

# Notes on Learning

Leigh Tesfatsion  
Department of Economics  
Iowa State University  
Ames, Iowa 50011-1070  
<http://www.econ.iastate.edu/tesfatsi/>  
tesfatsi@iastate.edu

19 February 2007

“Shopbots, agents that automatically search the Internet for goods and/or services on behalf of consumers, herald a future in which autonomous agents become an essential component of nearly every facet of electronic commerce. . . . Moreover, we predict the emergence of *pricebots* - economically motivated agents that set prices so as to maximize the profits of firms, just as shopbots seek prices that minimize costs for consumers.” Greenwald and Kephart (1999, p. 506).

## 1 Introduction

Roughly defined, *social learning* involves learning from the experiences of other agents whereas *individual learning* is learning on the basis of one’s own past and possibly unique experiences. Intermediate between these two polar cases are situations in which agents engage in aspects of each form of learning. For example, one might have *type learning* in which agents of several distinct types interact with each other and hence influence each other’s fitnesses through influences on payoffs, but the agents only learn from successful agents of their own type.

A key aspect of learning, whether social, individual, or something in between, is the amount of anticipation (look-ahead) that agents employ. At one extreme, an agent might rely entirely on *backward-looking stimulus and response learning* for choosing its actions.

A state-of-the-world (stimulus) occurs, and the agent responds to this state by choosing a particular action (response). The agent then observes an outcome, and it uses this outcome to either weaken or strengthen the association between the state and the action in the future. In this scenario, then, the agent continuously adapts to its environment by asking the question: If this state occurs, what action should I take? However, the agent does not deliberately attempt to modify its environment to suit its own purposes.

At the other extreme, an agent in a particular state  $s$  might instead rely on *anticipatory learning* for choosing a current action. For example, suppose the agent has a current estimate for the probability  $P(r_i|a, s)$  of each possible outcome  $r_i, i = 1, \dots, N$ , conditional on each possible choice of its current action  $a$  as well as the current state  $s$ . Let  $u(r_i|a, s)$  denote the utility that the agent would attain if the outcome  $r_i$  were realized subsequent to action  $a$  being taken, conditional on the current state  $s$ . The agent then chooses a current action  $a^* = a(s, P)$  that maximizes its *conditional expected utility*

$$U(a|s, P) = \sum_{i=1}^N u(r_i|a, s)P(r_i|a, s). \quad (1)$$

The frequency distributions of the actual outcomes realized over time for each given action  $a$  and state  $s$  are used to construct continually updated estimates for the outcome probability assessments  $P(r|a, s)$ , perhaps starting from initially given priors.

In the anticipatory learning case, the agent deliberately attempts to modify its environment to suit its own purposes by asking the following forward-looking question: If I take this action now, what outcomes might occur in the future? The agent chooses its actions in an attempt to increase the likelihood that the ensuing outcomes will be favorable to itself.

One complicating factor in the design of anticipatory learning algorithms is the well-known *dual decision problem* from adaptive control theory: namely, from a longer-run point of view, it might be optimal for an agent to sacrifice short-term reward for information gain through experimentation, for this might ultimately allow it to make more informed future

decisions because it has better estimates of the true outcome probability distributions. A second complicating factor arising in multi-agent decision contexts is potential infinite regress as each agent tries to take into account what other agents are doing; see Tesauro and Kephart (1998).

These notes briefly discuss several types of learning representations currently being used by ACE researchers. To facilitate understanding, a relatively simple market economy is first outlined in Section II. Section III presents and illustrates several basic reinforcement learning algorithms, and Section IV outlines the basic features of Q-learning. Section V briefly reviews the main features of the standard genetic algorithm (GA) and discusses the distinction between GA social learning and GA individual learning. In each of these sections, the discussion concludes by asking how the learning representation might be applied by the quantity and price setting firms in the market economy presented in Section II.

## 2 An Illustrative Market Economy

Consider an economy that exists over time periods  $T = 1, 2, \dots$ . The economy includes one consumer with income  $I = \$100$  per period who eats beans and hash (fried potatoes!). The utility the consumer receives from consuming beans and hash in amounts  $b$  and  $h$  is measured by a utility function  $U(b, h)$  that strictly increases with increases in either  $b$  or  $h$ .

The economy also includes a collection of hash-producing firms who sell hash in each period at a common publicly-known unit price  $p_H$  and who produce hash each period to satisfy whatever amount of hash the consumer demands. Finally, the economy includes two identically structured firms, F1 and F2, who produce cans of beans  $b_1$  and  $b_2$  at unit prices  $p_1$  and  $p_2$ , respectively. Each bean firm has a constant marginal cost of production given by  $MC = \$2$  per can, so that its total cost function is  $C(b) = MC \cdot b$ . Also, each bean firm has a maximum production capacity of  $BMax = 8$  cans per period. Finally, each bean firm has

an initial wealth of  $W(1) = \$1000$  at the beginning of period  $T = 1$ . Beans are a perishable good and so cannot be stored from one period to the next.

The bean industry operates as a seller posted-offer auction. Thus, at the beginning of each period  $T$ , firms F1 and F2 post quantity-price offers  $A_1(T)$  and  $A_2(T)$  respectively, where these offers take the form

$$A_1(T) = [b_1(T), p_1(T)] ; \tag{2}$$

$$A_2(T) = [b_2(T), p_2(T)] . \tag{3}$$

These posted offers are costlessly and immediately available to the consumer, along with the price  $p_H$  for hash.

At the beginning of each period  $T$ , after viewing the posted bean offers (2) and (3), along with the hash price  $p_H$ , the consumer solves the following utility-maximization problem: Maximize

$$U(b_1^d(T) + b_2^d(T), h^d(T)) \tag{4}$$

with respect to the choice of  $b_1^d(T)$  (demand for beans from F1),  $b_2^d(T)$  (demand for beans from F2), and  $h^d(T)$  (demand for hash) subject to the budget and feasibility constraints

$$p_1(T)b_1^d(T) + p_2(T)b_2^d(T) + p_H h^d(T) \leq I ;$$

$$0 \leq b_1^d(T) \leq b_1(T) ;$$

$$0 \leq b_2^d(T) \leq b_2(T) .$$

The solution to problem (4) yields the consumer's demands for beans and hash. By construction, these demands depend on the offers (2) and (3), the hash price  $p_H$ , and the consumer's income  $I$ . In particular, the demands for beans from firms F1 and F2 take the following forms:

$$b_1^d(T) = D_1(T) = D_1(A_1(T), A_2(T), p_H, I) ; \tag{5}$$

$$b_2^d(T) = D_2(T) = D_2(A_1(T), A_2(T), p_H, I) . \tag{6}$$

The consumer communicates the bean demand  $b_1^d(T)$  to firm F1 and the bean demand  $b_2^d(T)$  to F2 after the firms have posted their offers  $A_1(T)$  and  $A_2(T)$ .

At the end of each period  $T$ , all actual sales of beans to the consumer (separately by firm) become publicly available information. If the revenues of either bean firm in period  $T$  fail to cover its costs (i.e., if either firm has negative profits in period  $T$ ), this firm must pay the remainder of its costs out of its current wealth (i.e., its initial wealth  $W(1)$  plus accumulated profits and losses). If a bean firm's wealth level ever drops below zero, the firm is insolvent and must exit the market.

