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
jmvidal@umich.edu

Abstract

We describe a framework that can be used to model and predict the behavior of MASs with learning agents. It uses a difference equation for calculating the progression of an agent's error in its decision function, thereby telling us how the agent is expected to fare in the MAS. The equation relies on parameters which capture the agents' learning abilities (such as its change rate, learning rate and retention rate) as well as relevant aspects of the MAS (such as the impact that agents have on each other). We validate the framework with experimental results using reinforcement learning agents in a market system, as well as by other experimental results gathered from the AI literature.

1. Introduction

The analysis of multi-agent systems (MASs) composed of learning agents is an important problem because of the steady increase in the number of open MASs [2] [3] which allow for the use of learning agents. Up to now, most of the research in this area has consisted of experiments where multitudes of learning agents are placed in a MAS, then different learning/game parameters are varied, and the results are gathered and analyzed. We have ourselves learned a lot about the dynamics of market-based MASs using this approach [7]. In this paper, we will take a first step forward, beyond experimental results, and describe a framework that can be used to model and predict the behavior of MASs with learning agents. We give a difference equation that can be used to calculate the progression of an agent's error in its decision function, thereby telling us how the agent is expected to fare in the MAS. The equation relies on the values of parameters which capture the agents' learning abilities and the relevant aspects of the MAS. We validate the framework experimentally, as well as by using other experimental results gathered from the AI literature.

*This research has been funded in part by Digital Libraries Initiative under CERA IRI-9411287

We start by describing the type of world we assume the agents inhabit in Section 2. Next, we face the task of modeling the behavior of a learning algorithm; Section 3 describes the parameters we chose for this task. We believe that the parameters in our model will capture the dynamics of a wide variety of learning algorithms. Section 4 formally defines the concept of error, and gives the general equation for calculating an agent's error as time progresses. We can simplify this equation by making some assumptions about conditional independence, as shown in Section 5. This equation includes a term that measures how much the agent's target function changes: the volatility. In Section 6, we show how volatility can be calculated in terms of the other agents' error under the assumption that an agent's target function changes only as a consequence of other agents changing their decision functions (this being the type of MAS we are interested in). An example plot is given in Section 7.

Section 8 shows how our model matches experimental results using reinforcement learning agents in a market system, while Section 9 shows how we can apply our theory and predict experimental results seen elsewhere in the literature.

2. The World

We define a MAS as consisting of a finite number of agents, actions, and world states. We let N denote the set of agents in the system, W denote the set of of world states (i.e. distinct states of the world that the agents can perceive), and A_i, where $|A_i| \geq 2$, denotes the set of actions agent $i \in N$ can take. We assume discrete time, indexed in the various functions by the superscript t, where t is an integer greater than 0.

Each agent $i \in N$ has a **decision function**, given by $\delta_i^t : W \rightarrow A_i$. This function maps each state $w \in W$ to the action $a_i \in A_i$ that i will take in that state. The action that agent i *should* take, assuming knowledge of all other agents' decision functions and i's payoffs, is given by the **target function** $\Delta_i^t : W \rightarrow A_i$. The agent's learning task is to change its decision function so that it matches the target

function. The target function is assumed to be "myopic"; that is, it does not take into account the possibility of future encounters, but instead simply finds the action that maximizes the immediate payoff given the current situation.

3. Modeling a Learning Algorithm

The agents in the MAS are engaged in a discrete action/learn loop that works as follows: At time t the agents perceive a world $w^t \in W$ which is drawn from a fixed distribution $\mathcal{D}(w)$. They then each take the action dictated by their δ_i^t functions, and all of these actions are assumed to be taken in parallel. Lastly, they receive some sort of lesson or reward which their learning algorithm uses to change the δ_i^t so as to (hopefully) better match Δ_i^t . By time $t+1$, the agents have new δ_i^{t+1} functions and are ready to perceive the world again and repeat the loop. Notice that, at time t the Δ_i^t takes into account the δ_j^t of all other agents $j \in N_{-i}$.

The agent's learning algorithm is responsible for changing δ_i^t into δ_i^{t+1} so that it better matches Δ_i^t. Different machine learning algorithms will achieve this with different degrees of success. We have found a set of parameters that can be used to model the effects of a wide range of learning algorithms. We will also assume that each agent uses only one learning algorithm during its lifetime. This means that when we talk about an agent's learning capabilities, we are really talking about the capabilities of the agent's learning algorithm.

After agent i takes an action and receives some reward/lesson, it will try to change its δ_i^t to match Δ_i^t. We can expect that for some w's it was true that $\delta_i^t(w) = \Delta_i^t(w)$, while for some other w's this was not the case. That is, some of the $w \rightarrow a_i$ mappings given by $\delta_i^t(w)$ might have been incorrect. In general, a learning algorithm might affect both the correct and incorrect mappings. We will treat these two cases separately.

