Journal of Economic and Social Research 2 (1) 2000, 39-57

Reinforcement Learning and Dynamic Optimization[+]

Erdem Başçı[*] Mehmet Orhan[**]

Abstract. This paper is about optimization achieved through reinforced learning. First, the concept of an augmented value function for infinite horizon discounted dynamic programs is defined. Next, the issue of convergence and the speed of the convergence in the context of a cake-eating problem are studied. Finally, in numerical simulations it is observed that regardless of initial beliefs learning of the augmented value function and hence optimal behavior is attainable within reasonable time horizons under the presence of experimentation and slow cooling.

JEL Classification Codes: D83, D91.

Key Words: Learning; Dynamic programming; Classifier systems.

1. Introduction

In dynamic economic models it is usually assumed that an agent's behavior is in line with the solutions to dynamic optimization problems. Since such problems are quite difficult to handle, it is frequently argued that agents do not actually solve these problems, but, through a process of learning over time, they start to behave in accordance with the optimal solution.

As for reinforcement learning, it is possibly one of the most primitive learning methods. It does not require from the agents the forming of expectations or the use of sophisticated reasoning. Studies of reinforcement learning in repeated decision environments include Bush and

---

[+]We thank Neil Arnwine and Harald Uhlig for useful comments. The original version of this study is a discussion paper of Başçı and Orhan (1999).

[*]Department of Economics, Bilkent University, Ankara

[**]Department of Economics, Fatih University , İstanbul

Mosteller (1955), Cross (1973, 1979), Roth and Erev (1995), Börgers and Sarin (1997), Erev and Roth (1998), and Erev and Rapoport (1998). For dynamic decision environments, Lettau and Uhlig (1999) propose a learning algorithm based on classifier systems. Classifier system learning, introduced by Holland (1975) as a tool for machine learning, is also suitable for modeling reinforcement learning in economics. A classifier system consists of a list of *condition-action* statements, called classifiers, and a corresponding list of real numbers, called the *strengths* of the classifiers. Classifiers bid their strengths in competition for the right to guide the agent in each decision situation. The strengths are then updated according to the outcomes.

Classifier system learning is used in a number of economic models. Examples of its application in repeated static decision environments can be found in Arthur (1991), Beltrametti et al. (1997), Kirman and Vriend (1996), and Sargent (1993). In the context of the Kiyotaki-Wright model of money, a dynamic game with a recursive structure, Marimon et al. (1990) and Başçı (1999) use classifier systems in their simulations. Lettau and Uhlig (1999), on the other hand, analyze the connection between relevant dynamic programming problems and the asymptotic behavior of the corresponding classifier system learning. Lettau and Uhlig describe the limiting behavior of classifier strengths. They do not allow for experimentation by agents. They show that if the classifier system is sufficiently rich and the initial strengths high enough, the strengths of the asymptotically winning classifiers converge to the values given by the solution to the Bellman equation. They also show, however, that the strengths of the remaining classifiers may freeze at an arbitrary point in a dense subset of real numbers.

In this paper, we show through numerical simulations that allowing for experimentation, in the form of trembling hands, results in the convergence of the vector of all strengths to a unique vector of real numbers. We note this limit vector and call it the augmented value function. We also study the speed of the convergence, which is an issue of practical importance in computational economics but is not addressed by Lettau and Uhlig (1999).

In the section that follows, we define the augmented value function in the context of a simple cake eating problem with a stochastic twist. In section 3, we describe the details of the learning algorithm. In section 4, we present the results of our numerical simulations. In section 5, we do some robustness checks and sensitivity analysis. We conclude in section 6.

2. The Cake Eating Problem

We confine our presentation below to the cake-eating problem to facilitate illustration.[1] For learnability, we assume that the agents who eat up all of their cakes receive new ones as subsidies with a positive probability below one. This makes the dynamic optimization problem a *repeated* one.

We consider the optimization problem faced by a consumer with and infinitely long life who has $k_0 \in X = \{0,1,..., k^*\}$ units of cake available in Period 0. Here, X denotes the state space. The cake is perfectly storable and the consumer, in each period t, has the option of consuming an integer amount of cake, again, from the set X, subject to the availability condition, $c_t \leq k_t$.

We assume an instantaneous utility function, U: $X \rightarrow \Re$, which exhibits diminishing marginal utility from consumption. The lifetime utility is given by the expected infinite sum of current and future utilities from consumption, properly discounted by the factor $0< \beta <1$.

We also assume that there is a positive probability p_s of receiving a subsidy of k^* units of cake from the government, to be collected at the beginning of the following period, $t+1$, if and only if the consumer has 0 units of cake in hand at the end of the current period, t.

For this problem, we can write the following Bellman equation:

$$v(k) = max\{U(c) + \beta E\, v(k-c+s) \mid c \in X,\ c \leq k\} \quad (1)$$

for all $k \in X$, where s is a random variable that takes on a value of k^* whenever the subsidy is received, with probability p_s, by the consumer and 0 otherwise. Here, $v: X \rightarrow \Re$, which is called the *optimal value function*, gives the maximal amount of expected lifetime utility attainable by a consumer who starts off with a specified amount of cake in hand. Equation (1) can be numerically solved by using the value function iteration method discussed, for example, in Stokey and Lucas with Prescott (1989).

