Reinforcement learning and dynamic optimization

Erdem Başçı and Mehmet Orhan[1]

Department of Economics, Bilkent University, 06533, Bilkent, Ankara, Turkey

Abstract

This paper is about learning in dynamic environments. We introduce the concept of an augmented value function for infinite horizon discounted dynamic programs. In numerical simulations we show that presence of experimentation and 'slow cooling' enables learning, of the augmented value function and hence optimal behavior, over reasonable time horizons.

Keywords: Learning; Optimization; Classifier systems

JEL classification: D83; D91

1 Introduction

In dynamic economic models it is usually assumed that an agent's behavior is in line with solutions of a dynamic optimization problem. Since such problems are quite difficult to handle, it is frequently argued that the 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.

Reinforcement learning is possibly the most primitive learning method. It does not require the agents to form expectations or go through a sophisticated reasoning. Alternative policy choices are directly rewarded or penalized according to their consequences. Classifier systems, that are introduced by Holland (1975), are suitable tools for reinforcement learning by artificially intelligent agents. A classifier system consists of a list of *condition-*

[1]Corresponding author. Tel.: +90 312 2665716; fax: +90 312 2665140; e-mail: mehmet@bilkent.edu.tr

action statements, which are 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 systems learning is used in a number of economic models that have appeared in the literature. In repeated static decision environments, examples include Arthur (1991), Beltrametti et al. (1997) Kirman and Vriend (1996) and Sargent (1993). In the context of the Kiyotaki-Wright model of money, which is a dynamic game with a recursive structure, Marimon et al. (1990) and Başçı (1998) use classifier systems in their simulations. Lettau and Uhlig (1995), on the other hand, analyze the connection between the relevant dynamic programming problem and the asymptotic behavior of the corresponding classifier systems learning process. Lettau and Uhlig (1995) study only the limiting behavior of classifier strengths and do not allow experimentation by the agents. They show that if the classifier system is rich enough, then the strengths of the asymptotically winning classifiers converge to the values given by the solution to the Bellman's equation. They also show, however, that the strengths of the remaining classifiers may freeze in an arbitrary point in a large region of real numbers.

In the present paper, by numerical simulations, we show 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 characterize this limit vector and call it the *augmented value function*. We also study the speed of convergence, which is not addressed by Lettau and Uhlig (1995).

In the following section, we define the augmented value function in the context of a simple cake eating problem. In Section 3, we describe the details of the learning algorithm. In Section 4, we present the results of our numerical simulations. Section 5 concludes.

2 The Consumption-Savings Problem

Consider the optimization problem faced by an infinitely lived consumer who has $k_0 \in X = \{0, 1, ..., \bar{k}\}$ 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 a discrete amount of cake, again, from the set X, subject to the availability condition, $c_t \leq k_t$.

We assume an instantaneous utility function, $U : X \to \Re$, 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 assume that there is a positive probability p_s of receiving a subsidy of $\bar{k}$ 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's equation:

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

for all $k \in X$, where s is a random variable that takes on a value of $\bar{k}$ whenever the subsidy is received by the consumer and 0 otherwise. Here, $v : X \to \Re$, 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*, $\bar{v} : A \to \Re$, where $A = \{(k, c) \in X \times X \mid c \leq k\}$. The interpretation of $\bar{v}(k, c)$ will be the expected lifetime utility from consuming c units in this period, and following optimal policies thereafter. One can, therefore, define the augmented value function through the

formula:

$$\bar{v}(k,c) = U(c) + \beta E v(k-c+s), \tag{2}$$

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

$$v(k) = Max\{\bar{v}(k,c) \mid c \in X,\ c \leq k\} \tag{3}$$

for all $k \in X$.

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

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

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

In the simple numerical example that we will study here, 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, we numerically calculate the augmented value function as, $\bar{v}(2,0) = 46.27$, $\bar{v}(2,1) = 51.41$, $\bar{v}(2,2) = 50.23$, $\bar{v}(1,0) = 43.41$, $\bar{v}(1,1) = 48.23$, $\bar{v}(0,0) = 40.23$.

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. This statement follows from observing that $\bar{v}(2,1)$ is the highest among $\bar{v}(2,.)$, and $\bar{v}(1,1)$ exceeds $\bar{v}(1,0)$.

