. 11th Int. Conf. on Machine Learning*, San Francisco, CA.

  - Sen, S., Sekaran, M., and Hale, J. (1994). Learning to coordinate without sharing information. In *Proc. 12th Nation. Conf. on Artificial Intelligence*, Seattle, WA.

  - Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proc. 10th Int. Conf. on Machine Learning*, Amherst, MA.

Independent Learning

- **Claus, C. and Boutilier, C. (1998).** The dynamics of reinforcement learning in cooperative multiagent systems. In *Proc. 15th Nation. Conf. on Artificial Intelligence*, Madison, WI.
  - The authors examine **the conditions under which independent Q-learning leads to individual policies that form a Nash equilibrium** in a **single state coordination problem**
  - However, the **resulting equilibrium may not be Pareto optimal**

Independent Learners

- Initialize Q^i for each agent i
- Initialize current state s^i for each agent i
- Loop for each step:
  - Loop for each agent:
    - choose some action a^i (e.g., using ϵ-greedy)
  - Take joint action $a^1, a^{\cdots}, a^n$ and observe next state s'^i and reward r^i
  - Update Q estimate according to
    $$Q\left(s^i, a^i\right) \leftarrow Q\left(s^i, a^i\right) + \alpha[r + \gamma \max_b Q\left(s'^i, b\right) - Q(s^i, a^i)]$$
  - $s^i \leftarrow s'^i$

Examples

- Repeated game of Prisioner’s Dilemma

Prisoner 2

		Colaborate	***Defect***
Prisoner 1	***Colaborate***	-1, -1	-9, 0
	Defect	0, -9	-6, -6

- Repeated game of Matching Pennies

Agent 2

		Heads	***Tails***
Agent 1	***Head***	1, -1	-1, 1
	Tails	-1, 1	1, -1

Independent Learning Examples

- Q1=[0,0]
- Q2=[0,0]

- Play C, C

- Q1[C]=0+0.1*(-1-0)=-0.1
- Q2[C]=0+0.1*(-1-0)=-0.1

		Prisoner 2	
		Colaborate	***Defect***
Prisoner 1	***Colaborate***	-1, -1	-9, 0
	Defect	0, -9	-6, -6

Independent Learning Examples

- Q1=[-0.1,0]
- Q2=[-0.1,0]

- Play C, D

- Q1[C]=-0.1+0.1*(-9-(-0.1))=-0.99
- Q2[D]=0+0.1*(0-0)=0

Prisoner 2

Prisoner 1		***Colaborate***	***Defect***
	Colaborate	-1, -1	-9, 0
	Defect	0, -9	-6, -6

Independent Learning Examples

- Q1=[-0.99,0]
- Q2=[-0.1,0]

- Play D, D

- Q1[D]=-0.99+0.1*(-6-(-0.99))=-1,491
- Q2[D]=0+0.1*(-6-0)=-0.6

Prisoner 2

		Cooperate	Defect
Prisoner 1	Cooperate	-1, -1	-9, 0
	Defect	0, -9	-6, -6

Independent Learning Examples

- Q1=[-0.99, -1,491]
- Q2=[-0.1,-0.6]

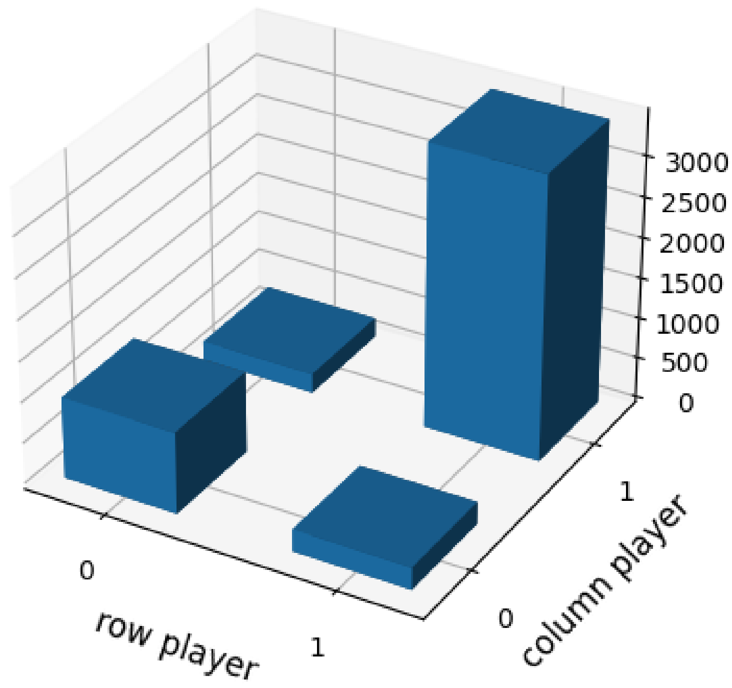
- Play D, D

- Q1[D]=-0.99+0.1*(-6-(-0.99))=-1,491
- Q2[D]=0+0.1*(-6-0)=-0.6

Prisoner 2

		Colaborate	***Defect***
Prisoner 1	***Colaborate***	-1, -1	-9, 0
	Defect	0, -9	-6, -6

Independent Learning Examples



Independent Learning Examples

- Q1=[0,0]
- Q2=[0,0]

- Play H, H

- Q1[H]=0+0.1*(1-0)=-.1
- Q2[H]=0+0.1*(-1-0)=-0.1

		Agent 2	
		Heads	***Tails***
Agent 1	***Head***	1, -1	-1, 1
	Tails	-1, 1	1, -1

Independent Learning Examples

- Q1=[.1,0]
- Q2=[-.1,0]

- Play H, T

- Q1[H]=0.1+0.1*(-1-0.1)=-0.01
- Q2[T]=0+0.1*(1-0)=0.1

Agent 2

		Heads	***Tails***
Agent 1	***Head***	1, -1	-1, 1
	Tails	-1, 1	1, -1

Independent Learning Examples

- Q1=[-0.01,0]
- Q2=[-.1,0.1]

- Play H, T

- Q1[H]=
- Q2[T]=

Agent 2

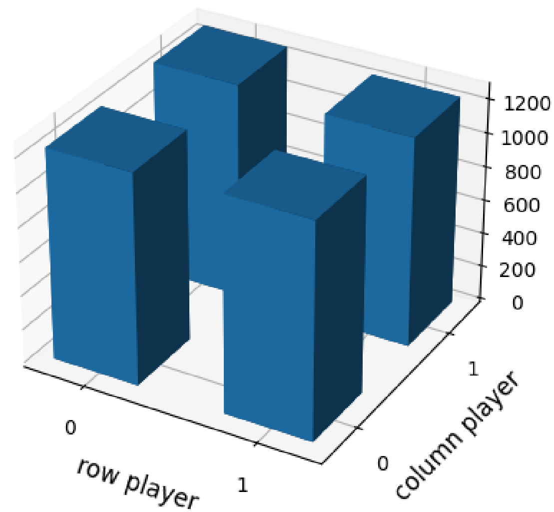
		Heads	Tails
Agent 1	Head	1, -1	-1, 1
	Tails	-1, 1	1, -1

Independent Learning Examples

- Q1=[-0.01,0]
- Q2=[-.1,0.1]

- Play H, T

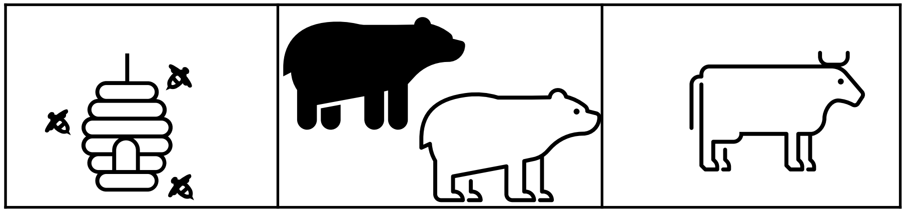
- Q1[H]=
- Q2[T]=



Agent 2

		Heads	***Tails***
Agent 1	***Head***	1, -1	-1, 1
	Tails	-1, 1	1, -1

Independent Learning in Markov Games

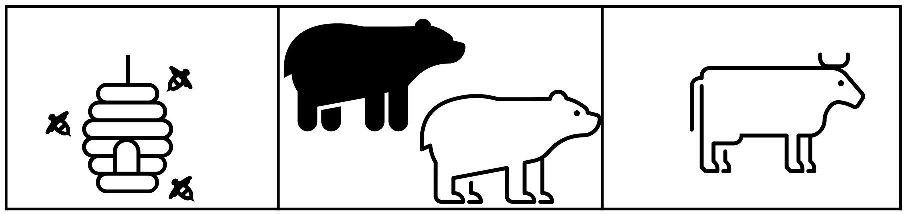


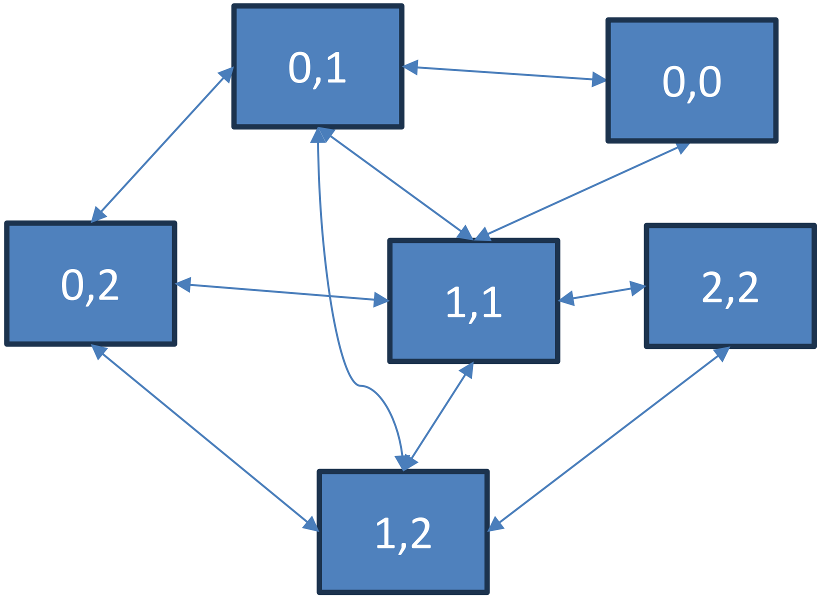
Imagine two bears that can get some honey from the bees or hunt a large animal. The honey they can get alone (+2 if alone, +1 if they go together). The large animal needs the collaboration of the two (they get 0 if going alone, +5 if they go together).

