Stochastic Game

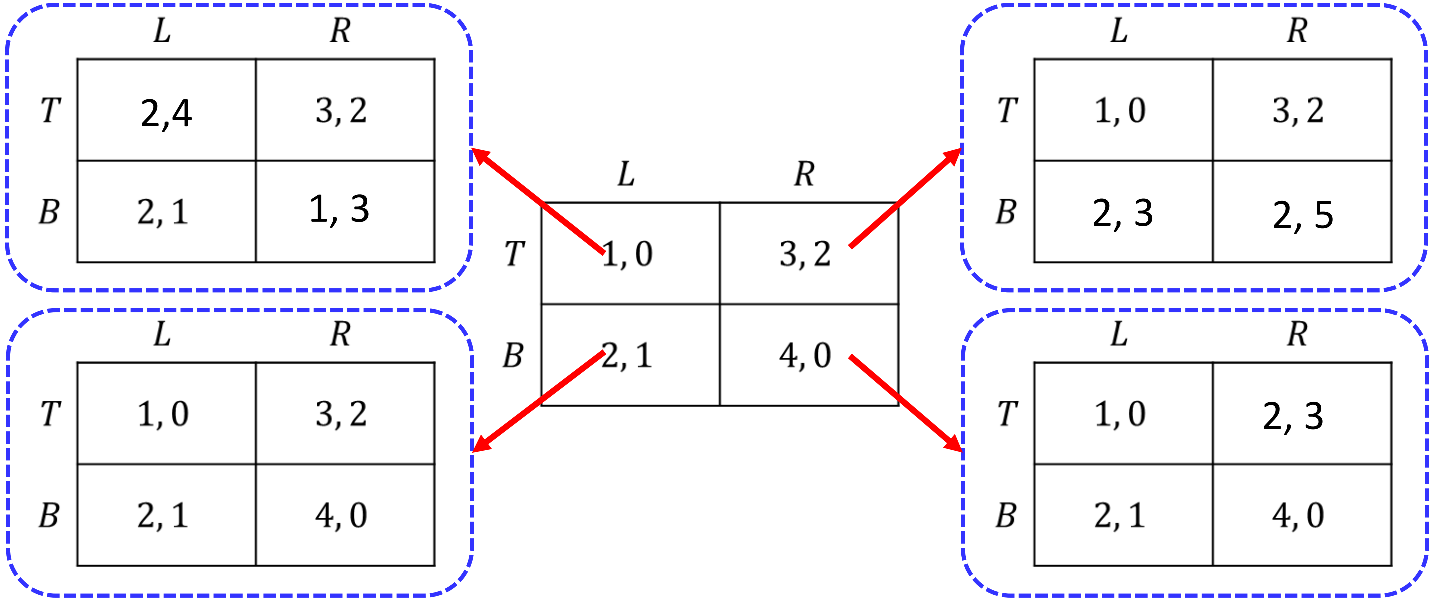
Motivations

What if we didn't always repeat back to the same stage game?

- A stochastic game is a generalization of repeated games
  - agents repeatedly play games from a set of normal-form games
  - the game played at any iteration depends on the previous game played and on the actions taken by all agents in that game

What if there are multiple decision makers in Markov Decision Process?

- A stochastic game is a generalized Markov decision process
  - there are multiple players one reward function for each agent
  - the state transition function and reward functions depend on the action choices of both players



- Stochastic game is a moral general setting where learning is taking place
  - The game transits to another game depending on the joint actions by agents
  - Same players and same actions sets are used through games
- Most of the techniques discussed in the context of repeated games are applicable more generally to stochastic games
  - ✓ specific results obtained for repeated games do not always generalize.

Formal Definition

Definition (Stochastic game)

A stochastic game is a tuple (N, S, A, R, T), where

- N is a finite set of n players
- S is a finite set of states (stage games),
- $A = A_1 \times \cdots \times A_n$, where A_i is a finite set of actions available to player i,
- $T : S \times A \times S \longmapsto [0,1]$ is the transition probability function; $T(s, a, s')$ is the probability of transitioning from state s to state s' after joint action a,
- $R = r_1 \dots, r_n$, where $r_i : S \times A \longmapsto \mathbb{R}$ is a real-valued payoff function for player i

Transition model

- All agents $(1, \ldots, n)$ share the joint state s
- The transition equation is similar to the Markov Decision Process decision transition:

$$\text{MDP}: \sum_{s'} T(s, a, s') = \sum_{s'} p(s' | a, s) = 1, \forall s \in S, \forall a \in A$$

$$\text{SG}: \sum_{s'} T(s, a_1, \ldots, a_i, \ldots, a_n, s') = \sum_{s'} p(s' | a_1, \ldots, a_i, \ldots, a_n, s) = 1$$

$$\forall s \in S, \forall a_i \in A_i, i = (1, \ldots, n)$$

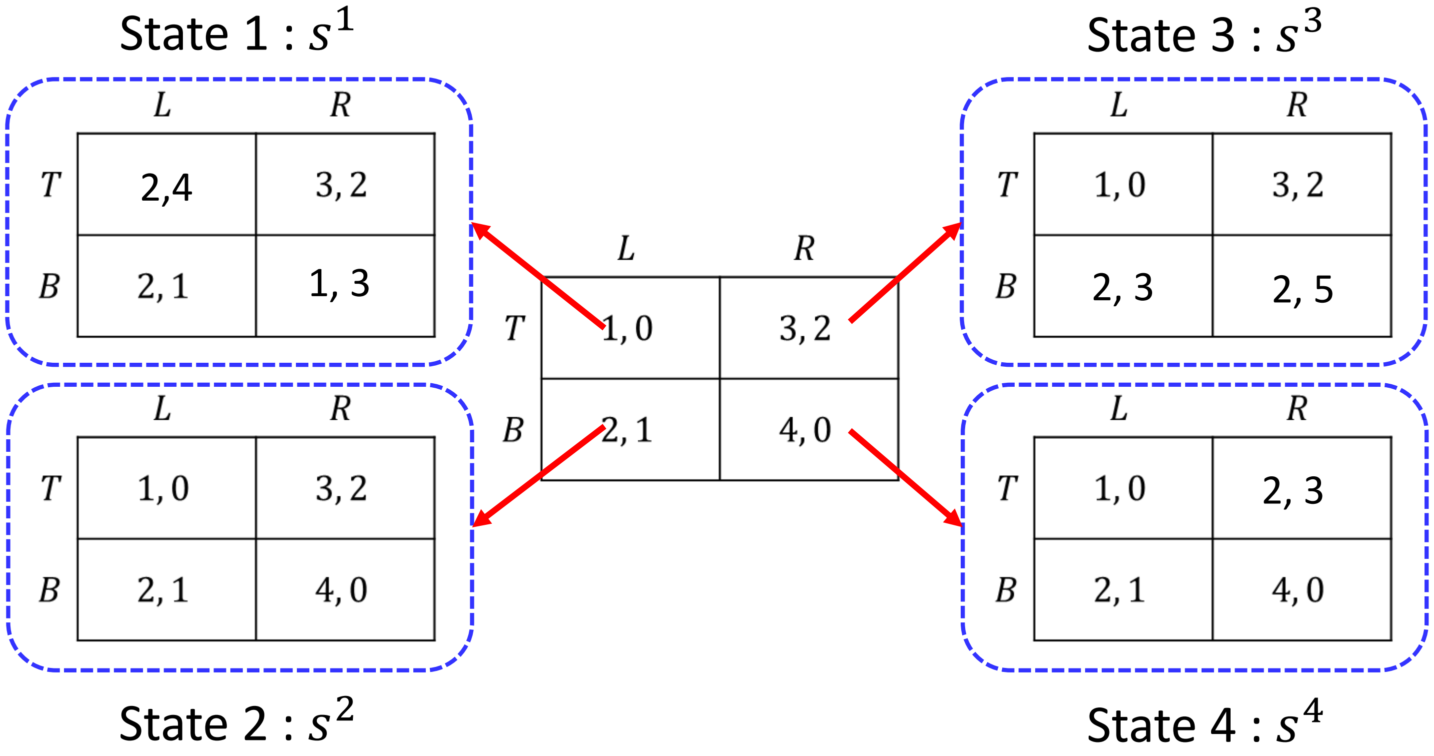
Reward function

- Reward function r_i for agent i depends on the current joint state s, the joint action $a = (a_1, \ldots, a_n)$, and the next joint future state s'

MDP : $r(s, a, s')$

SG: $r_i(s, a_1, \ldots, a_i, \ldots, a_n, s')$

Policy



- Policy π_1 will give the action that will be taken by player 1 at a given state (stage game):

$$a_1 = \pi_1(s),\ a_1 \in \{T, B\}$$

Value function

- As we did in MDP, we can define value function
- Let π_i be the policy of player $i \in N$. For a given initial state s, the value of state s for player i is defined as

$$V_i(s, \pi_1, \dots, \pi_i, \dots, \pi_n) = \sum_{t=0}^{\infty} \gamma^t E[r_{i,t} | \pi_1, \dots, \pi_i, \dots, \pi_n, s_0 = s]$$

  - ➢ The accumulated rewards depends on the policies of other agents
  - ➢ The immediate reward is expressed as expected value, because some policy π_i can be stochastic

- In a *discounted stochastic game*, the objective of each player is to maximize the discounted sum of rewards, with discount factor $\gamma \in [0,1)$.

Equilibrium strategy

Definition (Nash equilibrium policy in Stochastic game)

In a stochastic game $\Gamma = (N, S, A, R, T)$, a Nash equilibrium policy is a tuple of n policies $\pi^* = (\pi_1^*, \ldots, \pi_n^*)$ such that for all $s \in S$ and $i = 1, \ldots n$,

$$V_i(s, \pi_1^*, \ldots, \pi_i^*, \ldots, \pi_n^*) \geq V_i(s, \pi_1^*, \ldots, \pi_i, \ldots, \pi_n^*) \text{ for all } \pi_i \in \Pi_i$$

- A Nash equilibrium is a joint policy where each agent's policy is a best response to the others
- For a stochastic game, each agent's policy is defined over the entire time horizon of the game
- **A Nash equilibrium state value** $V_i(s, \pi_1^*, \ldots, \pi_n^*)$ is defined as the sum of discounted rewards when all agents following the Nash equilibrium policies $\pi^* = (\pi_1^*, \ldots, \pi_n^*)$
  - Notations: $V_i^*(s) = V_i^{\pi^*}(s) = V_i(s, \pi_1^*, \ldots, \pi_n^*)$

Equilibrium policy

Theorem (Fink 1964)

Every n −player discounted stochastic game processes at least one Nash equilibrium policy in stationary policies

- Action selection rule for non-stationary policy is different depending on time
  - $\pi_t(s) \neq \pi_{t+1}(s)$
- There are generally a great multiplicity of non-stationary equilibria, whose fact is partially demonstrated by Folk Theorems

Single agent

Q-values

$Q^\pi(s,a)$: The expected utility of taking action a from state s, and then following policy π

$$Q^\pi(s,a) = \mathbb{E}_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{t+k} \,\middle|\, S_t = s, A_t = a\right)$$

Optimal Q-values

$$Q^*(s,a) = \max_\pi Q^\pi(s,a)$$

$$= \max_\pi \mathbb{E}[r(s,a,s') + \gamma V^\pi(s') | s_t = s, a_t = a]$$

$$= \mathbb{E}\left[r(s,a,s') + \gamma \max_\pi V^\pi(s') \,|\, s_t = s, a_t = a\right]$$

$$= \mathbb{E}[r(s,a,s') + \gamma V^*(s') | s_t = s, a_t = a] \qquad \because V^*(s') \equiv \max_\pi V^\pi(s')$$

$$= \mathbb{E}\left[r(s,a,s') + \gamma \max_{a'} Q^*(s',a') \,|\, s_t = s, a_t = a\right] \qquad \because V^*(s') \equiv \max_{a'} Q^*(s',a')$$

Optimization over policy becomes greedy optimization over action!

- Optimal Q-value for a single-agent is the sum of the current reward and future discounted rewards when playing the optimal strategy from the next period onward

Multi agents

Q-values for agent i

$Q_i^\pi(s, a_1, \dots, a_n)$: The expected utility of taking joint action $(a_1, \dots, a_n)$ from state s, and then following policy π

$$Q_i^\pi(s, a_1, \dots, a_n) = \mathbb{E}_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{i,t+k} \;\middle|\; S_t = s, A = (a_1, \dots, a_n)\right)$$

Optimal Q-values for agent i

$$Q_i^*(s, a_1, \dots, a_n) = \max_{\pi_1,\dots,\pi_n} Q_i^\pi(s, a_1, \dots, a_n)$$

$$= \max_{\pi_1,\dots,\pi_n} \mathbb{E}[r_i(s, a_1, \dots, a_n, s') + \gamma V_i(s', \pi_1, \dots, \pi_n) | s_t = s, a_t = (a_1, \dots, a_n)]$$

$$= \mathbb{E}\left[r_i(s, a_1, \dots, a_n, s') + \gamma \max_{\pi_1,\dots,\pi_n} V_i(s', \pi_1, \dots, \pi_n) | s_t = s, a_t = (a_1, \dots, a_n)\right]$$

$$= \mathbb{E}[r_i(s, a_1, \dots, a_n, s') + \gamma V_i(s', \pi_1^*, \dots, \pi_n^*) | s_t = s, a_t = (a_1, \dots, a_n)]$$

$$= \mathbb{E}\left[r_i(s, a_1, \dots, a_n, s') + \gamma \max_{a_1,\dots,a_n} Q_i^*(s', a_1, \dots, a_n) \,| s_t = s, a_t = (a_1, \dots, a_n)\right]$$

- Optimal Q-value for agent i occurs when all agents are jointly coordinating to maximize agent i's accumulated reward
  - ➢ Rarely occurs! : Optimal Q-values for all agents are not achieved simultaneously

Multi agents

Q-values for agent i

$Q_i^\pi(s, a_1, \dots, a_n)$: The expected utility of taking joint action $(a_1, \dots, a_n)$ from state s, and then following policy π

$$Q_i^\pi(s, a_1, \dots, a_n) = \mathbb{E}_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{i,t+k} \mid S_t = s, A = (a_1, \dots, a_n)\right)$$

Optimal Q-values for agent i

$$\begin{aligned}
Q_i^*(s, a_1, \dots, a_n) &= \max_{\pi_1, \dots, \pi_n} Q_i^\pi(s, a_1, \dots, a_n) \\
&= \max_{\pi_1, \dots, \pi_n} \mathbb{E}[r_i(s, a_1, \dots, a_n, s') + \gamma V_i(s', \pi_1, \dots, \pi_n) | s_t = s, a_t = (a_1, \dots, a_n)] \\
&= \mathbb{E}\left[r_i(s, a_1, \dots, a_n, s') + \gamma \max_{\pi_1, \dots, \pi_n} V_i(s', \pi_1, \dots, \pi_n) | s_t = s, a_t = (a_1, \dots, a_n)\right] \\
&= \mathbb{E}[r_i(s, a_1, \dots, a_n, s') + \gamma V_i(s', \pi_1^*, \dots, \pi_n^*) | s_t = s, a_t = (a_1, \dots, a_n)] \\
&= \mathbb{E}\left[r_i(s, a_1, \dots, a_n, s') + \gamma \max_{a_1, \dots, a_n} Q_i^*(s', a_1, \dots, a_n) | s_t = s, a_t = (a_1, \dots, a_n)\right]
\end{aligned}$$

- Optimal Q-value for agent i occurs when all agents are jointly coordinating to maximize agent i's accumulated reward
  - Rarely occurs! : Optimal Q-values for all agents are not achieved simultaneously

Multi agents

Q-values for agent i

$Q_i^\pi(s, a_1, \ldots, a_n)$: The expected utility of taking joint action $(a_1, \ldots, a_n)$ from state s, and then following policy π

$$Q_i^\pi(s, a_1, \ldots, a_n) = \mathbb{E}_\pi\left(\sum_{k=0}^{\infty} \gamma^k r_{i,t+k} \;\middle|\; S_t = s, A = (a_1, \ldots, a_n)\right)$$

Nash Q-values for agent i

$$Q_i^*(s, a_1, \ldots, a_n) = \underset{\pi_1,\ldots,\pi_n}{\text{Nash}}\; Q_i^\pi(s, a_1, \ldots, a_n)$$

$$= \underset{\pi_1,\ldots,\pi_n}{\text{Nash}}\; \mathbb{E}[r_i(s, a_1, \ldots, a_n, s') + \gamma V_i(s', \pi_1, \ldots, \pi_n) | s_t = s, a_t = (a_1, \ldots, a_n)]$$

$$= \mathbb{E}\left[r_i(s, a_1, \ldots, a_n, s') + \gamma \underset{\pi_1,\ldots,\pi_n}{\text{Nash}}\; V_i(s', \pi_1, \ldots, \pi_n) | s_t = s, a_t = (a_1, \ldots, a_n)\right]$$

$$= \mathbb{E}[r_i(s, a_1, \ldots, a_n, s') + \gamma V_i(s', \pi_1^*, \ldots, \pi_n^*) | s_t = s, a_t = (a_1, \ldots, a_n)]$$

$$= \mathbb{E}\left[r_i(s, a_1, \ldots, a_n, s') + \gamma \underset{a_1,\ldots,a_n}{\text{Nash}}\; Q_i^*(s', a_1, \ldots, a_n) \,| s_t = s, a_t = (a_1, \ldots, a_n)\right]$$

Equilibrium over policies becomes stage game equilibrium over action!