Each bean firm's objective is to select its offers to maximize its profits (revenues minus costs) over time. Each bean firm knows the consumer's income in each period. However, at the beginning of period  $T = 1$ , neither bean firm has any direct information regarding the consumer's demand functions for beans. The problem for each bean firm, then, is to decide on a strategy for selecting its offers over time that is feasible given its actually available information, that keeps the bean firm in business, and that generates as much profit as possible.

Three distinct types of market conduct will be considered for the bean firms: pure cartel (single-price collusion); duopoly (simultaneous posting with no explicit collusion); and leadership pricing (one firm posts before the other).

## 2.1 Pure Cartel

A *cartel* is a group of firms who explicitly coordinate their actions in order to raise the market price [see Pepall et al. (Chapter 7, 1999)]. Suppose the bean firms F1 and F2 collude on a single bean price  $P_B(T)$  in each period  $T$ , i.e.,  $P_B(T) = p_1(T) = p_2(T)$ . Suppose also, for simplicity, that the consumer's aggregate bean demand  $B^d(T) = b_1^d(T) + b_2^d(T)$  in any period  $T$  takes the following simple linear form, where dependence on  $p_H$  and  $I$  has been

subsumed into the value of the demand function ordinate  $a$  and slope  $-v$ :

$$B^d(T) = D_B(A_1(T), A_2(T)) = \begin{cases} a - v \cdot P_B(T) & \text{if } a - v \cdot P_B(T) < b_1(T) + b_2(T); \\ b_1(T) + b_2(T) & \text{if } a - v \cdot P_B(T) \geq b_1(T) + b_2(T). \end{cases} \quad (7)$$

If the bean firms in period  $T$  knew the demand function (7), they could determine optimal levels for price  $P_B(T)$  and quantity  $B(T) = (b_1(T) + b_2(T))$  by solving the following problem:

Maximize joint profits

$$(P_B(T) - MC) \cdot D_B(A_1(T), A_2(T)) \quad (8)$$

with respect to the choice of  $P_B(T)$ ,  $A_1(T)$  and  $A_2(T)$  subject to the constraints

$$\begin{aligned} A_j(T) &= (b_j(T), p_j(T)) \quad , \quad j = 1, 2; \\ P_B(T) &= p_1(T) = p_2(T) \quad ; \\ b_1(T) + b_2(T) &= D_B(A_1(T), A_2(T)) \quad ; \\ MC &\leq P_B(T) \quad ; \\ 0 &\leq b_1(T) \leq BMax \quad ; \\ 0 &\leq b_2(T) \leq BMax \quad . \end{aligned}$$

Assuming the final three constraints in problem (8) are not binding, a necessary condition for  $B^*(T) = b_1^*(T) + b_2^*(T)$  and  $P_B^*(T)$  to solve this problem is that the bean firms' mark-up of price over cost satisfy:

$$\frac{P_B^*(T) - MC}{P_B^*(T)} = \frac{1}{e^*(T)} \quad , \quad (9)$$

where, letting  $\Delta$  denote "incremental change in," the aggregate bean demand elasticity  $e^*(T)$  is given by

$$e^*(T) = - \frac{\Delta B^*(T)}{\Delta P_B^*(T)} \cdot \frac{P_B^*(T)}{B^*(T)} \quad . \quad (10)$$

Note that  $P_B^*(T) > 0$  requires  $e^*(T) > 1$ , implying that optimal cartel production must take place along the elastic portion of the aggregate demand function for beans.

A problem for bean firms F1 and F2, however, is that they do not know the consumer's aggregate demand function for beans in each period  $T$ , as required to solve problem (8). Various ways the bean firms might try to learn over time to produce in accordance with problem (8) are examined in subsequent sections.

## 2.2 Symmetric Duopoly

Suppose, instead, that the bean firms F1 and F2 separately post quantities (production levels) and unit prices for beans at the beginning of each period  $T$ . These postings occur at the same time, so that neither firm has a strategic informational advantage. Suppose, also, that the consumer always prefers to buy beans from the bean firm with the *lowest* price.

For simplicity, consider the case in which bean capacity constraints are not binding. In this case the aggregate consumer demand for beans can be expressed as  $B^d = D_B(p_L)$ , where  $L$  denotes the bean firm (either 1 or 2) with the lowest price, and  $p_L$  denotes this lowest price. Consequently, the demand curve facing bean firm F1 can be expressed as

$$D_1(A_1, A_2) = \begin{cases} D_B(p_1) & \text{if } p_1 < p_2 \text{ ;} \\ D_B(p_1)/2 & \text{if } p_1 = p_2 \text{ ;} \\ 0 & \text{if } p_1 > p_2 \text{ .} \end{cases} \quad (11)$$

The demand curve facing bean firm F2 takes a similar form.

If both bean firms know the aggregate demand function  $D_B(p_L)$ , it follows from (11) that neither bean firm will want to set a higher price than the other. Moreover, if either bean firm's price  $p_j$  exceeds marginal cost  $MC$ , the other firm would generally be able to "steal the market" by undercutting this price by a suitably small amount  $\epsilon$  so that  $p_j - \epsilon \geq MC$  and  $D_B(p_j - \epsilon) > D_B(p_j)/2$ . Consequently, to be a (pure) Nash equilibrium, the bean firms' offers would essentially have to satisfy the "competitive pricing" condition  $p_1 = p_2 = MC$ .

Recall, however, that the bean firms in period  $T = 1$  are assumed *not* to know anything directly about consumer demand. How, then, should each bean firm choose its offer? Do the above observations imply that the bean firms have nothing to learn in this market setting,

and that they should simply set their prices equal to MC? Not necessarily, because this Nash equilibrium will generally be Pareto dominated by other feasible selections  $P_B$  for the common bean price level  $P_B = p_1 = p_2$ . For example, through some form of learning over time, the two bean firms might eventually be able to coordinate on the optimal pure cartel solution  $(B^*, P_B^*)$  (see above). A difficulty in ensuring coordination over time on this Pareto-dominating solution, however, is that it is *not* a Nash equilibrium. That is, as explained above, given any solution in which both bean firms are producing at a common price that exceeds their marginal cost  $MC$ , each firm will perceive short-term gains from cheating on this solution by an arbitrarily small undercutting of its rival's price.

### 2.3 Leader-Follower Duopoly

Finally, suppose in each period  $T$  that bean firm F1 must post its offer  $A_1(T)$  *before* firm F2 posts its offer  $A_2(T)$ . If the consumer's bean demand functions  $D_1(A_1(T), A_2(T))$  and  $D_2(A_1(T), A_2(T))$  were known to each bean firm in each period  $T$ , the problems facing each profit-maximizing firm would then take the following forms.

**Problem Facing bean firm 2 (the “Follower”) in Period  $T$ :** Given  $A_1(T) = (b_1(T), p_1(T))$ , solve:

$$\begin{aligned} \max_{A_2(T)} [p_2(T) - MC] \cdot D_2(A_1(T), A_2(T)) \quad \text{subject to} \quad (12) \\ A_2(T) &= (b_2(T), p_2(T)) \ ; \\ b_2(T) &= D_2(A_1(T), A_2(T)) \ ; \\ MC &\leq p_2(T) \ ; \\ 0 &\leq b_2(T) \leq BMax \ . \end{aligned}$$

The solution to problem (12) gives firm F2's profit-maximizing offer  $A_2(T)$  conditional on firm F1's offer  $A_1(T)$ . The function  $A_2 = R(A_1)$  expressing this dependence of  $A_2$  on  $A_1$  is called the *reaction function* of firm F2.