Define a Markov Game for this problem. Consider only two actions ←, C - capture, and →. They hunt/gather if they are in the target location and do the capture action.

The game restarts when one bear tries to capture one prey.

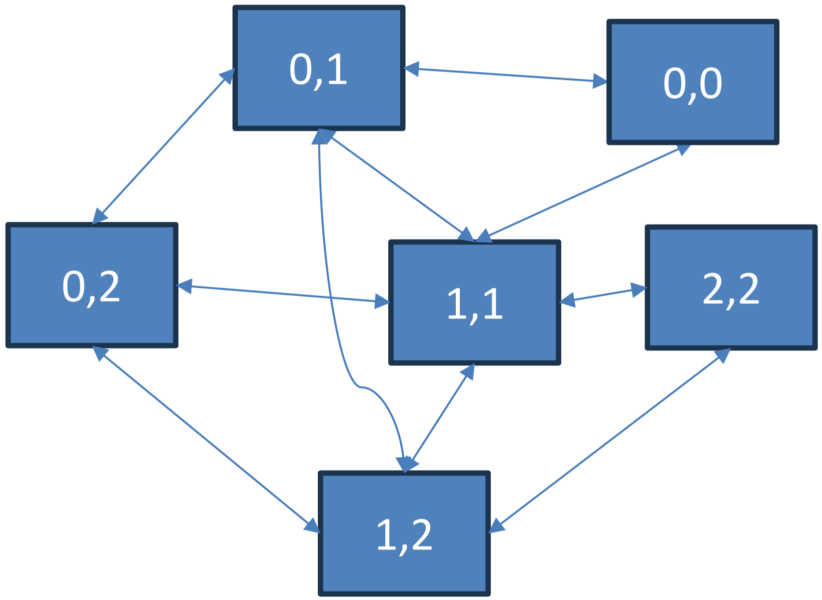
Independent Learning in Markov Games

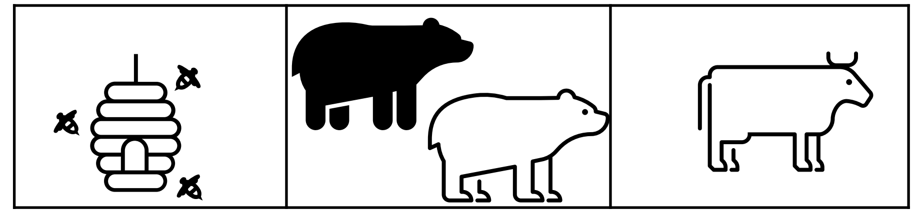




We are using the symmetry and ignore the repeated states, for instance, (1,2) and (2,1)

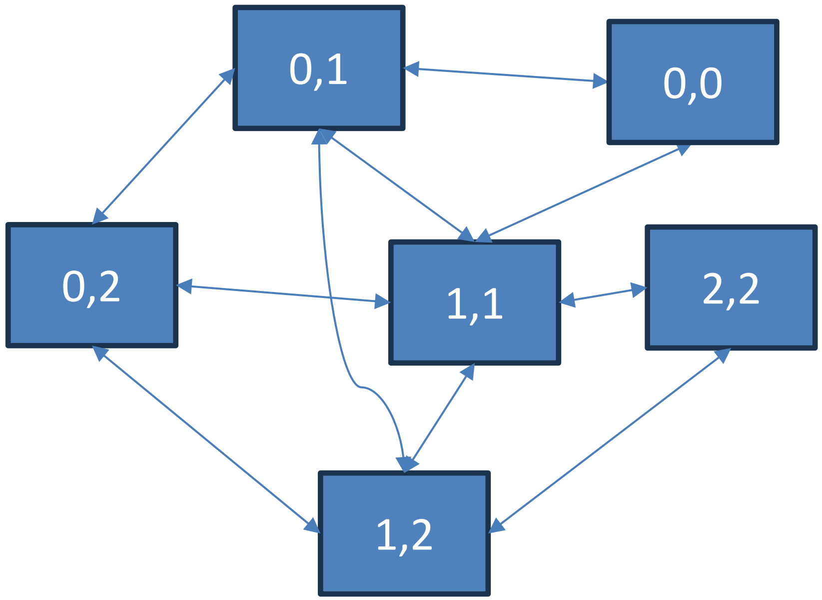
Independent Learning in Markov Games

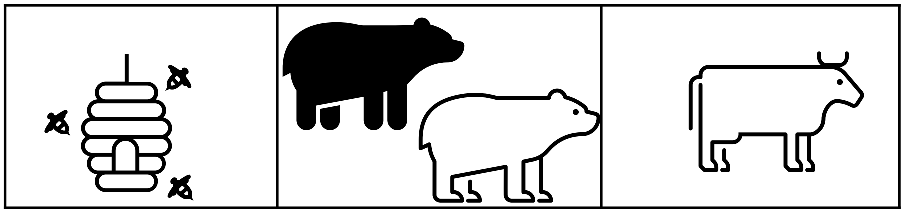




What will then learn if they always move to the left or always to the right?
Does it depend on what the others are doing?

Independent Learning in Markov Games

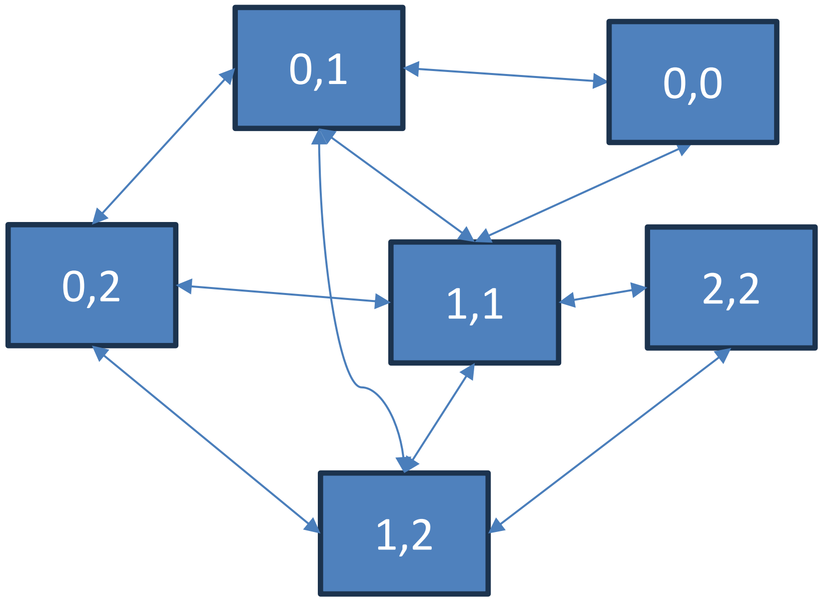


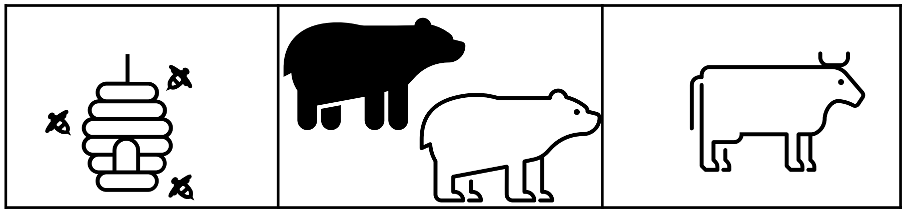


What will then learn if they always move to the left or always to the right?

If they both go left they will get the honey receiving +1
If they both go right they will capture the large prey both receiving +5
If one goes right and the other left, the one moving right gets 0 the one moving left gets +2

Independent Learning in Markov Games

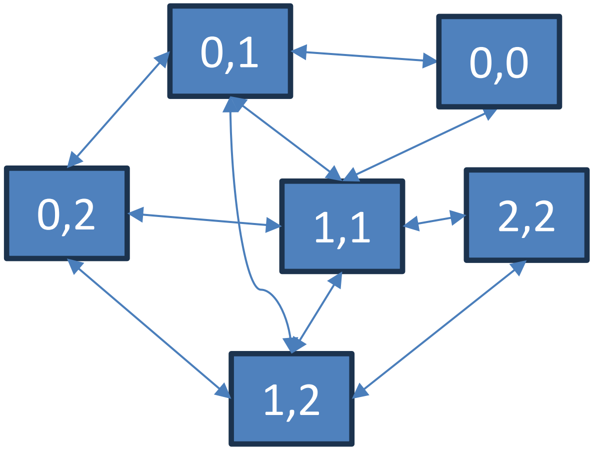


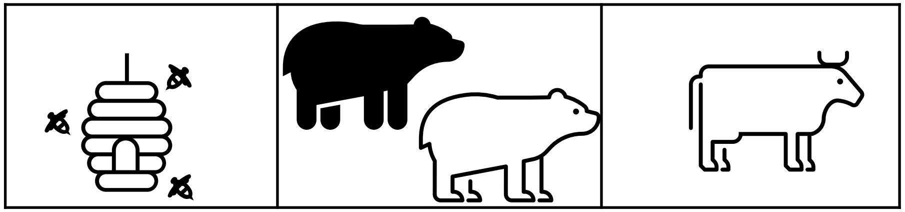


Does it depend on what the others are doing?

Yes, the values they receive include the impact of the actions of others. Let’s verify and compute the Q values.

