MultiAgent Dynamical Systems

Ben Johnson
Department of Mathematics

bjohnson@math.ucdavis.edu

June 6, 2008

Abstract We explore the learning dynamics of a multiagent system in the specific case of two-agent interactions, modeled via the rock-paper-scissors game. We develop a simulation that allows for the interactive modification of system parameters, and a deeper understanding of system dynamics. The hope is that this will allow us to draw conclusions about systems involving larger numbers of more complex agents.

Introduction Multiagent systems appear in various forms in fields such as computer science, engineering, and animal biology. They can be naturally occurring, such as a flock of birds or a herd of gazelle, or they can be artificially created, such as a distributed computer network, or a collection of programmable robots. Given the possible size of any of these systems, it is not unreasonable to assume that a detailed analysis of their dynamics would be quite difficult. Moreover, the agents involved are themselves very complicated, and would likely prove difficult to model in a realistic or meaningful way.

To this end, we look at what could be described as one of the simplest multiagent systems: the two agent system. It is certainly simple in terms of size: it consists only two agents in a static environment. But it is also simple in terms of agent complexity: agents can only choose from one of three actions, and their behavior is mitigated by three parameter values. These limitations on system size and agent complexity will hopefully make for a more feasible, yet still robust, analysis of dynamic behavior.

By studying this system, we hope to gain an understanding into how agents learn and how their behavior evolves over time. We would like to develop some manner of predictive ability about their future behavior, if not deterministically, then at least in some probabilistic way. The hope is that, if successful, this would allow for the development of multiagent systems that would exhibit a particular desired behavior. That is, it might be possible to create a simple agent, that when combined with other agents, is capable of accomplishing significantly more complex tasks than it could if acting alone.

Our results indicate that predicting the time evolution of a multiagent system is a very difficult task. Small parameter variations, even for our relatively simple agents, lead to significant changes in systems dynamics. It would seem likely that larger systems with more complex agents would exhibit similar behavior.

Background In the system we explored, an agent can be loosely described as an autonomous participant with a myopic view of the surrounding environment. This simply means that the agent makes its own decisions, and that it is unaware of the system in which it lives. In fact, the presence of the other agent is felt only in terms of the reward received for choosing a particular action. If we return to the examples of multiagent systems mentioned earlier, we have birds as the agents in the flock system, and individual computers as agents in the distributed network system. They are not agents in the exact same sense as the agents in our system, as they may be aware of other agents, but they are similar enough for the comparison to still be meaningful.

Each agent can choose one of three actions, which can be interpreted as rock, paper, or scissors. Just as in the classic game, rock beats scissors, scissors beat paper, and paper beats rock. Then for each action, there is a result: a 'win', a 'loss', or a 'draw'. As the agents are unaware of a game being played, they interpret these results in terms of a reward: positive, negative, and sometimes neutral. Each agent has a memory of its own actions and the corresponding result. Due to their myopic view, the agents behave in a self-interested manner, always seeking to maximize their own positive reward, and thus minimize the negative ones. It is the interplay between how quickly they learn and what they remember that produces the dynamics of this system.

Dynamical system We begin our description by explaining the discrete time equations that govern single agent behavior, then we extend this to a continuous time model and ultimately, the two agent model studied here. We describe the single agent model in the interest of greater understanding, noting that we did not perform any analysis of this model. We also present the two agent model in greater generality than it was

studied, making the appropriate clarifications where necessary. Also, remember that 'reward' in the context of this discussion does not have its usual connotation. It represents a reinforcement, not necessarily positive, for an agent's decision to choose a particular action.