For this consumption-savings problem, we now define the *augmented value function*, $v^*: X \rightarrow \Re$, where $A= \{(k,c) \in X \times X \mid c \leq k\}$. The interpretation of v*(k,c) will be the expected lifetime utility from consuming

---

[1]The same line of analysis could be conducted for more general dynamic programming problems with a similar structure.

c units in this period, and following optimal policies, thereafter. The augmented value function differs from Bellman's value function in that the former does not suppose optimal behavior in the current period. One can, therefore, define the augmented value function through the formula:

$$v^*(k,c) = U(c) + \beta E\, v(k-c+s) \quad (2)$$

for all $(k,c) \in A$. From (1) and (2) it follows that:

$$v(k) = Max\{ v^*(k,c) \mid c \in X,\ c \leq k\} \quad (3)$$

for all $k \in X$.

By using (2) and (3), one can easily show that the augmented value function satisfies the functional equation,

$$v^*(k,c) = U(c) + \beta E\, max\{ v^*(k-c+s,c') \mid c' \} \in X,\ c' \leq k-c+s \} \quad (4)$$

for all $(k,c) \in A$.

In fact, equation (4) provides an alternative method for determining the augmented value function without any need to know the optimal value function itself. The solution of equation (4) is unique and can be obtained by iterating on an arbitrary initial guess, v_0^*, via the contraction mapping $T: C(A) \rightarrow C(A)$ defined through,

$$T(f(k,c)) = U(c) + \beta E\, max\ \{f(k-c+s,c') \mid c' \in X,\ c' \leq k-c+s \} \quad (5)$$

for all $(k,c) \in A$. Here $C(A)$ denotes the space of continuous and bounded functions on the set A.[2]

In the simple numerical example that we will study in the next section, we take $X = \{0,1,2\}$, $U(0) = 0$, $U(1) = 8$, $U(2) = 10$, $\beta = 0.9$ and $p_s = 0.4$. Under these parameter values, the augmented value function can be calculated by either method as, $v^*(2, 0) = 46.27$, $v^*(2, 1) = 51.41$, $v^*(2, 2) = 50.23$, $v^*(1, 0) = 43.41$, $v^*(1, 1) = 48.23$, $v^*(0, 0) = 40.23$.

These numbers indicate that the optimal policy for a consumer with *2* units of cake is to consume *1* unit today and the remaining unit tomorrow.

---

[2]It is easy to see that the mapping T satisfies Blackwell's sufficient conditions for a contraction. For details the reader is referred to Stokey and Lucas with Prescott (1989, Thm. 3.3, p.54).

This statement follows from observing that $v^*(2,1)$ is the highest among $v^*(2,.)$ and that $v^*(1,1)$ exceeds $v^*(1,0)$.

3. The Learning Algorithm

Here we consider the learning problem of an agent who knows neither the augmented value function nor the optimal policies. It is assumed that the agent has subjective beliefs on the values of each possible state-action pair and is updating these beliefs through experience. We call this procedure of learning by doing *reinforcement learning*.

As already stated, classifier systems have a suitable structure for reinforcement learning. A classifier system consists of a potentially flexible list of condition-action statements together with their strengths. A strength is interpreted as the subjective belief about the value of a particular action under a particular condition. There are three main steps in the operation of a classifier system: 1. recognize your current condition, or state, and determine the list of classifiers applicable in the current condition (activation), 2. pick one classifier from among the activated ones based on the information conveyed by their strengths, follow its advice and bear the consequences (selection), and 3. according to the consequences, update the strength of the classifier responsible for these (update). Then, go back to Step 1.

In problems with a discrete and small state space, it is natural to assume that the agent can recognize precisely the current state. We, therefore, work with classifier systems that are complete. A classifier system is called complete[3] if it contains exactly one classifier for every conceivable action in every specific state. In the consumption-savings problem of section 2, there are a total of 3 states, namely 0, 1, 2, and a total of 1, 2, and 3 actions in these states respectively. Therefore, our consumer has a total of 6 specific classifiers. The strengths of these classifiers will be denoted by S_{im}, where i stands for the amount of cake in hand at the

[3]Allowing for incompleteness in classifier systems may be more relevant and interesting in explaining suboptimal human behavior. For example, Lettau and Uhlig (1999) provide an explanation to the empirical puzzle of excess sensitivity in consuming to income shocks by allowing the classifier system to be incomplete. Here, however, we concentrate on cases where learning optimal play *is* possible in the long run.

beginning of a period and *m* stands for the amount of consumption recommended.