- A **Nash Q value** $Q_i^*(s, a_1, \ldots, a_n)$ is the expected sum of discounted rewards when all agents take the joint action $a = (a_1, \ldots, a_n)$ at given state s and follow a Nash equilibrium strategy $\pi^* = (\pi_1^*, \ldots, \pi_n^*)$

Nash Bellman equation

For single agent:

$$V^*(s') = \max_a Q^*(s', a)$$

$$Q^*(s, a) = \mathbb{E}[r(s, a, s') + \gamma V^*(s') | s_t = s, a_t = a]$$

$$= \mathbb{E}\left[r(s, a, s') + \gamma \max_{a'} Q^*(s', a') \,| s_t = s, a_t = a\right]$$

For multiple agents:

$$V_i(s', \pi_1^*, \dots, \pi_n^*) = \underset{a_1,\dots,a_n}{\text{Nash}} \; Q_i^*(s', a_1, \dots, a_n)$$

$$Q_i^*(s, a_1, \dots, a_n) = \mathbb{E}[r(s, a, s') + \gamma V_i(s', \pi_1^*, \dots, \pi_n^*) | s_t = s, a_t = a]$$

$$= \mathbb{E}\left[r(s, a, s') + \gamma \underset{a_1,\dots,a_n}{\text{Nash}} \; Q_i^*(s', a_1, \dots, a_n) \,| s_t = s, a_t = (a_1, \dots, a_n)\right]$$

How to compute A Nash (equilibrium) state value $V_i(s, \pi_1^*, \ldots, \pi_n^*)$

For multiple agents:

$$V_i(s', \pi_1^*, \ldots, \pi_n^*) = \underset{a_1,\ldots,a_n}{\mathrm{Nash}}\ Q_i^*(s', a_1, \ldots, a_n)$$

$$Q_i^*(s, a_1, \ldots, a_n) = \mathbb{E}[r(s,a,s') + \gamma V_i(s', \pi_1^*, \ldots, \pi_n^*) | s_t = s, a_t = a]$$

$$= \mathbb{E}\left[r(s,a,s') + \gamma \underset{a_1,\ldots,a_n}{\mathrm{Nash}}\ Q_i^*(s', a_1, \ldots, a_n) \,|\, s_t = s, a_t = (a_1, \ldots, a_n)\right]$$

- Nash **equilibrium** Q value $\underset{a_1,\ldots,a_n}{\mathrm{Nash}}\ Q_i^*(s', a_1, \ldots, a_n)$ can be computed by computing player ith Nash equilibrium value for the stage game $[Q_i^*(s', a_1, \ldots, a_n), \ldots, Q_n^*(s', a_1, \ldots, a_n)]$

  - for example when $i = 1,2$

	a_2^1	a_2^2
a_1^1	$Q_1^*(s', a_1^1, a_2^1), Q_2(s', a_1^1, a_2^1)$	$Q_1^*(s', a_1^1, a_2^2), Q_2(s', a_1^1, a_2^2)$
a_1^2	$Q_1^*(s', a_1^2, a_2^1), Q_2(s', a_1^2, a_2^1)$	$Q_1^*(s', a_1^2, a_2^2), Q_2(s', a_1^2, a_2^2)$

Nash equilibrium

Simplifying Notation

For multiple agents:

$$r_i(s, a_1, \dots, a_n, s') \rightarrow r_i(s, \vec{a}, s')$$

$$V_i(s, \pi_1^*, \dots, \pi_n^*) \rightarrow V_i^*(s)$$

$$Q_i^*(s, a_1, \dots, a_n) \rightarrow Q_i^*(s', \vec{a}\,)$$

$$\begin{aligned} Q_i^*(s, a_1, \dots, a_n) &= \mathbb{E}[r_i(s, a_1, \dots, a_n, s') + \gamma V_i(s', \pi_1^*, \dots, \pi_n^*) | s_t = s, a_t = (a_1, \dots, a_n)] \\ &= \mathbb{E}\left[r_i(s, a_1, \dots, a_n, s') + \gamma \underset{a_1,\dots,a_n}{\text{Nash}}\ Q_i^*(s', a_1, \dots, a_n)\ | s_t = s, a_t = (a_1, \dots, a_n)\right] \end{aligned}$$

$$\begin{aligned} Q_i^*(s', \vec{a}\,) &= \mathbb{E}[r_i(s, \vec{a}, s') + \gamma V_i^*(s') | s_t = s, a_t = \vec{a}] \\ &= \mathbb{E}[r_i(s, \vec{a}, s') + \gamma\ \text{Nash}\ Q_i^*(s')\ | s_t = s, a_t = \vec{a}] \end{aligned}$$

$$\underset{a_1,\dots,a_n}{\text{Nash}}\ Q_i^*(s', a_1, \dots, a_n) = Q_i^*(s', \vec{a}_{NE}) = \text{Nash}\ Q_i^*(s')$$

Computing Nash Q-values analytically

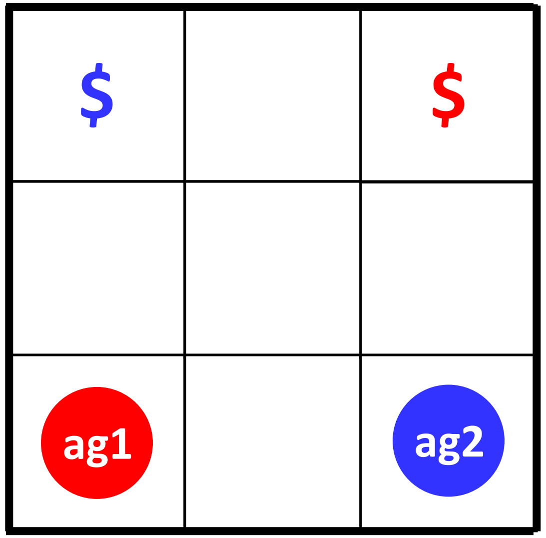
- If we know **Nash equilibrium policy** $\pi^* = (\pi_1^*, \dots, \pi_n^*)$, we can compute the Nash equilibrium state values $V_i(s, \pi_1^*, \dots, \pi_n^*)$ (i.e., policy evaluation)

$$V_i(s, \pi_1^*, \dots, \pi_n^*) = \sum_{t=0}^{\infty} \gamma^t E\left[r_{i,t} | \pi_1^*, \dots, \pi_n^*, s_0 = s\right]$$

- If we know **Nash equilibrium state value** $V_i(s, \pi_1^*, \dots, \pi_n^*)$ and transition models $p(s'|s, a_1, \dots, a_n)$, we can compute **Nash Q-values (i.e., Nash Q-function)** using backward induction (analytical approach)

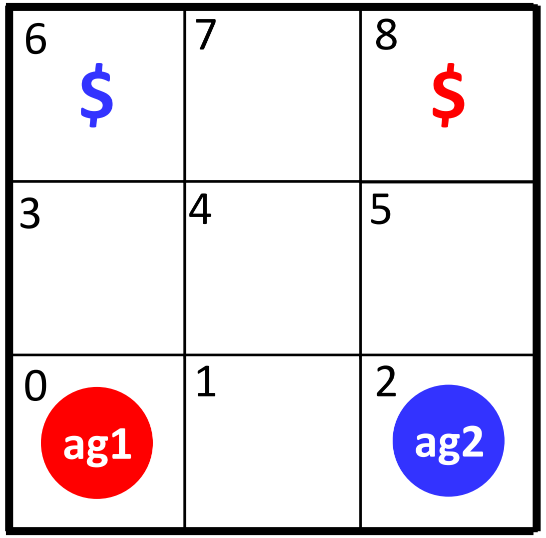
$$\begin{aligned} Q_i^*(s, a_1, \dots, a_n) &= \mathbb{E}[r_i(s, a_1, \dots, a_n, s') + \gamma V_i(s', \pi_1^*, \dots, \pi_n^*) | s_t = s, a_t = (a_1, \dots, a_n)] \\ &= r_i(s, a_1, \dots, a_n, s') + \sum_{s'} p(s'|s, a_1, \dots, a_n) V_i(s', \pi_1^*, \dots, \pi_n^*) \end{aligned}$$

Grid Game 1



- Grid game has deterministic moves
- Two agents start from respective lower corners, trying to reach their goal cells in the top row
- Agent can move only one cell a time, and in four possible directions: Left, Right, Up, Down
- If two agents attempt to move into the same cell (excluding a goal cell), they are bounced back to their previous cells
- The game ends as soon as an agent reaches its goal
  - The objective of an agent in this game is therefore to reach its goal with a minimum No. of steps
- Agents do not know
  - the locations of their goals at the beginning of the learning period
  - their own and the other agents' payoff functions
- Agent choose their action simultaneously and observe
  - the previous actions of both agents and the current joint state
  - the immediate rewards after both agents choose their actions

Grid Game 1 represented as stochastic game

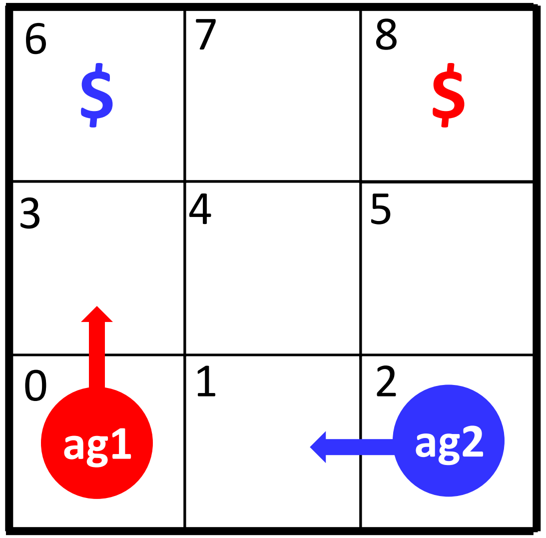


- The action space of agent i, $i = 1{,}2$, is $A_i = \{Left, Right, Down, Up\}$
- The sate space is $S = \{(0{,}1), (0{,}2), \dots, (8{,}7)\}$
  - $s = (l_1, l_2)$ represents the agents' joint location
  - $l_i \in \{0, 2, \dots, 8\}$ is the indexed location
- The reward function is, for $i = 1, 2$,

$$r_i = \begin{cases} 100 & \text{if } L(l_i, a_i) = Goal_i \\ -1 & \text{if } L(l_1, a_1) = L(l_2, a_2) \text{ and } L(l_i, a_i) \neq Goal_i \text{ for } i = 1{,}2 \\ 0 & \text{otherwise} \end{cases}$$

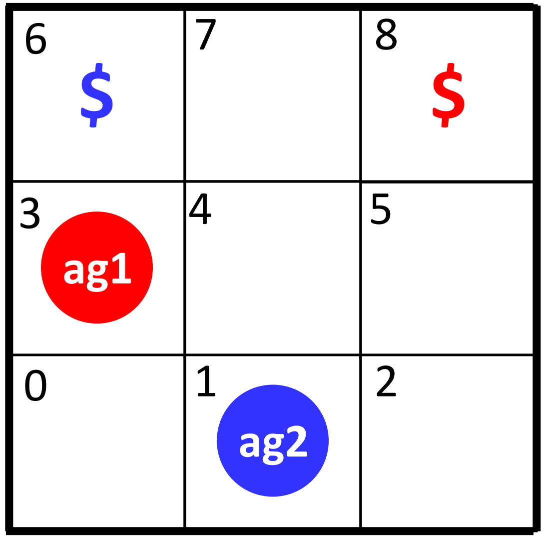
$l_i' = L(l_i, a_i)$ is the next location when executing a_i at l_i

Grid Game 1 represented as stochastic game



- $s = (l_1, l_2) = (0,2)$
- $a = (a_1, a_2) = (Up, Left)$

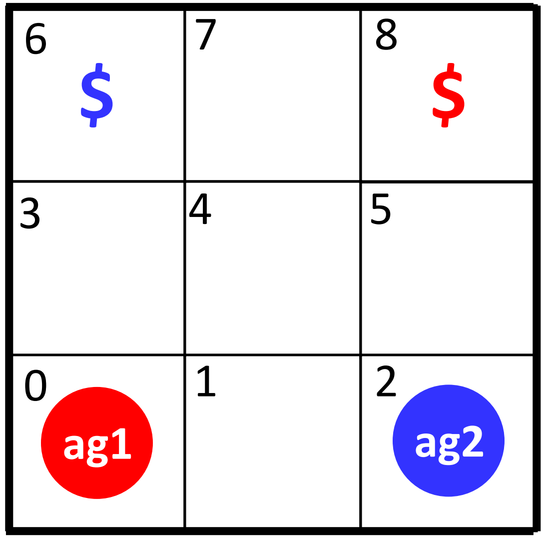
Grid Game 1 represented as stochastic game



- $s = (l_1, l_2) = (0,2)$
- $a = (a_1, a_2) = (Up, Left)$

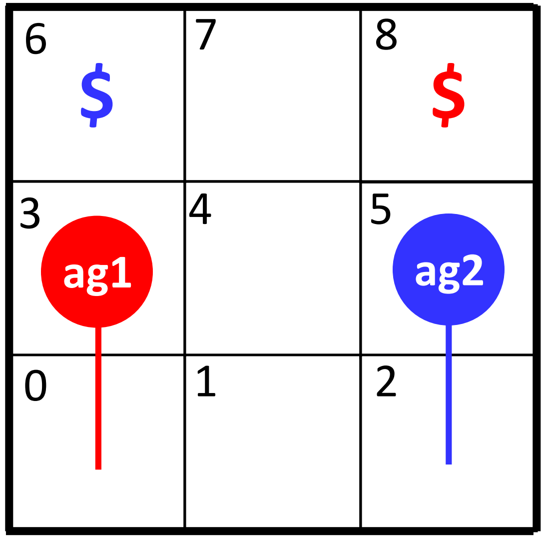
- $s' = \big(L(l_1, a_1), L(l_2, a_2)\big) = (3, 1)$
- $r_1 = 0$
- $r_2 = 0$

Grid Game 1 represented as stochastic game



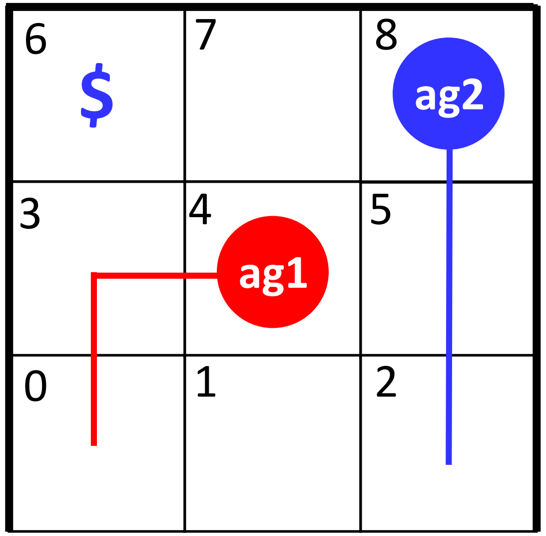
Nash Equilibrium strategies

Grid Game 1 represented as stochastic game



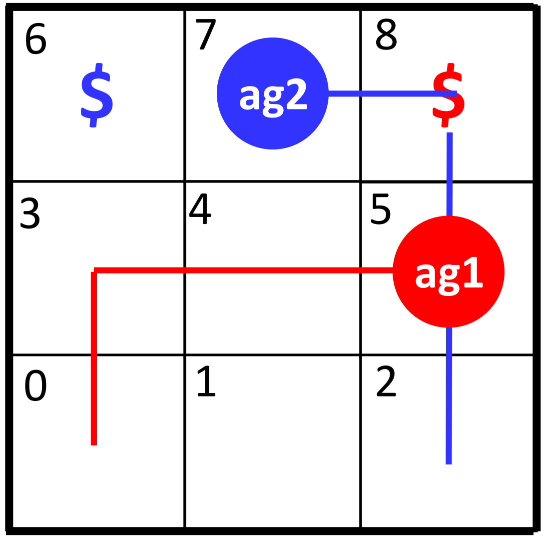
Nash Equilibrium strategies

Grid Game 1 represented as stochastic game



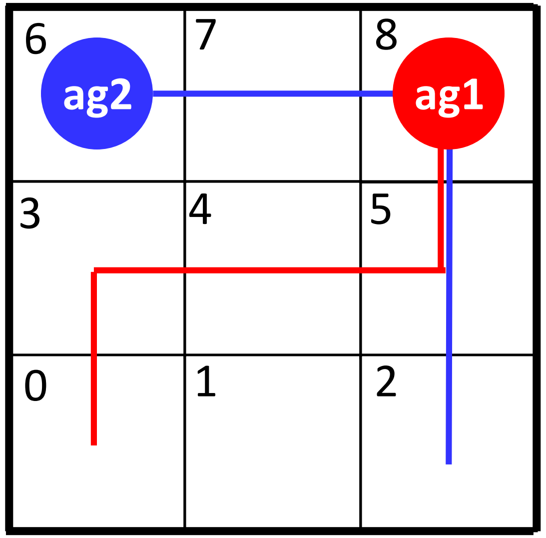
Nash Equilibrium policies

Grid Game 1 represented as stochastic game



Nash Equilibrium policies

Grid Game 1 represented as stochastic game



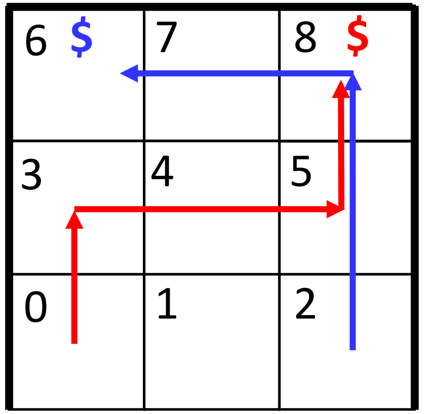
Nash Equilibrium policies

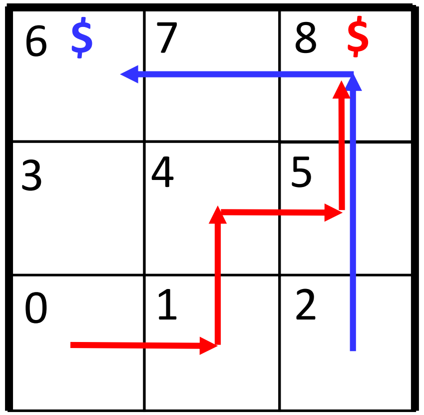
State s	$\pi_1(s)$
$(0, any)$	U
$(3, any)$	$Right$
$(4, any)$	$Right$
$(5, any)$	Up

