Agents Learning about Agents: A Framework and Analysis

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 provide a framework for the study of learning in certain types of multi-agent systems (MAS), that divides an agent's knowledge about others into different "types". We use concepts from computational learning theory to calculate the relative sample complexities of learning the different types of knowledge, given either a supervised or a reinforcement learning algorithm. These results apply only for the learning of a fixed target function, which would probably not exist if the other agents are also learning. We then show how a changing target function affects the learning behaviors of the agents, and how to determine the advantages of having lower sample complexity. Our results can be used by a designer of a learning agent in a MAS to determine which knowledge he should put into the agent and which knowledge should be learned by the agent.

Introduction

A designer of a learning agent in a multi-agent system (MAS) must decide how much his agent will know about other agents. He can choose to either implement this knowledge directly into the agent, or to let the agent learn this knowledge. For example, he might decide to start the agent with no knowledge and let it learn which actions to take based on it's experience, or he might give it knowledge about what to do given the actions of other agents and then have it learn which actions the others' take, or he might give it deeper knowledge about the others (if available), etc. It is not clear which one of the many options is better, especially if the other agents are also learning. In this paper we provide a framework for describing the MAS and the different types of knowledge in the agent. We characterize the knowledge as *nested agent models* and analyze the complexity of learning these models. We then study how the fact that other agents are also learning, and thereby changing their behavior, affects the effectiveness of learning the different agent models. Our framework and analysis can be used by the agent designer to help predict how well his agent will perform within a given MAS.

An example the reader can keep in mind is a market economy where agents are buying and selling from each other. Some agents might choose to simply remember the value they got when they bought good x for y dollars, or how profit they made when offering price z (remember, no sale equals zero profit). Others might choose to remember who they bought/sold from and the value/profit they received. Still others might choose to model how the other agents think about everyone else, and so on. It is clear that increased nested models require more computation, what is not so clear is how and when these deeper models will benefit the agent.

Different research communities have run across the problem of agents learning in a society of learning agents. The work of (Shoham & Tennenholtz 1992) focuses on very simple but numerous agents and emphasizes their emergent behavior. The work on agent-based modeling (Hübler & Pines 1994; Axelrod 1996) of complex systems studies slightly more complex agents that are meant as stand-ins for real world agents (e.g. insects, communities, corporations, people). Finally within the MAS community some work (Sen 1996; Tambe & Rosenbloom 1996; Vidal & Durfee 1996) has focused on how artificial AI-based learning agents would fare in communities of similar agents. We believe that our research will bring to the foreground some of the common observations seen in these research areas.

The world and its agents

We assume that the agents inhabit a discrete world with a finite number of states, denoted by the set W. The agents have common knowledge of the fact that every agent can see the state of the world and the actions taken by any other agent. There are n agents, numbered $\{1 \ldots n\} = N$. If we let $\phi(i,j)$ = "agent i can see the action taken by agent j", and $\rho(i,j)$ = "agent i and j both see the same world", then the notation in (Fagin *et al.* 1995) lets us express these ideas more succinctly in terms of common knowledge among the agents in N, using the two following statements: $C_N \forall_{i,j\in N}\phi(i,j)$ and $C_N \forall_{i,j\in N}\rho(i,j)$.

We group the set of all actions taken by all agents in $O = \{A_1, A_2, \ldots, A_n\}$, where A_i is the set of actions that can be taken by agent i and $a_i \in A_i$ is one particular action. We will sometimes assume that $\forall_{i \in N} |A_i| = |A|$. All agents take actions at discrete time intervals and these actions are considered simultaneous and seen by all.

Looking down on a such a system, we see that there is an oracle mapping $M_i(w)$ for each agent i, which returns the best action that agent i can take in state w. However, as we shall see, this function might be constantly changing, making the agents' learning task that much more difficult. We will also refer to a similar function $M_i(w, \vec{a}_{-i})$, which returns the best action in state w, if all other agents take the actions specified by $\vec{a}_{-i}$. It is assumed that the function M is "myopic", that is, it does not take into account the possibility of future encounters, it simply returns the action that maximizes the immediate payoff given the current situation. Some of the limitations imposed by this assumption are relaxed by the fact that agents learn from past experience.