We start by considering the incorrect mappings and define the **change rate** of the agent as the probability that the agent will change one of its incorrect mappings. Formally, we define the change rate c_i for agent i as

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

This tells us how likely the agent is to change an incorrect mapping into something else. This "something else" might be the correct action. The probability that the agent changes an incorrect mapping to the correct action is called the **learning rate** of the agent, which is defined as l_i where

$$\forall_w \; l_i = \mathbf{Pr}[\delta_i^{t+1}(w) = \Delta_i^t(w) \mid \delta_i^t(w) \neq \Delta_i^t(w)] \qquad (2)$$

The value of l_i must take into account the fact that the worlds seen at each time step are taken from $\mathcal{D}(w)$. There are two constraints that must always be satisfied by these two rates. Since changing to the correct mapping implies that a change was made, the value of l_i must be less than or equal to c_i, i.e. $l_i \leq c_i$ must always be true. Also, if $|A_i| = 2$ then $c_i = l_i$ since there are only two actions available, so the one that is not wrong must be right. The complimentary value of $1 - l_i$ gives us the probability that an incorrect mapping does not get fixed. As an example, $l_i = .5$ means that, if agent i initially has all mappings wrong, it will get half of them right after the first iteration.

We now consider the agent's correct mappings and define the **retention rate** as the probability that a correct mapping will stay correct in the next iteration. The retention rate is given by r_i where

$$\forall_w \; r_i = \mathbf{Pr}[\delta_i^{t+1}(w) = \Delta_i^t(w) \mid \delta_i^t(w) = \Delta_i^t(w)] \qquad (3)$$

We propose that the behavior of a wide variety of learning algorithms can be captured (or at least approximated) using appropriate values for c_i, l_i and r_i.

Finally, we define **volatility** to mean the probability that the oracle function will change. Formally, volatility is given by v_i where

$$\forall_w \; v_i = \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w)] \qquad (4)$$

In Section 6, we will show how to calculate v_i in terms of the error of the other agents.

4. Calculating the Agent's Error

We define the error of agent i's decision function δ_i^t at time t as:

$$e(\delta_i^t) = \sum_{w \in W} \mathcal{D}(w) \mathbf{Pr}[\delta_i^t(w) \neq \Delta_i^t(w)] \qquad (5)$$

where, as mentioned before, $\mathcal{D}(w)$ is a fixed probability distribution from which the worlds seen are taken. $e(\delta_i^t)$ gives us the probability that agent i will take an incorrect action. It assumes actions are either correct or incorrect.

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 (note: we will be referring to these functions as Δ_i^t and δ_i^t in order to make the equations shorter).

$$\begin{aligned} E[e(\delta_i^{t+1})] = \sum_{w \in W} \mathcal{D}(w) (& \mathbf{Pr}[\Delta_i^{t+1} = \Delta_i^t \mid \delta_i^t = \Delta_i^t] \\ & \cdot \mathbf{Pr}[\delta_i^t = \Delta_i^t] \cdot (1 - r_i) \\ & + \mathbf{Pr}[\Delta_i^{t+1} = \Delta_i^t \mid \delta_i^t \neq \Delta_i^t] \cdot \mathbf{Pr}[\delta_i^t \neq \Delta_i^t] \cdot (1 - l_i) \\ & + \mathbf{Pr}[\Delta_i^{t+1} \neq \Delta_i^t \mid \delta_i^t = \Delta_i^t] \\ & \cdot \mathbf{Pr}[\delta_i^t = \Delta_i^t] \cdot (r_i + (1 - r_i) \cdot B) \\ & + \mathbf{Pr}[\Delta_i^{t+1} \neq \Delta_i^t \mid \delta_i^t \neq \Delta_i^t] \cdot \mathbf{Pr}[\delta_i^t \neq \Delta_i^t] \cdot C \end{aligned} \qquad (6)$$

where we define

$$B = \mathbf{Pr}[\delta_i^{t+1} \neq \Delta_i^{t+1} \mid \delta_i^t = \Delta_i^t \wedge \Delta_i^{t+1} \neq \Delta_i^t \wedge \delta_i^{t+1} \neq \Delta_i^t] \quad (7)$$

$$C = l_i + (1 - c_i)D + (c_i - l_i)F \quad (8)$$

$$D = \mathbf{Pr}[\delta_i^t \neq \Delta_i^{t+1} \mid \delta_i^t \neq \Delta_i^t \wedge \Delta_i^{t+1} \neq \Delta_i^t] \quad (9)$$

$$F = \mathbf{Pr}[\delta_i^{t+1} \neq \Delta_i^{t+1} \mid \delta_i^t \neq \Delta_i^t \wedge \Delta_i^{t+1} \neq \Delta_i^t \wedge \delta_i^{t+1} \neq \Delta_i^t \wedge \delta_i^{t+1} \neq \delta_i^t] \quad (10)$$

Equation (6) will model any MAS whose agent learning can be described with the parameters presented Section 3 and whose action/learn loop is the same as we described. We can use (6) to calculate the successive expected errors for agent i, given values for all the parameters and probabilities. In the next section we show how this is done in a simple example game.