Since initially our consumer does not know the augmented values corresponding to these classifiers, we generate the initial strengths randomly from i.i.d. *N(46,20)*.[4] At any point in time, suppose that the consumer is in state $i \in \{0,1,2\}$, i.e. the consumer has entered the current period with *i* units of cake, and hence *i+1* different classifiers are activated. In the selection step, with a positive probability, $1-p_r$, it is assumed that the consumer will follow the advice of the classifier that has the highest strength among the activated ones. Here, $p_r > 0$ denotes the probability of experimentation in the form of random action in a given period. This randomness may be due to trembling hands or to a temporary memory block. Given such an occurrence, it is assumed that the consumer will randomly select one of the *i* activated classifiers with equal probability.

After the selection step, utility from consumption is realized, and if all of the cakes have been consumed, a subsidy of 2 units is received with probability p_s . These events determine the next period's state, *j*. The strength of the most recently activated classifier, *im*, is then updated in transition from state, *i*, to the next state, *j*, based on the realized utility, U_m, and the maximum classifier strength at the next period's state, $S^*_{jt}=max_n \{S_{jnt}\}$, according to the formula,[5]

$$S_{im,t+1} = S_{imt} + \alpha_{imt} (U_m + \beta S^*_{jt} - S_{imt}) \quad (6)$$

where *t* is the time subscript and α_{imt} is the t^{th} entry of the *cooling sequence* for classifier *im*. A *cooling sequence* is a non-increasing sequence of positive weights that converge to zero at a slow rate in such a way that,

$$\sum_t \alpha_{imt} = \infty$$

The intuition behind the strength update formula (6) lies in its connection to equation (4). If the sum of the current utility from consumption and the discounted future maximal strength is above the strength of the most recently selected classifier, then it is rewarded by an

---

[4]These numbers are selected so as to be around the v^* values for the given parameterization.

[5]While using this formula, the agent assumes away trembling hands in the time periods still to come.

increase in its strength. Otherwise, it is penalized by a reduction. Once the strengths of the classifiers have converged to their corresponding values in the augmented value function satisfying (4), the term in parentheses in (6) has an expectation of zero, which makes the expected change in S_{imt} zero. However, fluctuations due to the random subsidy term will remain. These fluctuations will be eliminated over time as the cooling sequence, α_{imt}, approaches zero.

In economic applications,[6] it is customary to take the cooling function in the form,

$$\alpha_{imt} = 1/(\tau_{imt} + 2) \quad (7)$$

where τ_{imt} is an *experience counter* recording the number of times that the particular classifier, *im*, has been selected up to time *t*. Initially, we set $\tau_{im0} = 0$ for all classifiers, *im*, so that the initial value of α_{im} becomes $1/2$.

In order to control the speed of convergence of α_{imt}, we use a positive integer denoted by *l*. Then the formula

$$\alpha_{imt} = 1 / ([l * \tau_{imt}] + 2) \quad (8)$$

is used to generate the cooling sequence. Here, *[.]* denotes the greatest integer function.[7] For example, for $l=1/5$, it takes 5 times longer for the α_{imt} sequence to reach any given $\varepsilon \in (0,1/2)$, compared to the case where $l=1$.

4. Simulation Results

During the discussion below we will distinguish the notions of learning optimal behavior from learning the correct classifier strengths, i.e. the augmented values. We have prepared a GAUSS program to implement the learning algorithm described in section 3. In a single run of the program with randomly generated initial strengths, $l=1$, and $p_r=5\%$, we were able to

---

[6]See, for example, Marimon et al. (1990).

[7]The use of the integer value function is not essential. Without it the decay of the cooling sequence becomes smoother without altering the simulation results. We prefer to keep it, in this section, however, since it essentially boils down to an exponentially weighted moving average during the periods α is held constant. In section 5, it will be dropped.

see the effects of fast cooling on the convergence pattern of classifier strengths.

n	S_{22}	S_{21}	S_{20}	S_{11}	S_{10}	S_{00}
0	37.31	75.36	21.68	60.58	51.15	18.96
0.1	47.72	48.71	43.57	45.54	40.53	37.57
1	48.14	49.29	44.14	46.13	41.27	38.12
5	48.44	49.62	44.47	46.44	41.61	38.44
10	48.56	49.74	44.59	46.56	41.73	38.56
15	48.62	49.8	44.66	46.62	41.8	38.62
20	48.67	49.85	44.71	46.67	41.85	38.67
Val.	50.23	51.41	46.27	48.23	43.41	40.23

Table I: Under fast cooling (l=1), the strengths at the end of period *n* (in millions), for a single run. The bottom row displays the target, which is the augmented value function.

As seen in Table 1, the ordering of the initial strengths is consistent with optimal behavior. However, the values of the strengths are far away from their targets depicted in the last line of Table 1. From the table, an extremely slow tendency for the strengths to converge to the augmented values is observed. Even at the end of *20* million trials the strengths are considerably far away from their targets. A linear extrapolation, from *15* million periods onwards, reveals that the agent would need *173.7* million more trials to reach the correct values for the classifiers at State 2. It is apparent that this number would, in fact, be much larger if the extrapolation were to take into account the decrease in the value of α_{imt} over time.