An agent chooses from one of N possible actions at each time step. (Remember, we are working with a discrete time model, and for our agent, $N = 3$). Let τ denote denote the number of steps from the initial system state. Then at step τ, the agent has a certain probability of choosing action i from the set of N actions. Denote this probability as $x_i(\tau)$. As the agent must choose from one of the N actions at each step, we have that $\sum_{i=1}^{N} x_i(\tau) = 1$. At each time step τ, there is a corresponding reward for choosing action i, denoted $r_i(\tau)$. Finally, the agent's memory for action i taken at step τ is denoted $Q_i(\tau)$. The memory is updated according to

$$Q_i(\tau+1) - Q_i(\tau) = \frac{1}{T}\left[\delta_i(\tau) r_i(\tau) - \alpha Q_i(\tau)\right) \qquad \text{where } \delta_i(\tau) = \begin{array}{ll} 1 & \text{if action } i \text{ is chosen at time } \tau \\ 0 & \text{otherwise} \end{array},$$

with the restriction that $Q_i(0) = 0$ for $i = 1, 2, ..., N$. This simply represents the agent starting with no memory of prior decisions. Here, $\alpha \in [0, 1)$ denotes the agent's memory loss, with $\alpha = 0$ representing perfect memory.

We can rearrange the equation for memory updates into a more intuitive form as

$$Q_i(\tau+1) = Q_i(\tau)\left(1 - \frac{\alpha}{T} Q_i(\tau)\right) + \frac{1}{T}\left(\delta_i(\tau) r_i(\tau)\right).$$

This allows us to understand the memory update in terms of previous memory state. We see that if $\alpha = 0$, the updated memory is the old memory plus a term that is nonzero when the agent takes that particular action, as we expect. We also see that for $\alpha \neq 0$, the agent only remembers a percentage of previous rewards, again as we expect for memory loss. Since the equations make at least some intuitive sense, we now extend the discrete model to a continuous-time version. We omit the steps that transform this to a continuous-time model, as they are not important to this analysis, and simply state the equations. Notationally, we use $x(\tau)$ to represent the vector $(x_1(\tau), x_2(\tau), ..., x_N(\tau))$, with $r(\tau)$ and $Q(\tau)$ taken to have analogous interpretations.

The governing equation for the single-agent model is then

$$\frac{\dot{x_i}}{x_i} = \beta\left(R_i - R\right) + \alpha\left(H_i - H\right),$$

where $R = \sum_{i=1}^{N} x_i R_i$ is the net reward averaged over all possible actions. Thus, as the agent wishes to increase positive rewards, it will seek to take an action where $R_i - R$ is positive. The other term $H = \sum_{i=1}^{N} x_i \log x_i$ represents the average informativeness in choosing action i. The difference, $H_i - H$, represents the relative surprise of the agent choosing action i. This term serves to increase the uncertainty in the agents actions (which intuitively makes sense, as there should be uncertainty due to memory loss). Then when $\alpha = 0$, we see that the agent's choice is based solely on which offers the greatest reward, as we would expect. The parameter $\beta \in [0, \infty)$ represents the agent's rate of adaptation or learning. A larger value for β represents faster adaptation and greater confidence in the decisions being made.

We now move on to the two agent case. The equations are very similar to the single agent case, except the equations are coupled, which represents the interaction between the two agents. Notationally, the terms in the two-agent model have the same meaning as in the single agent model, we simply denote with superscript

and subscript X's and Y's those parameters that belong to agent X and Y, respectively. The equations for the two agent system are:

$$\frac{\dot{x_i}}{x_i} = \beta_X \left(R_i^X - R^X\right) + \alpha_X \left(H_i^X - H^X\right) \qquad \text{for } i = 1, 2, ..., N$$

$$\frac{\dot{y_j}}{y_j} = \beta_Y \left(R_j^Y - R^Y\right) + \alpha_Y \left(H_j^X - H^Y\right) \qquad \text{for } j = 1, 2, ..., M$$

where

$$R^X = \sum_{k=1}^{N} x_k R_k^X, \qquad R^Y = \sum_{k=1}^{M} y_k R_k^Y$$

$$H^X = \sum_{k=1}^{N} x_k H_k^X, \qquad H^Y = \sum_{k=1}^{M} y_k H_k^Y.$$

While these equations do govern the dynamics of the system, it turns out that it is easier to visualize the system if we use what is called the information space. The equations of motion in this transformed space are:

$$x_i = \frac{e^{-u_i}}{\sum_{k=1}^{N} e^{-u_k}}$$

$$y_i = \frac{e^{-v_i}}{\sum_{k=1}^{M} e^{-v_k}}$$

$$\dot{\mathbf{u}} = -\beta_X A\mathbf{y} - \alpha_X \mathbf{u}$$

$$\dot{\mathbf{v}} = -\beta_Y B\mathbf{x} - \alpha_Y \mathbf{v}$$

where

$$A = \begin{bmatrix} \frac{2}{3}\epsilon_X & 1 - \frac{1}{3}\epsilon_X & -1 - \frac{1}{3}\epsilon_X \\ -1 - \frac{1}{3}\epsilon_X & \frac{2}{3}\epsilon_X & 1 - \frac{1}{3}\epsilon_X \\ 1 - \frac{1}{3}\epsilon_X & -1 - \frac{1}{3}\epsilon_X & \frac{2}{3}\epsilon_X \end{bmatrix} \qquad B = \begin{bmatrix} \frac{2}{3}\epsilon_Y & 1 - \frac{1}{3}\epsilon_Y & -1 - \frac{1}{3}\epsilon_Y \\ -1 - \frac{1}{3}\epsilon_Y & \frac{2}{3}\epsilon_Y & 1 - \frac{1}{3}\epsilon_Y \\ 1 - \frac{1}{3}\epsilon_Y & -1 - \frac{1}{3}\epsilon_Y & \frac{2}{3}\epsilon_Y \end{bmatrix}.$$

A and B are the respective normalized reward matrices for agent X and Y. Note here that this is the case for $N = 3$, not for a general multiagent system. This third parameter ϵ, represents the tie breaking parameter in the event of both agents choosing the same action (e.g. rock-rock). We require $\epsilon \in [-1, 1]$. We can see that depending on the values of ϵ_X and ϵ_Y, agents X and Y could interpret a tie as anything from a loss to a win.

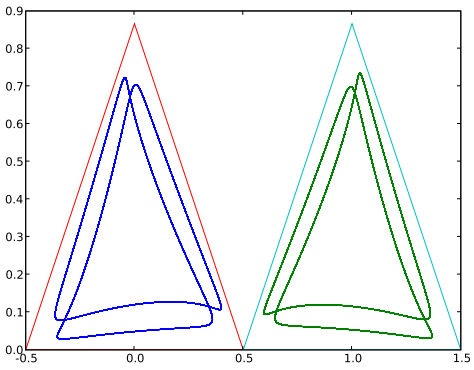
Methods Our first investigative approach was to create a model of the system using Python. We used the equations of motion for the information space, as mentioned above, and integrated the system with fixed time step $dt = .1$, using 4th-order Runge-Kutta. This allowed us to create static images of the information space, and by varying parameters, we could begin to elucidate the system dynamics. But we found this method of analysis somewhat lacking. Static images of the information space allowed us to see the the long term evolution of the system, but they didn't allow us to explore the immediate change in system dynamics due to parameter variation. As such, we created an interactive model of the system using Python's Tkinter GUI packages. This model plotted the 'real-time' dynamics of the system, which was a significant improvement to the initial version. But more importantly, it allowed us to interactively vary the parameters of the system, and see the immediate response in the agents' behavior. This tool allows the user to specify initial conditions for each agent, and it provides sliders to adjust the α and β parameters for each agent. Additional, but not implemented features of this program, would include the ability to vary the ϵ parameter for each agent, as well as using different time steps, and possible a different integrator, such as Euler's method.

Results We found that this system exhibited a wide range of dynamical and chaotic behavior. Small changes in any one of the parameters led to drastically different long term behavior of the system, as did a different choice of initial condition for a fixed set of parameters. We use a number of graphs below to highlight the salient features of the system, and the results of small parameter and initial condition variations. See Figures 1-3.