4.1. The Matching game

In this game, we assume that we have two agents i and j each of whom, in every world w, wants to play the same action as the other one. Their set of actions is $A_i = A_j$, where we assume $|A_i| > 2$ (for $|A_i| = 2$ the equation is simpler). After every time step, the agents both learn and change their decision functions in accordance to their learning rates, retention rates, and change rates. Given this information, we can find values for some of the probabilities in (6) (including values for (7) (8) (9) (10)) and rewrite (see Appendix A for derivation) it as:

$$\begin{aligned} E[e(\delta_i^{t+1})] = \sum_{w \in W} \mathcal{D}(w)(& r_j \cdot \mathbf{Pr}[\delta_i^t = \Delta_i^t] \cdot (1 - r_i) \\ &+ (1 - c_j) \cdot \mathbf{Pr}[\delta_i^t \neq \Delta_i^t] \cdot (1 - l_i) \\ &+ (1 - r_j) \cdot \mathbf{Pr}[\delta_i^t = \Delta_i^t] \cdot \left(r_i + (1 - r_i) \cdot \left(\frac{|A_i| - 2}{|A_i| - 1} \right) \right) \\ &+ c_j \cdot \mathbf{Pr}[\delta_i^t \neq \Delta_i^t] \\ &\quad \cdot \left(1 - l_j + \frac{c_i l_j (|A_i| - 1) + l_i (1 - l_j) - c_i}{|A_i| - 2} \right)) \end{aligned} \quad (11)$$

We can better understand this equation by plugging in some values and simplifying. For example, let's assume that $r_i = r_j = 1$ and $l_i = l_j = 1$, which implies that $c_i = c_j = 1$. This is the case where the two agents always change all the $\delta_i^t(w)$ and $\delta_j^t(w)$ that were wrong so that they match their respective target functions at time t. This results in agent i changing all its wrong mappings to match j, and j changing to match i so that all the mappings stay wrong (i ends up doing what j did before while j does what i did before) and the error stays the same. We can see this by plugging the values into (11). The first three terms will become 0 and the fourth term will simplify to the definition of error (5). We then end up with $E[e(\delta_i^{t+1})] = e(\delta_i^t)$.

We can also let c_i and l_i (keeping $c_j = l_j = 1$) be arbitrary numbers, which gives us $E[e(\delta_i^{t+1})] = c_i e(\delta_i^t)$. This tells us that the error will drop faster for a smaller change rate c_i. The reason is that i's learning (remember $l_i \leq c_i$) in this game is counter-productive because it is always made invalid by j's learning rate of 1. That is, since j is changing all its mappings to match i's actions, i's best strategy is to keep its actions the same (i.e. $c_i = 0$).

5. Further Simplification

We can further simplify (6) if we are willing to make two assumptions. The first assumption is that the new actions 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. By making this assumption we can find values for B, D, F and C, namely:

$$B = D = \frac{|A_i| - 2}{|A_i| - 1} \qquad F = \frac{|A_i| - 3}{|A_i| - 2} \quad (12)$$

which makes

$$C = \frac{|A_i| - 2 - c_i + 2l_i}{|A_i| - 1} \quad (13)$$

The second assumption we make is that the probability of $\Delta_i^t(w)$ changing, for a particular w, is conditionally independent of the probability that $\delta_i^t(w)$ was correct. In Section 4.1 we saw that in the matching game the probabilities of $\Delta_i^t(w)$ and $\delta_i^t(w)$ changing were correlated since, if $\delta_i^t(w)$ was wrong then $\delta_j^t(w)$ was also wrong, which meant j would probably change $\delta_j^t(w)$, which would change $\Delta_i^t(w)$.

The matching game is a degenerate example in exhibiting such tight coupling between the agents' oracle functions. In general, we can expect that there will be a number of MASs where the probability that any two agents i and j are correct is uncorrelated (or loosely correlated). For example, in a market system all sellers try to bid what the buyer wants, so the fact that one seller bids the correct amount says nothing about another seller's bid. Their bids are all uncorrelated.