For the same initial values, and the inclusion of a trembling-hand probability, but for a much smaller cooling rate given by *l=0.05*, the speed of convergence is observed to increase dramatically. Table 2 shows that after around *11,500* time periods, the strengths of the *correct* classifiers, S_{21}, $S_{11,}$ and S_{00}, have almost hit their target values. Moreover, the strengths of the remaining classifiers, S_{22}, $S_{20,}$ and S_{10}, subjected to a much smaller number of updates, have come close to their target values as well.

In order to make the observations in Table 2 statistically more precise, we conducted *1000* independent runs. Table 3 summarizes the convergence pattern for a trembling-hand probability of *5%* and a cooling parameter of

N	S_{22}	S_{21}	S_{20}	S_{11}	S_{10}	S_{00}
0	37.31	75.36	21.68	60.58	51.15	18.96
100	37.31	55.76	43.30	50.16	46.37	47.57
1000	46.91	49.35	47.39	45.15	41.62	48.59
3000	46.79	52.10	46.83	48.06	43.95	41.22
5000	48.93	50.17	45.22	46.81	42.39	40.18
7000	46.65	50.99	45.83	47.96	43.09	38.98
9000	47.90	51.78	46.23	48.46	43.88	40.66
11000	51.47	51.51	46.64	48.06	43.69	40.69
11505	49.80	51.40	46.41	48.23	43.57	40.21
Val.	50.23	51.41	46.27	48.23	43.41	40.23

Table II: Under slow cooling (*l=0.05*), the strengths at the end of period n for a single run. The bottom row displays the augmented value function.

l=0.05. Since the initial strengths are selected randomly across runs, initially only one third of the consumers who start with 2 units of cake choose the optimal consumption level of 1 unit. Similarly, only about a half of the consumers holding 1 unit of cake choose to consume it at the beginning. As learning proceeds, at around period 1,000 95 % of all consumers start to follow the optimal cake eating plan in their conscious choices. The convergence of strengths to the augmented value function, however, takes somewhat longer. In period 18,000 convergence is attained almost fully for the most frequently selected classifiers, and for the remaining ones, the tendency to converge is apparent.

To see how the trembling-hand and cooling parameters combine to affect the speed of learning, we set the initial value of S_{22} as high as 300 while initializing all other strengths at around their limit values. The first time period, at which $S_{21} > S_{22}$ is attained, is called the preliminary learning

N	S_{22}	S_{21}	S_{20}	S_{11}	S_{10}	S_{00}	R2.	R1.
0	46.29(19.42)	45.46(20.00)	46.54(19.89)	46.61(20.33)	45.22(19.44)	46.14(20.09)	0.320	0.519
100	43.77(11.90)	47.70(12.77)	40.27(11.27)	49.66(10.77)	38.16(14.82)	42.60(5.36)	0.608	0.869
500	46.48(6.05)	50.76(5.40)	43.96(5.91)	48.32(4.14)	39.76(9.37)	40.73(2.86)	0.870	0.985
1000	48.05(2.94)	51.59(2.45)	45.74(3.07)	48.45(2.28)	41.99(6.02)	40.62(2.06)	0.950	1.000
2000	48.81(1.95)	51.67(1.32)	46.47(1.30)	48.45(1.45)	43.31(3.33)	40.54(1.37)	0.984	1.000
3000	49.07(1.77)	51.68(0.91)	46.53(0.88)	48.47(1.04)	43.63(1.34)	40.52(1.03)	0.994	1.000
4000	49.22(1.65)	51.62(0.76)	46.49(0.72)	48.41(0.89)	43.62(0.89)	40.44(0.91)	0.996	1.000
5000	49.39(1.47)	51.56(0.68)	46.45(0.62)	48.36(0.79)	43.57(0.68)	40.40(0.78)	0.997	1.000
6000	49.46(1.34)	51.54(0.61)	46.41(0.56)	48.35(0.71)	43.54(0.61)	40.36(0.69)	0.998	1.000
7000	49.57(1.29)	51.53(0.56)	46.39(0.51)	48.35(0.66)	43.53(0.56)	40.37(0.67)	0.999	1.000
8000	49.70(1.25)	51.52(0.51)	46.38(0.46)	48.33(0.62)	43.52(0.51)	40.35(0.61)	0.996	1.000
9000	49.69(1.22)	51.51(0.49)	46.37(0.43)	48.32(0.59)	43.51(0.48)	40.31(0.58)	0.997	1.000
10000	49.71(1.14)	51.50(0.48)	46.36(0.42)	48.30(0.57)	43.50(0.48)	40.33(0.55)	0.998	1.000
11000	49.70(1.15)	51.49(0.45)	46.35(0.40)	48.32(0.52)	43.49(0.45)	40.31(0.53)	1.000	1.000
12000	49.75(1.11)	51.49(0.42)	46.35(0.38)	48.30(0.51)	43.49(0.42)	40.33(0.52)	0.997	1.000
13000	49.79(1.10)	51.48(0.42)	46.34(0.37)	48.29(0.49)	43.48(0.41)	40.32(0.49)	0.997	1.000
14000	49.78(1.10)	51.47(0.40)	46.33(0.36)	48.30(0.48)	43.47(0.40)	40.31(0.46)	0.996	1.000
15000	49.84(1.03)	51.47(0.38)	46.33(0.35)	48.29(0.44)	43.47(0.38)	40.31(0.44)	0.999	1.000
16000	49.84(1.03)	51.46(0.37)	46.32(0.33)	48.28(0.43)	43.17(0.37)	40.29(0.44)	1.000	1.000
17000	49.86(1.01)	51.46(0.36)	46.32(0.32)	48.28(0.43)	43.46(0.35)	40.29(0.42)	1.000	1.000
18000	49.91(0.98)	51.45(0.35)	46.32(0.31)	48.27(0.42)	43.45(0.35)	40.28(0.41)	0.999	1.000
Val.	50.23	51.41	46.27	48.23	43.41	40.23		