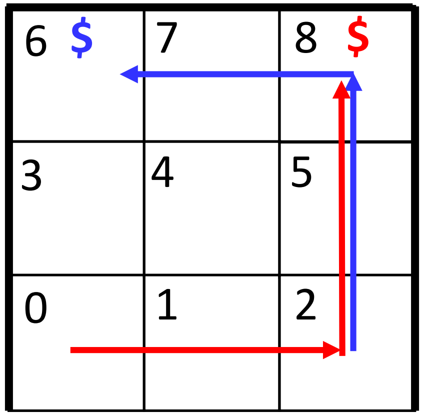
Nash strategy for agent 1

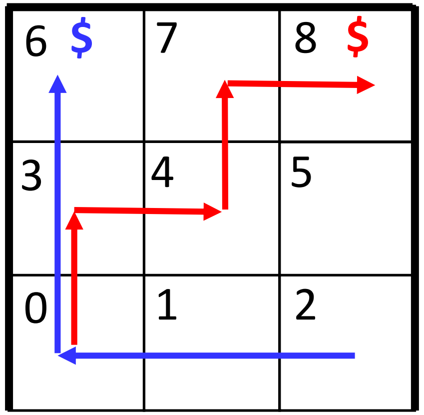
Grid Game 1 represented as stochastic game

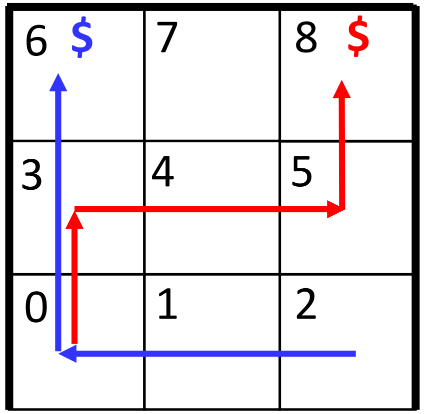
All Nash Equilibrium policies











Grid Game 1 represented as stochastic game

Nash Q values for the initial state $s_0 = (0,2)$

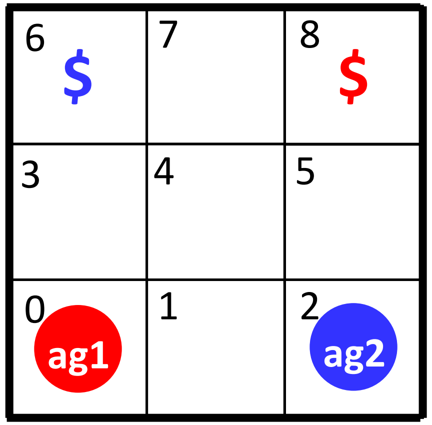
- The value of the game for agent 1 is defined as its accumulated reward when both agents follow their Nash equilibrium polices $\pi^* = (\pi_1^*, \dots, \pi_n^*)$:

$$V_1(s_0, \pi_1^*, \pi_2^*) = \sum_{t=0}^{\infty} \gamma^t E\left[r_{i,t} | \pi_1^*, \dots, \pi_n^*, s_0 = s\right]$$

- In Grid game 1 and initial state $s_0 = (0,2)$, this becomes, given $\gamma = 0.99$,

$$V_1(s_0, \pi_1^*, \pi_2^*) = 0 + 0.99 \times 0 + 0.99^2 \times 0 + 0.99^3 \times 100$$
$$= 97.0$$

$s_0 = (0,2)$



Grid Game 1 represented as stochastic game

Nash Q values for the initial state $s_0 = (0,2)$

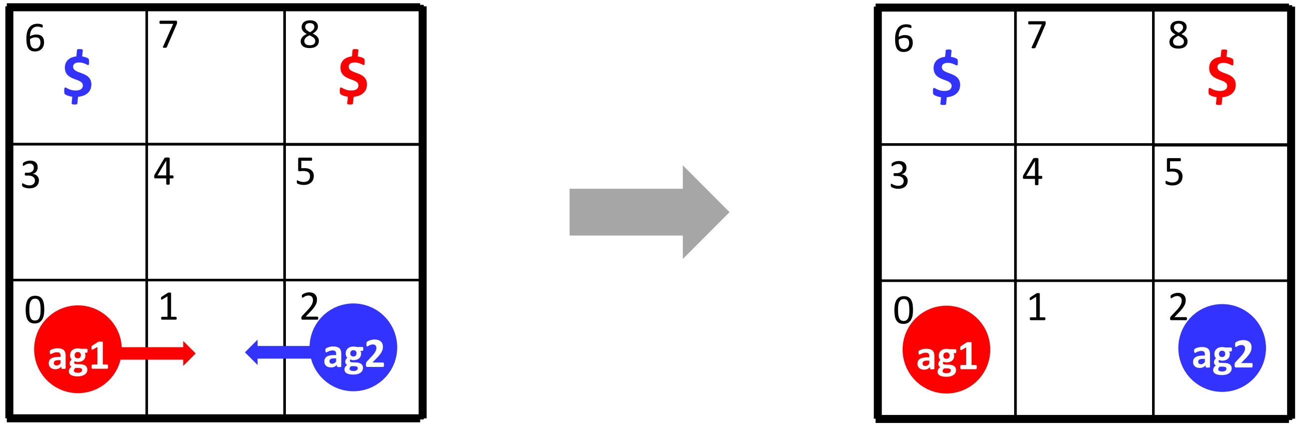
$$Q_1^*(s_0, a_1, a_2) = \mathbb{E}[r_1(s_0, a_1, a_2) + \gamma V_1(s', \pi_1^*, \pi_2^*) | s_t = s, a_t = (a_1, \dots, a_n)]$$

$$= r_1(s_0, a_1, a_2) + \gamma \sum_{s'} p(s'|s_0, a_1, a_2) V_1(s', \pi_1^*, \pi_2^*)$$

$$Q_1^*(s_0 = (0,2), Right, Left\,) = -1 + 0.99 \times V_1(s' = (0,2), \pi_1^*, \pi_2^*)$$

$$= -1 + 0.99 \times 97 = 95.1$$

$s_0 = (0,2)$ $\qquad$ $s' = s_0 = (0,2)$



Grid Game 1 represented as stochastic game

Nash Q values for the initial state $s_0 = (0,2)$

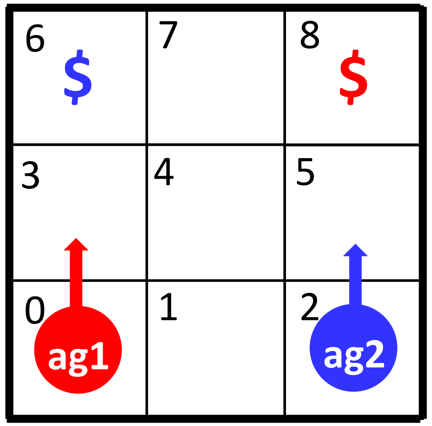
$$Q_1^*(s_0, a_1, a_2) = \mathbb{E}[r_1(s_0, a_1, a_2) + \gamma V_1(s', \pi_1^*, \pi_2^*) | s_t = s, a_t = (a_1, \dots, a_n)]$$

$$= r_1(s_0, a_1, a_2) + \gamma \sum_{s'} p(s' | s_0, a_1, a_2) V_1(s', \pi_1^*, \pi_2^*)$$

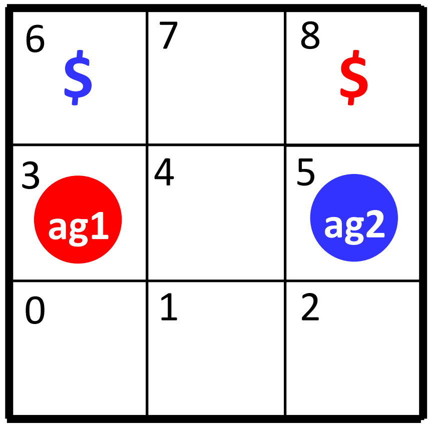
$$Q_1^*(s_0 = (0,2), Up, Up\,) = 0 + 0.99 \times V_1(s' = (3,5), \pi_1^*, \pi_2^*)$$

$$= 0 + 0.99 \times \{0 + 0.99 \times 0 + 0.99^2 \times 100\} = 97.0$$

$s_0 = (0,2)$



$s' = (3,5)$



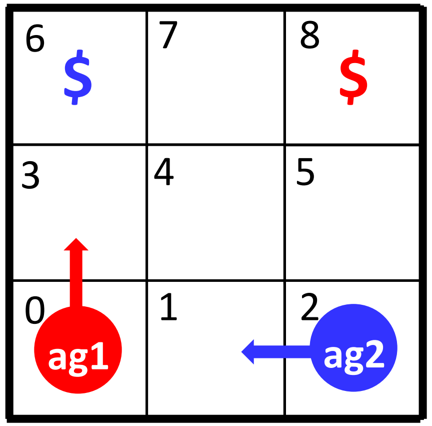
Grid Game 1 represented as stochastic game

Nash Q values for the initial state $s_0 = (0,2)$

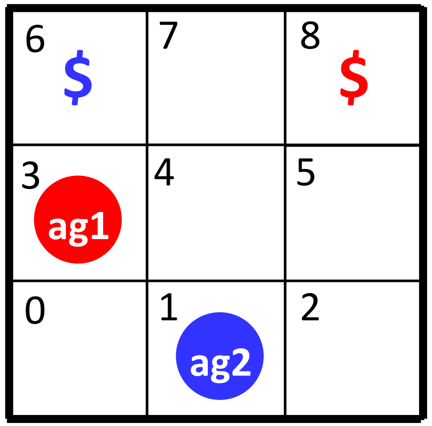
$$Q_1^*(s_0, a_1, a_2) = \mathbb{E}[r_1(s_0, a_1, a_2) + \gamma V_1(s', \pi_1^*, \pi_2^*) | s_t = s, a_t = (a_1, \dots, a_n)]$$
$$= r_1(s_0, a_1, a_2) + \gamma \sum_{s'} p(s' | s_0, a_1, a_2) V_1(s', \pi_1^*, \pi_2^*)$$

$$Q_1^*(s_0 = (0,2), Up, Left\,) = 0 + 0.99 \times V_1(s' = (3,1), \pi_1^*, \pi_2^*)$$
$$= 0 + 0.99 \times \{0 + 0.99 \times 0 + 0.99^2 \times 100\} = 97.0$$

$s_0 = (0,2)$



$s' = (3,1)$



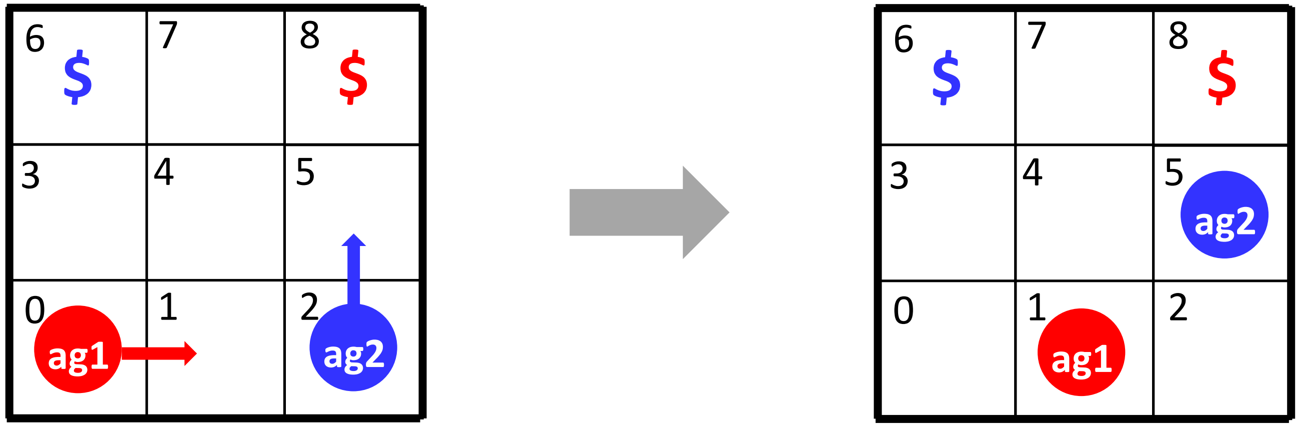
Grid Game 1 represented as stochastic game

Nash Q values for the initial state $s_0 = (0,2)$

$$
\begin{aligned}
Q_1^*(s_0, a_1, a_2) &= \mathbb{E}[r_1(s_0, a_1, a_2) + \gamma V_1(s', \pi_1^*, \pi_2^*) | s_t = s, a_t = (a_1, \dots, a_n)] \\
&= r_1(s_0, a_1, a_2) + \gamma \sum_{s'} p(s' | s_0, a_1, a_2) V_1(s', \pi_1^*, \pi_2^*)
\end{aligned}
$$

$$
\begin{aligned}
Q_1^*(s_0 = (0,2), Right, Up\,) &= 0 + 0.99 \times V_1(s' = (1,5), \pi_1^*, \pi_2^*) \\
&= 0 + 0.99 \times \{0 + 0.99 \times 0 + 0.99^2 \times 100\} = 97.0
\end{aligned}
$$

$s_0 = (0,2)$ $\qquad\qquad$ $s' = (1,5)$



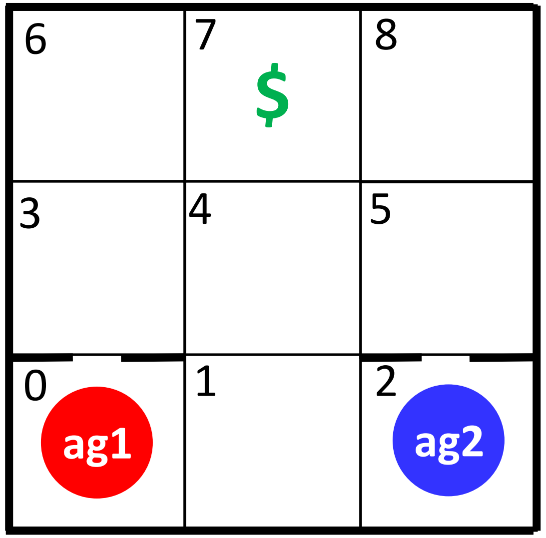
Grid Game 1 represented as stochastic game

Nash Q values for the initial state $s_0 = (0,2)$

	$a_2 = Left$	$a_2 = Up$
$a_1 = Right$	$Q_1^*(s_0, R, L), Q_2^*(s_0, R, L)$	$Q_1^*(s_0, R, U), Q_2^*(s_0, R, U)$
$a_2 = Up$	$Q_1^*(s_0, U, L), Q_2^*(s_0, U, L)$	$Q_1^*(s_0, U, U), Q_2^*(s_0, U, U)$

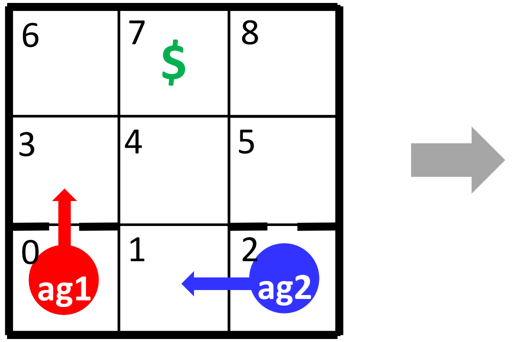
	$a_2 = Left$	$a_2 = Up$
$a_1 = Right$	$95.1, 95.1$	$97.0, 97.0$
$a_2 = Up$	$97.0, 97.0$	$97.0, 97.0$

Grid Game 2

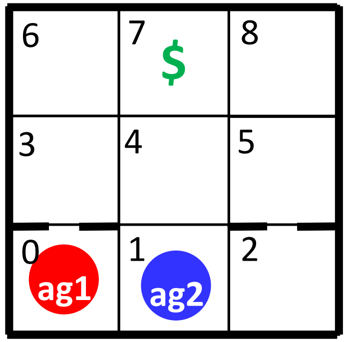


- First to reach goal gets $100
- If both reaches the money at the same time, both win
- Semi wall (50% go through)
- Cannot occupy the same grid

- Grid game has both **stochastic** and **deterministic** moves
- If agent choses Up from position 0 or 2, it moves up with probability 0.5 and remains in its previous position with probability 0.5