Independent Learning in Markov Games





If white bear goes to the right then if the black bear always goes left it gets:

Q(1,$\leftarrow$) = 0+ γ max_b Q(0,b)

Q(0,C)=2

Q(1,$\leftarrow$) = 2 γ

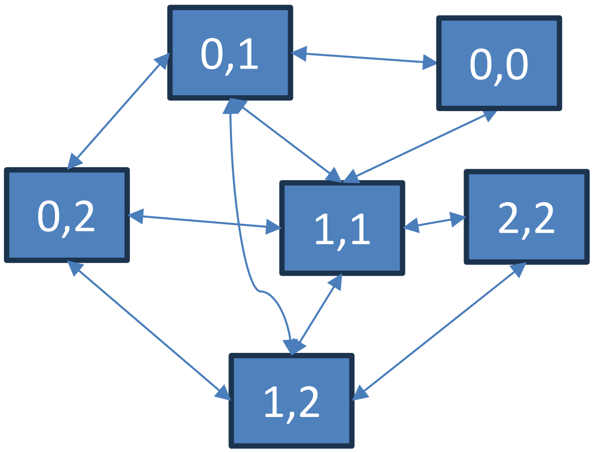
If both go to the left the black bear gets:

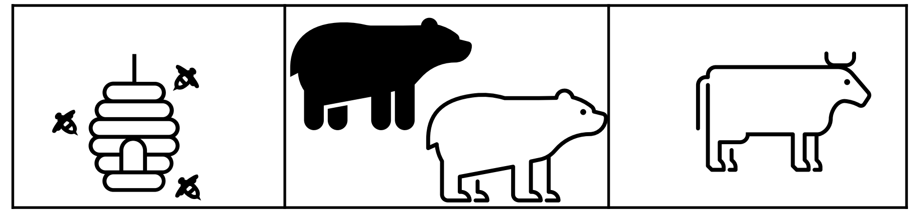
Q(1,$\leftarrow$) = 0+ γ max_b Q(0,b)

Q(0,C)=1

Q(1,$\leftarrow$) = γ

Independent Learning in Markov Games

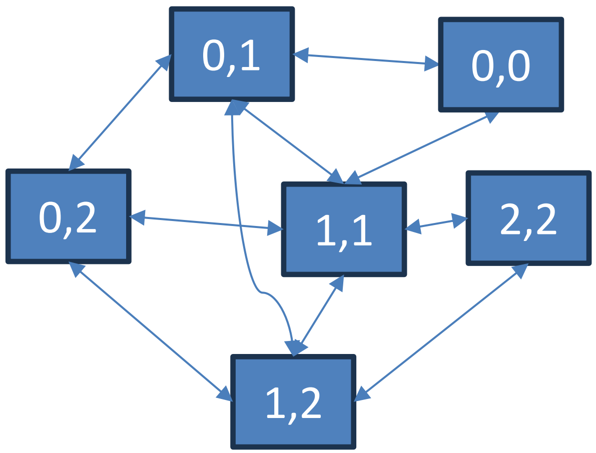


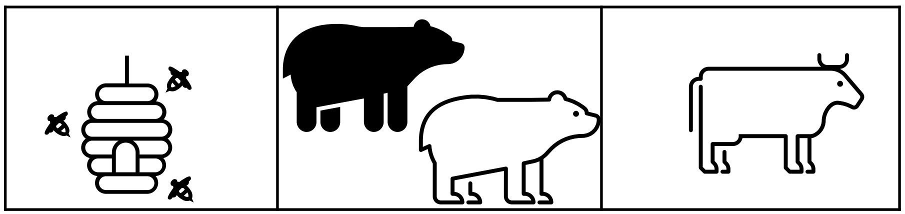


Imagine Black bear is in cell 2 and white bear is in cell 1. What should Black bear do?

Wait
Capture
Go to the honey?

Independent Learning in Markov Games





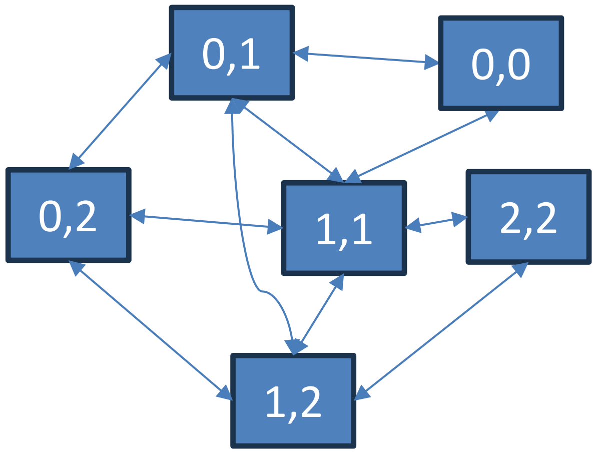
Imagine Black bear is in cell 2 and white bear is in cell 1. What should Black bear do?

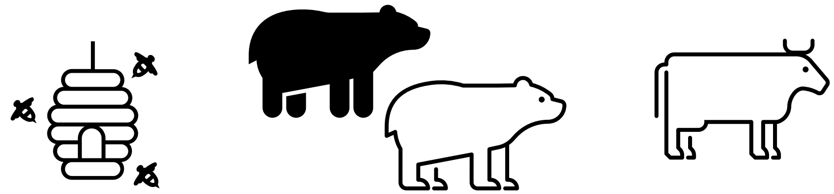
Wait
Capture
Go to the honey?

Black bear has no information about the location of White bear so unless they have practiced a lot this policy there is not enough information to decide.

→a better system would not be independent learning but distributed learning with full observation, i.e. the agents know the state of the other agents.

Independent Learning in Markov Games





With full observation

- Q((1,2),C) = ?
- Q((2,2),C) = ?

Independent learning

Q(1,C)=?

Q(2,C)=?

Learning in Markov Games

- a better system would not be independent learning but distributed learning with full observation, i.e. the agents know the state of the other agents.

- And if they see the actions of others?

Outline

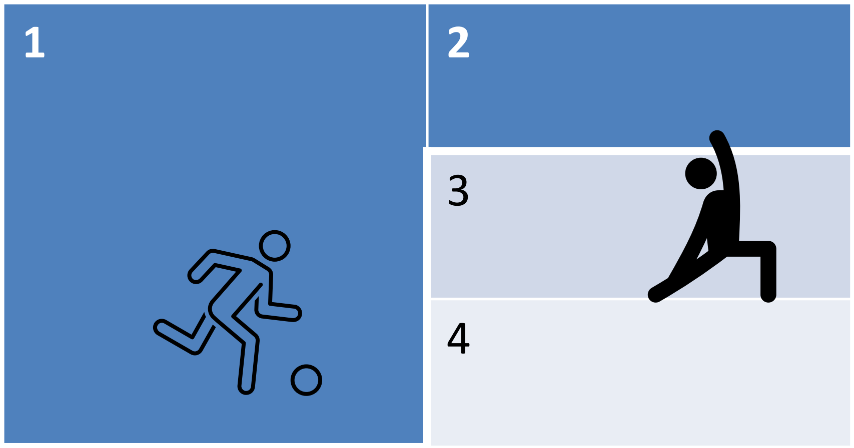
- Stochastic games
- Learning in repeated games
- Independent learning
- **Multiagent learning**
  - Joint-Action Learners
  - Agent-Modelling

Learning in Multi-Agent Systems

What should the criteria be?

- Assume the others are doing their best
- Are they considering my preferences
- Find a stable equilibria
- Improve the worst case scenario
- Ignore others
- Try to predict others

Goalkeeper



- Player 1 (on the left can move to the cells on the right with actions up, middle, bottom). Winning if Player 2 is not on the cell
- Player 2 (on the right, wants to block Player 1, moving up, down, or center)

Independent Learning

Agent 1
Q(1, ↑)
Q(1,→)
Q(1, ↓)

Agent 2
Q(2, ↑)
Q(2, →)
Q(2, ↓)
Q(3, ↑)
Q(3, →)
Q(3, ↓)
Q(4, ↑)
Q(4, →)
Q(4, ↓)

Agents do not observe the others

Agent 1	Agent 2
Q((1,2), ↑)	Q((1,2), ↑)
Q((1,2),→)	Q((1,2), →)
Q((1,2), ↓)	Q((1,2), ↓)
Q((1,3), ↑)	Q((1,3), ↑)
Q((1,3),→)	Q((1,3), →)
Q((1,3), ↓)	Q((1,3), ↓)
Q((1,4), ↑)	Q((1,4), ↑)
Q((1,4),→)	Q((1,4), →)
Q((1,4), ↓)	Q((1,4), ↓)

Agents observe the others as part of the environment, without assuming any intentionally by the other

Joint Action-Learning

Agent 1/Agent 2	Agent 1/Agent 2	Agent 1/Agent 2
Q((1,2), ↑ ↑)	Q((1,3), ↑ ↑)	Q((1,4), ↑ ↑)
Q((1,2),→ ↑)	Q((1,3),→ ↑)	Q((1,4),→ ↑)
Q((1,2), ↓ ↑)	Q((1,3), ↓ ↑)	Q((1,4), ↓ ↑)
Q((1,2), ↑ →)	Q((1,3), ↑ →)	Q((1,4), ↑ →)
Q((1,2),→ →)	Q((1,3),→ →)	Q((1,4),→ →)
Q((1,2), ↓ →)	Q((1,3), ↓ →)	Q((1,4), ↓ →)
Q((1,2), ↑ ↓)	Q((1,3), ↑ ↓)	Q((1,4), ↑ ↓)
Q((1,2),→ ↓)	Q((1,3),→ ↓)	Q((1,4),→ ↓)
Q((1,2), ↓ ↓)	Q((1,3), ↓ ↓)	Q((1,4), ↓ ↓)

Both agents have Q functions with the same shape, but each one computes them individually