Table III: The average strengths (*S*) of *1000* simulation runs with initial strengths coming from *N(46,20)* and the ratios (*R*) of the correct decisions at states 2 and 1, p_r*=0.05* and *l=0.05*. The numbers in parentheses are the standard deviations.

duration and is recorded as a statistic.[8] The average learning duration in *50* independent runs for each parameter pair is reported in Table 4. The table

[8]After this time period, $S_{22} > S_{21}$ can temporarily be observed again due to stochastic subsidy shocks. These episodes are expected to disappear as learning progresses further given the diminishing values of the cooling sequence.

indicates that preliminary learning of the correct order of strengths happens earlier as the trembling-hand probability increases and as the cooling rate decreases.[9]

$l \setminus p_r$	0.03	0.05	0.1	0.15	0.2	0.3
0.5	4741 (2343)	2578(1471)	1619 (1089)	1263 (1013)	890 (549)	713.1 (529)
0.2	903 (451)	652 (370)	351 (232)	255 (113)	197 (101)	144.6 (62)
0.1	402 (221)	334 (130)	214 (108)	158 (68)	146 (69)	98.1 (40)
0.05	192 (58)	174 (61)	146 (54)	126 (50)	102 (48)	86.5 (30)
0.025	135 (41)	136 (40)	122 (40)	105 (32)	96 (30)	79.3 (26)

Table 4: The average time for an agent to start consuming the optimal amount. The numbers in parentheses are standard deviations. S_{22} is intentionally set to *300* while all other strengths are initially around their limit values.

5. Robustness and Sensitivity Analysis

To study the robustness and sensitivity of the convergence patterns observed above, we changed (1) the "cooling rates," (2) the "concavity" of the utility function, and (3) the probability of a subsidy. Moreover, the alpha sequence was made to decline smoothly rather than stepwise as in section 4 according to $\alpha_{imt} = 1 / ([l * \tau_{imt}] + 2)$.

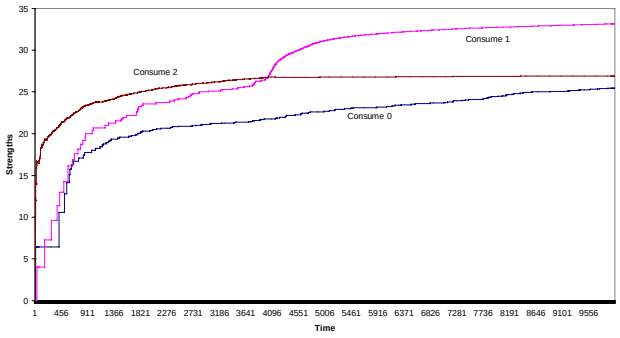
In the first three simulations, we set all of the initial strengths to zero. Then, we monitored the evolution of the strengths over time for three different cooling rates, $l=1$ (fast cooling), $l=0.05$ (slow cooling), and $l=0$ (no cooling). In cases of equality of strengths, which for instance happen initially, the tie breaking rule was the consumption of the smallest amount. This produces an initial bias for excessive savings, which disappears at the

[9]In further simulations we observed that higher trembling-hand probabilities, all the way up to *100%*, do increase the quality of *conscious behavior* at earlier times. The reason is related to the nature of the strength update formula used here. The agents assume away trembling hands in future periods in their mental accounting. This assumption of course gets more unrealistic as the actual trembling rate increases. Such high levels of mistakes, on the other hand, obviously lead to suboptimal *behavior* overall.

first instance of experimentation with any one of the other two classifiers. The trembling hand probability was set as $p_r=0.10$. The subsidy rate and the utility function values were kept at their previous values, $p_s=0.4$, $U(0)=0$, $U(1)=8$, and $U(2)=10$.