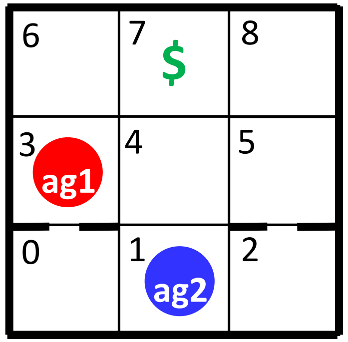


$p((0{,}1)|(0{,}2), Up, Left) = 0.5$

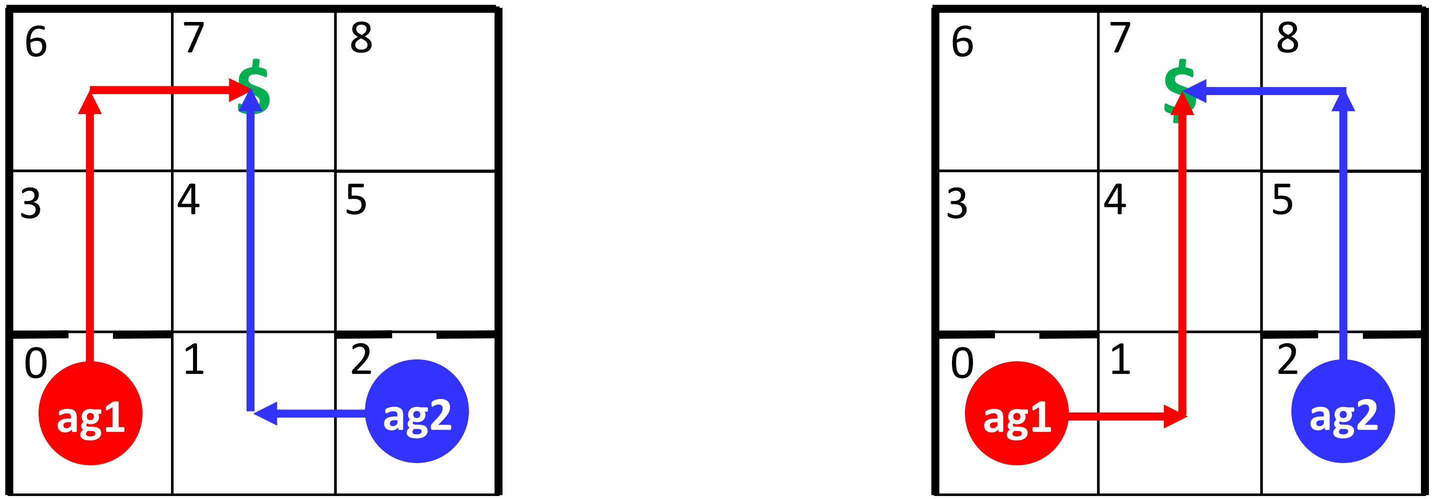


&

$p((3{,}1)|(0{,}2), Up, Left) = 0.5$



Grid Game 2



- There are two Nash equilibrium paths
- The value of the game for agent 1 is defined as its accumulated reward when both agents follow their Nash equilibrium strategies $\pi^* = (\pi_1^*, \ldots, \pi_n^*,)$:

$$V_1(s, \pi_1^*, \pi_2^*) = \sum_{t=0}^{\infty} \gamma^t E\left[r_{i,t} | \pi_1^*, \ldots, \pi_n^*, s_0 = s\right]$$

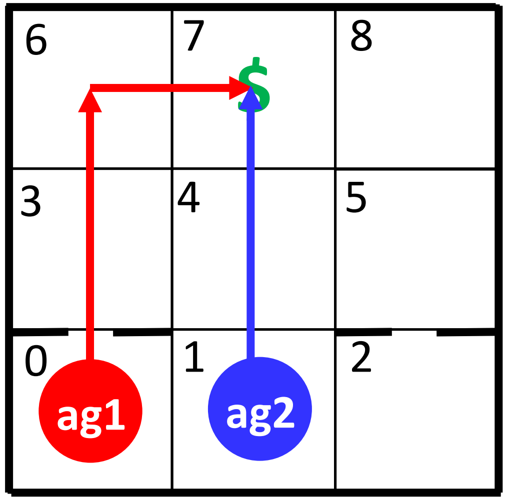
  - $V_1((0,1), \pi_1^*, \pi_2^*) = 0 + 0.99 \times 0 + 0.99^2 \times 0 = 0$
  - $V_1((0,x), \pi_1^*, \pi_2^*) = 0 \text{ for } x = 3,..,8$
  - $V_1((1,2), \pi_1^*, \pi_2^*) = 0 + 0.99 \times 100 = 99$
  - $V_1((1,3), \pi_1^*, \pi_2^*) = 0 + 0.99 \times 0 + 0.99^2 \times 0 = 0$
  - $V_1((1,x), \pi_1^*, \pi_2^*) = 0 + 0.99 \times 0 + 0.99^2 \times 0 = 0$

Grid Game 2

- The value of the game for agent 1 is defined as its accumulated reward when both agents follow their Nash equilibrium strategies $\pi^* = (\pi_1^*, \dots, \pi_n^*,)$:

$$V_1(s, \pi_1^*, \pi_2^*) = \sum_{t=0}^{\infty} \gamma^t E\left[r_{i,t} | \pi_1^*, \dots, \pi_n^*, s_0 = s\right]$$

  - $V_1((0,1), \pi_1^*, \pi_2^*) = 0 + 0.99 \times 0 + 0.99^2 \times 0 = 0$

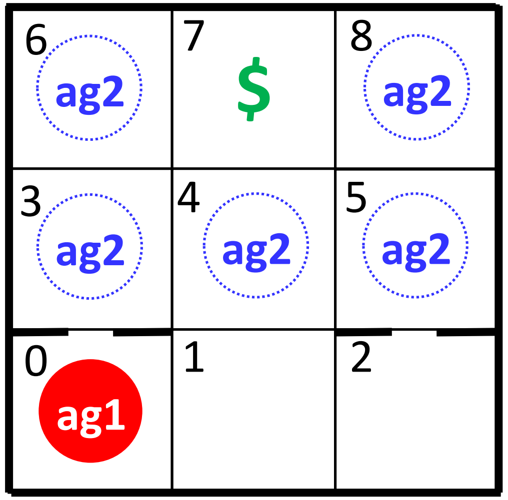


Grid Game 2

- The value of the game for agent 1 is defined as its accumulated reward when both agents follow their Nash equilibrium strategies $\pi^* = (\pi_1^*, \dots, \pi_n^*,)$:

$$V_1(s, \pi_1^*, \pi_2^*) = \sum_{t=0}^{\infty} \gamma^t E\left[r_{i,t} | \pi_1^*, \dots, \pi_n^*, s_0 = s\right]$$

  - $V_1((0, x), \pi_1^*, \pi_2^*) = 0 \quad \text{for} \;\; x = 3, .., 8$

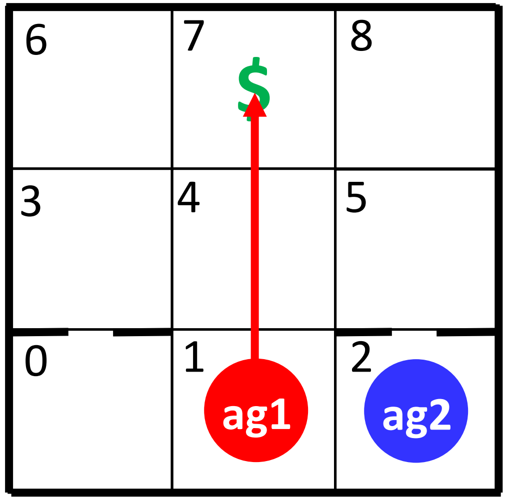


Grid Game 2

- The value of the game for agent 1 is defined as its accumulated reward when both agents follow their Nash equilibrium strategies $\pi^* = (\pi_1^*, \dots, \pi_n^*,)$:

$$V_1(s, \pi_1^*, \pi_2^*) = \sum_{t=0}^{\infty} \gamma^t E\left[r_{i,t} | \pi_1^*, \dots, \pi_n^*, s_0 = s\right]$$

  - $V_1((1,2), \pi_1^*, \pi_2^*) = 0 + 0.99 \times 100 = 99$

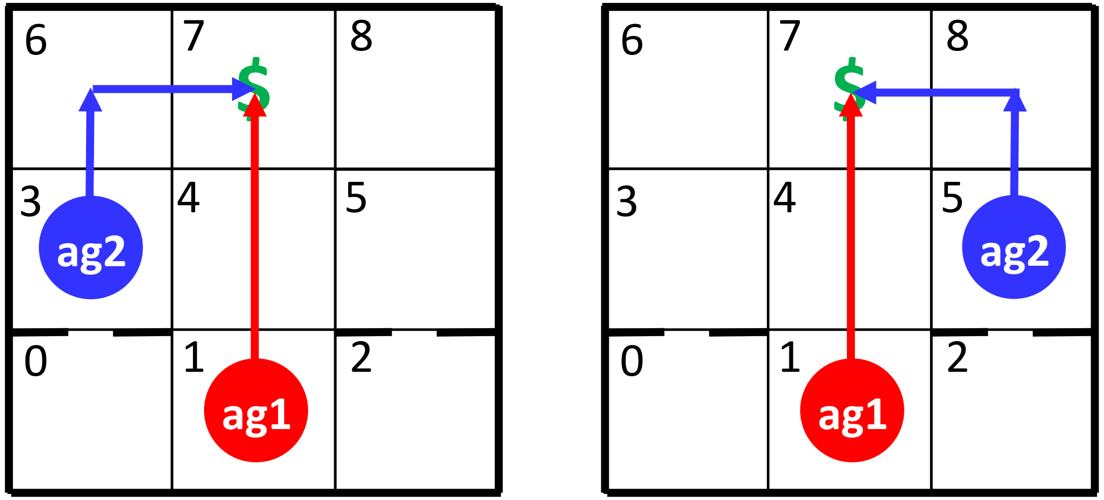


Grid Game 2

- The value of the game for agent 1 is defined as its accumulated reward when both agents follow their Nash equilibrium strategies $\pi^* = (\pi_1^*, \dots, \pi_n^*,)$:

$$V_1(s, \pi_1^*, \pi_2^*) = \sum_{t=0}^{\infty} \gamma^t E\left[r_{i,t} | \pi_1^*, \dots, \pi_n^*, s_0 = s\right]$$

- $V_1((1,3), \pi_1^*, \pi_2^*) = 0 + 0.99 \times 100 = 99 = V_1((1,5), \pi_1^*, \pi_2^*)$

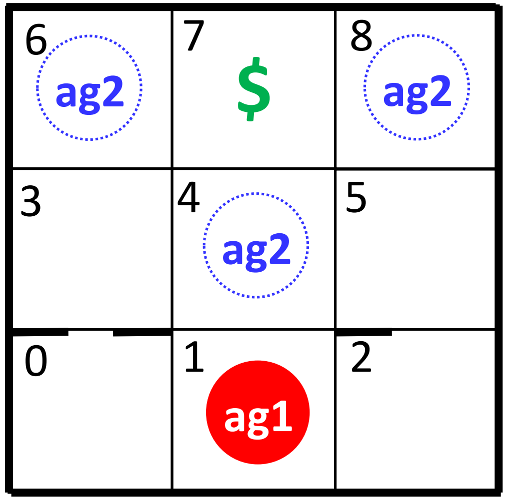


Grid Game 2

- The value of the game for agent 1 is defined as its accumulated reward when both agents follow their Nash equilibrium strategies $\pi^* = (\pi_1^*, \dots, \pi_n^*,)$:

$$V_1(s, \pi_1^*, \pi_2^*) = \sum_{t=0}^{\infty} \gamma^t E\left[r_{i,t} | \pi_1^*, \dots, \pi_n^*, s_0 = s\right]$$

  - $V_1((1, x), \pi_1^*, \pi_2^*) = 0 \quad \text{for} \quad x = 4, 6, 8$

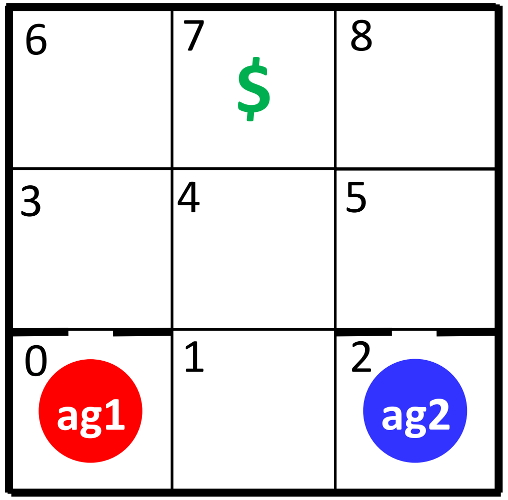


Grid Game 2

- The value of the game for agent 1 is defined as its accumulated reward when both agents follow their Nash equilibrium strategies $\pi^* = (\pi_1^*, \dots, \pi_{n,}^*)$:

$$V_1(s, \pi_1^*, \pi_2^*) = \sum_{t=0}^{\infty} \gamma^t E\left[r_{i,t} | \pi_1^*, \dots, \pi_n^*, s_0 = s\right]$$

  - $V_1((0,2), \pi_1^*, \pi_2^*) = V_1(s_0, \pi_1^*, \pi_2^*)$ can be computed only in expectation
  - We solve $V_1(s_0, \pi_1^*, \pi_2^*)$ from the state game $\left(Q_1^*(s_0, a_1, a_2), Q_2^*(s_0, a_1, a_2)\right)$



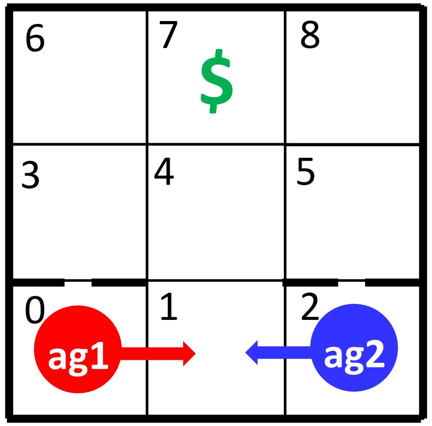
Grid Game 2

Nash Q values for the initial state $s_0 = (0,2)$

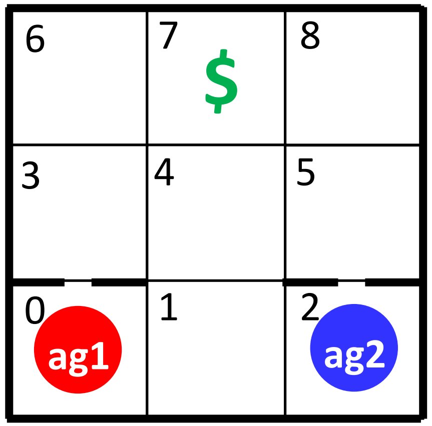
$$Q_1^*(s_0, a_1, a_2) = \mathbb{E}[r_1(s_0, a_1, a_2) + \gamma V_1(s', \pi_1^*, \pi_2^*) | s_t = s, a_t = (a_1, \dots, a_n)]$$
$$= r_1(s_0, a_1, a_2) + \gamma \sum_{s'} p(s' | s_0, a_1, a_2) V_1(s', \pi_1^*, \pi_2^*)$$

$$Q_1^*(s_0 = (0,2), Right, Left\ ) = -1 + 0.99 \times V_1(s_0, \pi_1^*, \pi_2^*)$$

$s_0 = (0,2)$



$s' = s_0 = (0,2)$



Grid Game 2

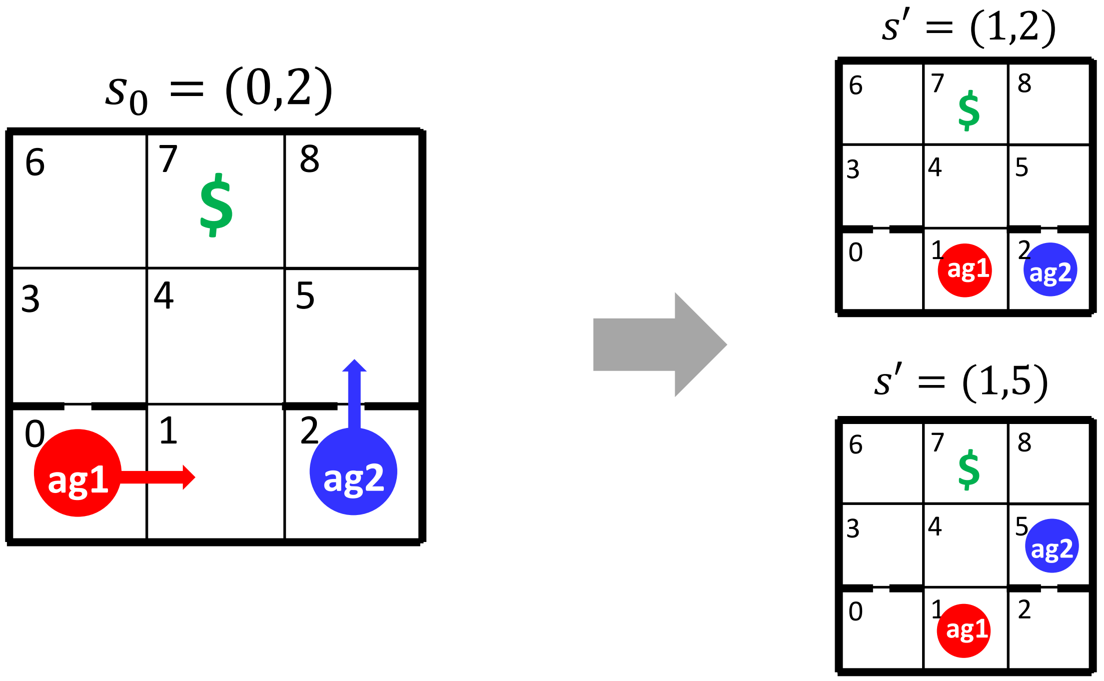
Nash Q values for the initial state $s_0 = (0,2)$

$$Q_1^*(s_0, a_1, a_2) = \mathbb{E}[r_1(s_0, a_1, a_2) + \gamma V_1(s', \pi_1^*, \pi_2^*) | s_t = s, a_t = (a_1, \dots, a_n)]$$

$$= r_1(s_0, a_1, a_2) + \gamma \sum_{s'} p(s' | s_0, a_1, a_2) V_1(s', \pi_1^*, \pi_2^*)$$

$$Q_1^*(s_0 = (0,2), Right, Up) = 0 + 0.99 \times \left\{\frac{1}{2} V_1\big((1,2), \pi_1^*, \pi_2^*\big) + \frac{1}{2} V_1\big((1,5), \pi_1^*, \pi_2^*\big)\right\}$$