**Problem Facing bean firm 1 (the “Leader”) in Period  $T$ :** Assume firm F1 knows (or can calculate) firm F2’s reaction function  $A_2 = R(A_1)$ . Then firm F1 solves:

$$\max_{A_1(T)} [p_1(T) - MC] \cdot D_1(A_1(T), R(A_1(T))) \quad \text{subject to} \quad (13)$$

$$\begin{aligned} A_1(T) &= (b_1(T), p_1(T)) \quad ; \\ b_1(T) &= D_1(A_1(T), R(A_1(T))) \quad ; \\ MC &\leq p_1(T) \quad ; \\ 0 &\leq b_1(T) \leq BMax \quad . \end{aligned}$$

Given known consumer bean demands and the *absence* of binding capacity constraints in period  $T$ , the follower bean firm F2 *must* set  $p_2(T) \leq p_1(T)$  if it is to have any chance for positive profits. On the other hand, if capacity constraints *can* bind, it might be profitable for F2 to set  $p_2(T) > p_1(T)$  in reaction to some offers  $A_1(T)$  in order to service remaining “residual bean demand” at a higher price.

How might bean firms F1 and F2 reasonably determine their offers in each period  $T$  if they lack any direct information about consumer bean demands in period  $T = 1$ ? Note that the approximate solution of problem (13) for bean firm F1 would now require F1 to form some kind of estimate of the reaction function of bean firm F2, which in turn would require F1 to consider the manner in which F2 is selecting its offers. This in general would represent a very difficult type of anticipatory learning problem for F1.

### 3 Reinforcement Learning

The basic intuition underlying *reinforcement learning (RL)* is that the tendency to implement an action should be strengthened (reinforced) if it results in favorable outcomes and weakened if it results in unfavorable outcomes [Sutton and Barto (2000)].

More precisely, RL is a simple type of learning algorithm in which agents construct associations between states and actions in the form of if-then behavioral rules: i.e., if this state occurs, then this is the action to be taken. When a particular if-then rule is implemented, the agent uses the ensuing outcome to update the probability that the if-then rule will be implemented again in the future. If the outcome is relatively good, the probability of future implementation is increased; if the outcome is relatively bad, the probability of future implementation is decreased.

RL choice problems can be divided into two types: *nonsequential* and *sequential*. In nonsequential choice problems, an agent must learn a mapping from states to actions that maximizes expected immediate reward. In sequential choice problems, the agent must again learn a mapping from states to actions, but now the actions selected by the agent may influence future situations and hence future rewards as well as immediate rewards. Consequently, it might be advantageous for the agent to engage in anticipatory evaluation of the future possible consequences of its current actions.

Sequential choice problems involve a *credit assignment problem*. A sequence of actions takes place before the long-term consequences are known. Credit for these long-term consequences must somehow be allocated among the individual actions in the action sequence. This is difficult because the effects of the individual actions on the long-term consequences might not be independent of each other; that is, there might be joint effects.

In the early RL literature, the focus was generally on a single agent playing a game against nature. That is, a single agent was assumed to face a choice problem in which his uncertainty regarding which action to choose arose from an exogenous source modelled as a probability distribution that was independent of the agent's action choices.

An example of a game against nature is the famous *two-armed bandit (TAB) problem*; see Berry and Fristedt (1985) for a survey. The TAB problem appears to have originated with Thompson (1933), but the study of the TAB problem only began in earnest following

a highly influential study by Robbins (1952). An agent must repeatedly decide whether to pull a left lever or a right lever on a two-armed bandit machine. Each lever corresponds to a different exogenously given probability of winning a specified reward, but the agent does not know these reward probabilities at the beginning of the game. The agent uses the rewards that actually ensue from his sequence of lever pulls to construct estimates of these reward probabilities. Ideally, the agent would like to be able to determine the arm corresponding to the “best” probability distribution for rewards (e.g. the arm offering the highest expected utility of return) so that all further lever pulls can be concentrated on this one arm only.

Recently, theoretical game theorists have begun to explore the use of RL in multi-agent contexts as a “selection principle” for determining the selection of a particular Nash equilibrium when multiple Nash equilibria exist (Fudenberg and Levine, 1998). Also, RL algorithms are being used to explain experimental data obtained from human subjects who are learning to play repeated games; see Roth and Er’ev (1995) and Er’ev and Roth (1998). Learning among multiple strategically interacting agents is far more difficult to study than learning in games against nature since the choice environment of each agent is now intrinsically nonstationary.

The following subsections summarize several different types of RL algorithms that are currently being used by agent-based computational modelers. The first “Derivative-Follower” algorithm is perhaps more correctly termed an *adaptive* algorithm rather than an RL algorithm since its initially specified state-action associations (in rate of change form) remain invariant over time. The remaining algorithms are RL algorithms that begin with an initially specified collection of state-action associations and then update these associations over time on the basis of experienced rewards.<sup>1</sup>

---

<sup>1</sup>There is an essential ambiguity in the distinction between adaptation and learning on the basis of whether or not structural changes are occurring over time in state-action associations. Suppose an agent begins with a particular state-action association  $S \rightarrow A$  in a given environment  $E$ . Suppose some change occurs in his environment ( $E \rightarrow E^*$ ) that induces the agent to change this association to  $S \rightarrow A^*$ . This appears to be a structural change in a state-action association, i.e., learning. However, if the state is extended to  $(S, E)$ , it

### 3.1 Derivative-Follower Algorithm

The *derivative-follower (DF) algorithm* developed by Greenwald and Kephart (1999) for selecting a scalar action is a computationally simple and informationally undemanding algorithm that focuses on reinforcement of action movement in desirable directions. The DF decision maker experiments with incremental increases (or decreases)  $\Delta a$  in some scalar action  $a$ , continuing to move  $a$  in the same direction until the observed reward level falls, at which point the direction of movement in  $a$  is reversed.

The selection of the step size  $\Delta a$  can be adapted to the problem at hand. In each of their reported experiments, Greenwald and Kephart (1999) randomly select  $\Delta a$  from a uniform probability distribution over the interval from 0.01 to 0.02 and then apparently maintain this same step size throughout the course of the experiment. However, as with any gradient ascent method, it might also be desirable to consider a step size that varies over the course of the experiment to protect against entrapment in the neighborhood of a local optimum.

Findings from computational market experiments reported by Kephart et al. (2000) suggest that homogenous populations of DF decision makers do relatively well in comparison with other types of homogeneous learning populations, maintaining the highest prices and the highest profits. However, when mixed together with other types of learners, DF decision makers do not necessarily fare the best. These preliminary findings are reminiscent of the findings for the simple Tit-for-Tat strategy for playing the iterated prisoner's dilemma game [Axelrod (1984)]. By construction, Tit-for-Tat cannot beat any rival; but it tends to do relatively well on average in games played against a wide variety of different strategy populations.

#### *DF-Algorithm Discussion Questions:*

Consider a variation of the economy outlined in Section II in which the bean firms

---

is seen there has been *no* change in the two associations  $(S, E) \rightarrow A$  and  $(S, E^*) \rightarrow A^*$ .

F1 and F2 post unit bean prices at the beginning of each period and then “produce to order” in response to received consumer demands for beans. How might the firms apply the DF algorithm in each of the three indicated cases (pure cartel, symmetric duopoly, leader-follower duopoly) to generate their posted prices in the absence of any a priori knowledge about consumer demand except the knowledge that demand curves are generally downward sloping?