Joint Action-Learning

Agent 1

Q((1,2), ↑ ↑) = -1

Q((1,2),→ ↑) = 1

Q((1,2), ↓ ↑) = 1

Q((1,2), ↑ →)=1

Q((1,2),→ →)=-1

Q((1,2), ↓ →)=1

Q((1,2), ↑ ↓)=1

Q((1,2),→ ↓)=1

Q((1,2), ↓ ↓)=-1

Agent 2

Q((1,2), ↑ ↑) = 1

Q((1,2),→ ↑) = -1

Q((1,2), ↓ ↑) = -1

Q((1,2), ↑ →)=-1

Q((1,2),→ →)=1

Q((1,2), ↓ →)=-1

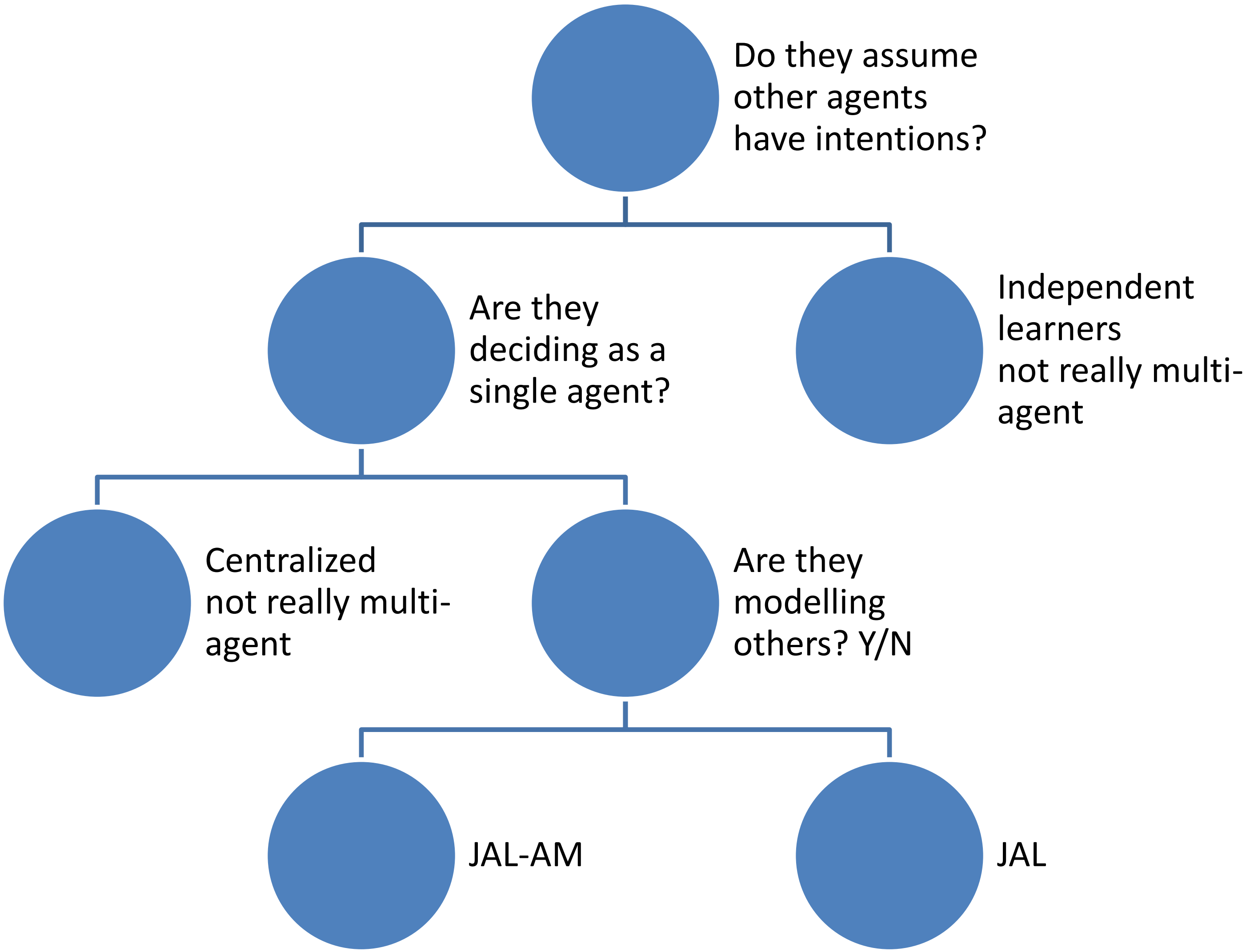
Q((1,2), ↑ ↓)=-1

Q((1,2),→ ↓)=-1

Q((1,2), ↓ ↓)=1

Both agents have Q functions with the same shape, but each one computes them individually

Learning in Markov Games



Joint Action Learners

- Initialize Q^i for each agent i

$$Q(s, a^1, a^{\cdots}, a^n)$$

- Initialize current state s
- Loop for each step:
  - Loop for each agent:
    - choose some action a^i (can we use ϵ-greedy over the Q values?)
  - Take joint action $a^1, a^{\cdots}, a^n$ and observe next state s'^i and reward r^i
  - Update Q estimate according to

$$Q(s^i, a^i) \leftarrow Q(s^i, a^i) + \alpha[r + \gamma \max_b Q(s'^i, ?) - Q(s^i, a^i)]$$

  - $s^i \leftarrow s'^i$

Joint Action Learners

- Initialize Q^i for each agent i

$$Q(s, a^1, a^{\cdots}, a^n)$$

- Initialize current state s
- Loop for each step:
  - Loop for each agent:
    - choose some action a^i (can we use ϵ-greedy over the Q values?)
  - Take joint action $a^1, a^{\cdots}, a^n$ and observe next state s'^i and reward r^i

    What value to use?

  - Update Q estimate according to

$$Q(s^i, a^i) \leftarrow Q(s^i, a^i) + \alpha[r + \gamma \max_b Q(s'^i, ?) - Q(s^i, a^i)]$$

  - $s^i \leftarrow s'^i$

Joint Action Learners

- can we use ϵ-greedy over the Q values

- No, the Q function is multidimensional $Q(s, a^1, a^{\cdots}, a^n)$ and so the best value depends on the actions of the others…

  - Update Q estimate according to
    $$Q\left(s^i, a^i\right) \leftarrow Q\left(s^i, a^i\right) + \alpha[r + \gamma \quad \max_b Q\left(s'^i, ?\right) - Q(s^i, a^i)]$$

  - Usually, the update is the estimate of the best value for next state, again, this depends on the actions of all other agents

Joint Action Learners

can we use ϵ-greedy over the Q values

- Now the Q function is multidimensional $Q(s, a^1, a^{\cdots}, a^n)$ and so the best value depends on the actions of the others…

Update Q estimate according to

$$Q\left(s^i, a^i\right) \leftarrow Q\left(s^i, a^i\right) + \alpha[r + \gamma \quad \max_b Q\left(s'^i, ?\right) - Q(s^i, a^i)]$$

- Usually, the update is the estimate of the best value for next state, again, this depends on the actions of all other agents

→ For both processes we need to make some assumptions about the other agents

Learning in Multi-Agent Systems

Joint Action Learners

Assume the others are doing their best
Find a stable equilibria
Improve the worst case scenario

Joint Action Learners with Agent Modelling

Are they considering my preferences
Ignore others
Try to predict others

Learning in Multi-Agent Systems

Joint Action Learners

- Assume the others are doing their best
- Improve the worst case scenario

→minmaxQ

- Find a stable equilibria

→nashQ

Joint Action Learners with Agent Modelling

- Are they considering my preferences
- Try to predict others

→JAL-AM (extension of ficticious play for Markov Games)

Joint Action Learners

- A standard approach is to have each agent i maintain:
  - An **individual value function** $V_i(s)$
  - An **individual Q-function** $Q_i(s, a)$
    - Where s is the state and a is the joint actions

Joint Action Learners

- With the individual value function and individual Q-function, the **standard value iteration generalizes to Stochastic games** as follows:

$$V_i(s) \coloneqq C\big(Q_1(s,a), \dots, Q_n(s,a)\big), \forall s$$

$$Q_i(s,a) \coloneqq R_i(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_i(s'),\ \forall s, \forall a$$

- Where ***C*** **is a function that applies some solution concept** to the strategic game formed by $Q_1(s,a), \dots, Q_n(s,a)$

Joint Action Learners

- When **the transition model is not available**, a corresponding coupled update scheme can be derived in which

$$Q_i(s,a) := R_i(s,a) + \gamma \sum_{s' \epsilon S} P(s'|s,a) V_i(s'),\ \forall s, \forall a$$

- is replaced by a Q-learning update, one per agent.

Joint Actions Learners

Where C **is a function that applies some solution concept** to the strategic game formed by $Q_1(s,a), \ldots, Q_n(s,a)$

What solutions concepts can we have?

-assume worst case;

-assume nash-equilibria;

-...

Nash Q-learning

Definition 5 *Agent i's* Nash Q-function *is defined over $(s, a^1, \ldots, a^n)$, as the sum of Agent i's current reward plus its future rewards when all agents follow a joint Nash equilibrium strategy. That is,*

$$Q_*^i(s, a^1, \ldots, a^n) = \\ r^i(s, a^1, \ldots, a^n) + \beta \sum_{s' \in S} p(s'|s, a^1, \ldots, a^n) v^i(s', \pi_*^1, \ldots, \pi_*^n), \tag{5}$$

where $(\pi_*^1, \ldots, \pi_*^n)$ is the joint Nash equilibrium strategy, $r^i(s, a^1, \ldots, a^n)$ is agent i's one-period reward in state s and under joint action $(a^1, \ldots, a^n)$, $v^i(s', \pi_*^1, \ldots, \pi_*^n)$ is agent i's total discounted reward over infinite periods starting from state s' given that agents follow the equilibrium strategies.

Nash Q-learning

Nash Q-learning algorithm

Initialize:

Let $t = 0$, get the initial state s_0.

Let the learning agent be indexed by i.

For all $s \in S$ and $a^j \in A^j$, $j = 1, \dots, n$, let $Q_t^j(s, a^1, \dots, a^n) = 0$.

Loop

Choose action a_t^i.

Observe $r_t^1, \dots, r_t^n; a_t^1, \dots, a_t^n$, and $s_{t+1} = s'$

Update Q_t^j for $j = 1, \dots, n$

$$Q_{t+1}^j(s, a^1, \dots, a^n) = (1 - \alpha_t) Q_t^j(s, a^1, \dots, a^n) + \alpha_t [r_t^j + \beta NashQ_t^j(s')]$$

where $\alpha_t \in (0, 1)$ is the learning rate, and $NashQ_t^k(s')$ is defined in (7)

Let $t := t + 1$.

Hu, J. and Wellman, M. P. (2004). Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069.

Nash Q-learning

Q1=[0,0;
0,0]

Q2=[0,0;
0,0]

	Colaborate	Defect
Colaborate	0,0	0,0
Defect	0,0	0,0

- Play C, C

- Q1[C,C]=0+0.1(-1-0)=-0.1
- Q2[C,C]=0+0.1(-1-0)=-0.1

Prisoner 2

Prisoner 1		Colaborate	Defect
	Colaborate	-1, -1	-9, 0
	Defect	0, -9	-6, -6

Nash Q-learning

Q1=[-0.1,0;
0,0]
Q2=[-0.1,0;
0,0]

	Colaborate	Defect
Colaborate	-0.1,-0.1	0,0
Defect	0,0	0,0

- Play D, D

- Q1[D,D]=0+0.1(-6)=-0.6
- Q2[D,D]=0+0.1(-6)=-0.6

Prisoner 2

Prisoner 1		Colaborate	Defect
	Colaborate	-1, -1	-9, 0
	Defect	0, -9	-6, -6

Nash Q-learning

What is the next equilibria?

