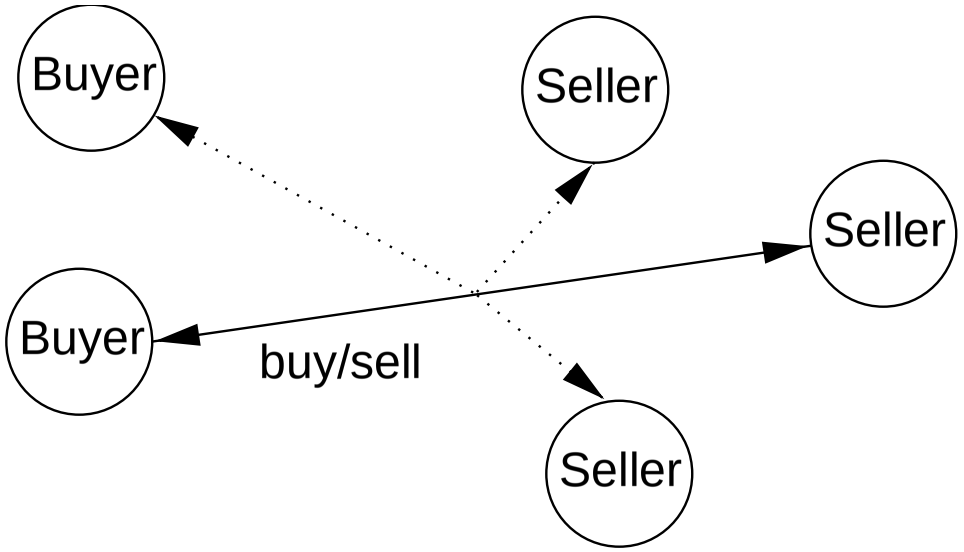
The Moving Target Function Problem in Multi-Agent Learning

José M.Vidal and Edmund H. Durfee
Artificial Intelligence Laboratory, University of Michigan
1101 Beal Avenue, Ann Arbor, MI 48109-2110
http://ai.eecs.umich.edu/people/jmvidal/
jmvidal@umich.edu