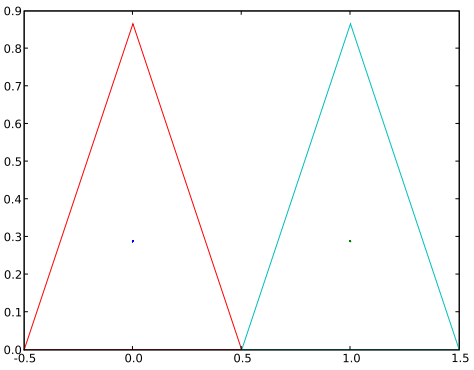
We also found that by varying certain parameters, would could gain a greater intuition for the behavior of the system. For example, if $\beta_Y \gg \beta_X$, agent Y's trajectories tends to stay toward the edges of the simplex, while X's trajectories tend to stay a little close to the interior. This simulate agent Y's greater confidence in it's choices, as compared to agent X. These results are found in Figures 4 and 5.

Conclusion It would seem that our goal of being able to predict long-term behavior in this system is ill-fated. Small changes to any of the system parameters can result in significantly different long term behavior. We found that given the right choice of parameters, the system could exhibit convergence to a fixed point or limit cycle, or it could never converge and be completely chaotic. As ours is one of the simplest multiagent systems with one of the simplest agent models possible, and we were still unable to concretely determine long-term behavior, it would be very difficult to imagine this being possible for a larger and more complex system.

Bibliography Y. Sato, E. Akiyama, and J. P. Crutchfield, "Stability and Diversity in Collective Adaptation," Journal of Theoretical Biology (2004)

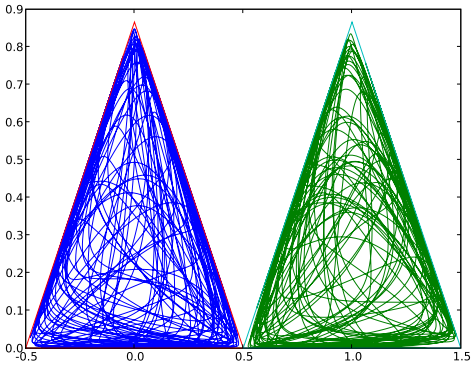


(a) $\alpha_X = .0198$, $\alpha_Y = .01$, $\beta_X = 1$, $\beta_Y = 1$, $\epsilon_X = .5$, $\epsilon_Y = -.5$.

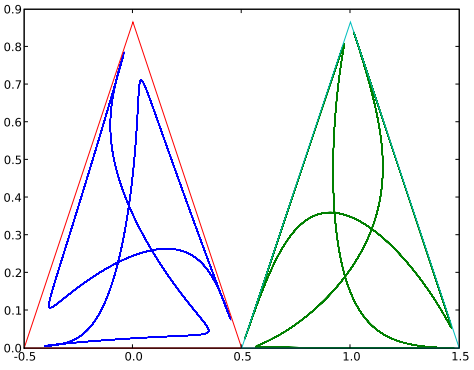


(b) Same parameters as (a) except $\alpha_Y = .02$. Settled to a fixed point, visible at the center of each simplex.

Figure 1: All system values except α_Y were held constant. Small variation changed trajectory from limit cycle to fixed point.



(a) Same parameters as 1(a), except $\epsilon_Y = -.3$.



(b) Same parameters as 1(a) except $\beta_Y = 2.0$.

Figure 2: By varying the parameters ϵ_Y and β_Y, would could significantly change the long term behavior of the system. This include the transition from a limit cycle to a chaotic orbit (a), or to a different limit cycle (b).

