Notes on Learning: Econ 308

Leigh Tesfatsion
Department of Economics
Iowa State University
Ames, Iowa 50011-1070
https://www2.econ.iastate.edu/tesfatsi/
tesfatsi@iastate.edu

14 February 2024

> "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).

Economics 308: Basic Course Information:

Title: Intro to Agent-Based Computational Economics (Undergrad/M.S.)

Homepage: https://www2.econ.iastate.edu/classes/econ308/tesfatsion

1 Overview of Notes

These notes briefly review several types of learning methods currently being used by ACE researchers to represent the learning capabilities of computationally modeled traders. A ppt presentation outlining and in some cases expanding upon the key points made in these notes is provided in Ref.[1] above.

Section 2 provides a brief discussion of key learning issues. An overview of reinforcement learning (RL) is presented in Section 3. Several illustrative types of reactive RL methods are discussed in Section 4, and a well-known anticipatory RL method — Q learning — is concretely illustrated in Section 5. Section 6 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 method might be applied by the quantity-setting and price-setting firms in the Computational Market Economy set out in Ref.[2].

2 Introduction to Learning

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 *reactive (stimulus-response) learning* for choosing its actions. A state-of-the-world (stimulus) occurs, and the agent reacts 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 $r_i, i = 1, \ldots N$, 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*

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

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 complication in the design of anticipatory learning methods 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 if this ultimately permits the agent to make more informed future decisions because it has better estimates of the true outcome probability distributions. A second complication arising in multi-agent decision contexts is potential infinite regress as each agent tries to take into account what other agents are doing, including the understanding that these other agents are trying to take into account what he is doing; see Tesauro and Kephart (1998).

3 Overview of 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, in its most basic form, RL is a relatively straight-forward type of learning method in which an agent constructs 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 methods 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 section illustrates several different types of reactive RL methods that are currently being used by agent-based computational modelers.

4 Reactive RL: Illustrative Examples

4.1 The Derivative-Follower Algorithm

The "Derivative-Follower" algorithm is perhaps more correctly termed an *adaptive* method rather than a reactive RL method since its initially specified state-action associations (in rate of change form) remain invariant over time.[1]

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) Δa 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 Δa can be adapted to the problem at hand. In each of their reported experiments, Greenwald and Kephart (1999) randomly select Δa 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 deci-

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

sion 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 Computational Market Economy outlined in Ref.[2] in which bean firms 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 bean firms apply the DF algorithm in the absence of any a priori knowledge about consumer demand except the knowledge that demand curves are generally downward sloping?

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.

For example, bean firms in Ref.[1] are actually assumed to determine their bean production levels in advance of sale. Specifically, at the beginning of each trading period t, each bean firm B_n produces a certain quantity of beans b_n and then posts a unit bean price p_{Bn} to attract consumers. Can a variant of the DF algorithm be constructed for the joint choice of (b_n, p_{Bn}) in each trading period t that would reliably ensure the survival and even the prosperity of bean firm B_n?

4.2 Gibbs/Boltzmann RL

Another type of reactive RL method maps past rewards into future actions through a *Gibbs (or Boltzmann) probability distribution*. The description below of this reactive RL method 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

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

$$p_{2t} = \frac{e^{w_{2t}/T_t}}{e^{w_{1t}/T_t} + e^{w_{2t}/T_t}} \quad . \tag{3}$$

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 μ (the stepsize) denote any number lying strictly between 0 and 1, a possible functional specification for the determination of T_t is

$$T_{t+1} \quad = \quad \max\{\mu T_t, \bar{T}\} \quad . \tag{4}$$

Given (4), 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:

$$w_{1,t+1} \quad = \quad \frac{[n_{1t} - I_{1t}]}{n_{1t}} \cdot w_{1t} \; + \; \frac{[I_{1t} \cdot r_{1t}]}{n_{1t}} \quad ; \tag{5}$$

$$w_{2,t+1} \quad = \quad \frac{[n_{2t} - I_{2t}]}{n_{2t}} \cdot w_{2t} \; + \; \frac{[I_{2t} \cdot r_{2t}]}{n_{2t}} \quad . \tag{6}$$

Gibbs/Boltzmann Learning Discussion Questions:

The Gibbs/Boltzmann RL method can in principle be generalized to handle action vectors of arbitrary finite dimension taking on arbitrarily many finite values. How might the bean and hash firms in the Computational Market Economy described in Ref.[2] fare if they all attempted to use such a learning method 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 some firms use the DF algorithm and other firms use Gibbs/Boltzmann learning?

