Learning in Multiagent Systems

José M Vidal

Department of Computer Science and Engineering, University of South Carolina

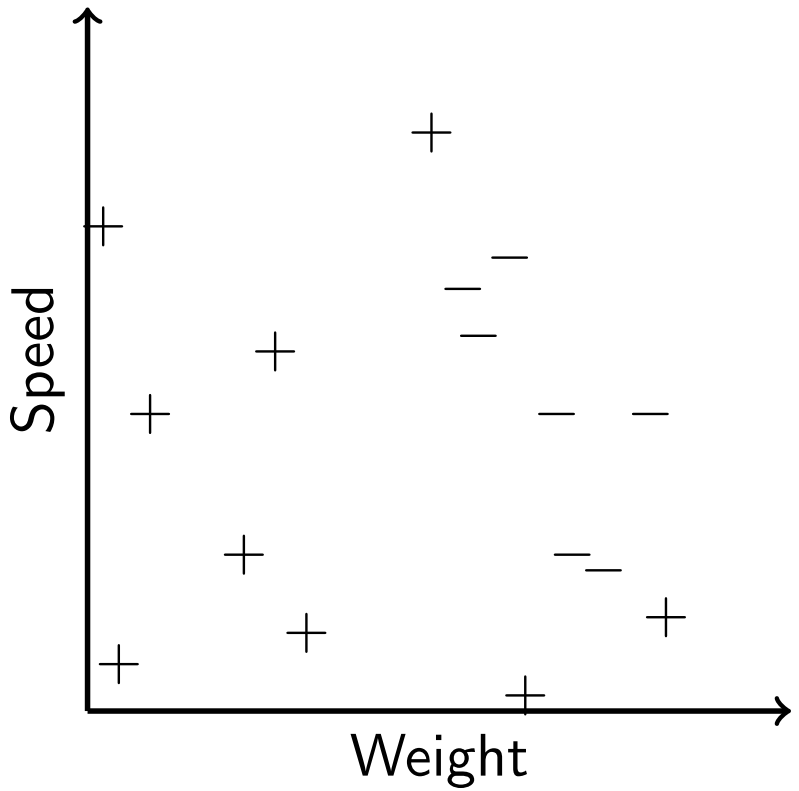
February 11, 2010

Abstract

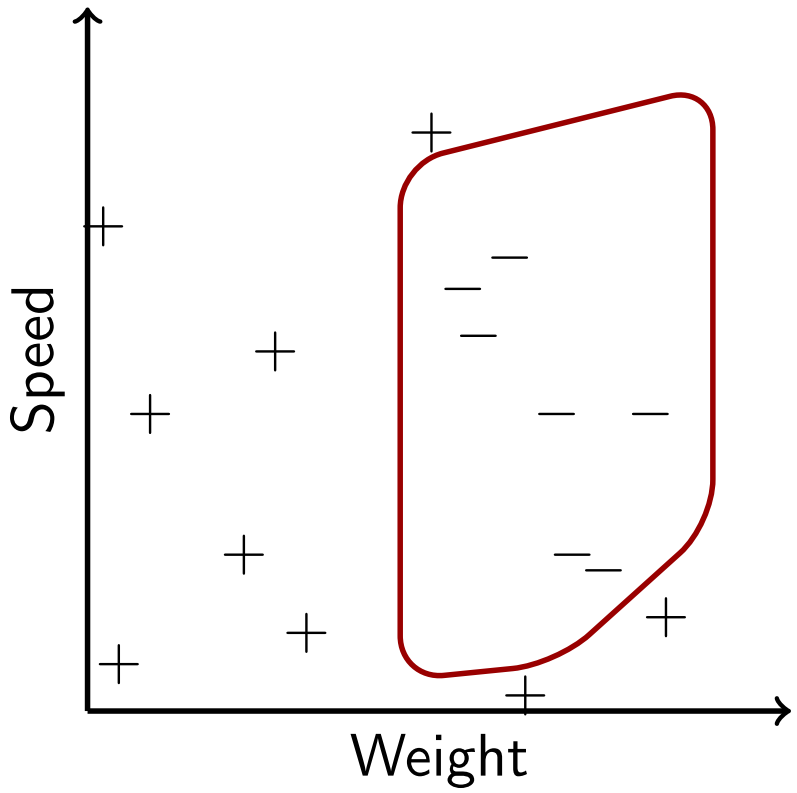
We introduce the topic of learning in multiagent systems and present recent results.