$$= 0 + 0.99 \times (0.5 \times 99 + 0.5 \times 99) = 98$$



Grid Game 2

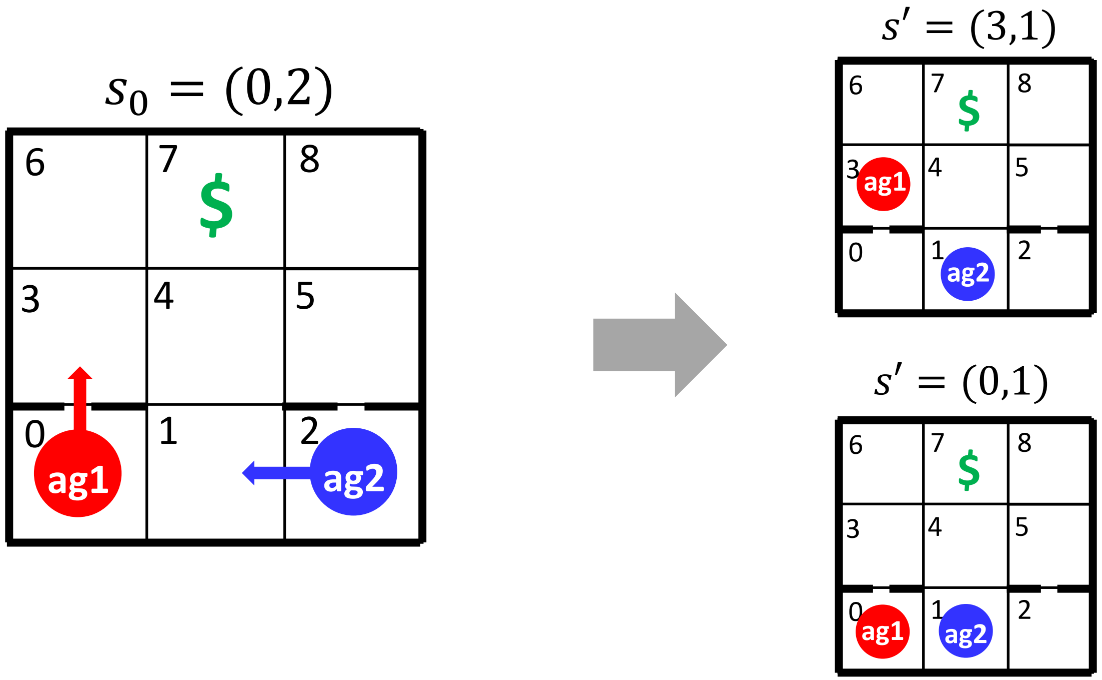
Nash Q values for the initial state $s_0 = (0,2)$

$$Q_1^*(s_0, a_1, a_2) = \mathbb{E}[r_1(s_0, a_1, a_2) + \gamma V_1(s', \pi_1^*, \pi_2^*) | s_t = s, a_t = (a_1, \dots, a_n)]$$

$$= r_1(s_0, a_1, a_2) + \gamma \sum_{s'} p(s' | s_0, a_1, a_2) V_1(s', \pi_1^*, \pi_2^*)$$

$$Q_1^*(s_0 = (0,2), Up, Left) = 0 + 0.99 \times \left\{ \frac{1}{2} V_1((3,1), \pi_1^*, \pi_2^*) + \frac{1}{2} V_1((0,1), \pi_1^*, \pi_2^*) \right\}$$

$$= 0 + 0.99 \times (0.5 \times 99 + 0.5 \times 0) = 49$$



Grid Game 2

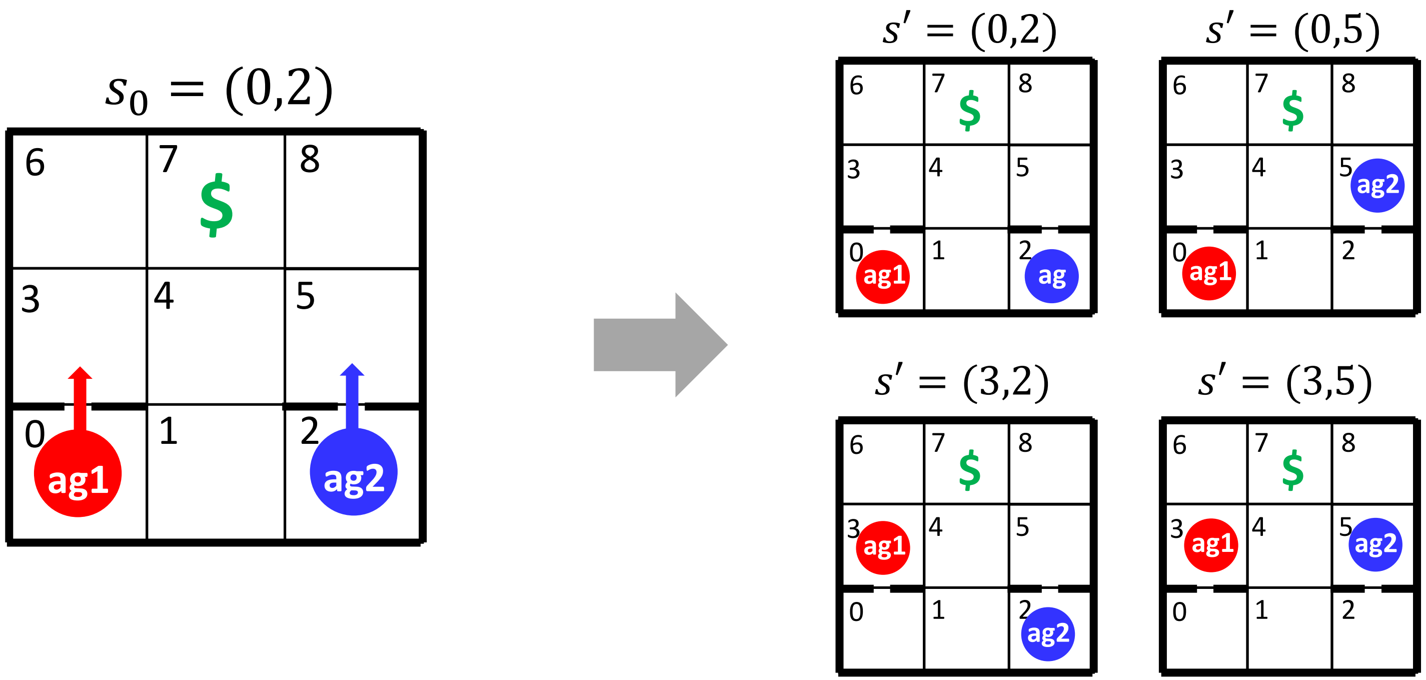
Nash Q values for the initial state $s_0 = (0,2)$

$$Q_1^*(s_0, a_1, a_2) = \mathbb{E}[r_1(s_0, a_1, a_2) + \gamma V_1(s', \pi_1^*, \pi_2^*) | s_t = s, a_t = (a_1, \dots, a_n)]$$

$$= r_1(s_0, a_1, a_2) + \gamma \sum_{s'} p(s' | s_0, a_1, a_2) V_1(s', \pi_1^*, \pi_2^*)$$

$$Q_1^*(s_0 = (0,2), Up, Up) = 0 + 0.99 \times \left\{ \frac{1}{4} V_1^*((0,2)) + \frac{1}{4} V_1^*((0,5)) + \frac{1}{4} V_1^*((3,2)) + \frac{1}{4} V_1^*((3,5)) \right\}$$

$$= 0 + 0.99 \times \left\{ \frac{1}{4} V_1^*(s_0) + \frac{1}{4} \times 0 + \frac{1}{4} \times 99 + \frac{1}{4} \times 99 \right\} = 0.99 \times \frac{1}{4} V_1^*(s_0) + 49$$



Nash Q values for the initial state $s_0 = (0,2)$

	$a_2 = Left$	$a_2 = Up$
$a_1 = Right$	$Q_1^*(s_0, R, L), Q_2^*(s_0, R, L)$	$Q_1^*(s_0, R, U), Q_2^*(s_0, R, U)$
$a_2 = Up$	$Q_1^*(s_0, U, L), Q_2^*(s_0, U, L)$	$Q_1^*(s_0, U, U), Q_2^*(s_0, U, U)$

	$a_2 = Left$	$a_2 = Up$
$a_1 = Right$	$-1 + 0.99V_1^*(s_0), -1 + 0.99V_2^*(s_0)$	$98, 49$
$a_2 = Up$	$49, 98$	$49 + \frac{0.99}{4}V_1^*(s_0), 49 + \frac{0.99}{4}V_2^*(s_0)$

Grid Game 2

Nash Q values for the initial state $s_0 = (0,2)$

	$a_2 = Left$	$a_2 = Up$
$a_1 = Right$	$-1 + 0.99V_1^*(s_0), -1 + 0.99V_2^*(s_0)$	$98, 49$
$a_2 = Up$	$\boxed{49, 98}$	$49 + \frac{0.99}{4}V_1^*(s_0), 49 + \frac{0.99}{4}V_2^*(s_0)$

$$V_1^*(s_0) = \text{Nash}\{Q_1^*(s_0, a_1, a_2), Q_2^*(s_0, a_1, a_2)\}$$

Case 1: $V_1^*(s_0) = 49$

	$Left$	Up
$Right$	$47, 96$	$98, 49$
Up	$49, 98$	$61, 73$

Grid Game 2

Nash Q values for the initial state $s_0 = (0,2)$

	$a_2 = Left$	$a_2 = Up$
$a_1 = Right$	$-1 + 0.99V_1^*(s_0), -1 + 0.99V_2^*(s_0)$	$\boxed{98, 49}$
$a_2 = Up$	$49, 98$	$49 + \frac{0.99}{4}V_1^*(s_0), 49 + \frac{0.99}{4}V_2^*(s_0)$

$$V_1^*(s_0) = \text{Nash}\{Q_1^*(s_0, a_1, a_2), Q_2^*(s_0, a_1, a_2)\}$$

Case 2: $V_1^*(s_0) = 98$

	$Left$	Up
$Right$	$96, 47$	$98, 49$
Up	$49, 98$	$73, 61$

Grid Game 2

Nash Q values for the initial state $s_0 = (0,2)$

	$a_2 = Left$	$a_2 = Up$
$a_1 = Right$	$-1 + 0.99V_1^*(s_0), -1 + 0.99V_2^*(s_0)$	$98, 49$
$a_2 = Up$	$49, 98$	$49 + \frac{0.99}{4}V_1^*(s_0), 49 + \frac{0.99}{4}V_2^*(s_0)$

$$V_1^*(s_0) = \text{Nash}\{Q_1^*(s_0, a_1, a_2), Q_2^*(s_0, a_1, a_2)\}$$

Case 3: $\{\pi_1(s_0), \pi_2(s_0)\} = (\{p(R) = 0.97, p(U) = 0.03\}, \{p(L) = 0.97, p(U) = 0.03\})$

	$Left$	Up
$Right$	$47.48, 47.48$	$98, 49$
Up	$49, 98$	$61.2, 61.2$

Optimal Q-function v.s. Nash Q-function

Definition (Optimal Q-function)

Optimal Q function is defined as

$$Q^*(s,a) = r(s,a,s') + \gamma \sum_{s' \in S} p(s'|s,a)V^*(s')$$

- $V^*(s') = \max_a Q^*(s',a)$
- With **optimum** policy $\pi^*(s) = \operatorname{argmax}_a Q^*(s,a)$

Definition (Nash Q-function)

Nash-Q function is defined as

$$Q_i^*(s,\vec{a}) = r_i(s,\vec{a},s') + \gamma \sum_{s' \in S} p(s'|s,\vec{a}) \underbrace{V_i^*(s')}_{Nash\, Q_i(s')}$$

- $V_i^*(s') = \text{Nash } Q_i^*(s')$ is Nash equilibrium value that can be computed by solving the following state game

$$(Q_1^*(s',\vec{a}), \dots, Q_n^*(s',\vec{a}))$$

Definition (Nash equilibrium policy in Stochastic game)

Compute the Nash equilibrium policies $\pi^* = (\pi_1^*, \pi_2^*)$ such that for all $s \in S$ and $i = 1, \dots 2$,

$$V_i(s, \pi_i^*, \pi_{-i}^*) \geq V_i(s, \pi_i^*, \pi_{-i}^*) \text{ for all } \pi_i \in \Pi_i$$

Value Function Based Multi-agent Reinforcement Learning

Multi Agent Q-learning Template

MultiQ(StochastiGame, f, γ, α, T)

Inputs equilibrium selection function f

discounting factor γ

learning rate α

total training time T

Outputs state − value functions V_i^*

action − value functions Q_i^*

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

for $t = 1:T$

1. simulate actions $\vec{a} = (a_1, \ldots, a_n)$in state s
2. observe rewards $r_1, \ldots, r_n$ and next state s'
3. for $i = 1$ to n (for each agent)
   (a) $V_i(s') = f_i(Q_1(s', \vec{a}), \ldots, Q_n(s', \vec{a}))$
   (b) $Q_i(s, \vec{a}) = (1 - \alpha_i)Q_i(s, \vec{a}) + \alpha_i[r_i + \gamma V_i(s')]$
4. agent choose actions $a_1', \ldots, a_n'$
5. $s = s', a_1 = a_1', \ldots, a_n = a_n'$
6. adjust learning rate $\alpha = (\alpha_1, \ldots, \alpha_n)$

Multi Agent Q-learning Template

Equilibrium selection function f : $V_i(s') = f_i(Q_1(s', \vec{a}), \ldots, Q_n(s', \vec{a}))$

- We going to study the following equilibrium concept:
  - Value function based (Bellman function based)
    - Single agent Q-learning
    - Independent Q learning by multiple agents
    - Nash-Q learning (Hu and Wellman 1998)
    - Minmax-Q learning (Littman 1994)
    - Friend-or-Foe Q learning (Littman 2001)
    - Correlated Q learning (Greenwald and Hall 2003)
  - Policy gradient methods (direct search for policy)
    - Wind-or-Learn-Fast Policy Hill Climbing (WOLF-PHC) (Policy gradient method)

Single agent Q learning

$$\text{(a) } V(s') = f\big(Q(s',a)\big) = \max_a Q(s',a)$$

$$\text{(b) } Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r + \gamma V(s')]$$

$$= (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_a Q(s',a)\right]$$

- Equivalent to Q-learning algorithm we have discussed couple weeks a go

Independent Q learning by multiple agents

$$\text{(a) } V_i(s') = f_i\big(Q_1(s',a_1),\dots,Q_i(s',a_i),\dots,Q_n(s',a_n)\big) = \max_{a_i} Q_i(s',a_i)$$

$$\text{(b) } Q_i(s,a_i) = (1-\alpha)Q_i(s,a_i) + \alpha[r_i + \gamma V_i(s')]$$

$$= (1-\alpha)Q_i(s,a_i) + \alpha\left[r_i + \gamma \max_{a_i} Q_i(s',a_i)\right]$$

- There are n agents whose Q-table is being independently updated regardless of the actions taken by other users
  - $Q_i(s',a_i) \sim Q_i(s',a_1,\dots,a_n)$
- Still the transition of joint state s depends on the all the actions taken by all agents, i.e., $p(s'|s,a_1,\dots a_i,\dots,a_n)$
  - Independent Q-learning thus ignore the effects of other agents' actions on state transition
    - ➢ treats other agents as a part of stochastic environment
    - ➢ Due to incomplete information on others' action, the agent cannot accurately learn the dynamic of the system

Nash-Q learning

Definition (Optimal Q-function)

Optimal Q function is defined as

$$Q^*(s,a) = r_i(s,a,s') + \gamma \sum_{s' \in S} p(s'|s,a) V_i^*(s')$$

- $V_i^*(s') = \max_a Q^*(s',a)$
- With **optimum** policy $\pi^*(s') = \operatorname{argmax}_a Q^*(s',a)$

Definition (Nash Q-function)

Nash-Q function is defined as

$$Q_i^*(s,a_1,\ldots,a_n) = r_i(s,a_1,\ldots,a_n,s') + \gamma \sum_{s' \in S} p(s'|s,a_1,\ldots,a_n) \underbrace{V_i(s',\pi_1^*,\ldots,\pi_n^*)}_{Nash\, Q_i(s')}$$

- $V_i(s',\pi_1^*,\ldots,\pi_n^*) = Q_i^*\big(s,\pi_1^*(s'),\ldots,\pi_n^*(s')\big) =$ Nash $Q_i(s')$
- with Nash **equilibrium** strategy $(\pi_1^*,\ldots,\pi_i^*,\ldots,\pi_n^*)$ satisfying for all s' and $i = 1,\ldots,n$

$$V_i(s',\pi_1^*,\ldots,\pi_i^*,\ldots,\pi_n^*) \geq V_i(s',\pi_1^*,\ldots,\pi_i,\ldots,\pi_n^*) \text{ for all } \pi_i \in \Pi_i$$

Nash-Q learning

- Q-learning directly find optimal Q-function (Q table) instead of optimum finding policy π^*

- Single agent Q-learning:
  - Iteratively find optimal Q values $Q^*(s,a)$ (table)

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r(s,a) + \gamma V(s')]$$
$$= (1-\alpha)Q(s,a) + \alpha\left[r(s,a) + \gamma \max_a Q(s',a)\right]$$

- Nah Q-learning:
  - Iteratively find Nash-Q values Nash $Q_i^*(s, a_1, \ldots, a_n)$ (table for each agent)

$$Q_i(s,\vec{a}) = (1-\alpha)Q_i(s,\vec{a}) + \alpha[r_i + \gamma V_i(s')]$$
$$= (1-\alpha)Q_i(s,\vec{a}) + \alpha[r_i + \gamma \text{ Nash } Q_i(s')]$$

$Q_i(s',\vec{a})$: Nash-Q values (state-action values)
Nash $Q_i(s')$: **Nash equilibrium value** of Nash-Q values

- the learning agent updates its Nash Q-value depending on the joint strategy of all the players and not only its own expected payoff.