4.3 Er'ev and Roth RL

This subsection briefly outlines two reactive RL methods developed by Roth and Er'ev (1995) and Er'ev and Roth (1998). The form of these RL methods is based on the human-subject experimental work of the psychologists Bush and Mosteller (1955). Camerer and Ho (1999) have extended these RL methods 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 kth pure strategy is given by a nonnegative *initial propensity* $q_{nk}(1)$, $k = 1, \ldots, M_n$. These initial propensities are assumed to be equal valued:

$$q_{nk}(1) \quad = \quad q_{nj}(1) \quad \text{for all pure strategies} \quad j, k \ . \tag{7}$$

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:

$$p_{nk}(1) = \frac{q_{nk}(1)}{\sum_j q_{nj}(1)} \quad = \quad 1/M_n. \tag{8}$$

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:

$$q_{nk}(t+1) \quad = \quad q_{nk}(t) \ + \ I_{nk}(t) r_{nk}(t) \quad . \tag{9}$$

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:

$$p_{nk}(t) = \frac{q_{nk}(t)}{\sum_j q_{nj}(t)} \quad . \tag{10}$$

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 (10) 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 (7) 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

$$q_n(1) \quad = \quad s(1)X_n/M_n \quad , \quad n = 1, \ldots, N, \tag{11}$$

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 (9), 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, $n = 1, \ldots N$. 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

$$q_{nj}(t+1) \quad = \quad [1-\phi]q_{nj}(t) + E_{nj}(\epsilon, k, t) \quad , \tag{12}$$

where ϕ 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=$k\pm1$, and the value 0 otherwise. Second, suppose

that player n's pure strategies have no apparent linear order by similarity. In this case (12) 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* ϕ acts as a damper on the growth of the propensities over time. The *experimentation parameter* ϵ 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 method is now characterized by three free parameters rather than just one: namely, the strength parameter $s(1)$, the forgetting parameter ϕ, and the experimentation parameter ϵ.

The Roth-Erev updating schemes (9) and (12) 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 (10) 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 mushing 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 ϵ 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 (12) 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 $j \neq k$,

$$ME_{nj} \quad = \quad q_{nj}(t)\frac{\epsilon}{M_n - 1} \quad . \tag{13}$$

Using this "modified Roth-Erev method," 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 methods can in principle be generalized to handle action vectors of arbitrary finite dimension taking on arbitrarily many finite values. How might the firms in

the Computational Market Economy described in Ref.[2] fare if all firms attempted to use such learning methods 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 different firms use different types of learning methods?

5 Anticipatory RL Illustration: Q-Learning

Q-Learning is a RL method developed by Watkins (1989) as a successive approximation technique for solving Bellman's equation in dynamic programming.[2] 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 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:

$$\Delta Q(s, a) = \alpha[r + \gamma \max_b Q(s', b) - Q(s, a)], \tag{14}$$

[2]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

where α 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 (14) indicates that the new Q-value is formed as a weighted average of old and new estimates as follows:

$$Q^N(s,a) \;=\; [1-\alpha]Q(s,a) + \alpha\left[r + \gamma \max_b Q(s',b)\right] \;. \tag{15}$$

3. Return to step 1.

This method 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 4.2 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

$$p(a|s,t) = \frac{e^{Q(s,a)/T_t}}{\sum_i e^{Q(s,a_i)/T_t}}, \tag{16}$$

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 (4).