The MAS model we have just described is general enough to encompass a wide variety of domains. Its two main restrictions are its discreteness, and the need for the world and the agents' actions to be completely observable by all agents. It does not, however, say anything about the agents and their structure. We propose to describe the possible agents at the knowledge level and characterize them as 0,1,2...-level modelers. The modeling levels refer to the types of knowledge that these agents keep.

A **0-level agent** is not capable of recognizing the fact that there are other agents in the world. The only way it "knows" about the actions of others is if their actions lead to changes in the world w, or in the reward it gets. At the knowledge level, we can say that a 0-level agent i knows a mapping from states w to actions a_i. This fact is denoted by $K_i(f_i(w))$, where $f_i(w) \rightarrow a_i$. We will later refer to this mapping as the function $g_i(w)$. The goal of the agent is to have $g_i(w) = M_i(w)$. The knowledge can either be known by the agent (i.e. pre-programmed), or it can be learned. We will talk about the complexity of learning in a later Section.

The reader will note that 0-level agents only look at the current world state w when deciding which action to take. It is possible that this information is not enough for making a correct decision. In these cases the 0-level agents are handicapped because of their simple modeling capabilities.

A **1-level agent** i recognizes the fact that there are other agents in the world and that they take actions, but it does not know anything more about them. Given these facts, the 1-level agent's strategy is to predict the other agents' actions based on their past behavior and any other knowledge it has, and use these predictions when trying to determine its best action. Essentially, it assumes that the other agents pick their actions by using a mapping from w to a. At the knowledge level, we say that it knows $K_i(f_i(w, \vec{a}_{-i}))$ where $f_i(w, \vec{a}_{-i}) \rightarrow a_i$, and $K_i K_j(f_{ij}(w))$ for all other agents j, where $f_{ij}(w) \rightarrow a_j$. Again, we can say that the agent's actions are given by the function $g_i(w)$.

Level	Type of Knowledge
0-level	$K_i(f_i(w))$
1-level	$K_i(f_i(w, \vec{a}_{-i}))$ $K_i K_j(f_{ij}(w))$
2-level	$K_i(f_i(w, \vec{a}_{-i}))$ $K_i K_j(f_{ij}(w, \vec{a}_{-j}))$ $K_i K_j K_k(f_{ijk}(w))$

Table 1: The type of knowledge the different agent levels are trying to acquire. They can acquire this knowledge using any learning technique.

A **2-level agent** i also recognizes the other agents in the world, but has some information about their decision processes and previous observations. That is, a 2-level agent has insight into the other agents' internal procedures used for picking an action. This *intentional* model of others allows the agent to dismiss "useless" information when picking its next action. At the knowledge level, we say that a 2-level agent knows $K_i(f_i(w, \vec{a}_{-i}))$, $K_i K_j(f_{ij}(w, \vec{a}_{-j}))$, and $K_i K_j K_k(f_{ijk}(w))$. A simple way a 1-level agent can become 2-level is by assuming that "others are like him", and modeling others using the same learning algorithms and observations the agent, itself, was using when it was a 1-level agent.

We can keep defining **n-level agents** with deeper models in a similar way. An n-level agent i would have knowledge of the type $K_i(f_i(w, \vec{a}_{-i}))$, $K_i K_j(f_{ij}(w, \vec{a}_{-j}))$, $\ldots$, $K_i \cdots K_y(f_y(w, \vec{a}_{-y}))$, $K_i \cdots K_z(f_z(w))$. The number of K's is $n+1$.

Convergence

If we direct our attention to the *impact* that agent actions have on others, we notice that if an agent's choice of best action is not impacted by the other agents' actions then its learning task reduces to that of learning to match the *fixed* function $M_i(w) \rightarrow a_i$. That is, M_i will be fixed if agent i's choice of action depends solely on the world state w. There are many learning algorithms available that allow an agent to learn such a function. We assume the agent uses one such algorithm. From this reasoning it follows that: If agent i's actions are not impacted by other agents and it is capable of learning a fixed function, then it will eventually learn $g_i(w) = M_i(w)$ and will stay fixed after that.

If, on the other hand, the agent's choice of action is impacted by the other agents' actions and the other agents are changing their behavior, then we find that there is no constant $M_i(w)$ function to learn. The MAS becomes a *complex adaptive system*. However, even in this case, it is still possible that all agents will all eventually settle on a fixed set of $g_i(w) = M_i(w)$. If this happens then eventually (and concurrently), the agents will all learn the set of best actions to take in each world state, and these will not change much, if at all. At this point, we say that the system has converged.