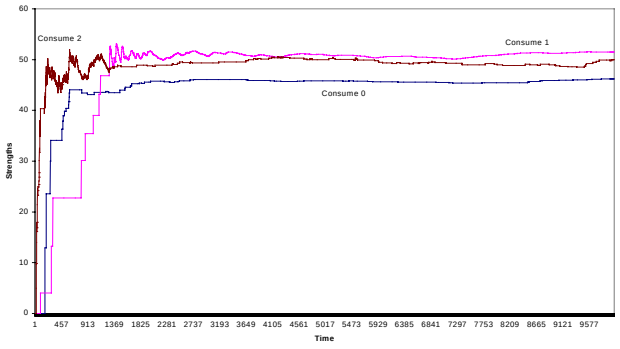
The learning pattern for the conventionally used fast cooling rate of $l=1$ (Marimon et al., 1990, Sargent, 1993) is shown in Figure 1. The figure plots the strengths of three classifiers that are activated in state 2. These are the ones that recommend zero, one, and two consumptions when the agent has two pieces of cake. The extremely slow and smooth convergence pattern is in line with the one we observed in Table 1. Even in period *10,000* the strength values are around *35*, far below their steady states which are around *50*. Nevertheless the correct ranking, hence optimal behavior, is learned around period *4,000*.

Figure 1. Classifier strengths under fast cooling (l=1)



When the cooling rate is decreased to l=0.05, adjustment of strengths becomes much faster. As seen in Figure 2, steady state values are

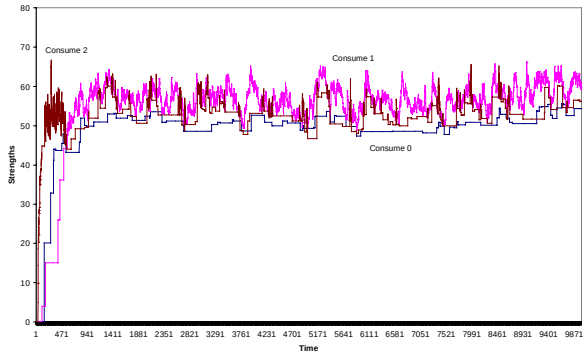
Figure 2. Classifier strengths under slow cooling (l=0.05)



approached at around period *2,000*. Nevertheless, fluctuations of strengths are more pronounced and do not completely die out until period *10,000*. The correct ranking, however, is learned at around period *1,400* and further fluctuations do not seem to be wide enough to alter the learned optimal behavior afterwards.

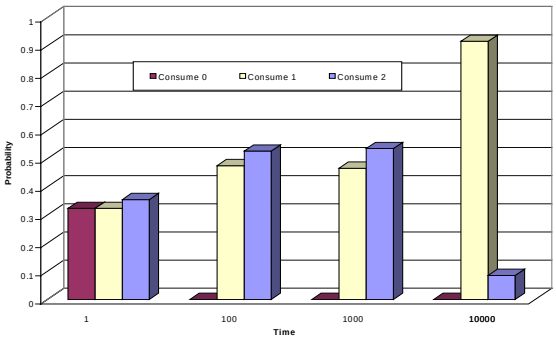
To speed up the initial convergence even further, one might consider setting the cooling rate to zero. This is the case where the consumer is always *hot* in the sense of being flexible while updating beliefs. Figure 3 indicates the validity of this expectation. Steady states are approached within the first *1,000* periods. The correct ranking of classifiers is attained before period *700*. However, fluctuations are observed to persist afterwards. This results in occasional changes in the ordering of S_{21} and classifier S_{22}. Therefore, a learning mistake due to giving too much weight to recent observations may result from time to time, in addition to the trembling hand type of mistakes. This mistake, however, is not that severe in terms of expected lifetime utility and would tend to become smaller if the constant α were fixed at a level below *1/2*.

Figure 3. Classifier strengths under no cooling (l=0)



These observations, of course, are made for a single run. To check their robustness, we conducted *1000* independent runs for each set of

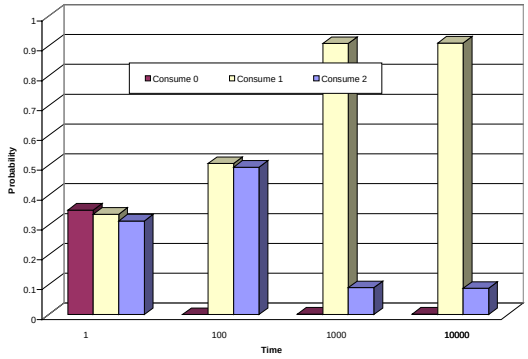
Figure 5. Learning to save one out of two cakes (fast cooling, l=1)



parameter values. This can be interpreted as a large society whose members are subject to idiosyncratic subsidy shocks. The initial strengths this time were selected independently across classifiers and consumers from *N(8,1)*. Therefore, initially a uniform distribution of behavior over three classifiers at state 2 was expected.