Outline

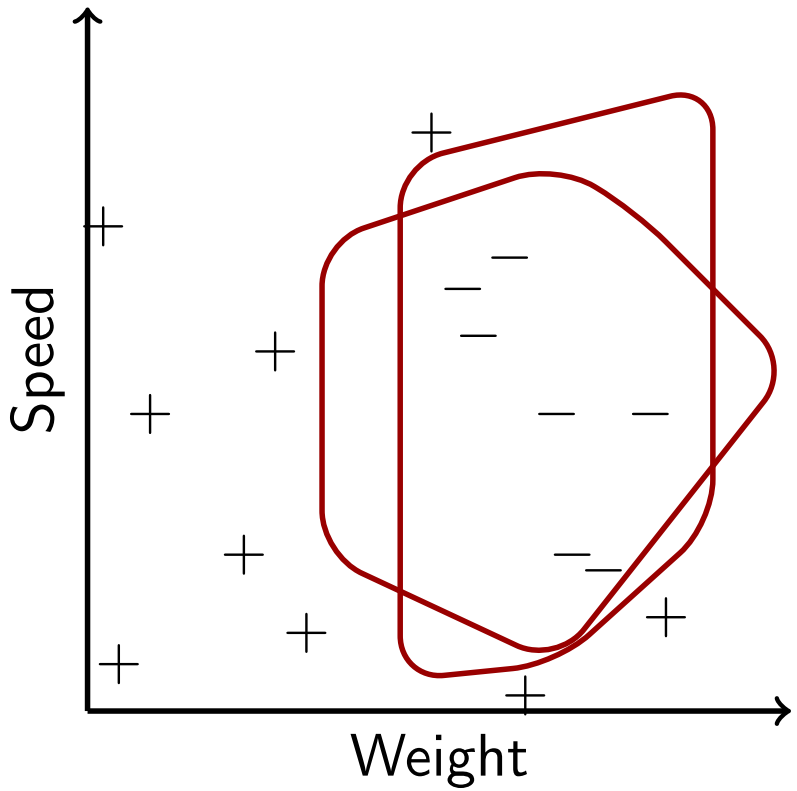
The Learning Problem



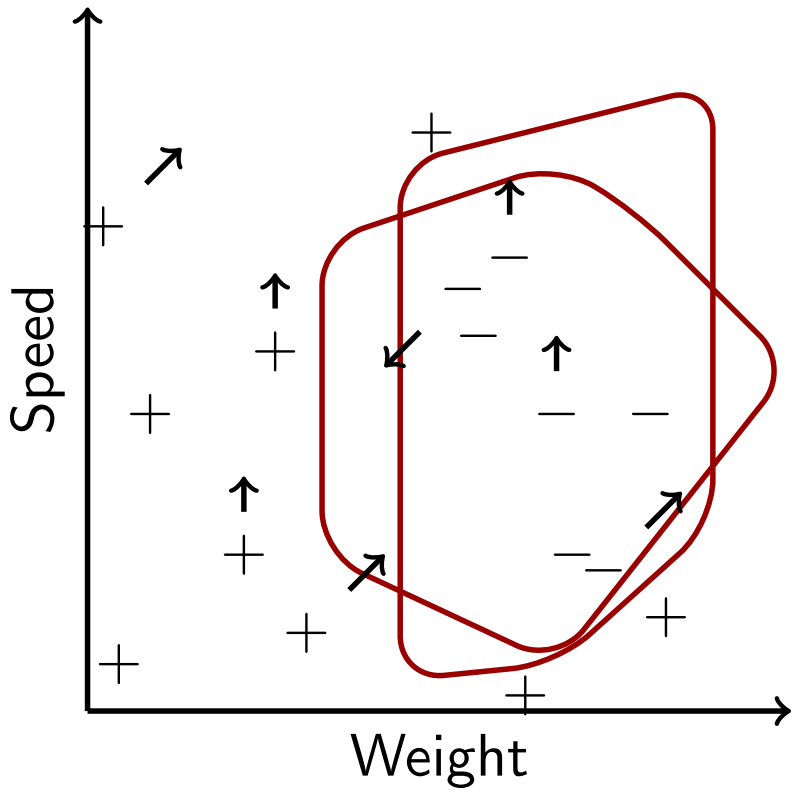
The Learning Problem



The Learning Problem



The Multiagent Learning Problem



Outline

Sharing Learned Knowledge

- Fairly easy with identical agent abilities.
- Largely unexplored for heterogeneous agents.

Sharing Learned Knowledge

- Fairly easy with identical agent abilities.
- Largely unexplored for heterogeneous agents.
- Generalizing learned knowledge seems very domain-specific (the induction problem again).