Q-Learning Discussion Questions:

How might Q-learning be modified to handle the type of anticipatory learning problems faced by firms in the Computational Market Economy presented in Ref.[2]? Would the firms do better or worse using Q-learning in comparison with DF decision making? with

Gibbs/Boltzmann RL learning? with Roth-Erev RL learning? What about mixed cases in which different firms use different types of learning methods?

6 Genetic Algorithms as Learning Methods

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 methods and use of GAs as individual learning methods, as clarified by Vriend (2000).

First developed by John Holland in the 1960s, the GA remains one of the most prominent types of methods 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 $\{1, 2, \ldots\}$. Recall that a string of natural numbers such as [3 | 2 | 4] represents the following arithmetic operation in decimal (base 10) arithmetic:

$$324 \equiv [\, 3 \,|\, 2 \,|\, 4 \,] \;=\; 3 \cdot 10^2 \;+\; 2 \cdot 10^1 \;+\; 4 \cdot 10^0 \;, \tag{17}$$

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:

$$101 \equiv [\, 1 \,|\, 0 \,|\, 1 \,] \;=\; 1 \cdot 2^2 \;+ 0 \cdot 2^1 \;+\; 1 \cdot 2^0 \;, \tag{18}$$

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**	**(3,3)**	**(0,5)**
	D	**(5,0)**	**(1,1)**

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:

State 1 = CC: Player 1 chose C, Player 2 chose C ; (19)

$$\textbf{State 2 = CD:} \quad \text{Player 1 chose C, Player 2 chose D} \quad ; \tag{20}$$

$$\textbf{State 3 = DC:} \quad \text{Player 1 chose D, Player 2 chose C} \quad ; \tag{21}$$

$$\textbf{State 4 = DD:} \quad \text{Player 1 chose D, Player 2 chose D} \quad . \tag{22}$$

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:

$$\begin{aligned} &\text{If in state 1 (CC), then choose C} \quad ; \\ &\text{If in state 2 (CD), then choose D} \quad ; \\ &\text{If in state 3 (DC), then choose C} \quad ; \\ &\text{If in state 4 (DD), then choose D} \quad . \end{aligned} \tag{23}$$

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 (23) 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 $f1$ and $f2$ of its two parents, e.g., $f = [L_1 \cdot f1 + L_2 \cdot f2]/4$, 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:

$$\text{Parent 1: } [\ 1 \mid 0 \mid 0 \mid 1\ ]\ ; \tag{24}$$

$$\text{Parent 2: } [\ 0 \mid 1 \mid 0 \mid 0\ ]\ . \tag{25}$$

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

$$\text{Divided Parent 1: } [\ 1\ ] \text{ and } [\ 0 \mid 0 \mid 1\ ]\ ; \tag{26}$$

$$\text{Divided Parent 2: } [\ 0\ ] \text{ and } [\ 1 \mid 0 \mid 0\ ]\ . \tag{27}$$

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

$$\text{Offspring 1: } [\ 1 \mid 1 \mid 0 \mid 0\ ]\ ; \tag{28}$$

$$\text{Offspring 2: } [\ 0 \mid 0 \mid 0 \mid 1\ ]\ . \tag{29}$$

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 would the firms in the Computational Market Economy presented in Ref.[2] fare if they use individual GA learning for determination of their quantity and price offers? Would use of social GA learning make any sense? Does GA learning appear to be better suited for the determination of the firms' quantity and price decisions than the other types of learning methods surveyed above? Why or why not?

References and Recommended Readings

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.

Tesfatsion, L. (2007a), "Learning Algorithms: Illustrative Examples", link at Syllabus Section III.B, directly accessible at
https://www2.econ.iastate.edu/classes/econ308/tesfatsion/LearnAlgorithms.LT.pdf

Tesfatsion, L. (2007b), "Constructing Computational Traders with Learning Capabilities", link at Syllabus Section III.B, directly accessible at
https://www2.econ.iastate.edu/classes/econ308/tesfatsion/ConstructLearningAgent.InClassDisc.pdf

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: Econ 308"

- **Social learning**
- **Individual learning**
- **Type learning**
- **Reactive (stimulus-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**