July 6, 1998

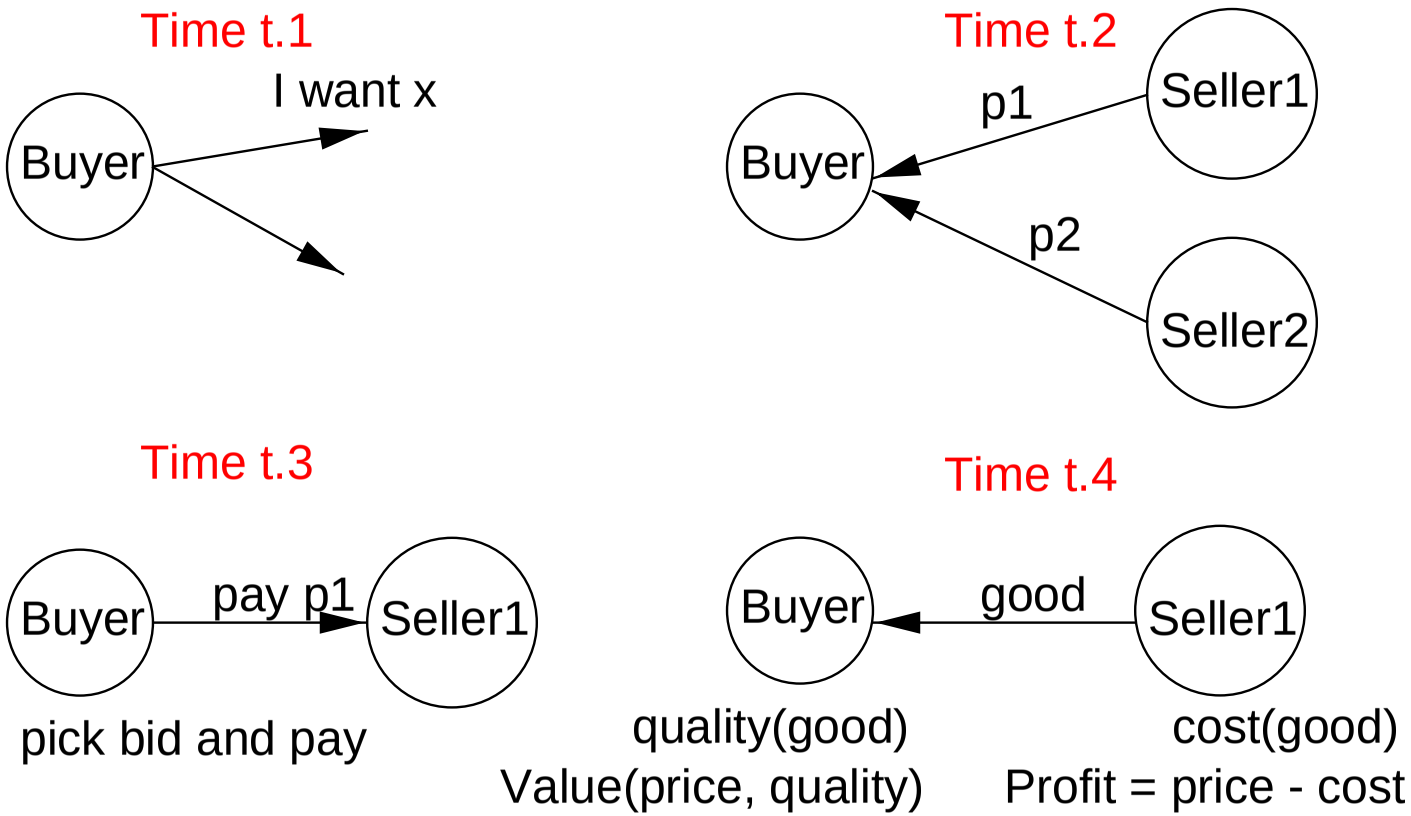
university of michigan artificial intelligence laboratory

Example: An Economic Multi-Agent System

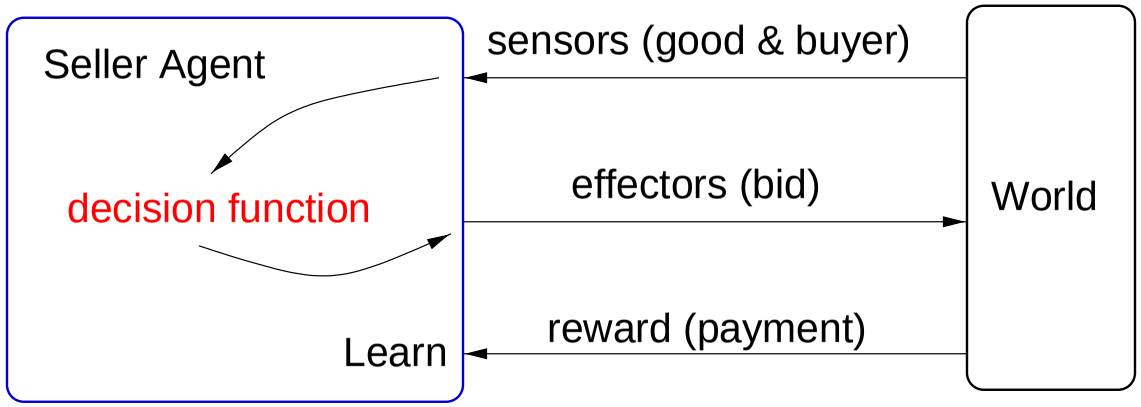


- Agents are selfish.
- Agents can use machine learning techniques.
- Agents develop individual preferences over who to buy/sell from.