Definition 1 *Once all agents have a fixed action function $g_i(w)$, we say that the system has* ***converged****.*

Convergence is a general phenomena that goes by many names. For example, if the system is an instance of the Pursuit Problem, we might say that the agents had agreed on a set of *conventions* for dealing with all situations, while in a market system, we would say that the system had reached a competitive *equilibrium*.

Unfortunately, we do not have any general rules for predicting which systems will converge, and which will not. These predictions can only be made by examining the particular system (e.g. under certain circumstances we can predict that a market system will reach a price equilibrium). Still, we can say that:

Theorem 1 *After a MAS system has converged, then all deeper models (i.e. more than 0-level) become useless. That is, an agent with deeper models can collapse them, keep only 0-level models, and still take the same actions, as long as the system maintains a fixed $M(w)$.*

Proof If M is fixed then, eventually, the knowledge of type $K_i \cdots K_j K_k(f_{i \cdots jk}(w))$ will become fixed, and so will the $K_i \cdots K_j(f_{i \cdots j}(w, \vec{a}_{-j}))$ such that the agent will actually just have a (very complicated) function of w. Therefore, all the knowledge can be collapsed into knowledge of the form $K_i(f_i(w))$ without losing any information. ∎

If, on the other hand, the system has not converged[1] then we find that deeper models are sometimes better than shallow ones, depending on exactly what knowledge the agents are trying to learn, how they are doing it and certain aspects of the structure of this knowledge, as we shall see in the next section.

Sample Learning Complexity

Lets say that a 0-level agent does not have perfect knowledge (i.e. its $K_i(f_i(w))$ does not match the oracle $M(w)$ function), then we know that some or all of it's $w \rightarrow a_i$ mappings must be wrong and need to be learned. If the agent is using some form of supervised learning (i.e. where a teacher tells it which action to take each time), then it is trying to learn one of $|A_i|^{|W|}$ possible 0-level models. If instead it is using some form of reinforcement learning, where it gets a reward (positive or negative) after every action, then it is trying to learn one of $|R|^{|W| \cdot |A_i|}$ possible models, where R is the set of rewards it gets ($|R| \geq 2$). This means that, if the agent is being taught which actions are better, then it just needs to learn the mapping from state w to action a. While, if it gets a reward for each action in each state, then it needs to learn the mapping from state-action (w, a) pairs to their rewards in order to determine which actions lead to the highest reward.

If, on the other hand, a 1-level agent is wrong, then the problem could be either in its $K_i(f_i(w, \vec{a}_{-i}))$, or in its $K_i K_j(f_{ij}(w))$. An interesting case is where we assume that the former knowledge is already known by the agent. This can happen in MASs where the designer knows what the agent should do given what all the other agents will do. So, assuming that $K_i(f_i(w, \vec{a}_{-i}))$ is always correct, we have $K_i K_j(f_{ij}(w))$ as the only source of the discrepancy. Since agents can observe each other's actions in all states, we can assume that they learn this knowledge using some form of supervised learning (i.e. the observed agent is the teacher because it "tells" others what it does in each w). Therefore, in learning this knowledge an agent will be picking from a set of $|A_j|^{|W|}$ possible models.

It should be intuitive (assuming $\forall_{i \in N} |A_i| = |A|$) that the learning problem that the 1-level agent has is the same magnitude as the one the 0-level agent using supervised learning has, but smaller than the reinforcement 0-level agent's problem. However, we can make this a bit more formal by noticing that we can use the size of the hypothesis (or concept) space to determine the *sample complexity* of the learning problem. This give us a rough idea of the number of examples that a PAC-learning algorithm would have to see before reaching an acceptable hypothesis (i.e. model).

We first define the error, at any given time, of agent i's action function $g_i(w)$, as:

$$\text{error}(g_i) = P(g_i(w) \neq M(w) | w \text{ drawn from D}) \leq \epsilon \tag{1}$$

where D is the distribution from which world states w are drawn, and $g_i(w)$ returns the action that agent i will take in state w, given its current knowledge (i.e. all the $K_i(\cdot)$ models). We also let γ be the upper bound we wish to set on the probability that i has a bad model, i.e. one with $\text{error}(g_i) > \epsilon$. The sample complexity is bounded from above by m, whose standard definition from computational learning theory is:

$$m \geq \frac{1}{\epsilon}(\ln \frac{|H|}{\gamma}) \tag{2}$$

where $|H|$ is the size of the hypothesis (i.e. model) space. Given these equations, we can plug in values for one particularly interesting case, and we get an interesting result.