Outline

		j	
		C	D
i	A	0,0	5,1
	B	-1,6	1,5

Outline

Weight Function

$$k_i^t(s_j) = k_i^{t-1}(s_j) + \begin{cases} 1 & \text{if } s_j^{t-1} = s_j, \\ 0 & \text{if } s_j^{t-1} \neq s_j. \end{cases}$$

Model of Opponent

$$\mathbf{Pr}_i^t[s_j] = \frac{k_i^t(s_j)}{\sum_{\tilde{s}_j \in S_j} k_i^t(\tilde{s}_j)}.$$

Best Response

Player i then determines the strategy that will give it the highest expected utility given that j will play each of its $s_j \in S_j$ with probability $\mathbf{Pr}_i^t[s_j]$.

Example

		j	
		C	D
i	A	0,0	1,2
	B	1,2	0,0

s_i	s_j	$k_i(C)$	$k_i(D)$	$\Pr_i[C]$	$\Pr_i[D]$
A	C	1	0	1	0

Example

		j: C	j: D
i	A	0,0	1,2
	B	1,2	0,0

s_i	s_j	$k_i(C)$	$k_i(D)$	$\Pr_i[C]$	$\Pr_i[D]$
A	C	1	0	1	0
B	D	1	1	.5	.5

Example

		j: C	j: D
i	A	0,0	1,2
	B	1,2	0,0

s_i	s_j	$k_i(C)$	$k_i(D)$	$\Pr_i[C]$	$\Pr_i[D]$
A	C	1	0	1	0
B	D	1	1	.5	.5
A	D	1	2	1/3	2/3

Example

		j: C	j: D
i	A	0,0	1,2
	B	1,2	0,0

s_i	s_j	$k_i(C)$	$k_i(D)$	$\Pr_i[C]$	$\Pr_i[D]$
A	C	1	0	1	0
B	D	1	1	.5	.5
A	D	1	2	1/3	2/3
A	D	1	3	1/4	3/4

Example

		j: C	j: D
i	A	0,0	1,2
	B	1,2	0,0

s_i	s_j	$k_i(C)$	$k_i(D)$	$\Pr_i[C]$	$\Pr_i[D]$
A	C	1	0	1	0
B	D	1	1	.5	.5
A	D	1	2	1/3	2/3
A	D	1	3	1/4	3/4
A	D	1	4	1/5	4/5

Theorem (Nash Equilibrium is Attractor to Fictitious Play)

If s is a strict Nash equilibrium and it is played at time t then it will be played at all times greater than t.

Theorem (Fictitious Play Converges to Nash)

If fictitious play converges to a pure strategy then that strategy must be a Nash equilibrium.

Infinite Cycle Example

		j	
		C	D
i	A	0,0	1,1
	B	1,1	0,0

s_i	s_j	$k_i(C)$	$k_i(D)$	$k_j(A)$	$k_j(B)$
		1	1.5	1	1.5

Infinite Cycle Example

		j: C	j: D
i	A	0,0	1,1
	B	1,1	0,0

s_i	s_j	$k_i(C)$	$k_i(D)$	$k_j(A)$	$k_j(B)$
		1	1.5	1	1.5
A	C	2	1.5	2	1.5

Infinite Cycle Example

		j	
		C	D
i	A	0,0	1,1
	B	1,1	0,0

s_i	s_j	$k_i(C)$	$k_i(D)$	$k_j(A)$	$k_j(B)$
		1	1.5	1	1.5
A	C	2	1.5	2	1.5
B	D	2	2.5	2	2.5

Infinite Cycle Example

		j: C	j: D
i	A	0,0	1,1
	B	1,1	0,0

s_i	s_j	$k_i(C)$	$k_i(D)$	$k_j(A)$	$k_j(B)$
		1	1.5	1	1.5
A	C	2	1.5	2	1.5
B	D	2	2.5	2	2.5
A	C	3	2.5	3	2.5

Infinite Cycle Example

		j: C	j: D
i	A	0,0	1,1
	B	1,1	0,0

s_i	s_j	$k_i(C)$	$k_i(D)$	$k_j(A)$	$k_j(B)$
		1	1.5	1	1.5
A	C	2	1.5	2	1.5
B	D	2	2.5	2	2.5
A	C	3	2.5	3	2.5
B	D	3	3.5	3	3.5

Outline

Fraction of Agents Playing s

let $\phi^t(s)$ be the number of agents using strategy s at time t. We can then define

$$\theta^t(s) = \frac{\phi^t(s)}{\sum_{s' \in S} \phi^t(s')}$$

Expected Utility for Playing s

$$u^t(s) \equiv \sum_{s' \in S} \theta^t(s') u(s, s'),$$

Reproduction Rate

$$\phi^{t+1}(s) = \phi^t(s)(1 + u^t(s)).$$

Population Dynamics

- Population size could change but we scale back or ignore.
- Game must be symmetric.
- A stable population of more than one strategy corresponds to a mixed strategy.