This second assumption we are trying to make can be formally represented by having (14) be true for all pairs of agents i and j in the system.

$$\mathbf{Pr}[\delta_i^t(w) = \Delta_i^t(w) \wedge \delta_j^t(w) = \Delta_j^t(w)] = \mathbf{Pr}[\delta_i^t(w) = \Delta_i^t(w)] \cdot \mathbf{Pr}[\delta_j^t(w) = \Delta_j^t(w)] \quad (14)$$

After making these two assumptions we can rewrite (6) as:

$$E[e(\delta_i^{t+1})] = \sum_{w \in W} \mathcal{D}(w)(\mathbf{Pr}[\Delta_i^{t+1} = \Delta_i^t] \cdot (\mathbf{Pr}[\delta_i^t = \Delta_i^t] \cdot (1 - r_i) + \mathbf{Pr}[\delta_i^t \neq \Delta_i^t] \cdot (1 - l_i)) + \mathbf{Pr}[\Delta_i^{t+1} \neq \Delta_i^t] \cdot (\mathbf{Pr}[\delta_i^t = \Delta_i^t] \cdot \left(r_i + (1 - r_i) \cdot \left(\frac{|A_i| - 2}{|A_i| - 1}\right)\right) + \mathbf{Pr}[\delta_i^t \neq \Delta_i^t] \cdot \left(\frac{|A_i| - 2 - c_i + 2l_i}{|A_i| - 1}\right))) \quad (15)$$

Some of the probabilities in this equation are just the definition of v_i, the others simplify to the agent's error. This means that we can simplify (15) to:

$$E[e(\delta_i^{t+1})] = 1 - r_i + v_i \left(\frac{|A_i| r_i - 1}{|A_i| - 1}\right) + e(\delta_i^t) \left(r_i - l_i + v_i \left(\frac{|A_i|(l_i - r_i) + l_i - c_i}{|A_i| - 1}\right)\right) \quad (16)$$

Eq. (16) is a difference equation that can be used to determine the expected error of the agent at any time by simply using $E[e(\delta_i^{t+1})]$ as the $e(\delta_i^t)$ for the next iteration. While it might look complicated, it is just the function for a line $y = mx + b$ where $x = e(\delta_i^t)$ and $y = e(\delta_i^{t+1})$. Using this observation, and the fact that $e(\delta_i^{t+1})$ will always be between 0 and 1, we determine the final convergence point for the error to be the point where (16) intersects the line $y = x$. The only exception is if the slope equals -1, in which case we will see the error oscillating between two points.

By looking at (16) we can also determine that there are two "forces" acting on the agent's error: volatility and the agent's learning abilities. The volatility tends to increase the agent's error past its current value while the learning reduces it. We can better appreciate this effect by separating the v_i terms in (16) and plotting the v_i terms (volatility) and the rest of the terms (learning) as two separate lines. By definition, these will add up to the line given by (16). We have plotted these three lines and traced a sample error progression in Figure 1. The error starts at .95 and then decreases to eventually converge to .44. We notice the learning curve always tries to reduce the agent's error, as confirmed by the fact that its line always falls below $y = x$. Meanwhile, the volatility adds an extra error. This extra error is bigger when the agent's error is small since, then, any change in the target function is likely to increase the agent's error.

6. Volatility and Impact

Eq. (16) is useful for determining the agent's error when we know the volatility of the system. However, it is likely

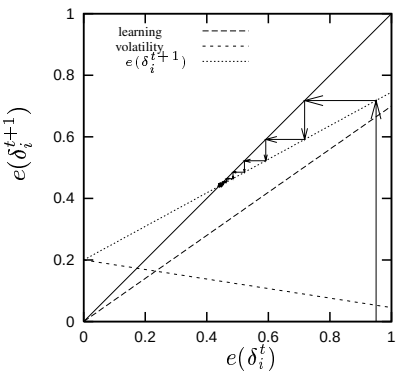


Figure 1. Error progression for agent i, assuming a fixed volatility $v_i = .2$, $c_i = 1$, $l_i = .3$, $r_i = 1$, $|A_i| = 20$. The error converges to .44.

that this value is not available to us (if we knew it we would already know a lot about the dynamics of the system). In this section we determine the value of v_i in terms of the other agents' changes in their decision functions. That is, in terms of $\mathbf{Pr}[\delta_j^{t+1} \neq \delta_j^t]$, for all other agents j.

In order to do this, we first need to define the **impact** I_{ji} that agent j's changes in its decision function have on i.

$$\forall_{w \in W}\ I_{ji} = \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w) \,|\, \delta_j^{t+1}(w) \neq \delta_j^t(w)] \quad (17)$$

We can now start by determining that, for two agents i and j, $\forall_{w \in W}\ v_i$ is defined as:

$$\begin{aligned} v_i^t &= \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w)] \\ &= \mathbf{Pr}[\Delta_i^{t+1} \neq \Delta_i^t \,|\, \delta_j^{t+1} \neq \delta_j^t] \cdot \mathbf{Pr}[\delta_j^{t+1} \neq \delta_j^t] \\ &+ \mathbf{Pr}[\Delta_i^{t+1} \neq \Delta_i^t \,|\, \delta_j^{t+1} = \delta_j^t] \cdot \mathbf{Pr}[\delta_j^{t+1} = \delta_j^t] \end{aligned} \quad (18)$$

The first thing to notice is that volatility is no longer constant, it varies with time (as indicated by the superscript). The first conditional probability in (18) is just I_{ji}, the second one we will set to 0 since we are specifically interested in MASs where the volatility arises *only* as a side-effect of the other agents' learning. That is, we assume that agent i's target function changes only when j's decision function changes. We ignore any other outside influences that might make the agent's target function change.