For example, consider the pure cartel case with a linear downward-sloping aggregate demand curve for beans given by (7). How might the perfectly colluding firms F1 and F2 use the DF algorithm to decide which bean price  $P_B$  to post in each period in an attempt to discover the optimal pure cartel bean price  $P_B^*$ ? Would the DF algorithm *always* permit them to discover this price? Why or why not?

The DF algorithm can in principle be generalized to handle action vectors of arbitrary finite dimension taking on arbitrarily many finite values. Note, however, that the specification of the step sizes then becomes much more problematic. Search essentially proceeds along a one-dimensional curve in a higher-dimensional action space, where the step sizes *jointly* determine the direction of the search curve at each point in time. In many problem contexts, some search curves will be more reasonable than others, hence it will generally not be desirable to specify the step sizes for the components of the action vector independently of one another.

Consider, for example, the pure cartel case in Section II in which the colluding bean firms F1 and F2 face the aggregate bean demand function (7) and must post both an aggregate bean quantity  $B$  and a unit bean price  $P_B$  at the beginning of each period. What the firms would presumably like to do is search along the demand function for the optimal pure cartel solution  $(B^*, P_B^*)$  where their joint profits are maximized. However, this demand function is a priori unknown to them. What can they do?

Suppose firms F1 and F2 at least know that their demand function is *linear*, i.e., it has

the general form depicted in (7) for some unknown values of  $a$  and  $v$ . In this case the firms can approximately locate the optimal pure cartel solution by applying a simple variant of the DF algorithm, modified to handle the choice of 2-dimensional action vectors  $(B, P_B)$ . They can first locate, by trial and error, two points  $(B', P')$  and  $(B'', P'')$  on the demand function, which will then allow them to determine the demand function slope  $-v$  by setting  $-v = [B' - B'']/[P' - P'']$ . They can then *jointly* set the step sizes  $\Delta B$  and  $\Delta P$  to satisfy  $\Delta B/\Delta P = -v$ . Starting from any point on the demand function, and using these step sizes, the DF algorithm will then guide the firms along the demand function in the “right” (i.e. profit increasing) direction until the optimal pure cartel solution is approximately located. The smaller the step sizes, the better the approximation will be.

What might the firms F1 and F2 do in the more difficult case in which they face an unknown demand function which is *not* necessarily linear? Can a variant of the DF algorithm still be constructed that will reliably guide the firms to the optimal pure cartel solution?

### 3.2 Gibbs/Boltzmann Reinforcement Learning

Another formulation of RL maps past rewards into future actions through a *Gibbs (or Boltzmann) probability distribution*. The description below of this RL algorithm closely follows Bell (2001).

For a simple illustration, consider a situation in which an agent can take only one of two possible actions in each time period. Given any time period  $t$ , the agent is characterized by: (a) a vector of weights for the two actions:  $w_t = (w_{1t}, w_{2t})$ ; and (b) a vector that records the number of times the agent has taken each action to date:  $n_t = (n_{1t}, n_{2t})$

The probabilities that the agent takes actions 1 and 2 at time  $t$  are given by

$$p_{1t} = \frac{e^{w_{1t}/T_t}}{e^{w_{1t}/T_t} + e^{w_{2t}/T_t}}; \tag{14}$$

$$p_{2t} = \frac{e^{w_{2t}/T_t}}{e^{w_{1t}/T_t} + e^{w_{2t}/T_t}}. \tag{15}$$

Here  $T_t$  denotes a time-dependent “temperature” parameter that decreases as a function of  $t$  starting from some positive initial value  $T_0$  and approaches some minimal nonnegative value  $\bar{T}$  as  $t$  becomes arbitrarily large. For example, letting  $\mu$  (the stepsize) denote any number lying strictly between 0 and 1, a possible functional specification for the determination of  $T_t$  is

$$T_{t+1} = \max\{\mu T_t, \bar{T}\} . \quad (16)$$

Given (16), note that it becomes more likely that the action  $j$  with the largest weight  $w_{jt}$  is chosen as  $t$  increases (hence  $T_t$  decreases).

Finally, let  $r_{it}$  denote the positive or negative reward at time  $t$  for taking action  $i$ ,  $i = 1, 2$ , and let  $I_{it}$  denote an indicator function that takes on the value 1 when action  $i$  is chosen at time  $t$  and 0 otherwise. The weights for the actions are then updated according to the following rules:

$$w_{1,t+1} = \frac{[n_{1t} - I_{1t}]}{n_{1t}} \cdot w_{1t} + \frac{[I_{1t} \cdot r_{1t}]}{n_{1t}} ; \quad (17)$$

$$w_{2,t+1} = \frac{[n_{2t} - I_{2t}]}{n_{2t}} \cdot w_{2t} + \frac{[I_{2t} \cdot r_{2t}]}{n_{2t}} . \quad (18)$$

*Gibbs/Boltzmann Learning Discussion Questions:* The Gibbs/Boltzmann RL algorithm can in principle be generalized to handle action vectors of arbitrary finite dimension taking on arbitrarily many finite values. How might the two bean firms in the market economy described in Section II fare if they both attempted to use such a learning algorithm for determination of their quantity and price offers? Would they tend to do better or worse in comparison with DF decision makers? What about the case in which one bean firm uses the DF algorithm and the other bean firm uses Gibbs/Boltzmann learning?

### 3.3 Er'ev and Roth Reinforcement Learning

This subsection briefly outlines two RL algorithms developed by Roth and Er'ev (1995) and Er'ev and Roth (1998). The form of these RL algorithms is based on the human-subject experimental work of the psychologists Bush and Mosteller (1955). Camerer and Ho (1999) have extended these RL algorithms to include anticipatory belief-based learning. See Feltovich (2000) for a detailed comparison between reinforcement-based and belief-based learning.

Consider a situation in which  $N$  players are engaged in a repeated game. In each period  $t$ , each player  $n$  can choose from among  $M_n$  pure strategies. In the initial period  $t=1$ , the initial propensity of player  $n$  to choose his  $k$ th pure strategy is given by a nonnegative *initial propensity*  $q_{nk}(1)$ ,  $k = 1, \dots, M_n$ . These initial propensities are assumed to be equal valued:

$$q_{nk}(1) = q_{nj}(1) \text{ for all pure strategies } j, k . \quad (19)$$

The probability that player  $n$  chooses any particular pure strategy  $k$  in the initial period 1 is then given by the *relative* propensity associated with  $k$ :

$$p_{nk}(1) = \frac{q_{nk}(1)}{\sum_j q_{nj}(1)} = 1/M_n . \quad (20)$$

Let  $r_{nk}(t)$  denote the immediate reward (positive or negative) attained by player  $n$  in period  $t$  if he chooses pure strategy  $k$  in period  $t$ , and let  $I_{nk}(t)$  denote an indicator function that takes on the value 1 if player  $n$  chooses pure strategy  $k$  in period  $t$  and 0 otherwise. At the end of each period  $t$ ,  $t \geq 1$ , the propensity that player  $n$  associates with pure strategy  $k$  is updated in accordance with the following rule:

$$q_{nk}(t+1) = q_{nk}(t) + I_{nk}(t)r_{nk}(t) . \quad (21)$$

Consequently, the current propensity associated with any pure strategy  $k$  by any player  $n$  is just the total sum of rewards received to date by player  $n$  from his past use of  $k$ , together with the initial propensity that he associated with  $k$ .