Theorem (Nash equilibrium is a Steady State)

Every Nash equilibrium is a steady state for the replicator dynamics.

Proof.

By contradiction. If an agent had a pure strategy that would return a higher utility than any other strategy then this strategy would be a best response to the Nash equilibrium. If this strategy was different from the Nash equilibrium then we would have a best response to the equilibrium which is not the equilibrium, so the system could not be at a Nash equilibrium. □

Theorem (Stable Steady State is a Nash Equilibrium)

A stable steady state of the replicator dynamics is a Nash equilibrium. A stable steady state is one that, after suffering from a small perturbation, is pushed back to the same steady state by the system's dynamics.

Theorem (Asymptotically Stable is Trembling-Hand Nash)

An asymptotically stable steady state corresponds to a Nash equilibrium that is trembling-hand perfect and isolated. That is, the stable steady states are a refinement on Nash equilibria—only a few Nash equilibria can are stable steady states.

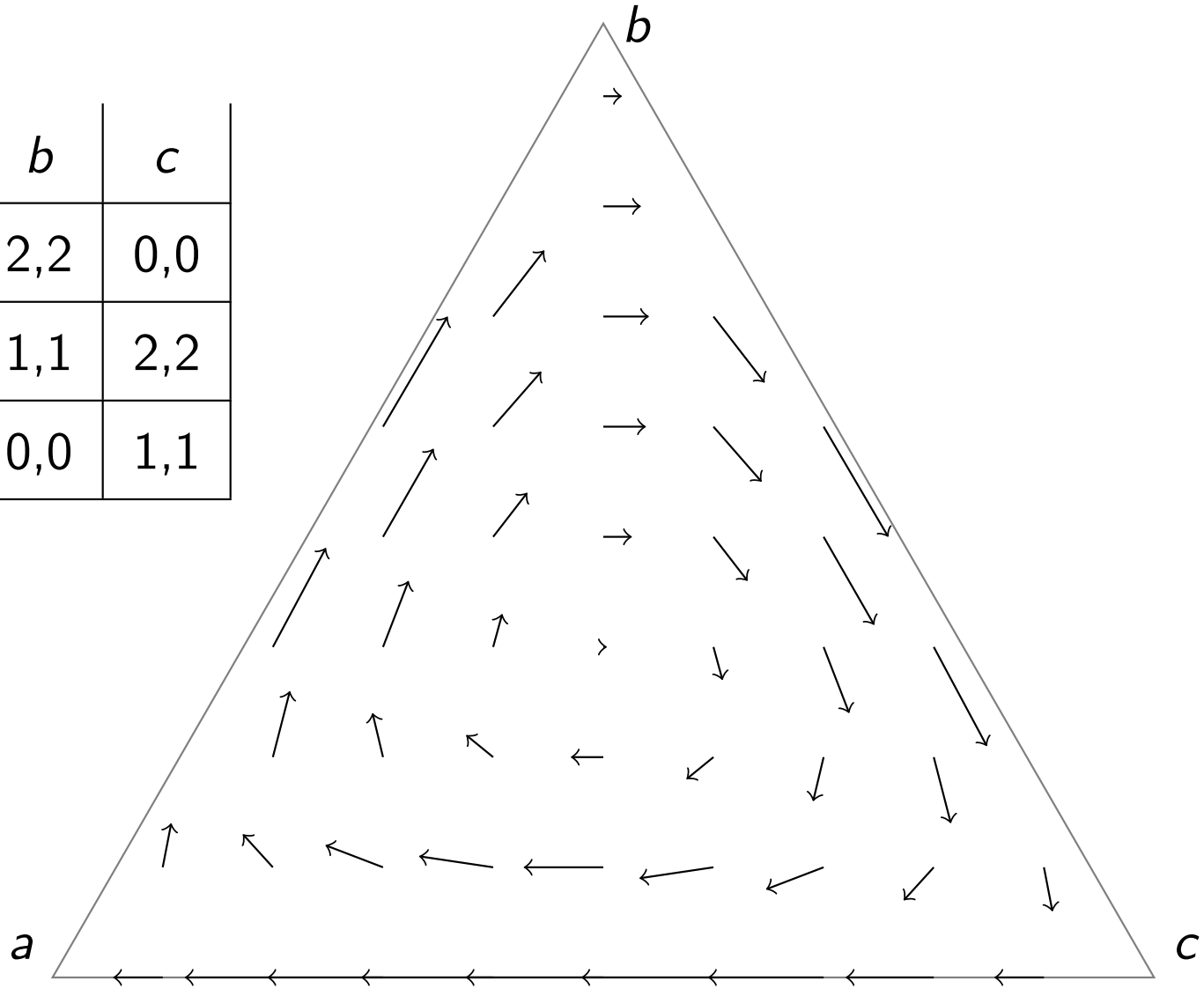
Definition (Evolutionary Stable Strategy)

An ESS is an equilibrium strategy that can overcome the presence of a small number of invaders. That is, if the equilibrium strategy profile is ω and small number ε of invaders start playing ω' then ESS states that the existing population should get a higher payoff against the new mixture $(\varepsilon\omega' + (1-\varepsilon)\omega)$ than the invaders.