The Nash Q-Learning algorithm

MultiQ(StochastiGame, f, γ, α, T)

Inputs equilibrium selection function f

discounting factor γ

learning rate α

total training time T

Outputs state – value functions V_i^*

action – value functions Q_i^*

Initialize $s, a_1, \dots, a_n$ and $Q_1, \dots, Q_n$

for $t = 1{:}T$

1. simulate actions $a_1, \dots, a_n$ in state s
2. observe rewards $r_1, \dots, r_n$ and next state s'
3. for $i = 1$ to n (for each agent)

   (a) $V_i(s') = f_i\big(Q_1(s', \vec{a}), \dots, Q_n(s', \vec{a})\big) = \mathrm{Nash}Q_i(s')$

   (b) $Q_i(s, \vec{a}) = (1 - \alpha_i)Q_i(s, \vec{a}) + \alpha_i[r_i + \gamma V_i(s')]$
4. agent choose actions $a_1', \dots, a_n'$
5. $s = s', a_1 = a_1', \dots, a_n = a_n'$
6. adjust learning rate $\alpha = (\alpha_1, \dots, \alpha_n)$

Nash-Q learning algorithm

For agent i

$$\text{(a) } V_i(s') = f_i\Big(\underbrace{Q_1(s',\vec{a}),\ldots,Q_i(s',\vec{a}),\ldots,Q_n(s',\vec{a})}_{\text{Nash Q values for agents } i\,=\,1:n}\Big) = \underbrace{\text{Nash } Q_i(s')}_{\text{Nash equilibrium value}}$$

$$\text{(b) } Q_i(s,\vec{a}) = (1-\alpha)Q_i(s,\vec{a}) + \alpha[r_i + \gamma V_i(s')]$$
$$= (1-\alpha)Q_i(s,\vec{a}) + \alpha[r_i + \gamma \text{ Nash } Q_i(s')]$$

- It uses the principle of the Nash equilibrium where "each player effectively holds a correct expectation about the other players' behaviors, and acts rationally with respect to this expectation
  - "rationally" means that the agent will have a strategy that is a best response for the other players' strategies.
- Nash Q-Learning is more complex than multiagent Q-learning because each player needs to keep track of the other players actions and rewards.
  - Each agent needs to track Nash Q values $Q_1(s',\vec{a}),\ldots,Q_n(s',\vec{a})$ for agents $i = 1{:}n$ to compute the Nash equilibrium value Nash $Q_i(s')$

How to compute Nash $Q_i(s')$?

1. At state s', agent i have the n Nash Q-values being tracked

$$\{Q_1(s',\vec{a}),\dots,Q_i(s',\vec{a}),\dots,Q_n(s',\vec{a})\}$$

   - In each state, the agent has to keep track of **every other agent's actions and rewards**.

2. Find the Nash equilibrium for the stage game $\{Q_1(s',\vec{a}),\dots,Q_i(s',\vec{a}),\dots,Q_n(s',\vec{a})\}$

   - For example, in two player general sum game, the agent i will build the matrix game for state s' with the reward matrix of both players as shown

	a_2^1	a_2^2
a_1^1	$Q_1(s',a_1^1,a_2^1),Q_2(s',a_1^1,a_2^1)$	$Q_1(s',a_1^1,a_2^2),Q_2(s',a_1^1,a_2^2)$
a_1^2	$Q_1(s',a_1^2,a_2^1),Q_2(s',a_1^2,a_2^1)$	$Q_1(s',a_1^2,a_2^2),Q_2(s',a_1^2,a_2^2)$

3. Compute the Nash equilibrium $\vec{a}_{NE}$ for the stage game (i.e., greedy optimization in single agent Q learning) and compute the Nash equilibrium value Nash $Q_i(s')$ for player i at state s'

$$\text{Nash } Q_i(s') = Q_i(s',\vec{a}_{NE})$$

How to compute Nash $Q_i(s')$?

4. Update Nash Q-values using the computed Nash equilibrium values

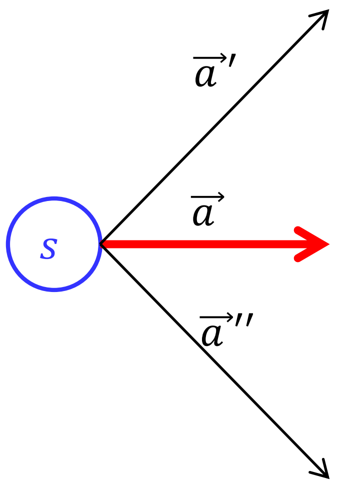
	a_2^1	a_2^2
a_1^1	$Q_1(s', a_1^1, a_2^1), Q_2(s', a_1^1, a_2^1)$	$Q_1(s', a_1^1, a_2^2), Q_2(s', a_1^1, a_2^2)$
a_1^2	$Q_1(s', a_1^2, a_2^1), Q_2(s', a_1^2, a_2^1)$	$Q_1(s', a_1^2, a_2^2), Q_2(s', a_1^2, a_2^2)$

$$Q_1(s, a_1, a_2) = (1 - \alpha) Q_1(s, a_1, a_2) + \alpha [r_1 + \gamma \text{ Nash } Q_1(s')]$$

$$Q_2(s, a_1, a_2) = (1 - \alpha) Q_2(s, a_1, a_2) + \alpha [r_2 + \gamma \text{ Nash } Q_2(s')]$$

Nash-Q learning : Procedure

S_t A_t R_{t+1} S_{t+1} $NashA_{t+1}$ S_{t+2}

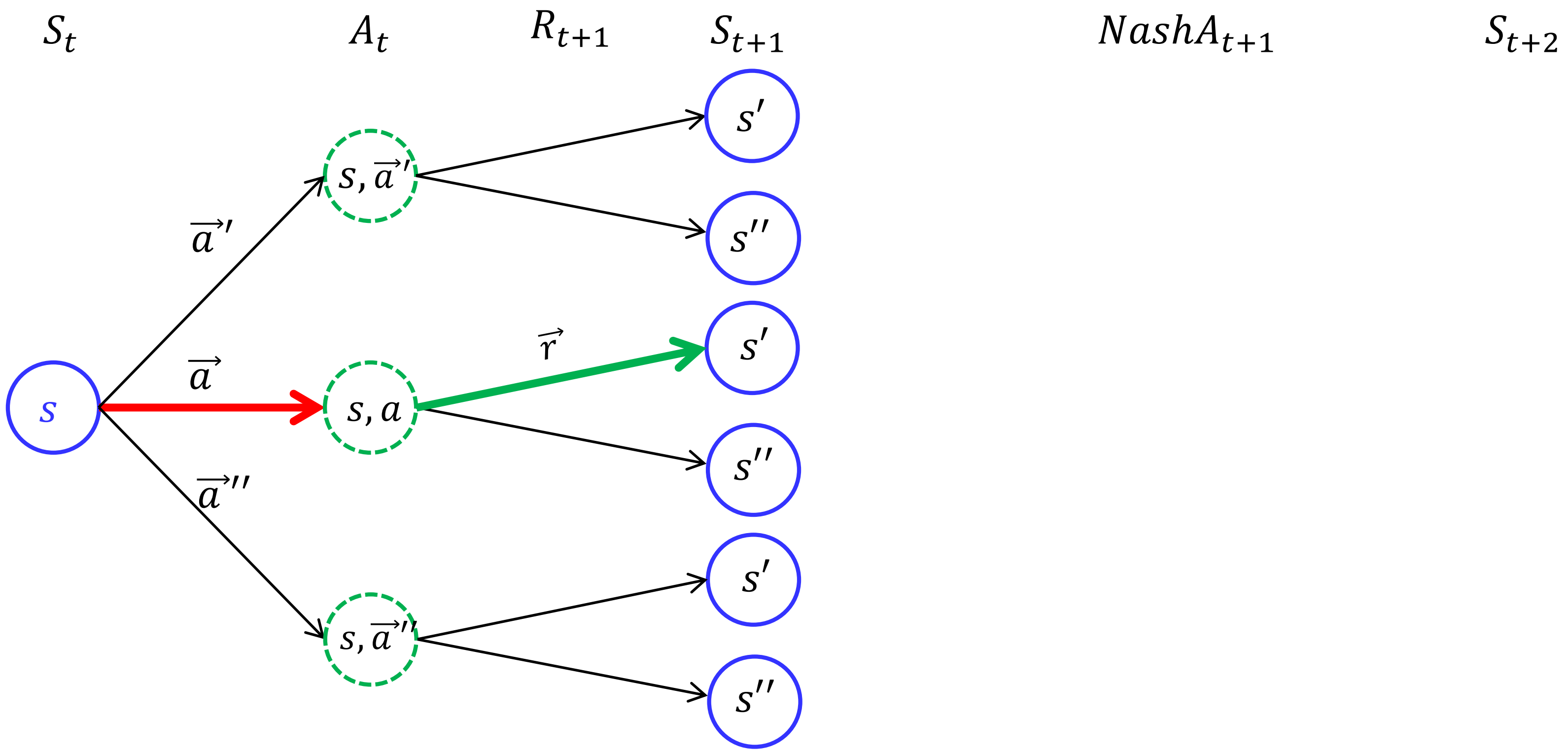


- Choose action $\vec{a} = (a_1, \dots, a_n)$ from s using current Nash Q values $(Q_1(s,\vec{a}), \dots, Q_n(s,\vec{a}))$

$$\vec{a} = \begin{cases} \vec{a}_{\mathrm{NE}} \text{ for } (Q_1(s,\vec{a}), \dots, Q_n(s,\vec{a})) & \text{with prob } 1-\epsilon \\ \text{random action} & \text{with prob } \epsilon \end{cases}$$

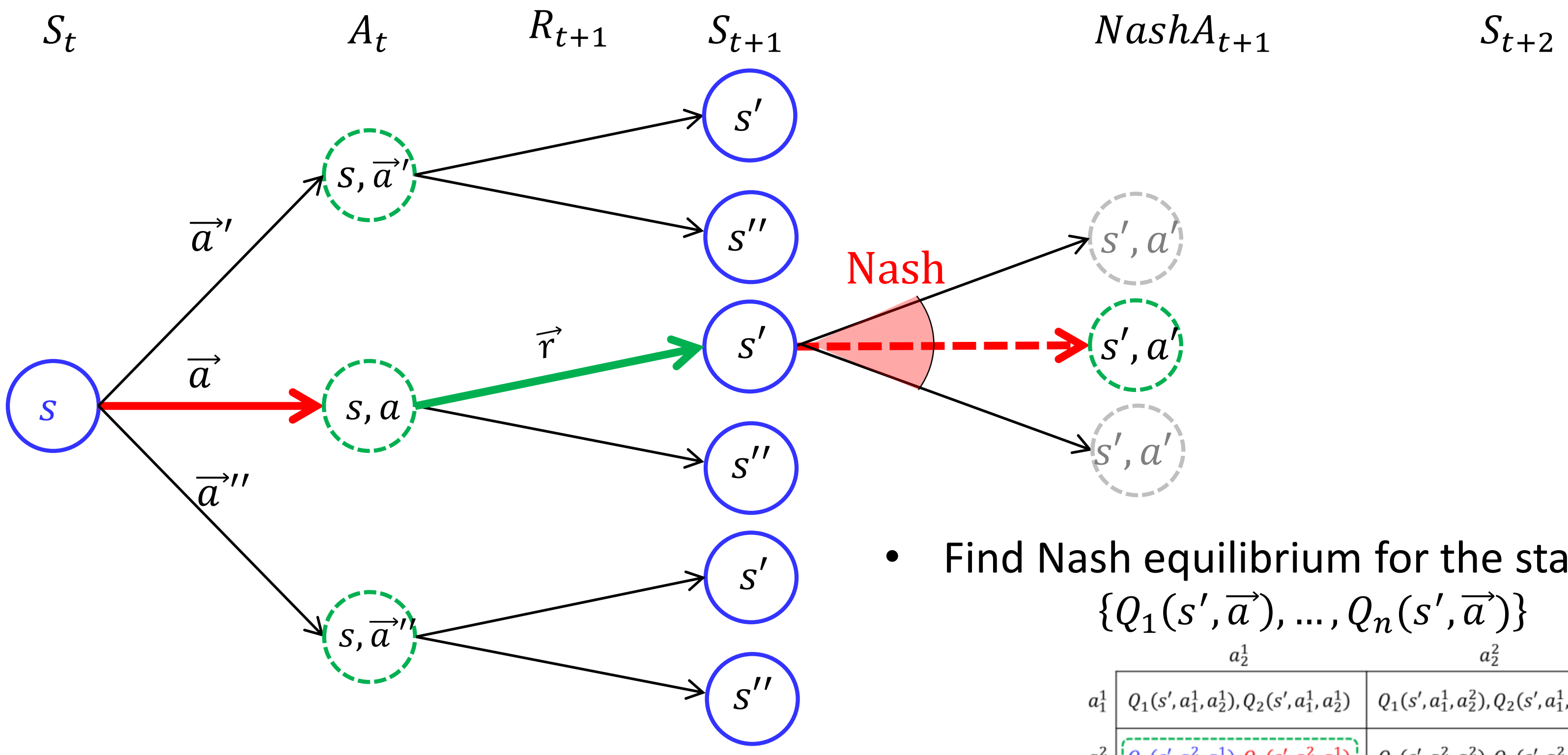
Any exploration policy can be used

Nash-Q learning : Procedure



- Take action$\vec{a} = (a_1, \dots, a_n)$ given s and observe reward $\vec{r} = (r_1, \dots, r_n)$ and the next state s'

Nash-Q learning : Procedure



- Find Nash equilibrium for the stage game

  $\{Q_1(s', \vec{a}), \dots, Q_n(s', \vec{a})\}$

	a_2^1	a_2^2
a_1^1	$Q_1(s', a_1^1, a_2^1), Q_2(s', a_1^1, a_2^1)$	$Q_1(s', a_1^1, a_2^2), Q_2(s', a_1^1, a_2^2)$
a_1^2	$Q_1(s', a_1^2, a_2^1), Q_2(s', a_1^2, a_2^1)$	$Q_1(s', a_1^2, a_2^2), Q_2(s', a_1^2, a_2^2)$

- The Nash equilibrium values are

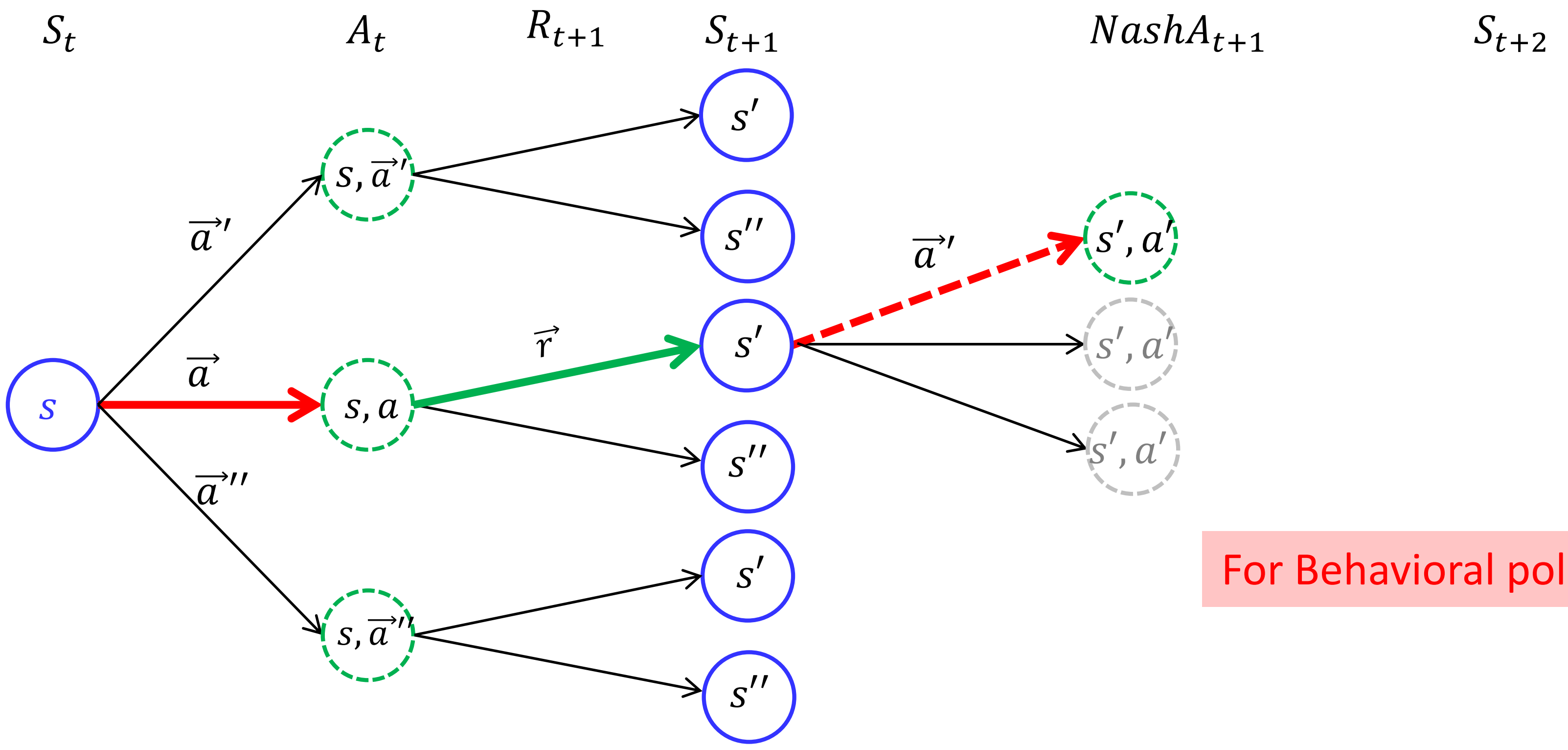
  Nash $Q_1(s')$, … , Nash $Q_n(s')$

- Update Nash-Q values, $Q_1(s, \vec{a}), \dots, Q_n(s, \vec{a})$, using the Nash equilibrium values