Theorem 2 *In a MAS where an agent can determine which move it should take, given that it knows what all other agents will do, $\forall_i |A_i| = |A|$, and 0-level agents use some form of reinforcement learning, we find that*

[1] If the system contains *unstable* states, then it will never converge. An unstable state is one where there is no set of actions for all agents that constitutes a Nash equilibrium.

the sample complexity of the 1-level agents' learning problem is $O(\ln(|A|^{|W|}))$, while for the 0-level agents' its $O(\ln(|R|^{|A|\cdot|W|}))$. The 0-level agent's complexity is bigger than the 1-level agent's complexity.

Proof We saw before that $|H| = |A|^{|W|}$ for the 1-level agent, and $|H| = |R|^{|A|\cdot|W|}$ for the 0-level with reinforcement-based learning. Using Equation 2 we can determine that the 1-level agent's sample complexity will be less than the 0-level reinforcement agent as long as $|R| > |A|^{1/|A|}$, which is always true because $|R| \geq 2$ and $|A| > 0$. ∎

This theorem tells us that, in these cases, the 1-level will have better models, on average, than the 0-level agent. In fact, we can calculate the size of the hypothesis space $|H|$ for all the different types of knowledge, as seen in Table 2. This table, along with Equation 2, can be used to determine the sample complexity of learning the different types of knowledge for any agent that uses any form of supervised or reinforcement learning. In this way, we can compare two agents to determine which one will have the more accurate models, on average. Please note that some of these complexities are independent of the number of agents (n). We can do this because we assume that all actions are seen by all agents so an agent can build $w \rightarrow a$ models of all other agents in parallel, and assume everyone else can do the same. However, the actual computational costs will increase linearly with each agent, since the agent will need to maintain a separate model for each other agent. The sample complexities rely on the assumption that, between each action, there is enough time for the agent to update its models.

A designer of an agent for a MAS can consult Table 2 to determine how long his agent will take to learn accurate models, given different combinations of implemented versus learned knowledge, and supervised versus reinforcement learning algorithms. However, we can further refine this table by noticing that if a designer has, for example, $K_i(f_i(w, \vec{a}_{-i}))$ knowledge he can actually apply this knowledge when building a 0-level agent. The use of this knowledge will result in a reduction in the size of the hypothesis space for the 0-level agent.

The reduction can be accomplished by looking at the $K_i(f_i(w, \vec{a}_{-i}))$ knowledge and determining which $w \rightarrow a_i$ pairings are impossible and eliminating these from the hypothesis space of the 0-level modeler. That is, for all $w \in W$ and $a_i \in A_i$, eliminate from the table of all possible mappings all the $w \rightarrow a_i$ mappings for which:

1. There does not exist an $\vec{a}_{-i} \in \vec{A}_{-i}$ such that $K_i(f_i(w, \vec{a}_{-i}))$ and $f_i(w, \vec{a}_{-i}) \rightarrow a_i$, i.e. the action a_i is never taken in state w, regardless of what the others do.
2. For all $\vec{a}_{-i} \in \vec{A}_{-i}$ it is true that $K_i(f_i(w, \vec{a}_{-i}))$ and $f_i(w, \vec{a}_{-i}) \rightarrow a_i$, i.e. the agent takes the same action a_i in w no matter what the others do.

After their application, we are left with a new table $T_i : W_i \rightarrow A_i$ with $|W_i| \leq |W|$, and each $w \in W_i$ has a set A_i^w associated with it. We can then determine that, if the new 0-level modeler uses supervised learning, the size of its hypothesis space will be $\prod_{w \in W_i} |A_i^w|$. While, if it uses reinforcement learning, its hypothesis size will be $|R_i|^{|T_i|}$. Table 3(a) summarizes the size of the hypothesis spaces for learning the different types of knowledge given that the designer uses the $K_i(f_i(w, \vec{a}_{-i}))$ knowledge to reduce the hypothesis spaces of other types of knowledge.