We can then simplify this equation and generalize it to N agents, under the assumption that the other agents' changes in their decision functions will not cancel each other out, making δ_i^t stay the same as a consequence. This approxi-

mation makes the equations simpler. v_i^t then becomes

$$\forall_{w \in W}\ v_i^t = \mathbf{Pr}[\Delta_i^{t+1}(w) \neq \Delta_i^t(w)] = 1 - \prod_{j \in N_{-i}} (1 - I_{ji}\mathbf{Pr}[\delta_j^{t+1}(w) \neq \delta_j^t(w)]) \quad (19)$$

We now need to determine the expected value of $\mathbf{Pr}[\delta_j^{t+1}(w) \neq \delta_j^t(w)]$ for any agent. Using i instead of j we have

$$\begin{aligned}\forall_{w \in W}\ &\mathbf{Pr}[\delta_i^{t+1}(w) \neq \delta_i^t(w)] \\ &= \mathbf{Pr}[\delta_i^t \neq \Delta_i^t] \cdot \mathbf{Pr}[\delta_i^{t+1} \neq \Delta_i^t | \delta_i^t \neq \Delta_i^t] \\ &+ \mathbf{Pr}[\delta_i^t = \Delta_i^t] \cdot \mathbf{Pr}[\delta_i^{t+1} \neq \Delta_i^t | \delta_i^t = \Delta_i^t]\end{aligned} \quad (20)$$

where the expected value is:

$$E[\mathbf{Pr}[\delta_i^{t+1} \neq \delta_i^t]] = c_i e(\delta_i^t) + (1 - r_i) \cdot (1 - e(\delta_i^t)) \quad (21)$$

We can then plug (21) into (19) in order to get the expected volatility

$$E[v_i^t] = 1 - \prod_{j \in N_{-i}} 1 - I_{ji}(c_j e(\delta_j^t) + (1 - r_j) \cdot (1 - e(\delta_j^t))) \quad (22)$$

We can use this expected value of v_i^t in (16) in order to find out how the other agents' learning will affect agent i. In MASs that have identical learning agents (i.e. their l, c, r, and I rates are all the same and they start with the same initial error) we can replace the multiplier in (22) with an exponent of $|N| - 1$. We use this simplification later in Appendix B.

7. An Example with Two Agents

In a MAS with just two agents i and j, we can use (22) to rewrite (16) as

$$\begin{aligned}E[e(\delta_i^{t+1})] = 1 &- r_i + I_{ji}(c_j e(\delta_j^t) + (1 - r_j) \cdot (1 - e(\delta_j^t))) \\ &\cdot \left(\frac{|A_i| r_i - 1}{|A_i| - 1}\right) \\ &+ e(\delta_i^t)\left(r_i - l_i + I_{ji}(c_j e(\delta_j^t) + (1 - r_j) \cdot (1 - e(\delta_j^t))) \right. \\ &\left. \cdot \left(\frac{|A_i|(l_i - r_i) + l_i - c_i}{|A_i| - 1}\right)\right)\end{aligned} \quad (23)$$

Let's say that $l_i = l_j = .2$, $c_i = c_j = 1$, $r_i = r_j = 1$, $|A_j| = |A_i| = 20$ and we let the impacts I_{ij} and I_{ji} vary between zero and one. Figure 2 shows the final error (after convergence) for this situation. It shows an area where the error is expected to be below .1, corresponding to low values for either I_{ij}, I_{ji} or both. This area represents MASs that are loosely coupled, i.e. one agent's change in behavior does not significantly affect the other one. In these systems we can expect that the error will eventually[1] reach a value close to zero. We see that as the impact increases the final error also increases, with a fairly abrupt transition between a final error of 0 and bigger final errors. This abrupt transition is characteristic of these type of systems where there is are tendencies to either converge or diverge, and both of them are self-enforcing behaviors. Notice also that the graph is not symmetric— I_{ij} has more weight in determining i's final error than I_{ji}. This result seems counterintuitive, until we realize that it is j's error that makes it hard for i to converge to a small error. If I_{ij} is high then, if i has a large error, then j's error will increase, which will make j change its decision function often and make it hard for i to reduce its error. If I_{ij} is low then, even if I_{ji} is high, j will probably settle down to a low error and as it does i will also be able to settle down to a low error.