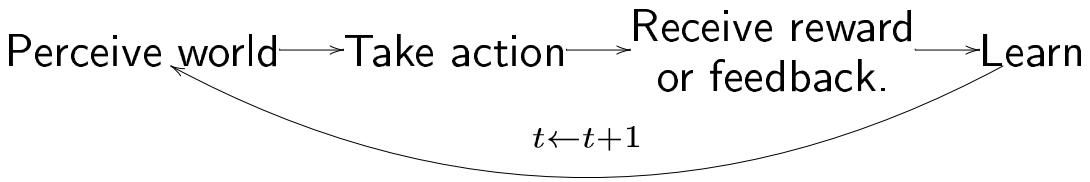
Transaction Protocol



Describing a Learning Seller-Agent



Action/Learn Loop for Agent



Framework for Describing a MAS

N the set of all agents, where $i \in N$.

W the set of possible states of the world, where state $w \in W$. We assume all agents perceive the same world state.

A_i the set of all actions that agent $i \in N$ can take, e.g., bid \$10.

$\delta_i^t : W \rightarrow A_i$ is the decision function for agent i. It tells us which action i will take in each world.

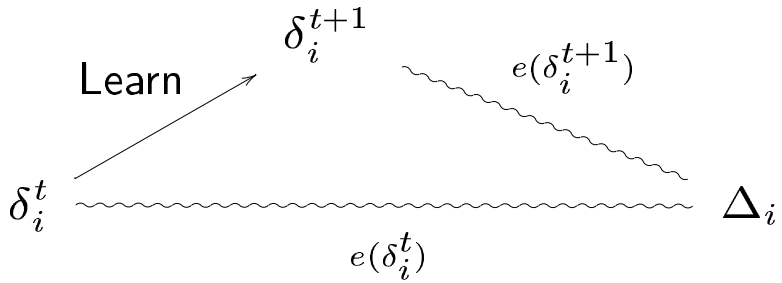
$\Delta_i^t : W \rightarrow A_i$ the target function for agent i. It tells us what action i should take in each world w. It takes into account the actions that other agents will take.

$e(\delta_i^t) = \mathbf{Pr}[\delta_i^t(w) \neq \Delta_i^t(w) \mid w \in \mathcal{D}]$ the error of agent i is the probability that it will take an incorrect action, given that the worlds w are taken from the fixed probability distribution $\mathcal{D}$.

- An agent's behavior can be described with $\delta_i^t(w)$, but this behavior can be implemented by different means.

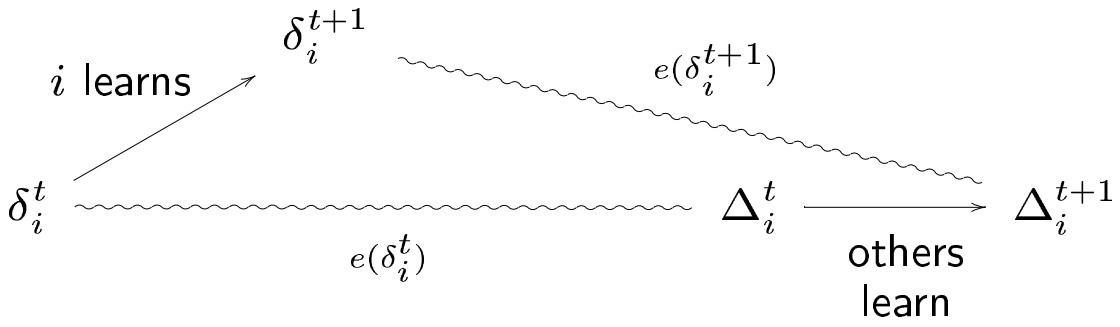
Visualizing the Learning Problem

- Imagine the various decision functions as points in the space of all possible decision functions.
- The distance between them is given by the error.



- In the traditional machine learning problem, the agent's learning changes the decision function so as to (hopefully) better match the target function.

The Moving Target Function Problem

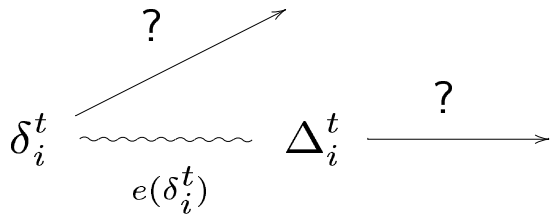


- The agent's learning moves the decision function closer to the target function.
- But, the fact that the other agents are also learning moves the target function in some direction, possibly away from decision function.
- Will the error ever be zero?
- What is the expected error as $t \rightarrow \infty$?