Similarly, if the designer also has the knowledge $K_i K_j(f_{ij}(w, \vec{a}_{-j}))$, he creates a reduced table T_j for all other agents. The new hypothesis spaces will then be given by Table 3(b).

For example, a designer for our example market economy MAS can quickly realize that he knows what price his agent should bid given the bids of all others and the probabilities that the buyer will pick each bid. That is, the designer has $K_i(f_i(w, \vec{a}_{-i}))$ knowledge. He also can determine that in a market economy he can not implement a 0-level supervised learning agent because, even after the fact, it is impossible for a 0-level to determine what it should have bid. Therefore, using Theorem 2, the designer will choose to implement a 1-level supervised learning agent and not a 0-level reinforcement learning agent. More complicated situations would be dealt with in a similar way using Table 3.

Learning a moving target

The sample complexities we have been talking about give us an idea of the time it would take for the learning algorithm to learn a *fixed* function. However, in a lot of MASs the function that the agents are trying to learn is constantly changing, mostly because other agents are also learning and changing their behaviors. Our results still apply because an agent that takes longer to learn a fixed function will also take longer to learn a changing function, since this problem is broken down to the problem of learning different fixed functions over time. Still, we wish to know more about the relative effectiveness of learning algorithms with different sample complexities, when learning target functions that change at different rates.

The first thing we need to do is to characterize the rate at which the learning algorithm learns, and the rate at which the target function changes. We do this using a differential equation that tells us what the error (as defined in Equation 1) of the model will be at time $t+1$ given the error at time t.

The learning algorithm is trying to learn the function f, where $f \in H$ is merely one of the possible models that the agent can have. If this function has an error of 1 it means that none of its mappings is correct (i.e. it takes the wrong action in every state). An agent with such a function at time t, will observe the next action and will definitely learn something since all its mappings are incorrect. Its error at time $t+1$ will

Level	Knowledge	Supervised Learning	Reinforcement Learning
0-level	$K_i(f_i(w))$	$\|A_i\|^{\|W\|}$	$\|R_i\|^{\|A_i\|\cdot\|W\|}$
1-level	$K_i(f_i(w, \vec{a}_{-i}))$	$\|A_i\|^{\|A_1\|\cdots\|A_{i-1}\|\|A_{i+1}\|\cdots\|A_n\|\|W\|}$	$\|R_i\|^{\|A_1\|\cdots\|A_n\|\|W\|}$
	$K_iK_j(f_{ij}(w))$	$\|A_j\|^{\|W\|}$	$\|R_j\|^{\|A_j\|\|W\|}$
2-level	$K_i(f_i(w, \vec{a}_{-i}))$	$\|A_i\|^{\|A_1\|\cdots\|A_{i-1}\|\|A_{i+1}\|\cdots\|A_n\|\|W\|}$	$\|R_i\|^{\|A_1\|\cdots\|A_n\|\|W\|}$
	$K_iK_j(f_{ij}(w, \vec{a}_{-j}))$	$\|A_j\|^{\|A_1\|\cdots\|A_{j-1}\|\|A_{j+1}\|\cdots\|A_n\|\|W\|}$	$\|R_j\|^{\|A_1\|\cdots\|A_n\|\|W\|}$
	$K_iK_jK_k(f_{ijk}(w))$	$\|A_k\|^{\|W\|}$	$\|R_k\|^{\|A_k\|\|W\|}$

Table 2: Size of the hypothesis spaces $|H|$ for learning the different sets of knowledge, depending on whether the agent uses supervised or reinforcement learning. A_i is the set of actions and R_i is the set of rewards for agent i, n is the number of agents, and W the set of possible world states.

Lvl	Knowledge	**(a)** Supervised Learning	**(a)** Reinforcement Learning	**(b)** Superv. Learning	**(b)** Reinf. Learning
0	$K_i(f_i(w))$	$\prod_{w \in W_i'} \|A_i^w\|$	$\|R_i\|^{\|T_i\|}$	$\prod_{w \in W_i'} \|A_i^w\|$	$\|R_i\|^{\|T_i\|}$
1	$K_i(f_i(w, \vec{a}_{-i}))$	Known	Known	Known	Known
	$K_iK_j(f_{ij}(w))$	$\|A_j\|^{\|W\|}$	$\|R_j\|^{\|A_j\|\|W\|}$	$\prod_{w \in W_j'} \|A_j^w\|$	$\|R_j\|^{\|T_j\|}$
2	$K_i(f_i(w, \vec{a}_{-i}))$	Known	Known	Known	Known
	$K_iK_j(f_{ij}(w, \vec{a}_{-j}))$	$\|A_j\|^{\|A_1\|\cdots\|A_{j-1}\|\|A_{j+1}\|\cdots\|A_n\|\|W\|}$	$\|R_j\|^{\|A_1\|\cdots\|A_n\|\|W\|}$	Known	Known
	$K_iK_jK_k(f_{ijk}(w))$	$\|A_k\|^{\|W\|}$	$\|R_k\|^{\|A_k\|\|W\|}$	$\|A_k\|^{\|W\|}$	$\|R_k\|^{\|A_k\|\|W\|}$