Theorem (ESS is Steady State of Replicator Dynamics)

ESS is an asymptotically stable steady state of the replicator dynamics. However, the converse need not be true—a stable state in the replicator dynamics does not need to be an ESS.

		j		
		a	b	c
i	a	1,1	2,2	0,0
	b	0,0	1,1	2,2
	c	2,2	0,0	1,1



Outline

AWESOME

1 $play\text{-}eq, play\text{-}sta \leftarrow \text{TRUE}; eq\text{-}rej \leftarrow \text{FALSE}; \phi \leftarrow \pi_i; t \leftarrow 0$
2 **while** *play-sta*
3 **do** play ϕ for N times in a row (an epoch)
4 $\forall_j$ update s_j given what they played in these N rounds.
5 **if** *play-eq*
6 **then if** some player j has $\max_a(s_j(a), \pi_j(a)) > \varepsilon_e$
7 **then** $eq\text{-}rej \leftarrow \text{TRUE}; \phi \leftarrow$ random action
8 **else if** $\neg\, eq\text{-}rej \wedge \exists_j \max_a(s_j^{\text{old}}(a), s_j(a)) > \varepsilon_s$
9 **then** $play\text{-}sta \leftarrow \text{FALSE}$
10 $eq\text{-}rej \leftarrow \text{FALSE}$
11 $b \leftarrow \arg\max_a u_i(a, s_{-i})$
12 **if** $u_i(b, s_{-i}) > u_i(\phi, s_{-i}) + n|A_i|\varepsilon_s^{t+1}\mu$
13 **then** $\phi \leftarrow b$
14 $\forall_j s_j^{\text{old}} \leftarrow s_j$
15 $t \leftarrow t+1$
16 **goto** 1

The Schedule

In order for the algorithm to always converge, ε_e and ε_s must be decreased and N must be increased over time using a schedule where

1. ε_s and ε_e decrease monotonically to 0,
2. N increases to infinity,
3. $\prod_{t \leftarrow 1,\ldots,\infty} 1 - \frac{\sum_i |A_i|}{N^t (\varepsilon_s^t)^2} > 0$
4. $\prod_{t \leftarrow 1,\ldots,\infty} 1 - \frac{\sum_i |A_i|}{N^t (\varepsilon_e^t)^2} > 0.$

It Converges

Theorem (AWESOME converges)

With a valid schedule, the AWESOME algorithm converges to best response if all the other players play fixed strategies and to a Nash equilibrium if all the other players are AWESOME players.

Outline

What is a Stochastic Game?

- One where the agents do not know the payoff they might get.
- That is, an unexplored MDP.

Outline

Reinforcement Learning Problem Definition

- s_t is a state, taken from S,
- a_t is an action, taken from A,
- $P(s_{t+1} \mid s_t, a_t)$ is the state transition function
- $r(s_t, a_t) \rightarrow \Re$ is the reward function.

Reinforcement Learning Problem Definition

- s_t is a state, taken from S,
- a_t is an action, taken from A,
- $P(s_{t+1} \mid s_t, a_t)$ is the state transition function
- $r(s_t, a_t) \rightarrow \Re$ is the reward function.

The problem is to find the policy $\pi(s) \rightarrow a$ which maximizes the discounted successive rewards r_t the agent receives if using π. That is, find

$$\pi^* = \arg\max_{\pi} \sum_{i=0}^{\infty} \gamma^i r_i$$

Q-LEARNING

1 $\forall_s \forall_a Q(s,a) \leftarrow 0$; $\lambda \leftarrow 1$; $\varepsilon \leftarrow 1$
2 $s \leftarrow$ current state
3 **if** $\text{RAND}() < \varepsilon$ $\triangleright$ Exploration rate
4 **then** $a \leftarrow$ random action
5 **else** $a \leftarrow \arg\max_a Q(s,a)$
6 Take action a
7 Receive reward r
8 $s' \leftarrow$ current state
9 $Q(s,a) \leftarrow \lambda(r + \gamma \max_{a'} Q(s',a')) + (1-\lambda)Q(s,a)$
10 $\lambda \leftarrow .99\lambda$
11 $\varepsilon \leftarrow .98\varepsilon$
12 **goto** 2