The probability that player  $n$  chooses any particular pure strategy  $k$  in period  $t$  is then given by the *relative* propensity currently associated with  $k$ :

$$p_{nk}(t) = \frac{q_{nk}(t)}{\sum_j q_{nj}(t)} . \quad (22)$$

Notice that players do *not* necessarily choose strategies with the highest accumulated rewards to date. The latter strategies have the highest *probability* of being chosen, but there is always a chance that other strategies with positive probability will be chosen instead. This ensures that each player continually experiments, which in turn helps to avoid premature fixation on suboptimal strategies that happen to generate relatively high rewards at the beginning of the decision process.

The initial propensity values  $q_{nk}(1)$  determine the scaling of the stepsize, i.e., the extent to which the attained rewards change the choice probabilities (22) over time. Note, however, that the magnitudes of the rewards also affect the rate of change of these probabilities over time. Thus, this RL rule is not independent of the units of measurement.

Let  $q_n(1)$  denote the common value for the initial propensities in (19) that player  $n$  associates in the initial period 1 with each of his  $M_n$  pure strategy choices. Er'ev and Roth (1998) further restrict these initial propensity values by assuming that

$$q_n(1) = s(1)X_n/M_n , \quad n = 1, \dots, N, \quad (23)$$

where  $s(1)$  is a positive *strength parameter*,  $X_n$  is the average absolute payoff for player  $n$  in the game, and  $M_n$  is the number of pure strategies available to player  $n$ . Consequently, the initial propensity values  $q_n(1)$  are determined by observable features of the game together with a single free parameter  $s(1)$ .

Roth and Er'ev (1998) also consider a second alternative way of updating the propensities in place of (21), as follows. Two cases are considered. First, suppose that player  $n$ 's pure strategies can be linearly ordered by similarity from 1 to  $M_n$ ,  $n = 1, \dots, N$ . Then, if player

$n$  chooses pure strategy  $k$  in period  $t$ , with resulting reward  $r_{nk}(t)$ , the propensity for any pure strategy  $j$  is updated in accordance with

$$q_{nj}(t+1) = [1 - \phi]q_{nj}(t) + E_{nj}(\epsilon, k, t) \quad , \quad (24)$$

where  $\phi$  is set at some small value lying between 0 and 1, and  $E_{nj}(\epsilon, k, t)$  takes on the value  $r_{nk}(t)[1 - \epsilon]$  if  $j=k$ , the value  $r_{nk}(t)\epsilon/2$  if  $j=k \pm 1$ , and the value 0 otherwise. Second, suppose that player  $n$ 's pure strategies have no apparent linear order by similarity. In this case (24) is again the general form of the updating rule, but now  $E_{nj}(\epsilon, k, t)$  is set equal to  $r_{nk}(t)[1 - \epsilon]$  if  $j$  equals  $k$  and is set equal to  $r_{nk}(t)\epsilon/[M_n - 1]$  for all  $j$  not equal to  $k$ .

The introduction of the *forgetting parameter*  $\phi$  acts as a damper on the growth of the propensities over time. The *experimentation parameter*  $\epsilon$  permits reinforcement to spill over from a chosen pure strategy to “similar” pure strategies. Given this alternative rule for updating the propensity values, it follows that the RL algorithm is now characterized by three free parameters rather than just one: namely, the strength parameter  $s(1)$ , the forgetting parameter  $\phi$ , and the experimentation parameter  $\epsilon$ .

The Roth-Erev updating schemes (21) and (24) turn out to have a serious problem when applied in market contexts in which the immediate rewards  $r_{nk}(t)$  can take on the value zero. Specifically, zero rewards leave the choice probabilities (22) unchanged, hence no learning takes place in response to zero-reward outcomes. As discussed in detail in Koesrindartoto (2001), using the double-auction electricity framework developed in Nicolaisen et al. (2001) for concrete illustration, this can cause inefficient mashing around in the early stages of a market process in which buyers and sellers are first struggling to learn how to make profitable price offers and failures to match (hence zero profit outcomes) are common. Indeed, in Koesrindartoto's experimental runs, the loss in market efficiency ranges from 0 to 80 percent as the experimentation parameter  $\epsilon$  is varied from 0 to 1.

Nicolaisen et al. (2001) avoid this zero-reward updating problem by replacing the original

Roth-Erev update function  $E_{nj}$  in equation (24) with a modified update function  $ME_{nj}$  defined as follows: In each choice period  $t$ ,  $ME_{nj}$  coincides with  $E_{nj}$  if  $j = k$ , where  $k$  denotes the selected strategy for period  $t$ ; otherwise, for all  $j \neq k$ ,

$$ME_{nj} = q_{nj}(t) \frac{\epsilon}{M_n - 1} . \quad (25)$$

Using this “modified Roth-Erev algorithm,” Nicolaisen et al. (2001) obtain market efficiency outcomes close to 100 percent in all tested treatments for their double auction experiment.

*Roth-Erev Learning Discussion Questions:* The Roth-Erev RL algorithms can in principle be generalized to handle action vectors of arbitrary finite dimension taking on arbitrarily many finite values. How might the bean firms in the market economy described in Section II fare if they both attempted to use such learning algorithms for determination of their quantity and price offers? Would they tend to do better or worse in comparison with DF agents? with Gibbs/Boltzmann RL learning? What about cases in which the two bean firms use distinct types of learning algorithms?

## 4 Q-Learning Rules

*Q-Learning* is a RL algorithm developed by Watkins (1989) as a successive approximation technique for solving Bellman’s equation in dynamic programming.<sup>2</sup> Q-learning does not need a model of the environment and can be used on-line in contexts where multiple agents are engaging in repeated non-zero sum games against unknown rivals and choosing their actions in an anticipatory way.

The following discussion depends heavily on Sandholm and Crites (1996). Alternative

---

<sup>2</sup>Q-learning has interesting connections with the *Criterion Filtering (CF)* method developed by this author in a series of studies beginning in 1977. CF is a sequential decision-making method that involves the direct updating of a dynamic programming value function via transitional return assessments in a manner analogous to Bayes’ rule for updating a probability distribution via transitional probability assessments. For a discussion of criterion filtering, see <http://www.econ.iastate.edu/tesfatsi/cfhome.htm>

motivations and justifications for the technique can be found in Jaakkola et al. (1994), Sutton and Barto (2000), and Tsitsiklis (1994).

Q-learning works by estimating the values of state-action pairs. The Q-value  $Q(s, a)$  is defined to be the expected discounted sum of future returns obtained by taking action  $a$  starting from state  $s$  and following an optimal action decision rule thereafter. Once these values have been learned, the optimal action from any state is the one with the highest Q-value.