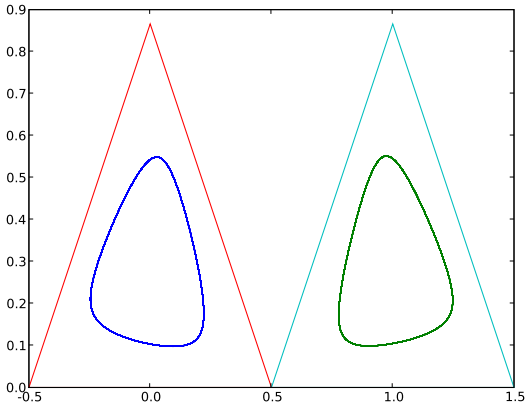
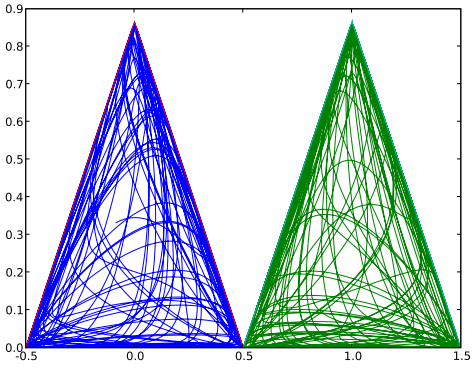
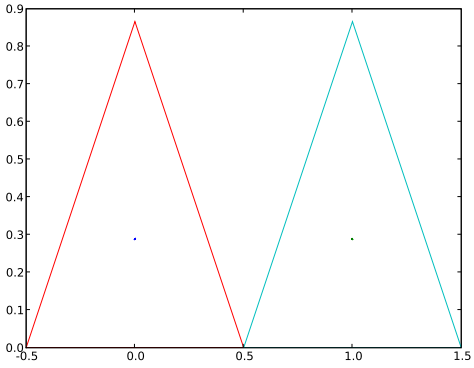


Figure 3: Same as 1(a), except with slightly different initial conditions.

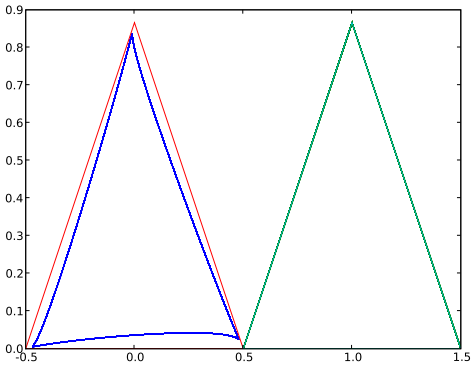


(a) Same parameters as 1(a), except $\alpha_X = 0$.

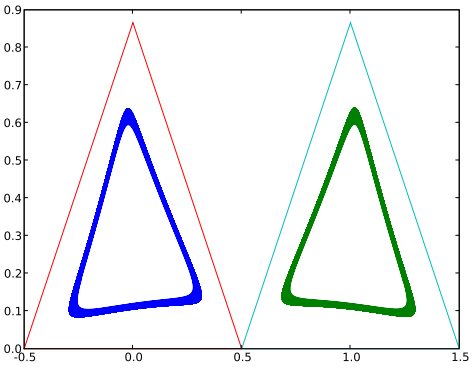


(b) Same parameters as 1(a) except $\alpha_X = .3$.

Figure 4: The effect of large and small values of α_X. Large values of α serve to randomize the decision process. We see that for $\alpha_X = .3$, the system converges to the fixed point $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, which is complete random decision making.



(a) Same parameters as 1(a), except $\beta_Y = 5$. The trajectories for agent Y stay on the of the simplex, while those of agent X stay noticably further away, due to agent X's relative lack of confidence in choosing a particular action.



(b) Same parameters as 1(a) except $\beta_X = \beta_Y = .85$.

Figure 5: The effect of large and small values of β_Y on the long term behavior of the system. For small β values, the trajectories of each agent stays close to the center of the simplex, evidence of the lack of confidence (and thus general randomness) of the choice being made. Large β has the trajectories much closer to 'pure' action choices.