Theorem (Q-LEARNING Converges)

Given that the learning and exploration rates decrease slowly enough, Q-LEARNING *is guaranteed to converge to the optimal policy.*

Definition (Nash Equilibrium Point)

A Nash equilibrium point is a tuple of n strategies $(\pi_1^*, \ldots, \pi_n^*)$ such that for all $s \in S$ and $i = 1, \ldots, n$,

$$\forall_{\pi_i \in \Pi_i} v_i(s, \pi_1^*, \ldots \pi_n^*) \geq v_i(s, \pi_1^*, \ldots \pi_{i-1}^*, \pi_i, \pi_{i+1}^*, \ldots, \pi_n^*)$$

Theorem (Nash Equilibrium Point Exists)

Every $n-$player discounted stochastic game possesses at least one Nash equilibrium point in stationary strategies.

NashQ-learning

1 $t \leftarrow 0$
2 $s_0 \leftarrow$ current state
3 $\forall_{s \in S} \forall_{j \leftarrow 1,\ldots,n} \forall_{a_j \in A_j} Q_j^t(s, a_1, \ldots, a_n) \leftarrow 0$
4 Choose action a_i^t
5 Observe $r_1^t, \ldots, r_n^t$; $a_1^t, \ldots, a_n^t$; $s_{t+1} = s'$
6 **for** $j \leftarrow 1, \ldots, n$
7 **do** $Q_j^{t+1}(s, a_1, \ldots, a_n) \leftarrow$
$(1 - \lambda^t) Q_j^t(s, a_1, \ldots, a_n) + \lambda^t (r_j^t + \gamma NashQ_j^t(s'))$
where $NashQ_j^t(s') = Q_j^t(s', \pi_1(s') \cdots \pi_n(s'))$
and $\pi_1(s') \cdots \pi_n(s')$ are Nash EP calculated from Q values
8 $t \leftarrow t + 1$
9 **goto** 4

Assumption

There exists an adversarial equilibrium for the entire game and for every game defined by the Q functions encountered during learning.

Assumption

There exists a coordination equilibrium for the entire game and for every game defined by the Q functions encountered during learning.

Assumption

There exists an adversarial equilibrium for the entire game and for every game defined by the Q functions encountered during learning.

Assumption

There exists a coordination equilibrium for the entire game and for every game defined by the Q functions encountered during learning.

Theorem (NashQ-learning Converges)

Under these assumptions NashQ-learning *converges to a Nash equilibrium as long as all the equilibria encountered during the game are unique.*

FRIEND-OR-FOE

1 $t \leftarrow 0$

2 $s_0 \leftarrow$ current state

3 $\forall_{s \in S} \forall_{a_j \in A_j} Q_i^t(s, a_1, \ldots, a_n) \leftarrow 0$

4 Choose action a_i^t

5 Observe $r_1^t, \ldots, r_n^t$; $a_1^t, \ldots, a_n^t$; $s_{t+1} = s'$

6 $Q_i^{t+1}(s, a_1, \ldots, a_n) \leftarrow$
$(1 - \lambda^t) Q_i^t(s, a_1, \ldots, a_n) + \lambda^t (r_i^t + \gamma NashQ_i^t(s'))$
where $NashQ_i^t(s') = \max_{\pi \in \Pi(X_1 \times \cdots \times X_k)} \min_{y_i, \ldots, y_l \in Y_1 \times \cdots \times Y_l}$
$\sum_{x_1, \ldots, x_k \in X_1 \times \cdots \times X_k} \pi(x_1) \cdots \pi(x_k) Q_i(s, x_1, \ldots, x_k, y_1, \ldots y_l)$
and X are actions for i's friends and Y are for the foes.

7 $t \leftarrow t + 1$

8 **goto** 4

Theorem (FRIEND-OR-FOE converges)

FRIEND-OR-FOE *converges.*

However, in general these do not correspond to a Nash equilibrium point.

Theorem (FRIEND-OR-FOE converges)

FRIEND-OR-FOE *converges.*

However, in general these do not correspond to a Nash equilibrium point. Still, we can show

Theorem

FOE-Q *learns values for a Nash equilibrium policy if the game has an adversarial equilibrium and* FRIEND-Q *learns values for a Nash equilibrium policy if the game has a coordination equilibrium. This is true regardless of opponent behavior.*

Outline

Moving Target Function Problem

- As the other agents change their behavior, what you need to do also changes.