Table 3: Size of the hypothesis spaces $|H|$ for learning the different types of knowledge. The (a) columns assume the designer already has $K_i(f_i(w, \vec{a}_{-i}))$ knowledge, while in (b) he also has $K_iK_j(f_{ij}(w))$. If knowledge is known then $|H| = 0$.

drop accordingly. But, as it learns more and more, the probability that the next action it observes is not one that it already knows keeps decreasing and, therefore, its error at time $t+1$ keeps getting closer and closer to its error at time t. We model this behavior by assuming that the error e_l of a learning algorithm l at time $t+1$ is given by

$$e_l(t+1) = \lambda \cdot e_l(t) \tag{3}$$

where $0 \leq \lambda \leq 1$ is the slope of the learning curve. λ can be determined from the sample complexity and the learning algorithm used. In general, it is safe to say that bigger sample complexities will lead to bigger λ values. If $\lambda = 1$ then the algorithm never learns anything, if $\lambda = 0$ it learns everything from just one example. We have plotted such a function in Figure 1 for[2] $\lambda = .6$.

In a similar way, the moving target will introduce an error to the current model at each time. If the current error is 1 then no amount of moving will make it bigger (by definition). While, as the error gets closer to 0, the moving target function is expected to introduce larger errors. We can characterize the error function e_m for the moving target M with the differential equation:

$$e_m(t+1) = v + (1-v)e_m(t) \tag{4}$$

[2]In reality, we expect λ to be much closer to 1. The smaller number simply makes the picture more readable.

v is where the line intersects the x-axis and the slope is $(1-v)$. We have also shown this curve in Figure 1, for $v = .1$. It is easy to see that if $v = 0$ then the target function is always fixed, while if $v = 1$ it is moving in the worst possible way. We use v as a measure of the *velocity* at which the target function moves away from the learned model.

In Figure 1 we trace the successive errors for a model given the two curves, assuming that the curves have been defined using the same time scales. We see that the error will converge to some point or, rather, a pair of points. These points can be determined by combining Equation 3 and 4 into one error function.

$$e(t+1) = \lambda v + \lambda(1-v)e(t) \tag{5}$$

The solution such that $e(t+1) = e(t)$ is

$$e_{\min} = \frac{\lambda v}{1 - \lambda(1-v)} \tag{6}$$

$$e_{\max} = v + e_{\min}(1-v) = \frac{v}{1-\lambda(1-v)} \tag{7}$$

Equation 6 gives us the minimum error that we expect given e_l and e_m. It corresponds to the error right after the agent has tried to learn the target function. Equation 7 gives the error right after the target moves. We now have the necessary math to examine the differences in the learning and the moving target error functions. Lets say we have two agents with different sample complexities and, therefore, different learning

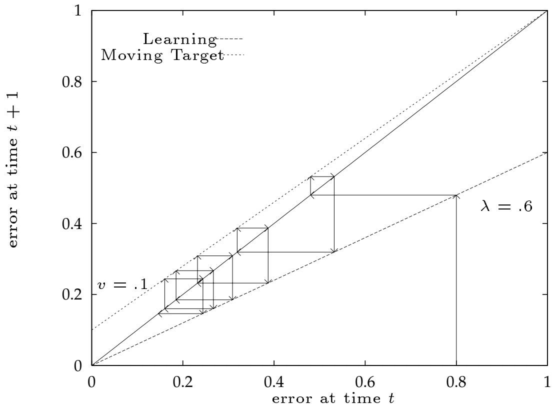

Figure 1: Plot of the errors for the Learning function e_l and the Moving Target function e_m for $\lambda = .6$, and $v = .1$.