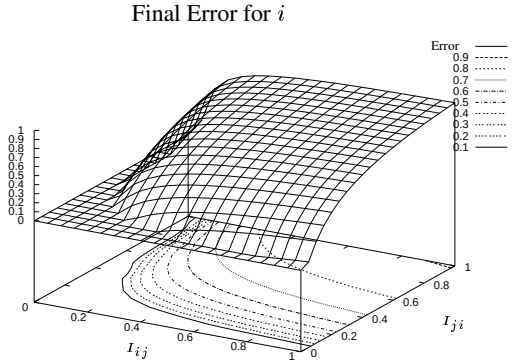


Figure 2. Plot of Final Error for agent i, given $l_i = l_j = .2$, $r_i = r_j = 1$, $c_i = c_j = 1$, $|A_j| = |A_i| = 20$.

Another view of the system is given by Figure 3 which shows a vector plot of the agents' errors. Higher errors are quickly reduced but the pace of learning decreases as the errors get closer to the convergence point. Notice that an agent's error need not change in a monotonic fashion.

8. A Simple Application

In order to demonstrate how our theory might be used, we tested it on a simple market-based MAS. The game con-

[1] Notice that we are not representing how long it takes for the error to converge. This can easily be done and is just one more of the parameters our theory allows us to explore.

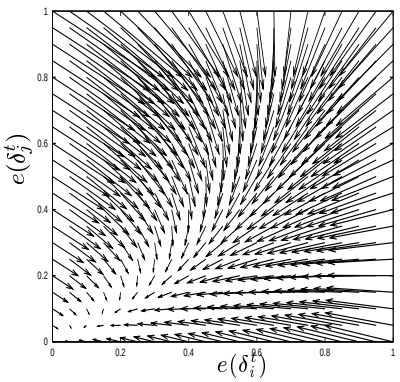


Figure 3. Vector plot for $e(\delta_i^t)$ and $e(\delta_j^t)$, where $|A_i| = |A_j| = 20$, $l_i = l_j = .2$, $r_i = r_j = 1$, $c_i = .5$, $c_j = 1$, $I_{ij} = .1$, $I_{ji} = .3$.

sists of three agents, one buyer and two seller agents i and j. The buyer will always buy at the cheapest price— but the sellers do not know this fact. The sellers can bid any one of 20 prices in an effort to maximize their profits. There is one good being sold ($|W| = 1$).

As predicted by economic theory, the price in this system settles to the sellers' marginal cost, but it takes time to get there due to the learning inefficiencies. The sellers are reinforcement learning agents [7]. We experimented with different α_j rates[2] for the reinforcement learning of agent j, while keeping $\alpha_i = .1$ fixed and plotted the running average of the error of agent i. A comparison is shown in Figure 4. Figure 4(a) gives the experimental results for three different values of α_j. It shows i's average error, over 100 runs, as a function of time. Since both sellers start with no knowledge, their initial actions are completely random which makes their error equal to .5. Then, depending on α_j, i's error will either start to go down from there or will first go up some and then down.

We tried to predict this behavior using (23). Based on the game description, we set $|A_i| = |A_j| = 20$, since there were 20 possible actions. We let $r_i = r_j = 1$ because, in reinforcement learning with fixed payoffs, once an agent is taking the correct action it will never change its decision function to take a different action. The agent might, however, still take a wrong action but only when its exploration rate dictates it.

We then let $I_{ij} = I_{ji} = 1/10$ based on the rough calculation that each agent has an equal probability of bidding any one of the 20 prices. If i is bidding 20 then, if j changes its bid price, i will have to change its price with probability $19/20$ (i.e. it will change it unless j just changed to 20). If i is bidding 19 then this probability is $18/20$, and so on ... The average of all of these is $1/10$. A more precise calculation of the impact would require us to actually run the system and find it via experimentation.

Finally, we chose $l_i = l_j = c_i = c_j = .005$ for the first curve (i.e. the one that compares with $\alpha_j = .1$). We knew that for such a low α_j the learning and change rate should be the same. The actual value was chosen via experimentation. The resulting curve is shown in Figure 4(b). At this moment, we do not possess a formal way of deriving learning and change rates from α-rates.

For the second curve ($\alpha_j = .3$), since only α_j had changed from the first experiment, we knew we should only change l_j and c_j. In fact, these two values should only be increased. We found their exact values, again by experimentation, to be $l_j = .04$, $c_j = .4$. For the third curve we found the values to be $l_j = .055$, $c_j = .8$.

One difference we noticed is that the experimental results show a longer delay before the error starts to decrease. We attribute this delay to the agent's initially high exploration rate. That is, the agents initially start by taking all random actions but progressively reduce this rate of exploration. As the exploration rate decreases, the discrepancy between our theoretical predictions and experimental results is minimized.