Outline

CLRI Notation

- $\delta_i^t(w) : W \rightarrow A$: decision function.
- $\Delta_i^t(w)$: target function
- $e(\delta_i^t) = \mathbf{Pr}[\delta_i^t(w) \neq \Delta_i^t(w) \,|\, w \in \mathscr{D}(W)$: error function.

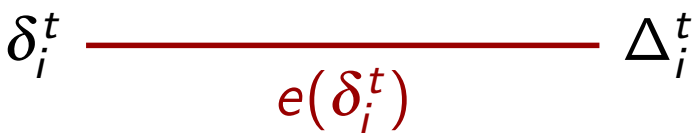


Figure: The moving target function problem.

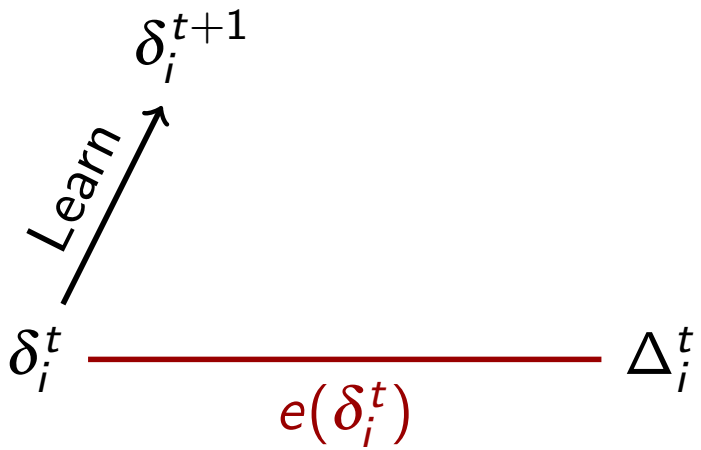


Figure: The moving target function problem.

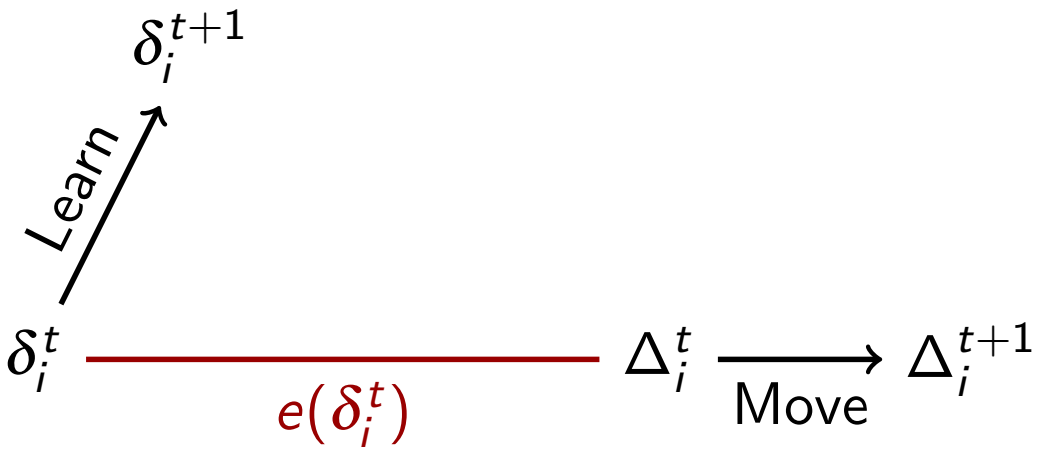


Figure: The moving target function problem.

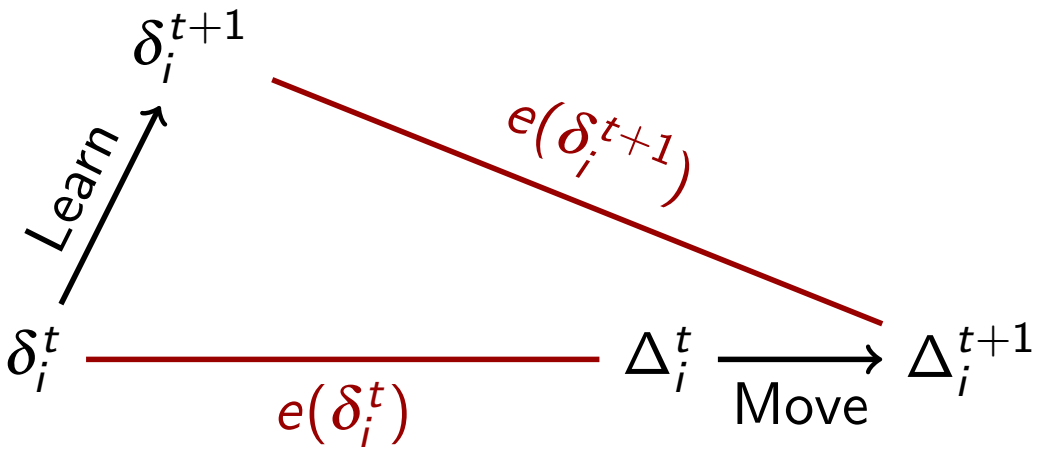


Figure: The moving target function problem.