  For $i = 1{:}n$

  $$Q_i(s, \vec{a}) \leftarrow (1 - \alpha) Q_i(s, \vec{a}) + \alpha [r_i + \gamma \text{ Nash } Q_i(s')]$$

Nash-Q learning : Procedure



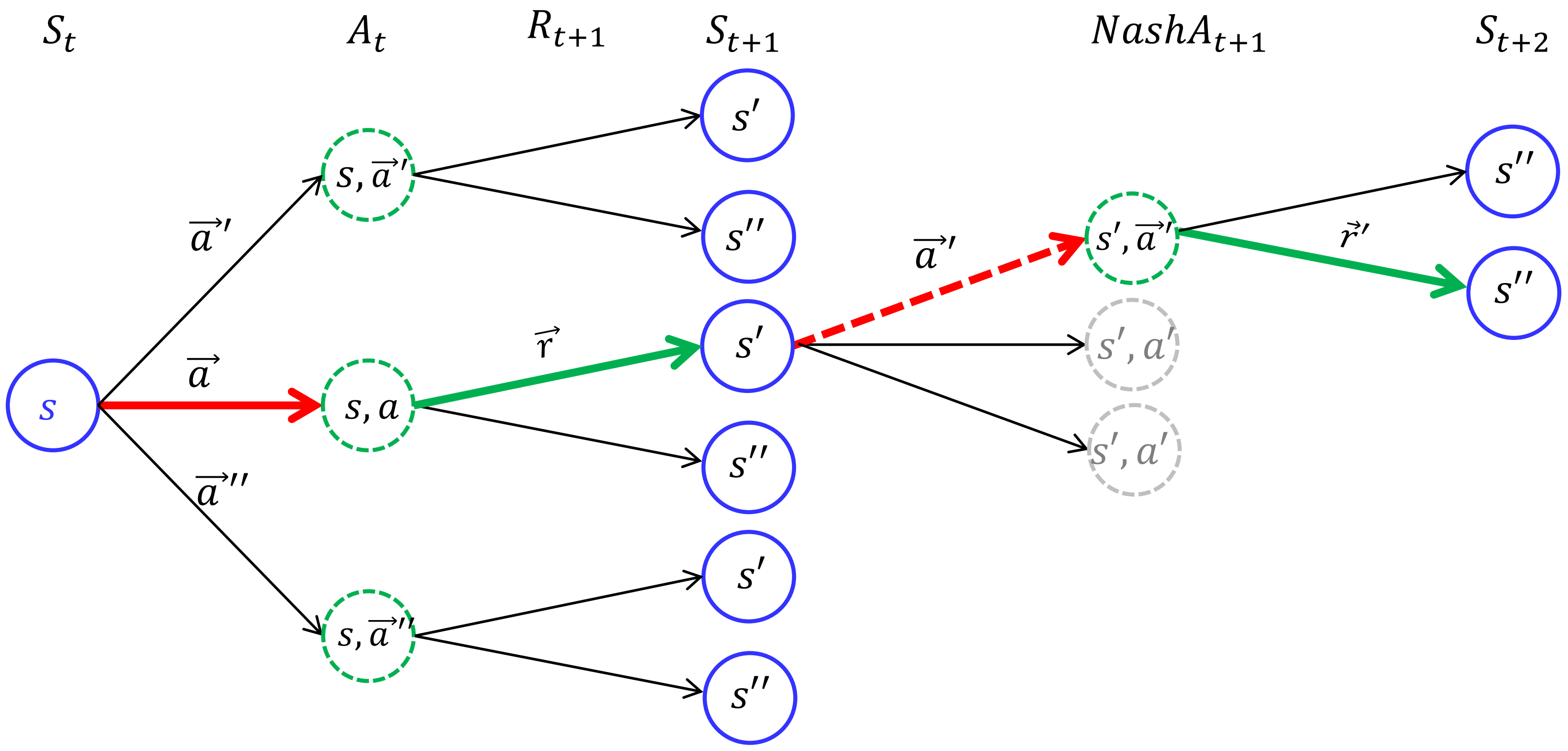
For Behavioral policy

- Choose action $\vec{a} = (a_1, \dots, a_n)$ from s using current Nash Q values $(Q_1(s', \vec{a}), \dots, Q_n(s', \vec{a}))$

$$\vec{a} = \begin{cases} \vec{a}_{\mathrm{NE}} \text{ for } (Q_1(s', \vec{a}), \dots, Q_n(s', \vec{a})) & \text{with prob } 1 - \epsilon \\ \text{random action} & \text{with prob } \epsilon \end{cases}$$

Any exploration policy can be used

Nash-Q learning : Procedure



- Take action $\vec{a}' = (a'_1, \ldots, a'_n)$ given s' and observe $\vec{r}' = (r'_1, \ldots, r'_n)$ and s''

Nash-Q learning: Convergence

The convergence of this Nash Q is based on three important assumptions:

- **Assumption 1**: Every state $s \in S$ and action $a_k \in A_k$ for $k = 1, \dots, n$, are visited infinitely often.

- **Assumption 2**: The learning rate α_t satisfies the following conditions for all $s, t, a_1, \dots, a_n$:
  - $0 \leq \alpha_t(s, a_1, \dots, a_n) < 1, \sum_t^{\infty} \alpha_t(s, a_1, \dots, a_n) = \infty, \sum_t^{\infty} [\alpha_t(s, a_1, \dots, a_n)]^2 < \infty$
  - $\alpha_t(s, a_1, \dots, a_n) = 0$ if $(s, a_1, \dots, a_n) \neq (s_t, a_1, \dots, a_n)$, meaning that agent will only update the Q-values for the present state and actions

- **Assumption 3**: One of the following conditions holds during learning:
  - Condition 1: Every stage game $(Q_1^t(s), \dots, Q_n^t(s))$, for all t and s, has a global optimal point, and agents' payoffs in this equilibrium are used to update their Q-functions
  - Condition 1: Every stage game $(Q_1^t(s), \dots, Q_n^t(s))$, for all t and s, has a saddle point, and agents' payoffs in this equilibrium are used to update their Q-functions

Minmax-Q learning

- The Minimax-Q algorithm was developed by Littman in 1994 when he adapted the value iteration method of Q-Learning from a single player to a two player zero sum game

- This is for a fully competitive game where players have opposite goals and reward functions ($R1 = -R2$).
  - Each agent tries to maximize its reward function while minimizing the opponent's.

Multi Agent Q-learning Template

MultiQ(StochastiGame, f, γ, α, T)

Inputs equilibrium selection function f

discounting factor γ

learning rate α

total training time T

Outputs state − value functions V_i^*

action − value functions Q_i^*

Initialize $s, a_1, \dots, a_n$ and $Q_1, \dots, Q_n$

for $t = 1{:}T$

1. simulate actions $\vec{a} = (a_1, \dots, a_n)$in state s
2. observe rewards $r_1, \dots, r_n$ and next state s'
3. for $i = 1$ to n (for each agent)
   (a) $V_i(s') = f_i(Q_1(s', \vec{a}), \dots, Q_n(s', \vec{a}))$
   (b) $Q_i(s, \vec{a}) = (1 - \alpha_i)Q_i(s, \vec{a}) + \alpha_i[r_i + \gamma V_i(s')]$
4. agent choose actions $a_1', \dots, a_n'$
5. $s = s', a_1 = a_1', \dots, a_n = a_n'$
6. adjust learning rate $\alpha = (\alpha_1, \dots, \alpha_n)$

Minmax-Q learning

For agent $i = 1{:}2$

(a) $V_i(s') = f_i(Q_1(s', a_i, a_{-i}), Q_2(s', a_i, a_{-i}),) = \max_{\pi_i(s',\cdot)} \min_{a_{-i} \in A_{-i}} \sum_{a_i \in A_i} Q_i(s', a_i, a_{-i}) \pi_i(s', a_i)$

$= \text{Maxmin}\, Q_i(s')$

(b) $Q_i(s, a_i, a_{-i}) = (1-\alpha) Q_i(s, a_i, a_{-i}) + \alpha[r_i + \gamma V_i(s')]$

$= (1-\alpha) Q_i(s, a_i, a_{-i}) + \alpha[r_i + \gamma\, \text{Maxmin}\, Q_i(s')]$

Action selection strategy:

$$\pi_i(s',\cdot) = \operatorname*{argmax}_{\pi_i(s',\cdot)} \min_{a_{-i} \in A_{-i}} \sum_{a_i \in A_i} Q_i(s', a_i, a_{-i}) \pi_i(s, a_i)$$

- Note that when computing the maxmin value, each agent can consider only its own action value function $Q_i(s', a_i, a_{-i})$
- But, still each agent need to track the action taken by the other agent for updating

Minmax-Q learning

For agent 1

(a) $V_1(s') = f_1(Q_1(s', a_1, a_2), Q_2(s', a_1, a_2),) = \max_{\pi_1(s',\cdot)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} Q_1(s', a_1, a_2)\pi_1(s', a_1)$

$= \text{Maxmin } Q_1(s')$

(b) $Q_1(s, a) = (1-\alpha)Q_1(s, a) + \alpha[r_1 + \gamma V_1(s')]$

$= (1-\alpha)Q_1(s, a) + \alpha[r_1 + \gamma \text{ Maxmin } Q_1(s')]$

For agent 2

(a) $V_2(s') = f_1(Q_1(s', a_1, a_2), Q_2(s', a_1, a_2),) = \max_{\pi_2(s',\cdot)} \min_{a_1 \in A_1} \sum_{a_2 \in A_2} Q_2(s', a_1, a_2)\pi_2(s', a_2)$

$= \text{Maxmin } Q_2(s')$

(b) $Q_2(s, a) = (1-\alpha)Q_2(s, a) + \alpha[r_1 + \gamma V_2(s')]$

$= (1-\alpha)Q_2(s, a) + \alpha[r_1 + \gamma \text{ Maxmin } Q_2(s')]$

Minmax-Q learning

- Because the property of a zero sum game, $Q_2(s', a_1, a_2) = -Q_1(s', a_1, a_2)$

$$\max_{\pi_2(s',\cdot)} \min_{a_1 \in A_1} \sum_{a_2 \in A_2} Q_2(s', a_1, a_2)\pi_2(s', a_2) = \max_{\pi_2(s',\cdot)} \min_{a_1 \in A_1} \sum_{a_2 \in A_2} -Q_1(s', a_1, a_2)\pi_2(s', a_2)$$

$$= \min_{\pi_2(s',\cdot)} \max_{a_1 \in A_1} \sum_{a_2 \in A_2} Q_1(s', a_1, a_2)\pi_2(s', a_2) = \text{minmax}\, Q_1(s')$$

- Therefore, player 2's Q-function can be updated using $Q_1(s, a)$

$$\text{(b) } Q_2(s, a) = (1-\alpha)Q_2(s, a) + \alpha[r_1 + \gamma V_2(s')]$$

$$= (1-\alpha)Q_2(s, a) + \alpha[r_1 + \gamma\, \text{Maxmin}\, Q_2(s')]$$

$$\text{(b) } -Q_1(s, a) = (1-\alpha)\{-Q_1(s, a)\} + \alpha[r_1 + \gamma V_2(s')]$$

$$= (1-\alpha)\{-Q_1(s, a)\} + \alpha[r_1 + \gamma\, \text{minmax}\, Q_1(s')]$$

- This result concludes that in Minmax-Q learning, we can only keep updating Q-function for player 1, $Q_1(s, a) \rightarrow Q(s, a)$

Minmax-Q learning

Updating rule for agent 1

(a) $$V(s') = f_1\big(Q(s', a_1, a_2)\big) = \max_{\pi_1(s',\cdot)} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} Q(s', a_1, a_2)\pi_1(s, a_1) = \text{Maxmin}\, Q(s')$$

(b) $$Q(s, a) = (1-\alpha)Q(s, a) + \alpha[r_1 + \gamma V(s')]$$

$$= (1-\alpha)Q_1(s, a) + \alpha[r_1 + \gamma\ \text{Maxmin}\, Q(s')]$$

Action selection rule for agent 1

$$\pi_1(s',\cdot) = \underset{\pi_1(s',\cdot)}{\operatorname{argmax}} \min_{a_2 \in A_2} \sum_{a_1 \in A_1} Q(s', a_1, a_2)\pi_1(s', a_1)$$

Updating rule for agent 2

Agent 2's Q function $Q_2(s, a)$=$-Q(s, a)$

Action selection rule for agent 2

$$\pi_2(s',\cdot) = \underset{\pi_2(s',\cdot)}{\operatorname{argmax}} \min_{a_1 \in A_1} \sum_{a_2 \in A_2} -Q(s', a_1, a_2)\pi_2(s, a_2)$$

$$= \underset{\pi_2(s',\cdot)}{\operatorname{argmin}} \max_{a_1 \in A_1} \sum_{a_2 \in A_2} Q(s', a_1, a_2)\pi_2(s, a_2)$$

Minmax-Q learning Algorithm

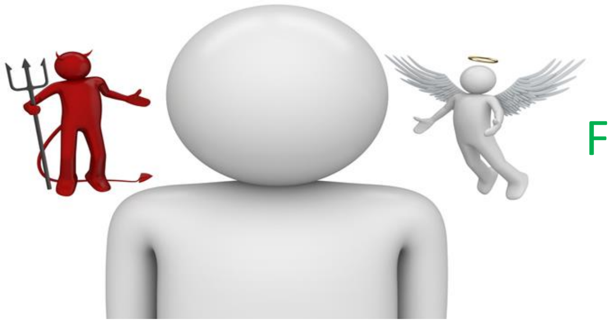
MultiQ(StochastiGame, f, γ, α, T)

Inputs	equilibrium selection function f
	discounting factor γ
	learning rate α
	total training time T
Outputs	state – value functions V_i^*
	action – value functions Q_i^*
Initialize	$s, a_1, \dots, a_n$ and $Q_1, \dots, Q_n$

for $t = 1 : T$

1. simulate actions $a_1, \dots, a_n$ in state s
2. observe rewards $r_1, \dots, r_n$ and next state s'
3. for $i = 1$ to n (for each agent)
   (a) $V_i(s') = f_i\big(Q_1(s', a), \dots, Q_n(s', a)\big) = \text{Minmax}Q_i(s')$
   (b) $Q_i(s, a) = (1 - \alpha_i)Q_i(s, a) + \alpha_i[r_i + \gamma V_i(s')]$
4. agent choose actions $a_1', \dots, a_n'$
5. $s = s', a_1 = a_1', \dots, a_n = a_n'$
6. adjust learning rate $\alpha = (\alpha_1, \dots, \alpha_n)$

Friend-or-For Q learning



- This algorithm was developed by Littman (1998) and tries to fix some of the convergence problems of Nash-Q Learning

- The main concern lies within assumption 3, where every stage game needs to have either a global optimal point or a saddle point.
  - ➢ These restrictions cannot be guaranteed during learning.

- To alleviate this restriction, this new algorithm is built to always converge by changing the update rules depending on the opponent.
  - ➢ The learning agent has to classify the other agent as “**friend**” or “**foe**”.
  - ➢ Player i’s friends are assumed to work together to maximize player i’s value
  - ➢ Player i’s foes are assumed to work together to minimize player i’s value

- Thus, n-player general-sum stochastic game can be treated as a two-player zero-sum game with an extended action set.