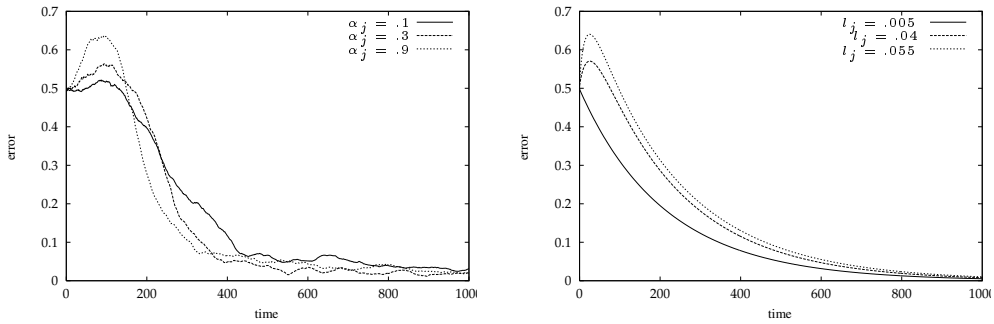
In summary, while it's true that we found l_j and c_j by experimentation, all the other values were calculated from the description of the problem. Even the relative values of l_j and c_j follow the intuitive relation with α_j that, as α_j increases, so does l_j and (even more) c_j. We believe that this experiment provides solid evidence that our theory can be used to approximately determine the quantitative behaviors of MASs with learning agents.

9. Application to Results from the Literature

We have also used our theory to reproduce experimental results from the literature. For example, Figure 5(a) shows experimental results from Shoham and Tennenholtz [6]. They show how the percentage of agents reaching a convention decreases as the delay between applications of the learning algorithm increases. We were able to translate from their units of measurement into ours (see Appendix B) and produce theoretical predictions that were, at worst, 5% off from their experimental results, as seen in Figure 5(b). This level of error can be justified by some of the approximations we had to make in the translation. Notice that the range for l_i is dictated by the experimenters' choice of testing update delays between 0 and 200. Learning rates $l_i \geq .167$ have no experimental counterpart since they imply a delay less than 0.

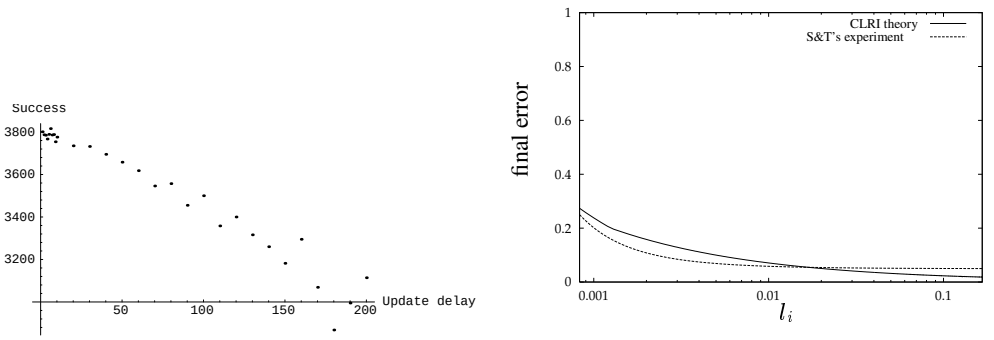
We have also reproduced some of the results in

[2] α is the relative weight the algorithm gives to the most recent reward. $\alpha = 1$ means that it will forget all previous experience and use only the latest reward to determine what action to take.



(a) Experiment (b) Theory

Figure 4. Comparison of observed and predicted error.



(a) Original Experiment (b) Theory and Experiment

Figure 5. Comparing theory (b) with results from Shoham and Tennenholtz (a) [6].

Claus and Boutelier [1], and believe that we can do the same for some of the experiments in Sen *et. al.* [5], and Ishida [4] (see `http://ai.eecs.umich.edu/people/jmvidal/papers/icmas98l.ps` for details on how these reproductions were achieved).

10. Summary

We have presented a framework for studying the behavior of MASs composed of learning agents. We believe that this framework captures the most important parameters that describe the agents' learning and the system's rules of encounter. Various simple applications of the theory were given, along with a comparison with experimental results using reinforcement learning agents. The theoretical predictions were shown to closely match the experimental results. We also reproduced experimental results published by others. These results allow us to more confidently state the effectiveness and accuracy of our theory to a wide variety of machine learning algorithms in different MASs.

Finally, it is unlikely that every single learning algorithm can be formally mapped into our framework. However, this should not deter one from using our theory since we believe that non-intuitive insights can be gained even if the parameter values are only approximations or educated guesses. We are currently exploring the possibility of formally mapping specific learning techniques into our framework, and trying to determine the sensitivity of our predictions to slight errors in the values of the parameters.

References

[1] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Pro-*

ceedings of Workshop on Multiagent Learning. AAAI Press, 1997. http://www.cs.ubc.ca/spider/cebly/Papers/multirl.ps .

[2] P. R. Cohen, A. Cheyer, M. Wang, and C. S. Baeg. An open agent architecture. In *Proceedings of the AAAI Spring Symposium on Software Agents*, pages 1–8. AAAI, 1994.