For the slow cooling rate of *l=0.05*, Figure 4 displays, over time, the proportions of consumers who would choose to consume 0, 1, and 2 units of cake when they possess 2 units. These proportions can also be interpreted as

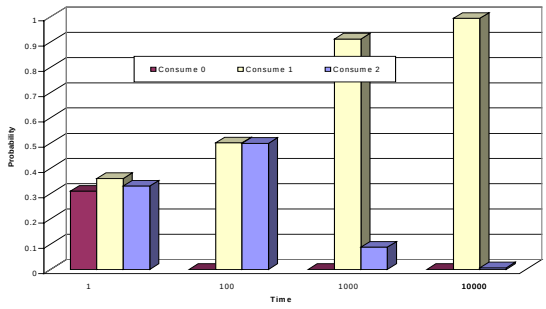
Figure 6. Learning to save one out of two cakes (no cooling, l=0)



an estimate of the probability of consuming *0*, *1*, or *2* units for a randomly selected consumer. The initial distribution is almost uniform, as expected. By period *100*, all agents learn to consume *1* or *2* pieces, at an equal degree of likelihood. Any initial tendency to save excessively quickly disappears while a tendency to consume excessively persists for at least *100* periods. This excess consumption tendency, however, diminishes all the way down to *9%* by period *1,000* and to around *1%* by period *10,000*.

For the conventionally used (fast) cooling rate of *l=1*, Figure 5 indicates that consuming excessively first increases and persists for at least *1,000* periods. It then dies out slowly all the way down to *8.5%* in period *10,000*. The probability of zero consumption becomes zero within the first *100* periods.

Figure 4. Learning to save one out of two cakes (slow cooling, l=0.05)



If, on the other hand, the cooling rate is set to zero, then the probability of consuming excessively rather quickly reaches a low level of *9%* by period *5000*, but as Figure 6 suggests, it stays there forever afterwards. This is due to persistent fluctuations in the strengths and the resulting change in the ranking of S_{21} and S_{22} observed from time to time during the experiences of each consumer. On the aggregate, this would show itself as excess consumption of a small magnitude, since saving excessively is ruled out altogether during the earlier stages of learning as the worst rule of thumb among the three.

The next numerical experiment was related to the "concavity" of the utility function. Here we set *U(1)=9* while keeping the cooling rate at the *l=0.05* level. This increases the marginal utility from the first piece consumed by one util while decreasing the marginal utility from the second piece by the same amount. This parameterization increases the incentives for saving. The results up to period *100* are not altered much. However, in period *1,000* the proportion of those consuming excessively drops to *2%* from *9%* which was observed under *U(0)=8*. This number becomes *0* as of time *10,000*, so that under stronger incentives for savings, full convergence takes place.

In contrast, increasing the subsidy rate, p_s has the consequence of reducing incentives for savings. For $p_s = 0.6$, i.e. when there is a slim incentive to save 1 unit, in period *10,000* the probability of saving one unit is only around *0.63*. When $p_s = 0.8$, the optimal behavior suggests zero savings. In this case the probability of consuming both of the two units become *0.891* and *1* in periods *1,000* and *10,000*, respectively.

5. Concluding Remarks

In this paper, we have studied classifier system learning in recursive dynamic decision problems. In numerical simulations, we have observed that once experimentation is allowed for, regardless of initial conditions, convergence to a unique vector of strengths takes place. The limit vector is observed to be consistent with the Bellman equation, so that asymptotic behavior becomes optimal. A mathematical analysis of such a convergence claim for more general settings seems worthwhile.

Secondly, we have observed that convergence of classifier strengths to their steady states becomes faster for higher trembling-hand probabilities. This result, which is somewhat surprising at first, is due to the mental

accounting system used. In formula (6), agents are assumed to pick the highest strength value for the next period's state, in updating the strength of their current classifier. This still is the case when, for instance, their trembling-hand probability is *100%*. They simply assume their hand will not tremble in the future. Therefore, mentally this becomes the fastest case for learning how to play optimally, but observationally it is totally random action.

Helpful, however, is the presence of experimentation, which appears in the form of trembling hands here, since such experimentation leads to experience with seemingly bad classifiers at early stages of learning. For instance, without trembling hands and for, say, $S_{21} < 0$ and $S_{im} > 0$ for all $(i,m) \neq (2,1)$ the consumer would never try consuming *1* unit of cake when *2* units are available, so the chance to update S_{21} would never be available. Of course, as done by Lettau and Uhlig (1999), if all of the initial strengths are taken *above* their steady state values, optimal *behavior* can always be reached. Nevertheless, learning of the augmented values corresponding to suboptimal classifiers will still not be possible. Moreover, under the absence of continual experimentation, if a structural change that increases the augmented value of a weak classifier takes place, the agents, of course, will not have a chance to detect it.

Likewise, for agents with arbitrary initial strengths, experimentation is essential to enable full learning of all strengths in the classifier system. We note, however, that experimentation never ceases here. Even after the agents learn the true values of specific actions, trembling hands will continue to lead them to mistakes. Other forms of experimentation, individual or social (cf. Başçı, 1999) may be suggested as well.