Multi Agent Q-learning Template

MultiQ(StochastiGame, f, γ, α, T)

Inputs	equilibrium selection function f
	discounting factor γ
	learning rate α
	total training time T
Outputs	state − value functions V_i^*
	action − value functions Q_i^*
Initialize	$s, a_1, \ldots, a_n$ and $Q_1, \ldots, Q_n$

for $t = 1{:}T$

1. simulate actions $\vec{a} = (a_1, \ldots, a_n)$in state s
2. observe rewards $r_1, \ldots, r_n$ and next state s'
3. for $i = 1$ to n (for each agent)

   (a) $V_i(s') = f_i(Q_1(s', \vec{a}), \ldots, Q_n(s', \vec{a}))$

   (b) $Q_i(s, \vec{a}) = (1 - \alpha_i)Q_i(s, \vec{a}) + \alpha_i[r_i + \gamma V_i(s')]$
4. agent choose actions $a_1', \ldots, a_n'$
5. $s = s', a_1 = a_1', \ldots, a_n = a_n'$
6. adjust learning rate $\alpha = (\alpha_1, \ldots, \alpha_n)$

Friend-or-For Q learning

For agent i

$$\text{(a) } V_i(s') = f_i\big(Q_1(s',\vec{a},\vec{o}), \dots, Q_i(s',\vec{a},\vec{o}), \dots, Q_n(s',\vec{a},\vec{o})\big)$$

$$= \max_{\pi_1(s',\cdot),\dots,\pi_{n_1}(s',\cdot)} \min_{o_1,\dots,o_{n_2} \in O_1 \times \cdots \times O_{n_2}} \sum_{a_i \in A_i} Q_i(s',\vec{a},\vec{o})\pi_1(s,a_1) \cdots \pi_{n_1}(s,a_{n_1})$$

$\vec{a} = (a_1, \dots, a_{n_1})$: actions for the friends agents

$\vec{o} = (o_1, \dots, o_{n_2})$: actions for the foe agents

$$\text{(b) } Q_i(s,\vec{a},\vec{o}) = (1-\alpha)Q_i(s,\vec{a},\vec{o}) + \alpha[r_i + \gamma V_i(s')]$$

$$= (1-\alpha)Q_i(s,\vec{a},\vec{o}) + \alpha[r_i + \gamma\, FoF\ Q_i(s')]$$

For two player case (described in terms of player 1)

(a) $V_1(s') = f_1\big(Q_1(s', a_1, a_2), Q_2(s', a_1, a_2)\big)$

$$= \begin{cases} \max\limits_{a_1 \in A_1, a_2 \in A_2} Q_1(s', a_1, a_2) & \text{If other player is fried:} \\ \max\limits_{\pi_1(s',\cdot)} \min\limits_{a_2 \in A_2} \sum\limits_{a_i \in A_i} Q_1(s', a_1, a_2)\pi_1(s, a_1) & \text{If other player is foe:} \end{cases}$$

(b) $Q_1(s, a_1, a_2) = (1-\alpha)Q_1(s, a_1, a_2) + \alpha[r_i + \gamma V_1(s')]$

$$= (1-\alpha)Q_1(s, a_1, a_2) + \alpha[r_i + \gamma FoFQ_1(s')]$$

Friend-or-For Q learning

$\mathrm{FoFQ}(\mathrm{StochastiGame}, f, \gamma, \alpha, T)$

Inputs	equilibrium selection function f = Friend or Foe
	discounting factor γ
	learning rate α
	total training time T
Outputs	state − value functions V_i^*
	action − value functions Q_i^*
Initialize	$s, a_1, \ldots, a_n$ and $Q_1, \ldots, Q_n$

for $t = 1\!:\!T$

1. simulate actions $\vec{a} = (a_1, \ldots, a_n)$in state s
2. observe rewards $r_1, \ldots, r_n$ and next state s'
3. for $i = 1$ to n (for each agent)

   (a) $V_i(s') = FoF(Q_1(s', \vec{a}), \ldots, Q_n(s', \vec{a}))$

   (b) $Q_i(s, \vec{a}) = (1 - \alpha_i)Q_i(s, \vec{a}) + \alpha_i[r_i + \gamma V_i(s')]$
4. agent choose actions $a_1', \ldots, a_n'$
5. $s = s', a_1 = a_1', \ldots, a_n = a_n'$
6. adjust learning rate $\alpha = (\alpha_1, \ldots, \alpha_n)$

- Correlated equilibrium

	Go	Wait
Go	-100, -100	10, 0
Wait	0, 10	-10, -10

Traffic game

- What is the natural solution here?
  - A traffic light: a fair randomizing device that tells one of the agents to go and the other to wait.
- Benefits:
  - the negative payoff outcomes are completely avoided
  - fairness is achieved
  - the sum of social welfare exceeds that of mixed Nash equilibrium

Correlated-Q learning

Definition

A joint probability distribution $\pi \in \Delta(A)$ is a correlated equilibrium of a finite game if and only if

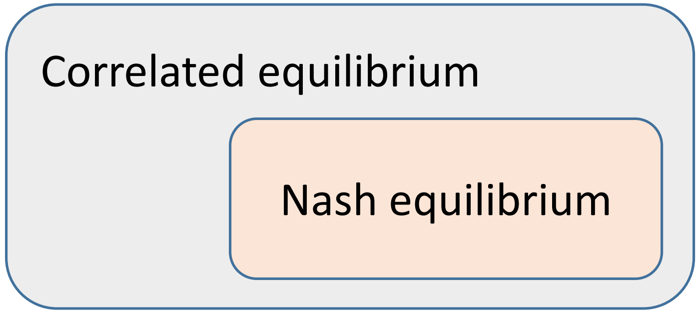
$$\sum_{a_{-i} \in A_{-i}} \pi(a) u_i(a_i, a_{-i}) \geq \sum_{a_{-i} \in A_{-i}} \pi(a) u_i(a_i', a_{-i})$$

For all players i, all $s_i \in S_i$, $t_i \in S_i$ such that $a_i' \in A_i$

Theorem (Correlated equilibrium)

For every Nash equilibrium σ^* there exists a corresponding correlated equilibrium σ

- Correlated equilibrium is a strictly weaker notion than Nash



Computing correlated equilibria : Example

	L	R
T	$6, 6$	$2, 8$
B	$8, 2$	$0, 0$

- Each correlated equilibrium corresponds to a probability distribution (a, b, c, d) over the possible pairs of actions, $\{(T, L), (T, R), (B, L), (B, R)\}$.
- The conditions needed to be correlated equilibrium, in addition to (a, b, c, d) being a probability distribution, are

$$(T \to B) \quad 6a + 2b \geq 8a + 0b$$

$$(B \to T) \quad 8c + 0d \geq 6c + 2d$$

$$(L \to R) \quad 6a + 2c \geq 8a + 0c$$

$$(R \to L) \quad 8b + 0d \geq 6b + 2d$$

  where, for example, the equation for $(T \to B)$ insures that the first player would not receive a higher expected payoff by using B whenever told to play T.
- The equations reduce to (a, b, c, d) is a probability vector such that $a \leq b, a \leq c. d \leq b,$ and $d \leq c$.

Multi Agent Q-learning Template

MultiQ(StochastiGame, f, γ, α, T)

Inputs	equilibrium selection function f
	discounting factor γ
	learning rate α
	total training time T
Outputs	state $-$ value functions V_i^*
	action $-$ value functions Q_i^*
Initialize	$s, a_1, \ldots, a_n$ and $Q_1, \ldots, Q_n$

for $t = 1{:}T$

1. simulate actions $\vec{a} = (a_1, \ldots, a_n)$in state s
2. observe rewards $r_1, \ldots, r_n$ and next state s'
3. for $i = 1$ to n (for each agent)
   (a) $V_i(s') = f_i(Q_1(s', \vec{a}), \ldots, Q_n(s', \vec{a}))$
   (b) $Q_i(s, \vec{a}) = (1 - \alpha_i)Q_i(s, \vec{a}) + \alpha_i[r_i + \gamma V_i(s')]$
4. agent choose actions $a_1', \ldots, a_n'$
5. $s = s', a_1 = a_1', \ldots, a_n = a_n'$
6. adjust learning rate $\alpha = (\alpha_1, \ldots, \alpha_n)$

Correlated-Q learning

For agent i

$$\text{(a) } V_i(s') = f_i\big(\underbrace{Q_1(s',\vec{a}), \dots, Q_i(s',\vec{a}), \dots, Q_n(s',\vec{a})}_{\text{Q values for agents } i\,=\,1:n}\big) = \underbrace{\text{CE } Q_i(s')}_{\text{Correlated equilibrium value}}$$

$$\text{(b) } Q_i(s,\vec{a}) = (1-\alpha)Q_i(s,\vec{a}) + \alpha[r_i + \gamma V_i(s')]$$

$$= (1-\alpha)Q_i(s,\vec{a}) + \alpha[r_i + \gamma \text{ CE } Q_i(s')]$$

Variants of Correlated-Q learning

How to compute $\mathrm{CE}\, Q_i(s')$?

1. First compute the Correlated equilibrium $\pi(s', \vec{a})$ by solving the following constrain satisfaction problem

$$\sum_{\vec{a} \in A | a_i \in \vec{a}} \pi(s, \vec{a}) Q_i(s', \vec{a}) \geq \sum_{\vec{a} \in A | a_i \in \vec{a}} \pi(s, \vec{a}) Q_i(s', a_i', a_{-i}), \forall i \in N, \forall a_i, a_i' \in A_i \quad (1)$$

$$\pi(s', \vec{a}) > 0, \ \forall \vec{a} \in A \quad (2)$$

$$\sum_{\vec{a} \in A} \pi(s', \vec{a}) = 1 \quad (3)$$

- Variables: $\pi(s', \vec{a})$, constants: $\{Q_i(s', \vec{a}), \ldots, Q_i(s', \vec{a}), \ldots, Q_n(s', \vec{a})\}$

2. With the correlated equilibrium strategy $\pi(s, \vec{a})$, compute the correlation equilibrium value $\mathrm{CE}\, Q_i(s')$ for player i at state s as

$$\mathrm{CE}\, Q_i(s') = \sum_{\vec{a} \in A} \pi(s', \vec{a}) Q_i(s', \vec{a})$$

Variants of Correlated-Q learning

- The difficulty in learning equilibria in Markov games stems from the equilibrium selection problem:
  - How can multiple agents select among multiple equilibria?
- We introduce four variants of correlated-Q learning, which determine a unique Eq.
  - Resolves the equilibrium selection problem with its respective choice of objective function

Variants of Correlated-Q learning

- Utilitarian equilibrium: an equilibrium which maximizes the sum of the expected payoffs of the players:

$$\sigma \in \underset{\sigma \in \mathrm{CE}}{\operatorname{argmax}} \sum_{i \in N} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a})$$

- Egalitarian equilibrium : an equilibrium which maximizes the minimum expected payoff of a player

$$\sigma \in \underset{\sigma \in \mathrm{CE}}{\operatorname{argmax}} \min_{i \in N} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a})$$

- Republican equilibrium : an equilibrium which maximizes the maximum expected payoff of a player

$$\sigma \in \underset{\sigma \in \mathrm{CE}}{\operatorname{argmax}} \max_{i \in N} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a})$$

- Libertarian i equilibrium: an equilibrium which maximizes the maximum of each individual player i's rewards: $\sigma = \prod_i \sigma^i$, where

$$\sigma_i \in \underset{\sigma \in \mathrm{CE}}{\operatorname{argmax}} \sum_{\vec{a} \in A} \sigma(\vec{a}) Q_i(s, \vec{a})$$

Correlated-Q learning

FoFQ(StochastiGame, f, γ, α, T)

Inputs equilibrium selection function f = Correlated eq.

discounting factor γ

learning rate α

total training time T

Outputs state − value functions V_i^*

action − value functions Q_i^*

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

for $t = 1{:}T$

1. simulate actions $\vec{a} = (a_1, \ldots, a_n)$in state s
2. observe rewards $r_1, \ldots, r_n$ and next state s'
3. for $i = 1$ to n (for each agent)

   (a) $V_i(s') = \mathrm{CE}(Q_1(s', \vec{a}), \ldots, Q_n(s', \vec{a}))$

   (b) $Q_i(s, \vec{a}) = (1 - \alpha_i) Q_i(s, \vec{a}) + \alpha_i [r_i + \gamma V_i(s')]$
4. agent choose actions $a_1', \ldots, a_n'$
5. $s = s', a_1 = a_1', \ldots, a_n = a_n'$
6. adjust learning rate $\alpha = (\alpha_1, \ldots, \alpha_n)$

Recall Policy Hill Climbing (PHC) for Repeated Game

- PHC algorithm has been discussed as a way to solve a repeated matrix game

Algorithm **Policy hill − climbing (PHC) algorithm for agent *i***

Initialize

learning rate $\alpha \in (0,1], \delta \in (0,1]$

discunt factor $\gamma \in (0,1)$

exploration rate ϵ

$Q_i(a_i) \leftarrow 0$ and $\pi_i(a_i) \leftarrow \frac{1}{|A_i|}\ \forall a_i \in A_i$

Repeat

(a) select an action a_i according to the straegy $\pi(a_i)$ with some exploration rate ϵ

(b) observe the immediate reward r_i

(c) update Q values:

$$Q_i(a_i) = (1-\alpha)Q_i(a_i) + \alpha\left(r_i + \gamma \max_{a_i'} Q_i(a_i')\right)$$

(d) Update the strategy $\pi_i(a_i)$ and constrain it to a legal probability distribution

$$\pi_i(a_i) = \pi_i(a_i) + \begin{cases} \delta & if\ a_i = \max_{a_i'} Q_i(a_i') \\ -\dfrac{\delta}{|A_i|-1} & otherwise \end{cases}$$

Policy Hill Climbing (PHC) for Stochastic Game

- We will expand the PHC algorithm so that it can be used for a general sum stochastic game

Algorithm Policy hill – climbing (PHC) algorithm for agent $\boldsymbol{i}$

Initialize

learning rate $\alpha \in (0,1], \delta \in (0,1]$

discunt factor $\gamma \in (0,1)$

exploration rate ϵ

$Q_i(s, a_i) \leftarrow 0$ and $\pi_i(s, a_i) \leftarrow \frac{1}{|A_i|}\ \forall a_i \in A_i$

Repeat

(a) select an action a_i according to the straegy $\pi(s, a_i)$ with some exploration rate ϵ

(b) observe the immediate reward r_i

(c) update Q values:

$$Q_i(s, a_i) = (1-\alpha)Q_i(s, a_i) + \alpha\left(r_i + \gamma \max_{a_i'} Q_i(s, a_i')\right)$$

(d) Update the strategy $\pi_i(s, a_i)$ and constrain it to a legal probability distribution

$$\pi_i(s, a_i) = \pi_i(s, a_i) + \begin{cases} \delta & if\ a_i = \max_{a_i'} Q_i(s, a_i') \\ -\frac{\delta}{|A_i| - 1} & otherwise \end{cases}$$

The WoLF-Policy Hill Climbing (PHC) for Stochastic Game

- The WoLF-PHC algorithm is an extension of the PHC algorithm
  - WoLF(win-or-learn-fast) allows variable learning rate → faster convergence

WoLF

(c) update Q values:

$$Q_i(s,a_i) = (1-\alpha)Q_i(s,a_i) + \alpha\left(r_i + \gamma \max_{a_i'} Q_i(s,a_i')\right)$$

update estimate of average policy $\bar{\pi}(s,a')$:

$$C(s) \leftarrow C(s) + 1$$

$$\forall a' \in A_i, \qquad \bar{\pi}_i(s,a') \leftarrow \pi_i(s,a') + \frac{1}{C(s)}\left(\pi_i(s,a') - \bar{\pi}_i(s,a')\right)$$

(d) Update the strategy $\pi_i(s,a_i)$ and constrain it to a legal probability distribution

$$\pi_i(s,a_i) = \pi_i(s,a_i) + \begin{cases} \delta & \text{if } a_i = \max_{a_i'} Q_i(s,a_i') \\ -\dfrac{\delta}{|A_i|-1} & \text{otherwise} \end{cases}$$

where

$$\delta = \begin{cases} \delta_w & \text{if } \sum_{a_i} \pi_i(s,a_i)Q_i(s,a_i) > \sum_{a_i} \bar{\pi}_i(s,a_i)Q_i(s,a_i) \\ \delta_l & \text{otherwise} \end{cases}$$

The WoLF-Policy Hill Climbing (PHC) for Stochastic Game

- This algorithm has two different learning rates
  - ✓ When the algorithm is wining
  - ✓ When the algorithm is losing
- The losing learning rate δ_l is larger than wining learning rate δ_w
  - When an agent is losing, it learns faster than when it is winning
  - This causes the agent to adapt quickly to the changes in the strategies of the other agents when it is doing more poorly than expected
  - Learns cautiously when it is doing better than expected
  - Also gives the other agents the time to adapt to the agent's strategy changes
- The different between the average strategy and the current strategy is used as a criterion to decide when the algorithm wins or loses
- The WoLF-PHC algorithm exhibits the property of convergence as it makes the agent converge to one of its Nash equilibria (no proof, but empirical results)
- The algorithm is also a rational learning algorithm as it makes the agent converge to its optimal strategy when its opponent plays a stationary strategy

Comparison of MARL algorithms

Algorithm	Applicability	Rationality	Convergence	Required info
Minmax-Q	Zero-sum SGs	NO	YES	Other agent's action, rewards
Nash-Q	general sum SGs	NO	YES	Other agent's action, rewards
Friend-or-foe Q	general sum SGs	NO	YES	Other agent's action, rewards
Correlated-Q	general sum SGs	NO	YES	Other agent's action, rewards
WoLF-PHC	General sum SGs	YES	NO	Own action, reward

- The WoLF-PHC algorithm does not need to observe the other player's strategies and actions
- The WoLF-PHC does not require to solve Linear programming nor quadratic programming

The latest paper on Multi-agent actor critic algorithm

Team game⊂ Cooperative gameTeam game ⊂ Non−cooperativeTeam game

Actor-Critic algorithm is used	Counterfactual MAPG (Oxford)	Fully-decentralized MARL with networked agent (UIUC)	MADDPG (OpenAI)	Correalted-DDPG (KAIST)
Reward function	Common reward $R^i(s,a) = R(s,a)\ \forall i$	Independent $R^i(s,a), i = 1, \dots, n$	Independent $R^i(s,a), i = 1, \dots, n$	Independent $R^i(s,a), i = 1, \dots, n$
Q function	Common central Q $Q(s,a) = \sum_{t=0}^{T} \gamma^t r_t$	Common central Q $Q(s,a) = \sum_{t=0}^{T} \sum_{i=1}^{N} \gamma^t r_t^i$	Independent Q $Q^i(s,a) = \sum_{t=0}^{T} \gamma^t r_t^i$, $i = 1, \dots, n$	Independent Q $Q^i(s,a) = \sum_{t=0}^{T} \gamma^t r_t^i$, $i = 1, \dots, n$
Game type	**Team game**	**Cooperative game**	**Non-cooperative game**	**Non-cooperative game**
Equilibrium concept	Optimum Q (single)	Optimum Q (single)	Nash Q	Correlated Q
Policy $\pi(s,a) = \prod_{i=1}^{N} \pi^i(s,a^i)$	Decentralized independent policy $\pi^i(o^i,a^i), o^i = h^i(s)$	Decentralized independent policy $\pi^i(s,a^i)$	Decentralized independent policy $\pi^i(o^i,a^i), o^i = h^i(s)$	Decentralized independent policy $\pi^i(s,a^i)$
Consensus mechanism (how the mutual interaction is modeled?)	Each agent Learn central $Q(s,a)$ and train $\pi^i(o^i,a^i)$ Automatically each agent will have the common $Q(s,a)$	Each agent Learn central $Q(s,a)$ using its own local reward by sharing $Q(s,a)$ parameters and train $\pi^i(o^i,a^i)$ independently **(distributed optimization)**	$Q^i(s,a) = Q^i(s,\pi^1(s),\dots,\pi^N(s))$ Learn other player's policy and use that to estimate $Q^i(s,a)$ **Imitation learning + Best response principle**	Use collective gradient or coordinated gradient **(coordination is considered through gradient)**
Limitation		Take long time to reach consensus in $Q(s,a)$	**Separate training for π and Q in** $Q^i(s,\pi^1(s),\dots,\pi^N(s))$	