A Predictive Framework for Learning MAS

- There has been a lot of experimental research into the behavior of systems with learning agents. In the general case, the system is a complex adaptive system and no predictive theory exists.
- We noticed certain recurring phenomena in our experiments and in the experiments of others.
- Since the agents are using machine learning techniques we should be able to precisely describe their learning abilities.
- Using these descriptions we can calculate an agent's expected error for successive times.
- We call our framework CLRI—an acronym for the parameters we use to characterize the agents' and the system.

Modeling Learning



c_i is the change rate of agent i, which gives us the probability that the agent will change one of its incorrect $W \rightarrow A$ mappings.

$$\forall_w \; c_i = \mathbf{Pr}[\delta_i^{t+1}(w) \neq \delta_i^t(w) \mid \delta_i^t(w) \neq \Delta_i^t(w)]$$

l_i is the learning rate. It gives us the probability that an incorrect $W \rightarrow A$ mapping in the decision function gets changed to the correct mapping in the next time step.

$$\forall_w \; l_i = \mathbf{Pr}[\delta_i^{t+1}(w) = \Delta_i^t(w) \mid \delta_i^t(w) \neq \Delta_i^t(w)]$$

Notice that it must always be true that $l_i \leq c_i$.

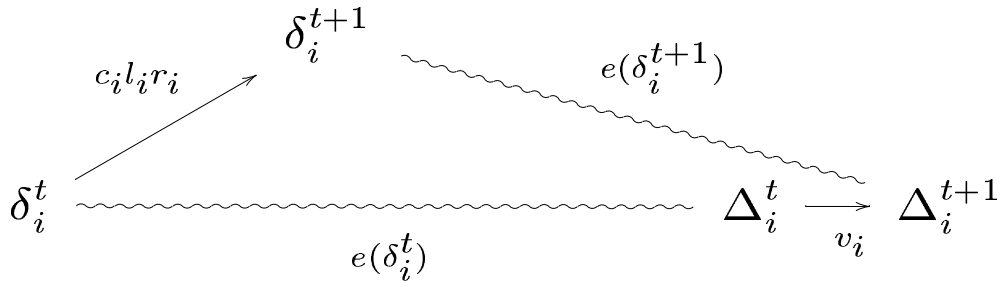
Modeling Learning

r_i is the retention rate. It is the probability that a correct mapping will stay correct in the next iteration.

$$\forall_w \; r_i = \mathbf{Pr}[\delta_i^{t+1}(w) = \Delta_i^t(w) \mid \delta_i^t(w) = \Delta_i^t(w)]$$

v_i is the volatility. It measures how far the target function moves each time.

$$\forall_w \; v_i = \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w)]$$



Calculating the Agent's Error

- The agent's expected error at time $t+1$ can be calculated by considering the four possibilities of whether $\Delta_i^t(w)$ changes or not, and whether $\delta_i^t(w)$ was correct or not.
- We can write a difference equation for the expected error:

$$E[e(\delta_i^{t+1})] = E[\sum_{w \in W} \mathcal{D}(w)\mathbf{Pr}[\delta_i^{t+1}(w) \neq \Delta_i^{t+1}(w)]]$$

$$= \sum_{w \in W} \mathcal{D}(w)($$

(Prob. target fn. stays the same and decision fn. was right) $\cdot (1 - r_i)$

$+$ (Prob. target fn. stays the same and decision fn. was wrong) $\cdot (1 - l_i)$

$+$ (Prob. target fn. changes and decision fn. was right) $\cdot (r_i + (1 - r_i) \cdot B)$