The Q-values are estimated on the basis of experience, starting from arbitrary initial values. The estimation process consists of three basic steps:

1. From the current state  $s$ , select an action  $a$ , receiving an immediate reward  $r$ , and arrive at a next state  $s'$ ;
2. Based on this experience, update  $Q(s, a)$  to  $Q(s, a) + \Delta Q(s, a)$  using the following updating rule:

$$\Delta Q(s, a) = \alpha[r + \gamma \max_b Q(s', b) - Q(s, a)], \quad (26)$$

where  $\alpha$  is the learning rate and  $0 \leq \gamma \leq 1$  is the discount factor. Equivalently, letting  $Q^N(s, a) = Q(s, a) + \Delta Q(s, a)$  denote the new Q-value, a suitable manipulation of (26) indicates that the new Q-value is formed as a weighted average of old and new estimates as follows:

$$Q^N(s, a) = [1 - \alpha]Q(s, a) + \alpha \left[ r + \gamma \max_b Q(s', b) \right]. \quad (27)$$

3. Return to step 1.

This algorithm is guaranteed to converge to the correct Q-values with probability one under certain specified conditions. These conditions include: no action is neglected forever; the learning rate is suitably decreased over time; the environment is stationary; and the state

transition probabilities are Markovian in the sense that the probability of transiting from  $s$  to  $s'$  depends only on  $s$ ,  $s'$ , and the current action  $a$ , and not on previous history. Note that the latter condition essentially rules out multi-agent contexts, since in such contexts the probability of transiting from a state  $s$  to a new state  $s'$  would typically depend on the actions undertaken by all agents, not just on the action  $a$  undertaken by the agent who is doing the Q-value estimation.

Q-learning does not specify which action should be chosen at each time step. In practice, action choice rules are usually selected to ensure sufficient exploration while still favoring actions with higher Q-value estimates. The Gibbs/Boltzmann probability distribution introduced in Section 3.2 provides one such action choice rule, as follows: at each time step  $t$ , if the current state is  $s$ , then the probability of selecting action  $a$  is given by

$$p(a|s, t) = \frac{e^{Q(s,a)/T_t}}{\sum_i e^{Q(s,a_i)/T_t}}, \quad (28)$$

where  $T_t$  is a “temperature” parameter that controls the amount of exploration and is usually “annealed” (decreased) gradually over time. For example,  $T_t$  might be defined as in (16).

*Q-Learning Discussion Questions:* How might Q-learning be modified to handle the type of anticipatory learning problem faced by bean firm F1 (the “leader”) in the leader-follower duopoly market outlined in Section II, assuming bean firm F2 also uses Q-learning? Would firms F1 and F2 tend to do better or worse using Q-learning in comparison with DF decision making? with Gibbs/Boltzmann with RL learning? with Roth-Erev RL learning? What about mixed cases in which firms F1 and F2 use distinct types of learning processes?

## 5 Genetic Algorithms as Learning Rules

An excellent introduction to genetic algorithm (GA) learning can be found in Holland (1992) and Mitchell (1995). Consequently, only a brief review is given here, based largely on Mitchell

(1995). The section ends with a discussion of the important distinction between use of GAs as social learning algorithms and use of GAs as individual learning algorithms, as clarified by Vriend (2000).

First developed by John Holland in the 1960s, the GA remains one of the most prominent types of algorithms used in evolutionary computation. *Evolutionary computation* is the use of computer programs that not only self-replicate but also evolve over time in an attempt to increase their fitness for some purpose. Fitness might represent:

- the ability to *solve some problem*, such as a math problem, or an architectural design problem;
- the ability to *repeatedly perform some task*, such as handwriting recognition, facial recognition, or chess playing;
- the ability to *survive and prosper in some computationally specified virtual world*, such as a computational city or a computational market.

A GA is an abstraction of biological evolution. It is a method for evolving a new population of entities (e.g., candidate solutions for a math problem) from an existing population of entities, with evolution biased in favor of more fit entities (“survival of the fittest”). The evolution proceeds via genetic operations (recombination, mutation,...) that act directly upon the structural characteristics of the population members.

#### *GA Representation of Population Members:*

The standard GA represents population members as *bit strings*, i.e., as sequences of 0’s and 1’s. A bit string can be used to represent a wide variety of entities.

For example, a bit string can be used to represent in binary (base 2) arithmetic any arbitrary natural number  $n$  in  $\{1, 2, \dots\}$ . Recall that a string of natural numbers such as

$[ 3 | 2 | 4 ]$  represents the following arithmetic operation in decimal (base 10) arithmetic:

$$324 \equiv [ 3 | 2 | 4 ] = 3 \cdot 10^2 + 2 \cdot 10^1 + 4 \cdot 10^0 , \quad (29)$$

where, by convention, any natural number  $n$  raised to the power 0 equals 1. Similarly, a string of 0's and 1's such as  $[ 1 | 0 | 1 ]$  represents the following arithmetic operation in binary (base 2) arithmetic:

$$101 \equiv [ 1 | 0 | 1 ] = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 1^0 , \quad (30)$$

which is equivalent to 5 in decimal arithmetic. Clearly any finite natural number  $n$ , however large, can be represented as a bit string of suitably large (but finite) length.

As another example, bit strings can be used to represent possible strategies for playing the iterated prisoner's dilemma (IPD) game. Suppose the payoff matrix for each stage of the IPD is as depicted in Table 1, where "C" denotes cooperation and "D" denotes defection.

		Player 2	
		C	D
Player 1	C	<b>(3,3)</b>	<b>(0,5)</b>
	D	<b>(5,0)</b>	<b>(1,1)</b>

**Table 1. One-Stage Payoff Matrix for the Iterated Prisoner's Dilemma Game**

Suppose the memory of each player is *one* previous turn. In this case there are *four* possible states (move configurations) that could result from this previous turn, as follows:

$$\text{State 1} = \text{CC: Player 1 chose C, Player 2 chose C ;} \quad (31)$$

$$\text{State 2 = CD: Player 1 chose C, Player 2 chose D ;} \quad (32)$$

$$\text{State 3 = DC: Player 1 chose D, Player 2 chose C ;} \quad (33)$$

$$\text{State 4 = DD: Player 1 chose D, Player 2 chose D .} \quad (34)$$

A strategy for Player 1 (or Player 2) is then a rule specifying what move (C or D) the player should choose if he finds himself in each possible state 1, 2, 3, or 4. An example of such a strategy would be:

$$\text{If in state 1 (CC), then choose C ;} \quad (35)$$

$$\text{If in state 2 (CD), then choose D ;}$$

$$\text{If in state 3 (DC), then choose C ;}$$

$$\text{If in state 4 (DD), then choose D .}$$

Given  $s = 4$  possible states, and  $m = 2$  two possible actions in each possible state, note that the total number of distinct IPD strategies for this game is  $m^s = 2^4 = 16$ .

Suppose the four possible states are always ordered in the same manner, from 1 through 4. Then each possible IPD strategy can be represented as a bit string of length four. For example, letting 1 denote cooperation (C) and 0 denote defection (D), the strategy depicted above in (35) can be represented as the bit string [1 | 0 | 1 | 0], where the first position gives the required move 1 in state 1, the second position gives the required move 0 in state 2, the third position gives the required move 1 in state 3, and the fourth position gives the required move 0 in state 4.

#### *GA Population Evolution:*

Recall that the GA is a method for evolving a new population of entities from an existing population of entities, with evolution biased in favor of more fit entities. The standard GA represents each entity in the population as a bit string and evolves a new entity population

from an existing entity population by applying the following four steps to these bit string representations:

1. *Evaluation Step:* A fitness score is assigned to each entity in the existing population.
2. *Replacement Step:* Some percentage  $q$  of the most fit (“elite”) entities in the existing population are retained for the new entity population.
3. *Recombination Step:* The remaining  $[1 - q]$  percent of the existing population are replaced with offspring entities (new ideas) constructed by combining the genetic material (structural characteristics) of pairs of parent entities chosen from among the entities in the existing population, with selection biased in favor of more fit entities.
4. *Mutation Step:* Additional variations (new ideas) are introduced into the new population by mutating the structural characteristics of each offspring entity with some small probability.