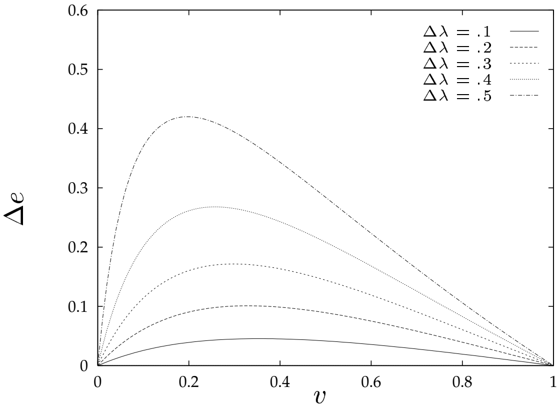

Figure 2: Difference in the error of an agent with high sample complexity ($\lambda + \Delta\lambda$) minus one with low sample complexity (λ), as given by Equation 8, plotted as a function of v.

rates λ and $\lambda + \Delta\lambda$. Using Equation 7 we can calculate the difference in their expected maximum errors.

$$\Delta e = \frac{v}{1 - (\lambda + \Delta\lambda)(1 - v)} - \frac{v}{1 - \lambda(1 - v)} \qquad (8)$$

This is a rather complicated equation so we have plotted it in Figure 2 for different values of $\Delta\lambda$ and $\lambda = .4$. We notice that, for small values of v, Δe increases linearly as a function of v, but it quickly reaches a maximum and starts to decrease. This means that the difference in the maximum expected error between the agent with the lower sample complexity (i.e. smaller λ), and the one with higher sample complexity, will increase as a function of v for small values of v, and will decrease for bigger values. Still, we notice that Δe is always a positive value, and is only 0 for $v = 0$ and $v = 1$, an unlikely situation.

Theorem 3 *The advantages of agents with lower sample complexities will be more evident in moderately dynamic/changing systems.*

Proof Equation 8 shows that $\Delta e > 0$ for all legal values of $\Delta\lambda$, given our assumptions about the learning and moving target error functions. Note that, even thought the lower sample complexities always lead to smaller errors, only for the smaller values of v do we find a correlation between increased v and increased difference in error Δe. ∎

This theorem leads us to look for metrics (e.g. *price volatility* in a market system) that can be used to determine how fast the system changes (i.e. the v) and, in turn, how much better the models of the agents with smaller sample complexity will be. For example, the designer of a 1-level agent in a market system might, after some time has passed, notice that the price volatility in the system is close to zero. He should then, using Theorem 3, consider making his agent a 0-level modeler.

Discussion

We presented a framework for structuring the knowledge of an agent learning about agents, and then determined the complexities of learning the different types of knowledge, and the advantages of having lower complexity in a MAS where the other agents are also learning. These results can be used by a designer of such an agent to help decide which knowledge should be put into the agent and which one learned. Our analysis also takes a first step into trying to elucidate some of the *emergent* phenomena we might expect in these types of MAS, e.g. the results predicted by Theorem 3 were previously observed in a MAS implementation by (Vidal & Durfee 1996), and are reminiscent of similar results in (Hübler & Pines 1994).

References

Axelrod, R. 1996. The evolution of strategies in the iterated prisoner's dilemma. Cambridge University Press.

Fagin, R.; Halpern, J. Y.; Moses, Y.; and Vardi, M. Y. 1995. *Reasoning About Knowledge.* MIT Press.

Hübler, A., and Pines, D. 1994. *Complexity: Methaphors, Models and Reality.* Addison Wesley. chapter Prediction and Adaptation in an Evolving Chaotic Environment, 343–379.

Sen, S., ed. 1996. *Working Notes for the AAAI Symposium on Adaptation, Co-evolution and Learning in Multiagent Systems.*

Shoham, Y., and Tennenholtz, M. 1992. Emergent conventions in multi-agent systems. In *Proceedings of Knowledge Representation.*

Tambe, M., and Rosenbloom, P. S. 1996. Agent tracking in real-time dynamic environments. In *Intelligent Agents Volume II.*

Vidal, J. M., and Durfee, E. H. 1996. The impact of nested agent models in an information economy. In *Proceedings of the Second International Conference on Multi-Agent Systems.* `http://ai.eecs.umich.edu/people/jmvidal/papers/amumdl/`.