$+$ (Prob. target fn. changes and decision fn. was wrong) $\cdot C$

Calculating the Agent's Error

- The agent's expected error at time $t+1$ can be calculated by considering the four possibilities of whether $\Delta_i^t(w)$ changes or not, and whether $\delta_i^t(w)$ was correct or not.
- We can write a difference equation for the expected error:

$$E[e(\delta_i^{t+1})] = E[\sum_{w \in W} \mathcal{D}(w)\mathbf{Pr}[\delta_i^{t+1}(w) \neq \Delta_i^{t+1}(w)]]$$

$$= \sum_{w \in W} \mathcal{D}(w)($$

$$\mathbf{Pr}[\Delta_i^{t+1}(w) = \Delta_i^t(w) \wedge \delta_i^t(w) = \Delta_i^t(w)] \cdot (1 - r_i) \quad (1)$$

$$+ \mathbf{Pr}[\Delta_i^{t+1}(w) = \Delta_i^t(w) \wedge \delta_i^t(w) \neq \Delta_i^t(w)] \cdot (1 - l_i)$$

$$+ \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w) \wedge \delta_i^t(w) = \Delta_i^t(w)] \cdot (r_i + (1 - r_i) \cdot B)$$

$$+ \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w) \wedge \delta_i^t(w) \neq \Delta_i^t(w)] \cdot C$$

where we define

$$B = \mathbf{Pr}[\delta_i^{t+1}(w) \neq \Delta_i^{t+1}(w) | \delta_i^t(w) = \Delta_i^t(w) \wedge \Delta_i^{t+1}(w) \neq \Delta_i^t(w) \wedge \delta_i^{t+1}(w) \neq \Delta_i^t(w)]$$

$$\begin{aligned} C = {} & l_i \\ & + (1 - c_i)\mathbf{Pr}[\delta_i^t(w) \neq \Delta_i^{t+1}(w) | \delta_i^t(w) \neq \Delta_i^t(w) \wedge \Delta_i^{t+1}(w) \neq \Delta_i^t(w)] \\ & + (c_i - l_i)\mathbf{Pr}[\delta_i^{t+1}(w) \neq \Delta_i^{t+1}(w) | \delta_i^t(w) \neq \Delta_i^t(w) \wedge \Delta_i^{t+1}(w) \neq \Delta_i^t(w) \\ & \quad \wedge \delta_i^{t+1}(w) \neq \Delta_i^t(w) \wedge \delta_i^{t+1}(w) \neq \delta_i^t(w)] \end{aligned}$$

- This equation applies to a wide variety of MASs.
- A lot of these probabilities can be reduced to functions of c_i, l_i, r_i and v_i.

Simplification

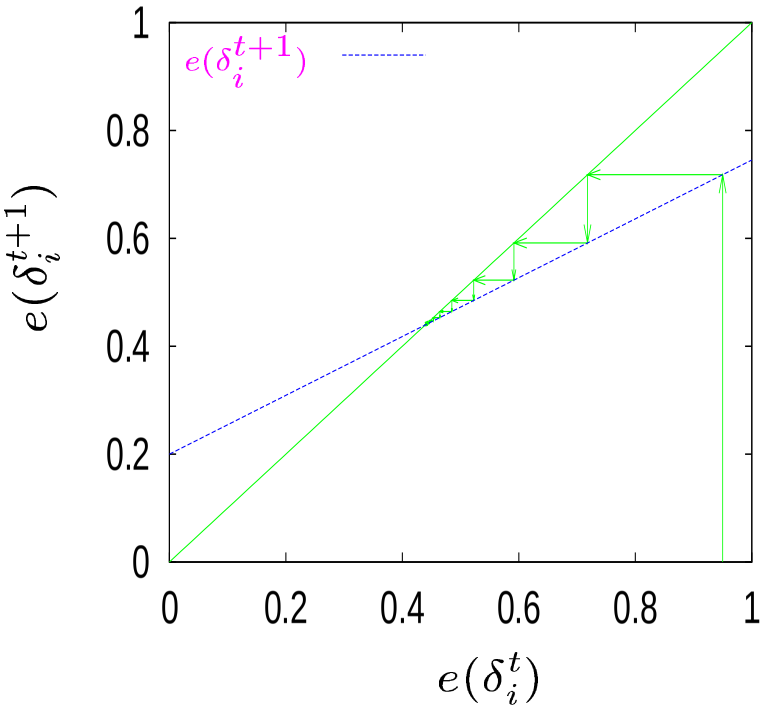
We can simplify this equation, as long as we make the following assumptions:

1. The new action chosen when either $\delta_i^t(w)$ changes (and does not match the target), or when $\Delta_i^t(w)$ changes, are both taken from flat probability distributions over A_i.
2. The probability of $\Delta_i^t(w)$ changing, for a particular w, is independent of the probability that $\delta_i^t(w)$ was correct.

Once we make these two assumptions we can rewrite (1) as:

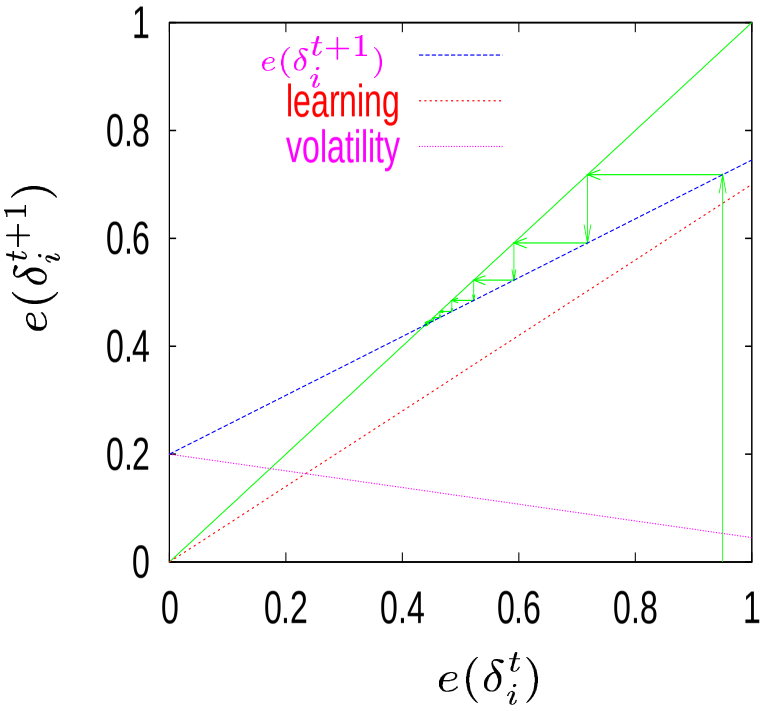
$$E[e(\delta_i^{t+1})] = 1 - r_i + v_i \left(\frac{|A_i| r_i - 1}{|A_i| - 1} \right) + e(\delta_i^t) \left(r_i - l_i + v_i \left(\frac{|A_i|(l_i - r_i) + l_i - c_i}{|A_i| - 1} \right) \right) \quad (2)$$

Error Progression Plot



- We can trace the expected error progression for agent i, assuming a fixed volatility $v_i = .8$, $l_i = .2$, $c_i = 1$, $r_i = 1$. The error converges to .51.
- We can calculate the final error for an agent, given the other parameters.

Learning and Volatility Components



- We can separate the v_i and l_i from equation (2), and plot these two lines separately. By definition, they add to (2).
- The learning curve decreases the error.
- The volatility increases the error.

Volatility and Impact

- v_i is a property of the system, not a parameter we can set.
- If we assume that the only reason that an agent's target function changes is because of changes in the others' decision function then we can calculate the expected v_i. We first define:

I_{ji} the impact that agent j's changes in its decision function have on i's target function.