Each of the four steps can be implemented in a variety of ways. For example, consider the previously discussed IPD illustration. Suppose the initial entity population consists of 10 IPD strategies, each represented as a bit string of length four, where the four positions in each bit string correspond to the four possible states 1, 2, 3, and 4.

Regarding step one, set the *fitness* associated with each strategy in the initial population at the beginning of period  $T = 1$  equal to 0, and set the fitness associated with each strategy in the population at the beginning of each period  $T > 1$  equal to the total accumulated payoffs currently associated with this strategy from previous IPD play. Also, at the beginning of each period  $T > 1$ , set the *total population fitness* of the existing strategy population equal to the sum of fitnesses of all 10 strategies in the existing population, and set the *relative fitness* of each strategy in the existing population equal to its own fitness divided by total population fitness.

Regarding step two, suppose the elite percentage rate  $q$  is fixed at 20%. Let the members of the existing strategy population be ranked by fitness. Then the two strategies comprising the top 20% in this fitness ranking are deemed to be the *elite strategies*, meaning they will be transferred without modification to the new strategy population.

Regarding step three, one of the simplest recombination implementations is *two-point cross-over*. Randomly select two distinct *cross-over points*  $COP_1$  and  $COP_2$  from among the set of values  $\{1, 2, 3, 4\}$ . At the beginning of each period  $T > 1$ , let each strategy (length-four bit string) in the existing strategy population be interpreted as a four-position loop with its ends pasted together, and let the left boundary for state position  $j$  be labeled boundary  $j$ ,  $j = 1, 2, 3, 4$ . Select two parent strategies from the existing strategy population, where the probability that any strategy is selected to be a parent is given by its relative fitness. Divide each parent strategy into two portions by splitting it at the boundaries  $COP_1$  and  $COP_2$ . Form two new offspring strategies from the two parents by taking the first portion of the first parent bit string and pasting it to the second portion of the second parent bit string, and taking the second portion of the first parent bit string and pasting it to the first portion of the second parent bit string. Repeat this operation four times in total, so that a total of 8 new offspring strategies are formed. The fitness  $f$  of each offspring strategy is taken to be some weighted average of the fitnesses  $f_1$  and  $f_2$  of its two parents, e.g.,  $f = [L_1 \cdot f_1 + L_2 \cdot f_2]/4$ , where  $L_i$  is the bit string length (number of state positions) incorporated from parent  $i$ .

To illustrate this two-point cross-over operation more concretely, suppose the cross-over points are randomly determined to be boundaries 1 and 2. Suppose the following two parent strategies have been selected from the existing population for cross-over:

$$\text{Parent 1: } [ 1 \mid 0 \mid 0 \mid 1 ] \ ; \quad (36)$$

$$\text{Parent 2: } [ 0 \mid 1 \mid 0 \mid 0 ] \ . \quad (37)$$

Each parent strategy is split into two portions by dividing its bit string at boundaries 1 and 2, as follows:

$$\text{Divided Parent 1: } [ 1 ] \text{ and } [ 0 | 0 | 1 ] ; \quad (38)$$

$$\text{Divided Parent 2: } [ 0 ] \text{ and } [ 1 | 0 | 0 ] . \quad (39)$$

Two offspring are then formed by recombining these portions as follows:

$$\text{Offspring 1: } [ 1 | 1 | 0 | 0 ] ; \quad (40)$$

$$\text{Offspring 2: } [ 0 | 0 | 0 | 1 ] . \quad (41)$$

A simple way to implement the final step four is to select a *mutation probability*  $p_m$  lying strictly between 0 and 1. Each strategy in the new population is then subjected to bit mutation. That is, each bit in the strategy's bit string representation is flipped to its opposite value (from 0 to 1, or from 1 to 0) with probability  $p_m$  and left unchanged with probability  $1 - p_m$ .

A new population of 10 strategies is then formed by combining the 2 elite strategies from the existing strategy population with the 8 newly constructed and mutated offspring strategies.

#### *Individual versus Social GA Learning:*

There are two basic approaches to implementing a GA as a learning rule in multi-agent contexts. The GA can either be applied as a social learning (mimicry) rule across agents, or it can be applied as an individual RL rule for each agent separately. Vriend (2000) explores the implications of these two different approaches for a simple oligopoly (multiple-firm) market.

Specifically, Vriend (2000) considers a collection of  $N$  firms producing a homogeneous commodity  $q$  in a market with a fixed aggregate demand curve that is reconstituted at the beginning of every trading period. Each firm  $i$  has a total cost function  $C_i(q)$  and a maximum production capacity  $QMax_i$ . The only choice to be made by each firm  $i$  in each

trading period  $T$  is the particular quantity  $q_i$  that it is willing to supply at each given market price for  $q$ .

The first approach is to use the GA as a model of social learning. At any given time, each firm  $i$  participating in the market is characterized by a single supply rule that determines its supply quantity  $q_i$  as a function  $q_i(P)$  of the market price  $P$  subject only to one restriction: namely,  $q_i$  cannot exceed firm  $i$ 's maximum capacity  $QMax_i$ . The supply rule for each firm is represented as a bit string (i.e., a string of 1s and 0s) of fixed length. At the beginning of each trading period  $T$  each firm submits its current supply rule to a central clearing house, a uniform market price  $P_T$  is determined by the clearinghouse by setting aggregate supply equal to aggregate demand, and each firm's profits are then determined as its revenues  $P_T q_i(P_T)$  minus its costs  $C_i(q_i(P_T))$ .

At the end of the trading period, a new population of  $N$  supply rules is constructed from the current population of  $N$  supply rules by the application of a GA involving elitism, recombination, and mutation operations that are biased in favor of more fit (profitable) rules. The basic idea is that the most successful supply rules are retained unchanged (elitism) while the less successful supply rules are replaced by variants (recombinations and mutations) of the more successful supply rules. Each firm is then assigned a supply rule from the new population of rules, and a new trading period commences.

The second approach is to use the GA as a model of individual reinforcement learning. One way to do this is to use the *classifier system* approach originally developed by Holland (1992). Each firm  $i$  is now individually modelled as a collection of potential supply rules, where each rule is again modelled as a bit string. Attached to each rule is a fitness or "strength" that reflects the degree to which the rule has been successful in generating profits for firm  $i$  in past usage instances. At the beginning of each trading period, firm  $i$  selects one supply rule from among its current collection of supply rules, where the probability that any particular supply rule is selected is proportional to its current strength.

As in the social learning case, after receiving a supply rule from each firm at the beginning of any trading period  $T$ , the clearinghouse determines the period- $T$  market price  $P_T$  which in turn determines the period- $T$  profit level for each firm in the market. Each firm then uses its own attained profit level to update the strength of the supply rule it is currently using. A GA is then separately applied to the collection of supply rules for each of the  $N$  firms in the same way that a single GA was previously applied to the collection of supply rules for the collection of traders as a whole. Thus, instead of comparing the profit performance of its current supply rule against the supply rules currently in use by other firms, each firm now compares the profit performance of its current supply rule only against the past profit performance of all other supply rules in its own collection.

Vriend (2000) considers the special case in which each firm  $i$  at the beginning of each period  $T$  must produce and offer for sale a quantity  $q_i$  prior to the actual determination of the period- $T$  market price, so that  $q_i(P) = q_i$  (no dependence of period- $T$  supply on the period- $T$  market price). For this case, he demonstrates experimentally that the two types of GA learning can result in essential differences in market outcomes. Specifically, for the particular parameter values he studies, social learning tends to lead to a more socially desirable outcome than individual learning.