Q1=[-0.1,0;
0,-0.6]

Q2=[-0.1,0;
0,-0.6]

	Colaborate	*Defect*
Colaborate	-0.1,-0.1	0,0
Defect	0,0	-0.6,-0.6

- Play D, D

- Q1[D,D]=0+0.1(-6)=-0.6
- Q2[D,D]=0+0.1(-6)=-0.6

Prisoner 2

Prisoner 1		*Colaborate*	*Defect*
	Colaborate	-1, -1	-9, 0
	Defect	0, -9	-6, -6

Nash Q-learning

Q1=[-0.1,0;
0,-0.6]

Q2=[-0.1,0;
0,-0.6]

	Colaborate	***Defect***
Colaborate	-0.1,-0.1	0,0
Defect	0,0	-0.6,-0.6

- Play C, D

- Q1[C,D]=0+0.1(-9)=-0.9
- Q2[C,D]=0+0.1(0)=0

Prisoner 2

Prisoner 1		***Colaborate***	***Defect***
	Colaborate	-1, -1	-9, 0
	Defect	0, -9	-6, -6

Nash Q-learning

Q1=[-0.1,-0.9;
0,-0.6]

Q2=[-0.1,0;
0,-0.6]

	Colaborate	*Defect*
Colaborate	-0.1,-0.1	-0.9,0
Defect	0,0	-0.6,-0.6

- Play D, C

- Q1[D,C]=0+0.1(0)=0
- Q2[D,C]=0+0.1(-9)=-0.9

Prisoner 2

Prisoner 1		*Colaborate*	*Defect*
	Colaborate	-1, -1	-9, 0
	Defect	0, -9	-6, -6

Nash Q-learning

Did it converge?

Q1=[-0.1,-0.9;
0,-0.6]

Q2=[-0.1,0;
-0.9,-0.6]

	Colaborate	***Defect***
Colaborate	-0.1,-0.1	-0.9,0
Defect	0,-0.9	-0.6,-0.6

Prisoner 2

Prisoner 1		***Colaborate***	***Defect***
	Colaborate	-1, -1	-9, 0
	Defect	0, -9	-6, -6

minimaxQ

```
Initialize:
For all s in S, a in A, and o in O,
  Let Q[s,a,o] := 1
For all s in S,
  Let V[s] := 1
For all s in S, a in A,
  Let pi[s,a] := 1/|A|
Let alpha := 1.0

Choose an action:
With probability explor, return an action uniformly at random.
Otherwise, if current state is s,
  Return action a with probability pi[s,a].

Learn:
After receiving reward rew for moving from state s to s'
      via action a and opponent's action o,
  Let Q[s,a,o] := (1-alpha) * Q[s,a,o] + alpha * (rew + gamma * V[s'])
  Use linear programming to find pi[s,.] such that:
   pi[s,.] := argmax{pi'[s,.], min{o', sum{a', pi[s,a'] * Q[s,a',o']}}}
  Let V[s] := min{o', sum{a', pi[s,a'] * Q[s,a',o']}}
  Let alpha := alpha * decay
```

Figure 1: The minimax-Q algorithm.

Only for zero-sum games

Coupled Learning

Correlated-Q Learning

MULTIQ(MarkovGame, γ, α, S, T)

Inputs: discount factor γ
learning rate α
decay schedule S
total training time T

Output: action-value functions Q_i^*

Initialize: $s, a_1, \ldots, a_n$ and $Q_1, \ldots, Q_n$

for $t = 1$ **to** T

1. simulate actions $a_1, \ldots, a_n$ in state s
2. observe rewards $r_1, \ldots, r_n$ and next state s'
2. **for** $i = 1$ **to** n
   (a) compute $V_i(s')$
   (b) update $Q_i(s, a_1, \ldots, a_n)$
   i. $Q_i(s, a_1, \ldots, a_n) =$
   $(1 - \alpha)Q_i(s, a_1, \ldots, a_n) + \alpha[r_i + \gamma V_i(s')]$
4. agents choose actions $a_1', \ldots, a_n'$
5. $s = s'$, $a_1 = a_1'$, $\ldots$, $a_n = a_n'$
6. decay α according to S

A **correlated equilibrium** is a generalization of a mixed-strategy Nash equilibrium where the mixed strategies of the agents can be correlated

Greenwald, A. and Hall, K. (2003). Correlated-Q learning.
In *Proc. 20th Int. Conf. on Machine Learning*, Washington, DC, USA

Coupled Learning

- The algorithms require that **a learning agent must know**:
  - The actions of other agents
  - The rewards of the other agents
- **Complexity**:
  - The learning agent has to maintain n Q-tables, where each table has the size $|S| \times |A|^n$ (i.e., exponential in the number of agents)
  - The running time of the function that applies the solution concept can also be exponential

Outline

- Stochastic games
- Learning in repeated games
- Independent learning
- **Multiagent learning**
  - Joint-Action Learners
  - Agent-Modelling

Fictitious Play

- Fictitious play is an **instance of model-based** learning
  - The learning agent **explicitly maintains beliefs about the opponent's actions**
- In fictitious play, the **learning agent is oblivious to the payoffs obtained by other agents**

Learning in Repeated Games

- How can an agent **learn to play a repeated game**?
  - A famous learning algorithm is called **Fictitious play**
    - It was actually not proposed initially as a learning model
    - Originally, it was used as an iterative method for computing Nash equilibria in zero-sum games

Fictitious Play

Fictitious play algorithm:

Initialize beliefs about the opponent’s action

repeat

- Play a best response to the assessed action of the opponent
- Observe the opponent’s actual play and update beliefs accordingly

Fictitious Play

- In Fictitious play, we assume:
  - the **agent does not know the payoffs of the other agents**
  - the **agent only needs to know his own payoff matrix**
    - i.e., the payoff he would get in each joint action profile, whether or not encountered in the past
- Also, in fictitious play, **the learning agent believes that his opponent is playing the mixed strategy** given by the empirical distribution of the opponent's previous actions

Fictitious Play

- Let:
  - A be the set of the opponent's actions
  - $w(a)$ be the number of times that the opponent has played action a, for every $a \in A$

- Hence, the **learning agent assesses the probability of a in the opponent's mixed strategy** as:

$$P(a) = \frac{w(a)}{\sum_{a' \in A} w(a')}$$

Fictitious Play

- For example, in a **repeated Prisoner's Dilemma**, if the opponent has selected the following actions in the first five games:
  - *Confess, Confess, Not Confess, Confess, Not Confess*

- Before the sixth game, the learning agent has $w(Confess) = 3$ and $w(Not\ Confess) = 2$ and thus
  - The agent thus assumes that the opponent is playing the mixed strategy $(0.6, 0.4)$

		Prisoner 2	
		Not confess	***Confess***
Prisoner 1	***Not confess***	-1, -1	-9, 0
	Confess	0, -9	-6, -6

Fictitious Play

- Note that **we can represent an opponent's beliefs** with either a **probability measure** or with the **set of counts** $(w(a_1), \ldots, w(a_k))$

- Note that **different versions of fictitious play exist** depending on the **tie-breaking method** used to select an action when **there is more than one best response**

  - In general, the **tie-breaking rule chosen has little effect on the results of fictitious play**

Fictitious Play

- However, fictitious play is **very sensitive to the players' initial beliefs**
  - The initial beliefs can be interpreted as action counts that were observed before the start of the game
  - These **initial beliefs can have a radical impact on the learning process**
- Note that **one must pick some nonempty prior belief** for each agent
  - the prior beliefs cannot be (0, . . . , 0) since this does not define a meaningful mixed strategy

Fictitious Play

- **Example**:
  - Repeated game of Matching Pennies

Agent 2

Agent 1		*Heads*	*Tails*
	Head	1, -1	-1, 1
	Tails	-1, 1	1, -1

Fictitious Play

- Repeated game of Matching Pennies:

	Heads	*Tails*
Head	1, -1	-1, 1
Tails	-1, 1	1, -1

Round	1's action	2's action	1's beliefs about 2	2's beliefs about 1
0			(1.5,2)	(2,1.5)

Fictitious Play

- Repeated game of Matching Pennies:

	Heads	*Tails*
Head	1, -1	-1, 1
Tails	-1, 1	1, -1

Round	1's action	2's action	1's beliefs about 2	2's beliefs about 1
0			(1.5,2)	(2,1.5)
1	T	T	(1.5,**3**)	(2,**2.5**)

Update beliefs

Fictitious Play

- Repeated game of Matching Pennies:

	Heads	***Tails***
Head	1, -1	-1, 1
Tails	-1, 1	1, -1

Round	1's action	2's action	1's beliefs about 2	2's beliefs about 1
0			(1.5,2)	(2,1.5)
1	T	T	(1.5,3)	(2,2.5)
2	T	H	(**2.5**,3)	(2,**3.5**)

Update beliefs

Fictitious Play

- Repeated game of Matching Pennies:

	Heads	*Tails*
Head	1, -1	-1, 1
Tails	-1, 1	1, -1

Round	1's action	2's action	1's beliefs about 2	2's beliefs about 1
0			(1.5,2)	(2,1.5)
1	T	T	(1.5,3)	(2,2.5)
2	T	H	(2.5,3)	(2,3.5)
3	T	H	(**3.5**,3)	(2,**4.5**)

Update beliefs

Fictitious Play

- Repeated game of Matching Pennies:

	Heads	***Tails***
Head	1, -1	-1, 1
Tails	-1, 1	1, -1

Round	1's action	2's action	1's beliefs about 2	2's beliefs about 1
0			(1.5,2)	(2,1.5)
1	T	T	(1.5,3)	(2,2.5)
2	T	H	(2.5,3)	(2,3.5)
3	T	H	(3.5,3)	(2,4.5)
4	H	H	(**4.5**,3)	(**3**,4.5)

Update beliefs

Fictitious Play

- Repeated game of Matching Pennies:

	Heads	Tails
Head	1, -1	-1, 1
Tails	-1, 1	1, -1

Round	1's action	2's action	1's beliefs about 2	2's beliefs about 1
0			(1.5,2)	(2,1.5)
1	T	T	(1.5,3)	(2,2.5)
2	T	H	(2.5,3)	(2,3.5)
3	T	H	(3.5,3)	(2,4.5)
4	H	H	(4.5,3)	(3,4.5)
5	H	H	(**5.5**,3)	(**4**,4.5)

Update beliefs

Fictitious Play

- Repeated game of Matching Pennies:

	Heads	*Tails*
Head	1, -1	-1, 1
Tails	-1, 1	1, -1

Round	1's action	2's action	1's beliefs about 2	2's beliefs about 1
0			(1.5,2)	(2,1.5)
1	T	T	(1.5,3)	(2,2.5)
2	T	H	(2.5,3)	(2,3.5)
3	T	H	(3.5,3)	(2,4.5)
4	H	H	(4.5,3)	(3,4.5)
5	H	H	(5.5,3)	(4,4.5)
6	H	H	(**6.5**,3)	(**5**,4.5)

Update beliefs

Fictitious Play

- Repeated game of Matching Pennies:

	Heads	*Tails*
Head	1, -1	-1, 1
Tails	-1, 1	1, -1

Round	1's action	2's action	1's beliefs about 2	2's beliefs about 1
0			(1.5,2)	(2,1.5)
1	T	T	(1.5,3)	(2,2.5)
2	T	H	(2.5,3)	(2,3.5)
3	T	H	(3.5,3)	(2,4.5)
4	H	H	(4.5,3)	(3,4.5)
5	H	H	(5.5,3)	(4,4.5)
6	H	H	(6.5,3)	(5,4.5)
7	H	T	(6.5,**4**)	(**6**,4.5)
…	…	…	…	…

Update beliefs

Fictitious Play

- As we can see, each player ends up alternating back and forth between playing heads and tails

- In fact, as the number of rounds tends to infinity, the empirical distribution of each player will converge to $(0.5, 0.5)$

- If we take this distribution to be the mixed strategy of each player
  - **the play converges to the unique Nash equilibrium** of the normal form game
  - In the repeated Matching Pennies, each player plays the mixed strategy (0.5, 0.5)

Joint Action Learning with Agent Modelling (JAL-AM)

How to expand ficticious play for stochastic games?

$$\hat{\pi}_j(a_j \mid s) = \frac{C(s, a_j)}{\sum_{a'_j} C(s, a'_j)}.$$

Is it good to use this equation? If the policy changes how fast it will converge?

$$AV_i(s, a_i) = \sum_{a_{-i} \in A_{-i}} Q_i(s, \langle a_i, a_{-i} \rangle) \prod_{j \neq i} \hat{\pi}_j(a_j \mid s)$$

Joint Action Learning with Agent Modelling (JAL-AM)

Algorithm 8 Joint Action Learning with Agent Modelling (JAL-AM)

// Algorithm controls agent i

Initialise:
    $Q_i(s, a) = 0$ for all $s \in S, a \in A$
    Agent models $\hat{\pi}_j(a_j \mid s) = \frac{1}{|A_j|}$ for all $j \neq i, a_j \in A_j, s \in S$
Repeat for every episode:
for $t = 0, 1, 2, \ldots$ **do**
    Observe current state s^t
    With probability ϵ: choose random action a_i^t
    Else: choose best-response action $a_i^t \in \arg\max_{a_i} AV_i(s^t, a_i)$
    Observe joint action $a^t = (a_1^t, \ldots, a_n^t)$, reward r_i^t, next state s^{t+1}
    for all $j \neq i$ **do**
        Update agent model $\hat{\pi}_j$ with new observations (e.g. (s^t, a_j^t))
    $Q_i(s^t, a^t) \leftarrow Q_i(s^t, a^t) + \alpha \left[r_i^t + \gamma \max_{a_i'} AV_i(s^{t+1}, a_i') - Q_i(s^t, a^t) \right]$

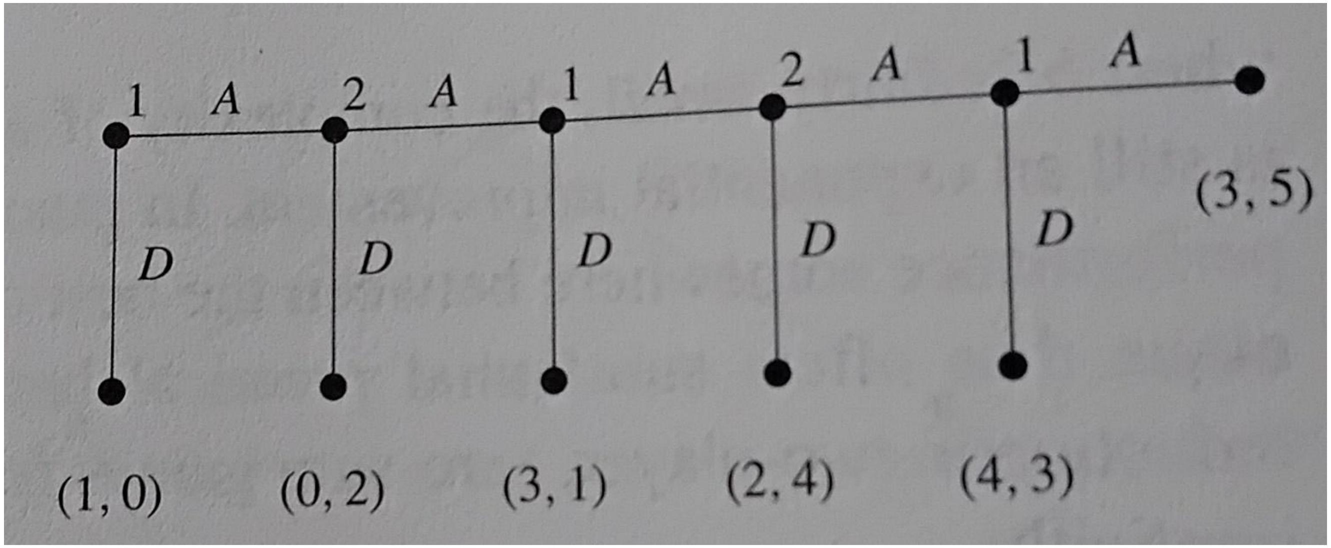
https://www.marl-book.com/

Are Equilibria Good Predictor of Human Behavior?

- In prisioner's dilema do people always end in the Nash equilibria?
- What is your expectation?

Are Equilibria Good Predictor of Human Behavior?

- What is the equilibria in the following game?



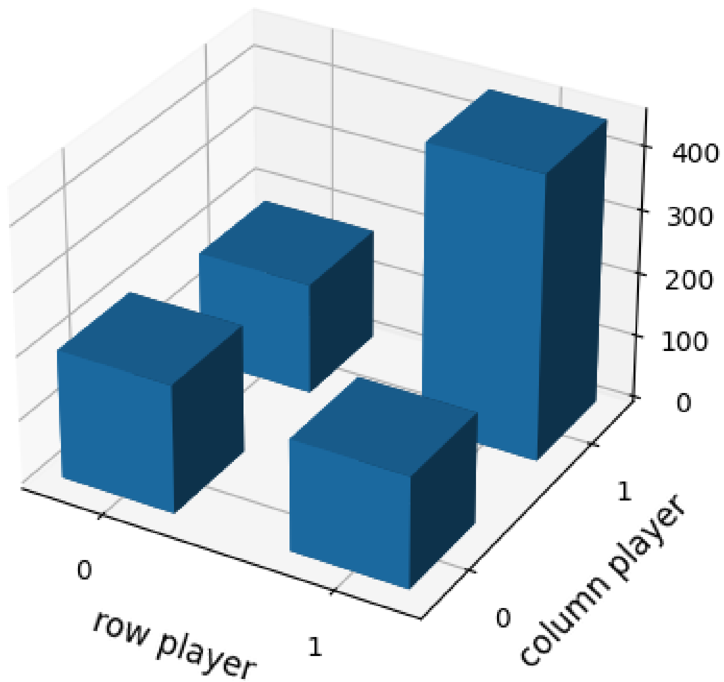
- What would you expect to happen with 2 people?

Are Equilibria Good Predictor of Human Behavior?

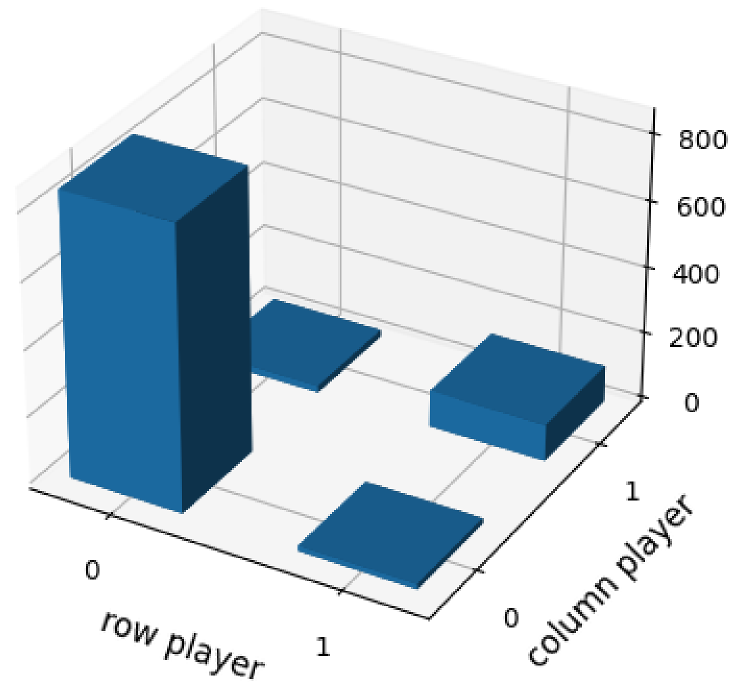
- Consider two strategies in PD
  - Nash-Equilibria
  - Tit-for-Tat – Defect after defect
  - Tit-for-2Tats – Defect after 2 defects
- What would you expect to happen with 2 people?

Are Equilibria Good Predictor of Human Behavior?

Tit-for-TaT



Tit-for-2Tats



Behavior Profiles

- The behavior of people tend to be more pro-social than what pure utility based approaches predict.
- Over many interactions we might start to expect specific behaviors of people.
- Ficticious play cannot address profiles of people and always acts as with the best response for the mixed strategy observed up to the moment.