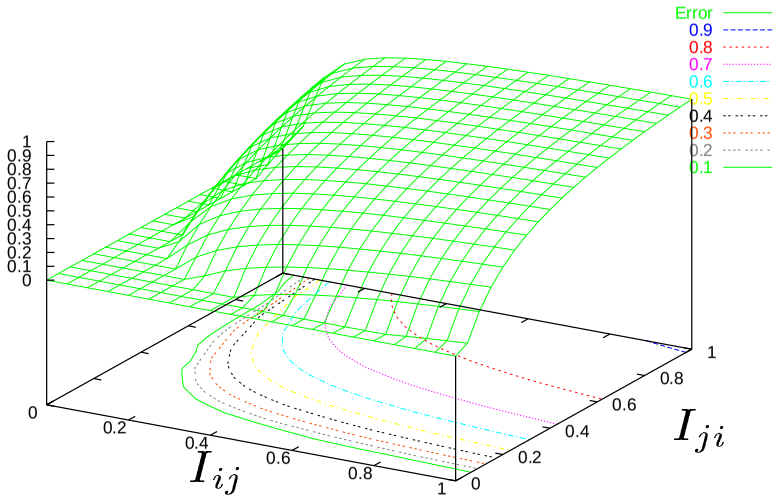
$$\forall_{w \in W}\ I_{ji} = \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w) \mid \delta_j^{t+1}(w) \neq \delta_j^t(w)]$$

- We can then find the expected volatility:

$$E[v_i^t] = 1 - \prod_{j \in N_{-}i} 1 - I_{ji}(c_j e(\delta_j^t) + (1 - r_j) \cdot (1 - e(\delta_j^t))) \qquad (3)$$

An Example with Two Agents

Final Error for i



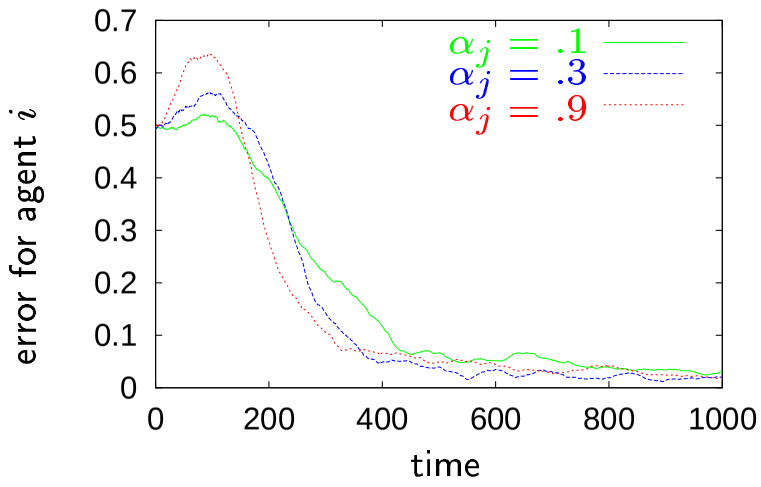
- We plot the final error for agent i after convergence (it always converges).
- Plot of Final Error for agent i, given $l_i = l_j = .2$, $r_i = r_j = 1$, $c_i = c_j = 1$, $|A_j| = |A_i| = 20$.

- We notice that there is a fairly abrupt transition between final errors of 0 and 1. This is characteristic for systems with self-enforcing behaviors.
- The graph is not symmetric. I_{ij} has more weight in determining i's final error than I_{ji}.

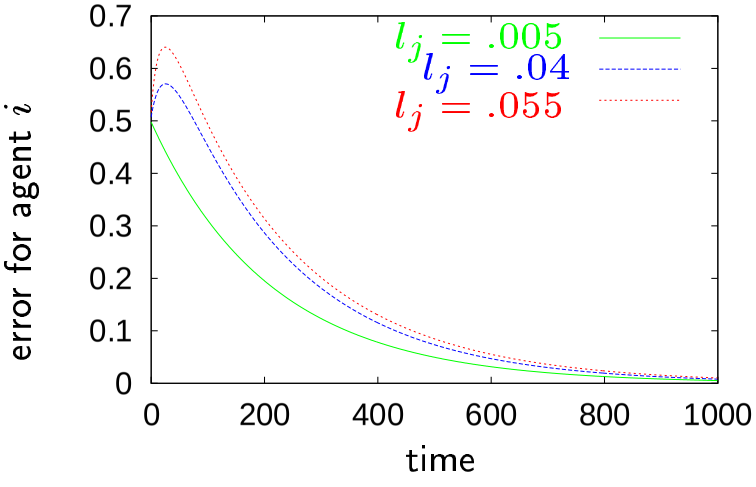
Experiment: Reinforcement Learning Agents