[3] E. H. Durfee, D. L. Kiskis, and W. P. Birmingham. The agent architecture of the University of Michigan Digital Library. *IEE Proceedings on Software Engineering*, 144(1):61–71, 1997.

[4] T. Ishida. *Real-Time Search for Learnnig Autonomous Agents*. Kluwer Academic Publishers, 1997.

[5] S. Sen, M. Sekaran, and J. Hale. Learning to coordinate without sharing information. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994.

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

[7] J. M. Vidal and E. H. Durfee. Learning nested models in an information economy. *Journal of Experimental and Theoretical Artificial Intelligence: Special Issue Learning in DAI Systems*, 1998. to appear. http://ai.eecs.umich.edu/people/jmvidal/papers/jetai.ps .

A. Derivation for matching game

If we can assume that the action chosen when an agent changes $\delta_i^t(w)$ and the result does not match $\Delta_i^t(w)$ (for some specific w) is taken from a flat probability distribution, then we can say that:

$$B = \frac{|A_i| - 2}{|A_i| - 1} \qquad D = 1 - l_j \tag{24}$$

$$F = l_j + (1 - l_j)\left(\frac{|A_i| - 3}{|A_i| - 2}\right) \tag{25}$$

First, since having $|A_i| = 2$ implies that $c_i = l_i$, this means that for this case we have $C = l_i + (1 - c_i)(1 - l_j)$.

For the case where $|A_i| > 2$, which is the case we are interested in, we can plug in the values for D and F and simplify C to be:

$$C = 1 - l_j + \frac{c_i l_j(|A_i| - 1) + l_i(1 - l_j) - c_i}{|A_i| - 2} \tag{26}$$

B. Shoham and Tennenholtz

In [6], the authors introduce a learning algorithm (strategy-selection rule) called *highest cumulative reward* (HCR) which their agents use for learning conventions. We reproduced experimental results from their Section 4.1 where they study the "coordination game" which is the same as our matching game but with only two actions.

The experiment in question involves 100 agents, all of them identical and all of them using HCR. At each time instant the agents take one of two available actions. The aim is for every pair of chosen agents to take the same action as each other. The authors do not state how the agents are paired up so we will assume that they are all randomly matched. The agents update their behaviors (i.e. apply HCR) after a given delay. The authors try a series of delays (from 0 to 200) and show that increasing the update delay decreases the percentage of trials where, after 1600 iterations, at least 95% of the agents reached a convention. The authors show surprise at finding this phenomenon. Their results are reproduced in Figure 5(a) (cf. Figure 1 in [6]).

The number of actions is $|A_i| = 2$, which implies that $l_i = c_i$. By examining HCR, we determine that $r_i = 1$ (i.e if an agent took the right action, it will only get more support for it). Since there are 100 agents and only pairs of them interact at every time instant, we have $I_{ij} = 1/99$.

In Figure 5(a), we see their x-axis is called the *update delay*. We will refer to it as d. d is the number of time units that pass before the agent is allowed to learn. This means that we must set $l_i = \frac{1}{p(d+1)}$ where $p > 0$. We solve for d and get $d = 1/pl_i - 1$. The value of p depends on their learning algorithm's performance, but we know that it must be a small number (< 50) greater than 0. Through some experimentation we settled on $p = 6$ (other values close to this one give similar results). Since in their graph they look at $0 \leq d \leq 200$, we must then look at l_i where $\frac{1}{1206} \leq l_i \leq \frac{1}{6}$.

The y-axis of Figure 5(a) is the *success* (we call it s), i.e. number of trials, out of 4000, where at least 95% of the agents reached a convention. We know that in $s/4000$ percentage of the trials *at least* 95% of the agents have error near 0 (i.e. reaching a convention means that the agents take the right action almost all the time). We can approximately map this to an error by saying that in $s/4000$ of the trials the error was 0 (a slight underestimate), while in $1 - s/4000$ of the trials the error was 1 (a slight overestimate). We add these two and arrive at an equation that maps s to $e(\delta_i^t)$.

$$e(\delta_i^t) \approx ((4000 - s)/4000) \tag{27}$$

The mapping from d to s is given by their actual data, which can be fitted using the function:

$$s = 3900 - 4d - (d - 100)^2/100 \tag{28}$$

Putting the value of d into (28), and the result into (27), we finally arrive at a function that maps their experimental results into our units (for the range $\frac{1}{1206} \leq l_i \leq \frac{1}{6}$).

$$e(\delta_i) = \frac{4000 - \left(3900 - 4(1/pl_i - 1) - \left(\frac{(1/pl_i - 1 - 100)^2}{100}\right)\right)}{4000} \tag{29}$$

This equation is used to plot the experiment curve in Figure 5(b). For the theory curve, we used (16) and (22), iterated 1600 times and plotted the error at that time.