CLRI Parameter

- Change rate (c) is the probability that an agent will change at least one of its incorrect mappings in $\delta^t(w)$.
- Learning rate (l) is the probability that the agent changes an incorrect mapping to the correct one.
- Retention rate (r) represents the probability that the agent will retain its correct mapping.
- Impact (I_{ij}) is the impact that i's learning has on j's target function. Specifically, it is the probability that $\Delta_j^t(w)$ will change given that $\delta_i^{t+1}(w) \neq \delta_i^t(w)$.

CLRI Equation

$$E[e(\delta_i^{t+1})] = 1 - r_i + v_i \left(\frac{|A_i| r_i - 1}{|A_i| - 1} \right) + e(\delta_i^t) \left(r_i - l_i + v_i \left(\frac{|A_i|(l_i - r_i) + l_i - c_i}{|A_i| - 1} \right) \right) \quad (1)$$

Outline

I Think that You Think that I Think that...

- 0-level agent is one that does not recognize the existence of other agents in the world.
- 1-level agent recognizes that there are other agents in the world whose actions affect its payoff. It also has some knowledge that tells it the utility it will receive given any set of joint actions.
- 2-level agent believes that all other agents are 1-level agents.

Decreasing Returns of Thinking

- n-level always beats $(n-1)$-level agents.
- Marginal utility gains grow smaller with each extra level.
- Computational costs grow exponentially with each extra level.

Decreasing Returns of Thinking

- n-level always beats $(n-1)$-level agents.
- Marginal utility gains grow smaller with each extra level.
- Computational costs grow exponentially with each extra level.
- Many times, it doesn't pay to think about your opponent.

Outline

COllective INtelligence

- Idea: start with the global utility function $U(s, \vec{a})$ and determine from it the individual utilities.

Define Preferences

We define i's preference over $s, \vec{a}$ as

$$P_i(s, \vec{a}) = \frac{\sum_{\vec{a}' \in \vec{A}} \Theta[r_i(s, \vec{a}) - r_i(s, \vec{a}')]}{|\vec{A}|},$$

where $\Theta(x)$ is the Heaviside function which is 1 if x is greater than or equal to 0, otherwise it is 0. S
Similarly, we define the global preference function as

$$P(s, \vec{a}) = \frac{\sum_{\vec{a}' \in \vec{A}} \Theta[U(s, \vec{a}) - U(s, \vec{a}')]}{|\vec{A}|}.$$

The Easier Case

- A system where, for all agents i it is true that $P_i(s, \vec{a}) = P(s, \vec{a})$ is called factored.

The Easier Case

- A system where, for all agents i it is true that $P_i(s, \vec{a}) = P(s, \vec{a})$ is called factored.
- But, might still not converge because agents' actions change other agents' target function.

Opacity

We define the opacity Ω_i for agent i as

$$\Omega_i(s, \vec{a}) = \sum_{\vec{a}' \in \vec{A}} \mathbf{Pr}[\vec{a}'] \frac{|u_i(s, \vec{a}) - u_i(s, \vec{a}'_{-i}, \vec{a}_i)|}{|u_i(s, \vec{a}) - u_i(s, \vec{a}_{-i}, \vec{a}'_i)|}.$$

System Categorization

- If factored and zero opacity then its easy to solve, but rare.

System Categorization

- If factored and zero opacity then its easy to solve, but rare.
- If zero opacity then it amounts to multiple parallel learning problems.

System Categorization

- If factored and zero opacity then its easy to solve, but rare.
- If zero opacity then it amounts to multiple parallel learning problems.
- Goal: Find low opacity reward functions that are highly factored.

Wonderful Life

- The wonderful life utility function gives each agent:

$$u_i(s, \vec{a}) = U(s, \vec{a}) - U(s, \vec{a}_{-i}, 0),$$

Aristocrat Utility

- Another solution is the aristocrat utility

$$u_i(s, \vec{a}) = U(s, \vec{a}) - \sum_{\vec{a}' \in \vec{A}} \mathbf{Pr}[\vec{a}'] U(s, \vec{a}_{-i}, \vec{a}'_i),$$

where $\mathbf{Pr}[\vec{a}']$ is the probability that $\vec{a}'$ happens.

COIN Tests

- Both utility functions have been shown to perform better than $u_i = U$ and other hand-tailored utility functions.