Thirdly, we have studied various values of the cooling rate. The effect of the cooling rate on the speed of convergence is tricky. In stochastic dynamic decision problems, in contrast to static ones, there are two things to learn: the expected values of payoffs and the values of the states. Without the concept of the value of a state, formulating a reinforcement learning model, of the type considered by Roth and Erev (1995) for instance, would not be possible. Also, since there are two types of objects to learn by means of a common learning algorithm, the importance of the cooling rate increases. One update is for the values as in the contraction mapping idea, and the second is for averaging out the randomness in payoffs.

Slower cooling does help in bringing the strengths around to their steady states earlier. Nevertheless, if the rate is too small, fluctuations in the

strengths around their steady states stay there for longer. Hence, there is a trade-off between bringing your beliefs into the neighborhood of their true values quickly and freezing them at these values as early as possible. The second aspect becomes more important as the extent of randomness in the system is increased while the first aspect bears more importance if non-systematic structural changes in the system take place rather frequently.

Overall, the speed of convergence is observed to be very sensitive to the "cooling rate" used in the strength update formula. Under the cooling rate used by Marimon et al. (1990), for example, convergence is unreasonably slow, taking millions of periods in our simulations. This might explain their non-convergence result for one parameterization of the Kiyotaki-Wright model of commodity money. The lesson is that if you "cool off" too early, then you essentially stop learning because you no longer update your classifiers by a sizeable amount.

In contrast, it is usually observed in experimental studies that the human subject tends to learn much faster than the conventional learning algorithms would suggest. Our suggestion is to leave the cooling rate as a free parameter to affect the learning speed. This could bring forth the possibilities of "estimating" or "calibrating" the cooling rate as well as the trembling-hand probability to deal with the behavioral observations more successfully.

Another interesting idea to pursue would be to link the cooling sequence to the trembling-hand probability in such a way that both decrease together. This would reduce the degrees of freedom by one, provide a reasonable model of learning for recursive environments, and bring a testable restriction for empirical work.

However, if the system is not really recursive, if, for instance, only occasional structural changes occur, a fixed cooling rate and a fixed experimentation probability could provide a sound rule of thumb to follow. In this case the strength update takes place as an exponentially weighted moving average (EWMA) of past perceived payoffs.[10]

---

[10]An example in the literature of the use of an EWMA is Cripps' (1991) in modeling inflation forecasts of economic agents under the structural change justification.

References

Arthur, W. B. (1991) "Designing Economic Agents That Act Like Human Agents: A Behavioral Approach to Bounded Rationality." *American Economic Review* 81:352-359.

Başçı, E. (1999) "Learning by Imitation." *Journal of Economic Dynamics and Control* 23:1569-1585.

Başçı, E. and Orhan, M. (1999) "Reinforcement Learning and Dynamic Optimization" *Bilkent University Discussion Paper* 99-8, Bilkent University, 06533, Ankara, Turkey.

Beltrametti, L., Fiorentini, R., Marengo, L., and Tamborini, R. (1997) "A Learning-to-Forecast Experiment on the Foreign Exchange Market with a Classifier System." *Journal of Economic Dynamics and Control* 21:1543-1575.

Börgers, T. and Sarin, R. (1997) "Learning Through Reinforcement and Replicator Dynamics." *Journal of Economic Theory* 77:1-14.

Bush, R. and Mosteller, F. (1955) *Stochastic Models for Learning*. New York: Wiley.

Cripps, M. (1991) "Learning Rational Expectations in a Policy Game." *Journal of Economic Dynamics and Control* 15:297-315.

Cross, J. G. (1973) "A Stochastic Model of Economic Behavior." *Quarterly Journal of Economics* 87:239-366.

Cross, J. G. (1979) "Reinforcement Theory and the Consumer Model." *Review of Economics and Statistics* 61:190-98.

Erev, I. and Rapoport, A. (1998) "Coordination, `Magic,' and Reinforcement Learning in a Market Entry Game." *Games and Economic Behavior* 23:146-75.

Erev, I. and Roth, A. E. (1998) "Predicting How People Play Games: reinforcement learning in experimental games with unique, mixed strategy equilibria." *American Economic Review* 4:848-881.

Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.

Kirman, A.P. and Vriend, N. J. (1996) "Evolving Market Structure: a model of price dispersion and loyalty." Paper presented at the Econometric Society European Meeting, Istanbul.

Lettau, M. and Uhlig, H. (1999) "Rules of Thumb and Dynamic Programming." *American Economic Review* 89:148-174.

Marimon, R., McGrattan, E., and Sargent, T. J. (1990) "Money as a Medium of Exchange in an Economy with Artificially Intelligent Agents." *Journal of Economic Dynamics and Control* 14:329-373.

Roth, A.E. and Erev, I. (1995) "Learning in Extensive-Form Games: experimental data and simple dynamic models in the intermediate term." *Games and Economic Behavior* 8:164-212.

Sargent, T. J. (1993) *Bounded Rationality in Macroeconomics.* Oxford: Oxford University Press.

Stokey, N. L., Lucas, R. E., Jr., and Prescott, E. C. (1989) *Recursive Methods in Economic Dynamics.* Cambridge: Harvard University Press.