→ How to act when we know that there are several profiles of people?

Rational Learning

Let's assume that there are two profiles of people, nice and not nice.

Nice people collaborate 90% of the time

$p(C|N)=0.9$

Not nice people collaborate only 10% of the time

$p(C|\tilde{N})=0.1$

Rational Learning

If during play one player collaborates what is the probability that they are Nice?

$$p(N|C) = \frac{p(C|N)p(N)}{\sum_b p(C|b)p(b)} \propto p(C|N)p(N)$$

$$p(N|C) \propto 0.9 \times 0.5$$
$$p(\tilde{N}|C) \propto 0.1 \times 0.5$$
$$p(N|C) = \frac{p(N|C)}{p(N|C) + p(\tilde{N}|C)} = 0.9$$

Rational Learning

If during play one player plays C,D,C, what is the probability that they are Nice?

$$p(N|C,D,C) \propto p(C,D,C|N)p(N)$$
$$\propto p(C|N)p(D|N)p(C|N)p(N)$$
$$\propto 0.9 \times 0.1 \times 0.9 \times 0.5 = 0.0405$$
$$p(N|C,D,C) = \frac{0.0405}{0.0405 + 0.0045} = 0.9$$

Assuming independe

Rational Learning

If during play one player plays C,D,C, what is the probability that they are Nice?

$$p(N|C,D,C) \propto p(C,D,C|N)p(N)$$
$$\propto p(C|N)p(D|N)p(C|N)p(N)$$

The rule can be applied incrementally and at each step the prior is the posterior of the previous step.

Rational Learning

How to compute the best response?

In ficticious play we assume that they are playing the mixed strategy that we computed up to the current moment. Now we have probabilities over behaviors.

Again a Bayesian risk analysis can be used.

Again U_1(a_1,a_2) it is the utility for agent 1 when the players play a_1,a_2

p(action = a|profile = b) it is the probability of playing a if the player profile is b

Rational Learning

How to compute the best response?

Again a Bayesian risk analysis can be used.

Again U_1(a_1,a_2) it is the utility for agent 1 when the players play a_1,a_2

p(action = a|profile = b) it is the probability of playing a if the player profile is b

Utility(Cooperate) =
$\sum_b \sum_a U(C, a)$p(action = a|profile = b)$p(profile = b)$

Other Learning Approaches

Policy gradient

Instead of learning the value and then choose the actions based on a given criteria, the values obtained are used to change directly the policy/strategy (using gradient ascent).

Policy-Based Learning

- Assume the payoffs for each agent

$$\mathcal{R}_i = \begin{bmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & r_{2,2} \end{bmatrix} \qquad \mathcal{R}_j = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix}$$

- And the following strategies (one advantage is that we can reason directly with mixed strategies)

$$\pi_i = (\alpha, 1-\alpha) \qquad \pi_j = (\beta, 1-\beta), \qquad \alpha, \beta \in [0, 1]$$

- What is the expected payoff for each agent?

Policy-Based Learning

$$\mathcal{R}_i = \begin{bmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & r_{2,2} \end{bmatrix} \qquad \mathcal{R}_j = \begin{bmatrix} c_{1,1} & c_{1,2} \\ c_{2,1} & c_{2,2} \end{bmatrix}$$

$$\pi_i = (\alpha, 1-\alpha) \qquad \pi_j = (\beta, 1-\beta), \qquad \alpha, \beta \in [0, 1]$$

The expected payoff for each is:

$$U_i(\alpha, \beta) = \alpha\beta r_{1,1} + \alpha(1-\beta)r_{1,2} + (1-\alpha)\beta r_{2,1} + (1-\alpha)(1-\beta)r_{2,2}$$

$$= \alpha\beta u + \alpha(r_{1,2} - r_{2,2}) + \beta(r_{2,1} - r_{2,2}) + r_{2,2}$$

$$U_j(\alpha, \beta) = \alpha\beta c_{1,1} + \alpha(1-\beta)c_{1,2} + (1-\alpha)\beta c_{2,1} + (1-\alpha)(1-\beta)c_{2,2}$$

$$= \alpha\beta u' + \alpha(c_{1,2} - c_{2,2}) + \beta(c_{2,1} - c_{2,2}) + c_{2,2}$$

$$u = r_{1,1} - r_{1,2} - r_{2,1} + r_{2,2}$$

$$u' = c_{1,1} - c_{1,2} - c_{2,1} + c_{2,2}.$$

Policy-Based Learning

The expected payoff for each is:

$$U_i(\alpha,\beta) = \alpha\beta r_{1,1} + \alpha(1-\beta)r_{1,2} + (1-\alpha)\beta r_{2,1} + (1-\alpha)(1-\beta)r_{2,2}$$

$$= \alpha\beta u + \alpha(r_{1,2} - r_{2,2}) + \beta(r_{2,1} - r_{2,2}) + r_{2,2}$$

$$u = r_{1,1} - r_{1,2} - r_{2,1} + r_{2,2}$$

The gradient update: $$\alpha^{k+1} = \alpha^k + \kappa \frac{\partial U_i(\alpha^k,\beta^k)}{\partial \alpha^k}$$

Where $$\frac{\partial U_i(\alpha,\beta)}{\partial \alpha} = \beta u + (r_{1,2} - r_{2,2})$$

Policy-Based Learning

The expected payoff for each is:

$$U_i(\alpha, \beta) = \alpha\beta r_{1,1} + \alpha(1-\beta) r_{1,2} + (1-\alpha)\beta r_{2,1}$$

$$= \alpha\beta u + \alpha(r_{1,2} - r_{2,2}) + \beta(r_{2,1}$$

$$u = r_{1,1} - r_{1,2} - r_{2}$$

To do this we need to know all the utility matrices and the policy of the other agent at each step...

The gradient update: $\alpha^{k+1} = \alpha^{k} +$

Where $$\frac{\partial U_i(\alpha, \beta)}{\partial \alpha} = \beta u + (r_{1,2} - r_{2,2})$$

Generalised Infinitesimal Gradient Ascent

Given a policy π_i for agent i and an action a_j for agent j, the expected reward for agent i against action a_j is

$$U_i(\pi_i, a_j) = \sum_{a_i \in A_i} \pi_i(a_i)\mathcal{R}_i(a_i, a_j). \tag{6.46}$$

Therefore, the gradient of this expected reward with respect to policy π_i is simply the vector of rewards for each of agent i's available actions $1, 2, 3 \ldots$,

$$\nabla_{\pi_i} U_i(\pi_i, a_j) = \left[\begin{array}{cccc} \frac{\partial U_i(\pi_i, a_j)}{\partial \pi_i(1)}, & \frac{\partial U_i(\pi_i, a_j)}{\partial \pi_i(2)}, & \frac{\partial U_i(\pi_i, a_j)}{\partial \pi_i(3)}, & \ldots \end{array} \right] \tag{6.47}$$

$$= \left[\begin{array}{cccc} \mathcal{R}_i(1, a_j), & \mathcal{R}_i(2, a_j), & \mathcal{R}_i(3, a_j), & \ldots \end{array} \right]. \tag{6.48}$$

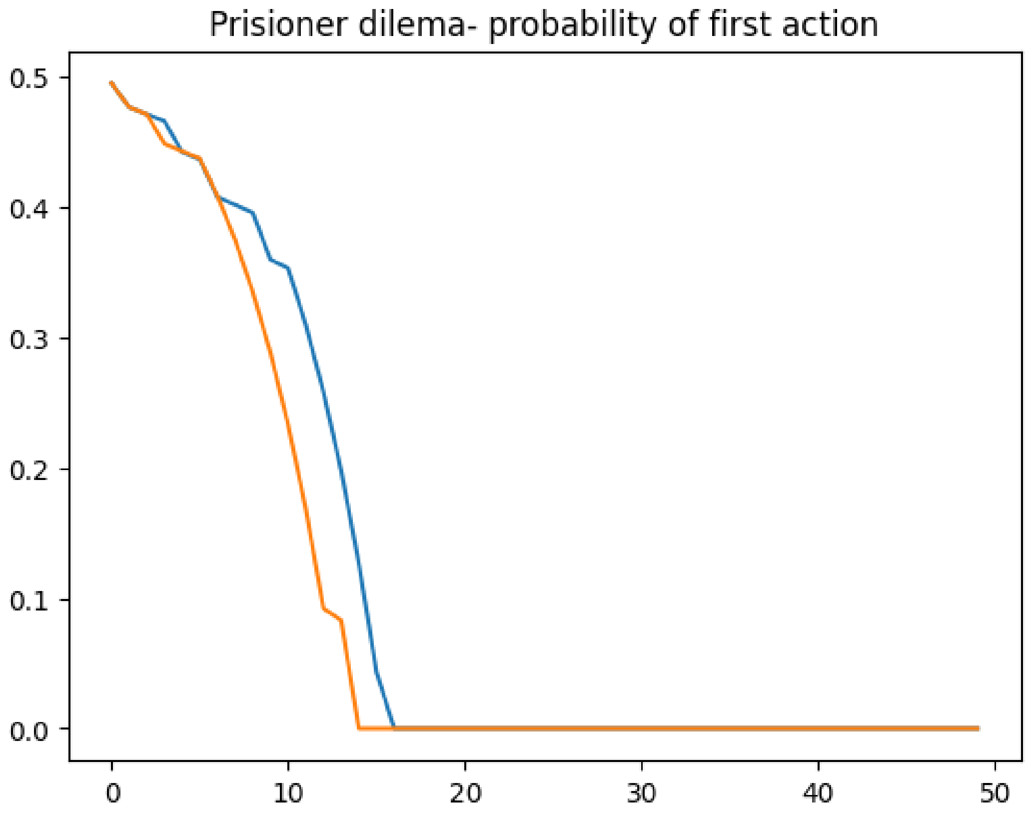
Given policy π_i^k and observed action a_j^k in episode k, GIGA updates π_i^k via two steps:

$$\begin{aligned} &(1) \quad \tilde{\pi}_i^{k+1} \leftarrow \pi_i^k + \kappa^k \nabla_{\pi_i^k} U_i(\pi_i^k, a_j^k) \\ &(2) \quad \pi_i^{k+1} \leftarrow P(\tilde{\pi}_i^{k+1}) \end{aligned} \tag{6.49}$$