- The game consists of three agents, one buyer and two seller agents i and j.
- The sellers are reinforcement learning agents, with α rates.
- We varied the α_j and found l_j's to match. The other parameters were set according to the problem description. We plot $e(\delta_i^t)$ as a function of time.

Experimental Results



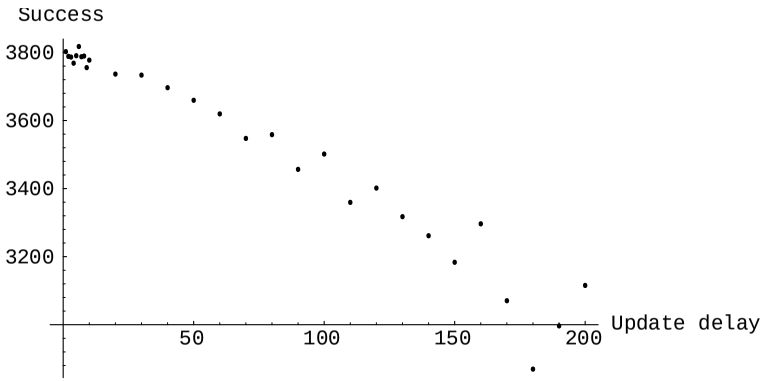
Theoretical Prediction



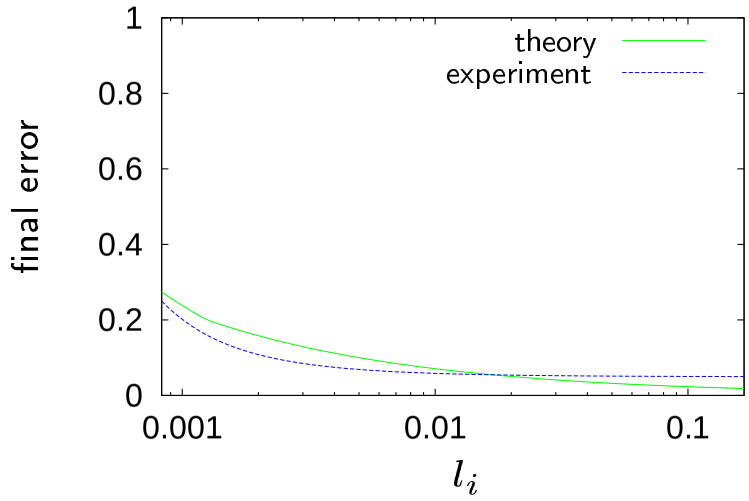
Experiment: An Application to Existing Experimental Results

- We also tested our theory against experimental results from the literature.
- In (Shoham and Tennenholtz, 1997) the authors study a coordination game with 100 agents. They try to see if all agents end up taking the same action.

Experimental Results



Experiment and Theory



Our predictions were, at worst, 5% from their experimental results.

Conclusions

- We gave a predictive framework (CLRI) for calculating the expected error of learning agents in MASs.
- We corroborated our predictions with experimental results.
- A designer of MASs can use CLRI to determine the expected behavior of any one agent in the system, without having to build the whole system.
- We can calculate lower bounds on some of the CLRI parameters, but only for PAC-learning agents.
- **Future work:**
  - Use error distributions instead of expected error (easy?).
  - Formalize mapping from particular learning algorithms to CLRI parameters.
  - Failing that, extend CLRI framework to capture other machine learning techniques (e.g. genetic algorithms, neural networks).

References

Shoham, Y. and Tennenholtz, M. (1997). On the emergence of social conventions: modeling, analysis, and simulations. *Artificial Intelligence*, 94(1):139–166.