Vriend attributes this difference to a structural feature of the underlying oligopoly model which he calls the *spite effect*: namely, a firm can act in a way that decreases its own profits but that decreases the profits of its rivals more. Note that this spite effect exists independently of the particular learning processes of the firms.

When the firms are social learners, the spite effect can potentially influence the learning processes of the firms. Specifically, it can lead to a *relatively* better profit (fitness) outcome for a spiteful firm, which favors the reproduction (social mimicry) of the spiteful firm's particular quantity choice in the evolutionary step. Vriend demonstrates experimentally that this influence tends to coordinate the quantity choices of the socially learning firms

on the “Walrasian” aggregate quantity outcome where total social surplus (consumer plus producer surplus) is maximized.

In contrast, when the firms are individual learners, the spite effect is still present, but the exercise of spitefulness by any one firm does not give that firm any evolutionary advantage relative to other firms. This is because each firm individually determines its current quantity choice by comparing the past profitability of this quantity choice to the past profitability of other past quantity choices *by this same firm*. However, the *actual* profitability of the current quantity choice of any one firm is determined by the current set of quantity choices across all firms and not by past quantity choices. Consequently, there is no reason for a firm to act to decrease its own profits in order to decrease its rivals’ profits more since it does not secure any evolutionary advantage from this spiteful behavior. Vriend’s experimental findings demonstrate that the individual learning firms tend to coordinate on an aggregate quantity outcome that yields higher firm profits (producer surplus) but lower total social surplus in comparison with the social learning outcome.

*GA Learning Discussion Questions:* How might the bean firms in the market economy described in Section II attempt to use GA learning for determination of their quantity and price offers? Does GA learning appear to be well suited for the determination of pure cartel, symmetric duopoly, or leader-follower duopoly solutions in comparison with other types of learning algorithms? Why or why not?

## References

- Axelrod, Robert (1984), *The Evolution of Cooperation*, Basic Books, Incorporated, New York, N.Y.
- Bell, Ann M. (2001), “Reinforcement Learning Rules in a Repeated Game,” *Computational Economics*, Vol. 18, No. 1, August.

- Berry, Donald A., and Bert Fristedt (1985), *Bandit Problems: Sequential Allocation of Experiments*, Chapman-Hall.
- Bush, R., and R. Mosteller (1955), *Stochastic Models for Learning*, John Wiley, New York, N.Y.
- Camerer, Colin, and Teck-Hua Ho (1999), “Experience-Weighted Attraction Learning in Normal Form Games,” *Econometrica*, Vol. 67, No. 4, 827–874.
- Er’ev, Ido, and Alvin Roth (1998), “Predicting How People Play Games: Reinforcement Learning in Experimental Games with Unique, Mixed Strategy Equilibria,” *American Economic Review* 8(4), 848–881.
- Feltovich, Nick (2000), “Reinforcement-Based Vs. Belief-Based Learning in Models in Experimental Asymmetric-Information Games,” *Econometrica*, Vol. 68, No. 3, 605–641.
- Fudenberg, Drew, and David K. Levine (1998), *The Theory of Learning in Games*, The MIT Press, Cambridge, MA.
- Greenwald, Amy R., and Jeffrey O. Kephart (1999), “Shopbots and Pricebots,” Sixteenth International Joint Conference on Artificial Intelligence, Stockholm, Sweden, August, pp. 506-511.
- Holland, John (1992), *Adaptation in Natural and Artificial Systems*, Second Edition, The MIT Press, Cambridge, MA.
- Jaakkola, T., M. Jordan, and S. Singh (1994), “Convergence of Stochastic Iterative Dynamic Programming Algorithms,” *Neural Information Processing Systems* 6, pp. 703–710.
- Kephart, Jeffrey O., James E. Hanson, and Amy R. Greenwald (2000), “Dynamic Pricing by Software Agents,” *Computer Networks*, Vol. 32(6), 731–752.
- Koesrindartoto, Deddy (2001), “Discrete Double Auctions with Artificial Adaptive Agents: A Case Study of an Electricity Market Using a Double Auction

Simulator,” M.S. Creative Component, Economics Dept., Iowa State University, June 14.

Mitchell, Melanie (1995), “Genetic Algorithms: An Overview,” *Complexity*, Vol. 1(1), 1995, 31–39.

Nicolaisen, James, Valentin Petrov, and Leigh Tesfatsion (2001), “Market Power and Efficiency in a Computational Electricity Market with Double-Auction Discriminatory Pricing,” *IEEE Transactions on Evolutionary Computation* 5, pp. 504-523.

Lynne Pepall, Daniel J. Richards, and George Norman, *Industrial Organization: Contemporary Theory and Practice*, South-Western College Publishing, New York, N.Y., 1999. [This book develops the essentials of modern industrial organization (IO). The authors stress the understanding of strategic interaction among firms in an industry or across industries, making use of elementary tools and concepts from game theory. The text is relatively non-technical. It is primarily intended for undergraduate students who have completed an intermediate microeconomics course, and as a background text for graduate students taking their first course in industrial organization.]

Robbins, Harold (1952), “Some Aspects of the Sequential Design of Experiments,” *Bulletin of the American Mathematical Society* 58, pp. 527–535.

Roth, Alvin, and Ido Er’ev (1995), “Learning in Extensive Form Games: Experimental Data and Simple Dynamic Models in the Intermediate Term,” *Games and Economic Behavior* 8, 164–212.

Sandholm, Tuomas, and Robert H. Crites (1996), “Multiagent Reinforcement Learning in the Repeated Prisoner’s Dilemma,” *BioSystems* 37(1-2).

Sutton, R. S., and A. G. Barto (2000), *Reinforcement Learning: An Introduction*,

Third Printing, The MIT Press, Cambridge, MA.

Tesauro, Gerald J., and Jeffrey O. Kephart (1998), “Foresight-Based Pricing Algorithms in an Economy of Software Agents,” pp. 37-44 in *Proceedings of the First International Conference on Information and Computational Economics*, ACM Press.

Thompson, W. R. (1933), “On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples,” *Biometrika* 24, 275–294.

Tsitsiklis, J. (1994), “Asynchronous Stochastic Approximation and Q-Learning,” *Machine Learning* 16, pp. 185–202.

Vriend, Nicolaas (2000), “An Illustration of the Essential Differences Between Individual and Social Learning, and its Consequences for Computational Analyses,” *Journal of Economic Dynamics and Control* 24, 1–19.

Watkins, C. (1989), “Learning from Delayed Rewards,” Ph.D. Thesis, Cambridge University, UK.

## Appendix

### Key Concepts from “Notes on Learning”

- Social learning
- Individual learning
- Type learning
- Backward-looking stimulus and response learning
- Anticipatory look-ahead learning
- Conditional expected utility
- Reinforcement learning
- Nonsequential choice problem
- Sequential choice problem
- Credit assignment problem
- Two-armed bandit (TAB) problem

- **Derivative-Follower Algorithm**
- **Gibbs/Boltzmann reinforcement learning algorithm**
- **Roth-Erev reinforcement learning algorithms**
- **Q-learning**
- **Genetic algorithm (GA) learning**
- **Classifier system**

Copyright © 2006 by Leigh Tesfatsion. All rights reserved.