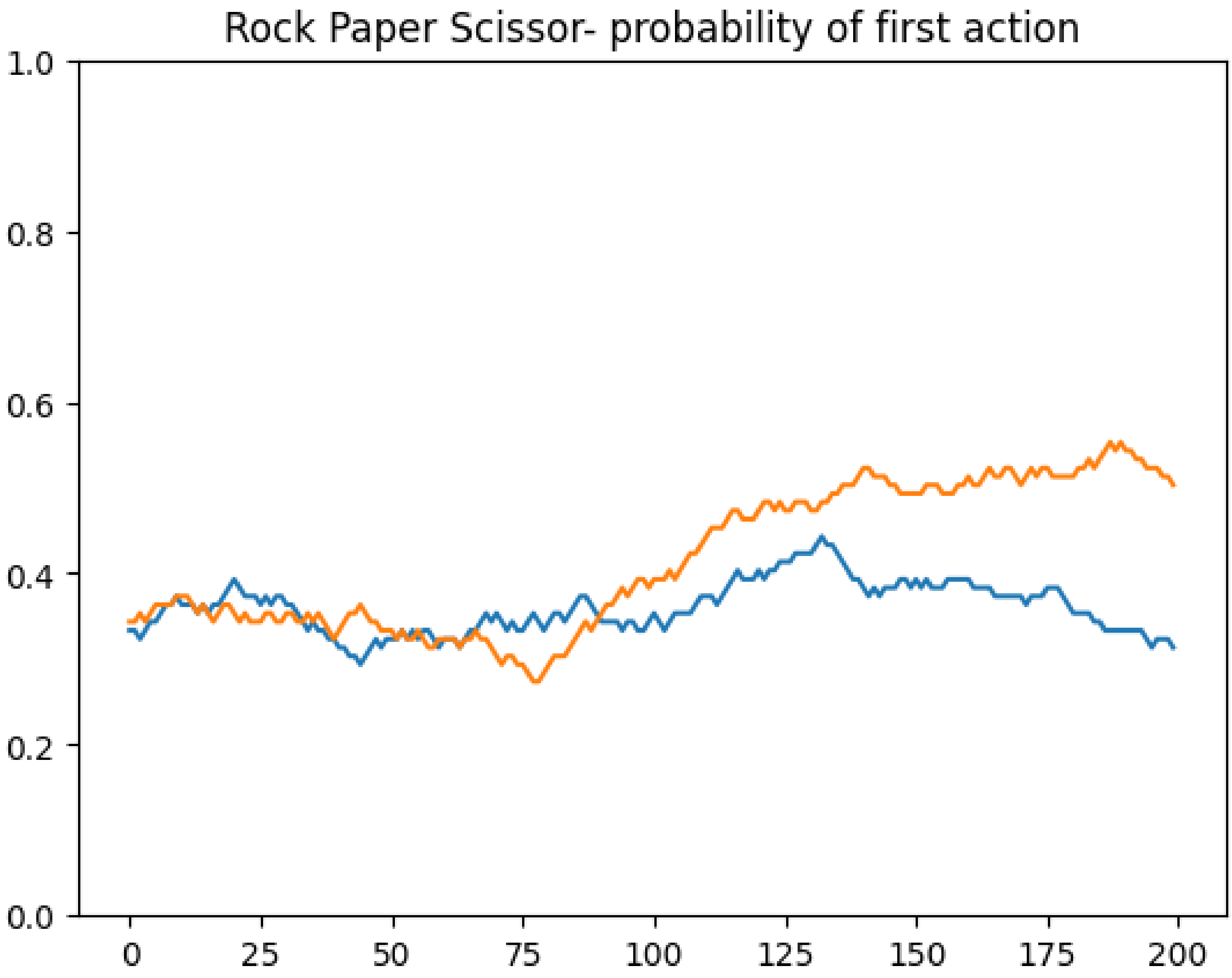
Generalised Infinitesimal Gradient Ascent

```
    pols =[[],[]]
    pols[0] = np.ones(n0)/n0
    pols[1] = np.ones(n1)/n1

    A = [-1,-1]
    for ii in range(200):
        #select action for each agent, $\epsilon-greedy$ might 
work
        A[0] = np.random.choice(n0, p=pols[0])
        A[1] = np.random.choice(n1, p=pols[1])

        r = get_value(g,A[0],A[1])

        d0 = g.payoff_matrices[0][:,A[1]]
        d1 = g.payoff_matrices[1][A[0],:]

        pols[0] = np.clip(pols[0] + 0.01 * d0,0,1)
        pols[1] = np.clip(pols[1] + 0.01 * d1,0,1)
        pols[0] *= 1/np.sum(pols[0])
        pols[1] *= 1/np.sum(pols[1])
```

Generalised Infinitesimal Gradient Ascent



Generalised Infinitesimal Gradient Ascent



Learning in Multi-Agent Systems

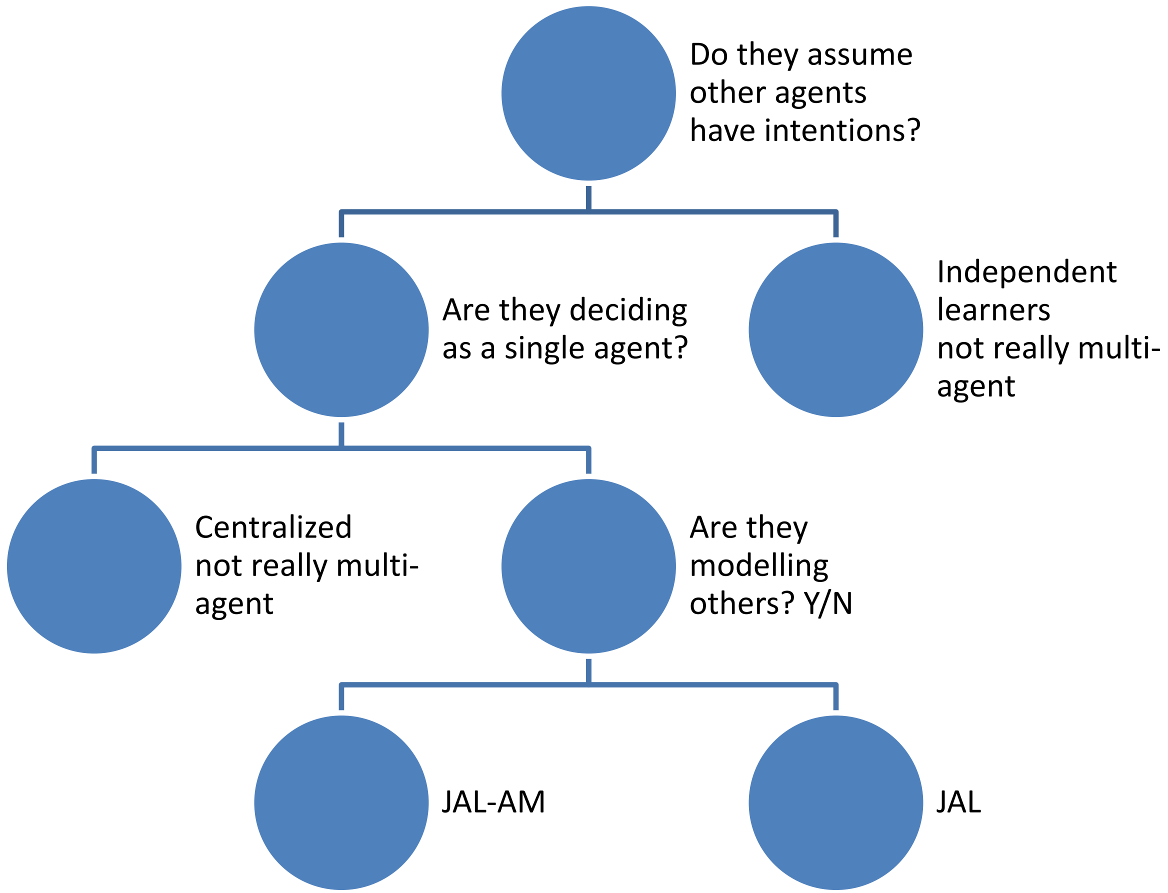
What should the criteria be?

- Assume the others are doing their best
- Are they considering my preferences
- Find a stable equilibria
- Improve the worst case scenario
- Ignore others
- Try to predict others

https://openspiel.readthedocs.io/en/latest/algorithms.html

AlphaZero (C++/LibTorch)	MARL	Silver et al. '18	
AlphaZero (Python/TF)	MARL	Silver et al. '18	
Correlated Q-Learning	MARL	Greenwald & Hall '03	~
Asymmetric Q-Learning	MARL	Kononen '04	~
Deep CFR	MARL	Brown et al. '18	
DiCE: The Infinitely Differentiable Monte-Carlo Estimator (LOLA-DiCE)	MARL	Foerster, Farquhar, Al-Shedivat et al. '18	~
Exploitability Descent (ED)	MARL	Lockhart et al. '19	
(Extensive-form) Fictitious Play (XFP)	MARL	Heinrich, Lanctot, & Silver '15	
Learning with Opponent-Learning Awareness (LOLA)	MARL	Foerster, Chen, Al-Shedivat, et al. '18	~
Nash Q-Learning	MARL	Hu & Wellman '03	~
Neural Fictitious Self-Play (NFSP)	MARL	Heinrich & Silver '16	
Neural Replicator Dynamics (NeuRD)	MARL	Omidshafiei, Hennes, Morrill, et al. '19	X
Regret Policy Gradients (RPG, RMPG)	MARL	Srinivasan, Lanctot, et al. '18	
Policy-Space Response Oracles (PSRO)	MARL	Lanctot et al. '17	
Q-based ("all-actions") Policy Gradient (QPG)	MARL	Srinivasan, Lanctot, et al. '18	
Regularized Nash Dynamics (R-NaD)	MARL	Perolat, De Vylder, et al. '22	
Regression CFR (RCFR)	MARL	Waugh et al. '15, Morrill '16	
Rectified Nash Response (PSRO_rn)	MARL	Balduzzi et al. '19	~

Summary:
Learning in Markov Games



Reference: www.marl-book.com
Chapter 6 – 6.1, 6.2, 6.3, 6.4