3 The Learning Algorithm

We will consider the learning problem of an agent who knows neither the augmented value function, nor the optimal policies. The agent will be assumed to have subjective values for each possible state-action pair and to update these values through experience. We call such a procedure of learning by doing, *reinforcement learning*. Classifier systems, developed as flexible and powerful tools for machine learning have a structure suitable for modeling reinforcement learning.

A classifier system consists of a potentially flexible list of condition-action statements together with their strengths. 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 among the activated ones based on the information conveyed by their strengths, follow its advice and bear the consequences (selection), 3. According to the consequences, update the strength of the classifier responsible from these (update). Then, go 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* 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, 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 in the 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). At any point in time, suppose that the consumer is at state i, i.e. has entered the current period with i units of cake, and hence i different classifiers are activated. In the selection step, with a positive probability, $1 - p_m$, the consumer will be assumed to follow the advice of the classifier which has the highest strength among the activated ones. Here, $p_m > 0$ denotes the probability of mistake taking place in a given period. The mistakes are in the form of trembling hands. Upon their realization, the consumer is assumed to randomly select one of the i activated classifiers, with uniform probability.

After the selection step, utility from consumption is realized and if all the cake has 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 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,

$$S_{im,t+1} = S_{imt} + \alpha_{imt}(U_m + \beta S^*_{jt} - S_{imt}) \tag{5}$$

where, t is the time subscript and α_{imt} is the t^{th} element 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,

$$\sum_{t=0}^{\infty} \alpha_{imt} = \infty.$$

The intuition behind the strength update formula (6) lies in its connection with 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 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 parenthesis 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,[2] it is customary to take the cooling function in the form,

$$\alpha_{imt} = \frac{1}{\tau_{imt} + 2}, \tag{6}$$

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.

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

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

$$\alpha_{imt} = \frac{1}{[\tau_{imt}/l] + 2} \tag{7}$$

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

4 Simulation Results

We have prepared a GAUSS program to implement the learning algorithm discussed above. In a single run of the algorithm with randomly generated initial strengths, $l = 1$, and $m = 5\%$, we were able to see the effects of fast cooling on the convergence pattern of classifier strengths.

Insert Table 1 about here.

As seen in Table 1, the ordering of the initial strengths is consistent with the optimal behavior. However, the values of strengths are far away from their targets depicted in the last line of Table 1. From the Table, an extremely slow tendency for 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 value for the classifiers at State 2. It is apparent that this number would, in fact, be much higher if the extrapolation were to take into account the decrease in the value of α_{imt} over time.

Insert Table 2 about here.

For the same initial values, and mistake rate but for a much slower cooling rate given by $l = 20$, the speed of convergence is observed to increase dramatically. Table 2 shows that

after around 11500 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}, which were subject to much smaller number of updates, came close to their target values.

Insert Table 3 about here.

In order to see the robustness of the observations made in Table 2, we have conducted 1000 independent runs. Table 3 summarizes the convergence pattern for a mistake rate of 5% and a cooling parameter of $l = 20$. Since the initial strengths are selected randomly across runs, initially only one third of the consumers, who start with 2 units of cake, would choose the optimal consumption level of 1 unit. Similarly only about a half of the consumers holding 1 units of cake would consume it, at the beginning. As learning proceeds, at around period 1000, 95% of all consumers have started to follow the optimal cake eating plan. The convergence of strengths to the augmented value function, however, took somewhat longer. In period 18000, the convergence was attained almost fully for the most frequently selected classifiers. And for the remaining ones, the tendency of convergence is apparent.

Insert Table 4 about here.

To see how the mistake and the cooling parameters affect the speed of learning, we have 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 learning duration and is recorded as a statistic. In 50 independent runs for each parameter pair, the average learning duration is reported in Table 4. The Table indicates that learning becomes faster as the mistake rate increases and as the cooling speed decreases.

5 Concluding Remarks

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

The presence of trembling hands is essential in learning, since they lead to experimentation 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 that the chance to update S_{21} would never be available. This shows that experimentation is essential in classifier systems learning. We note, however, that experimentation never ceases here. Even after the agents learn the true values of specific actions, the trembling hands will continue leading them to mistakes.

The speed of convergence is seen to be very sensitive to the 'cooling rate' in the strength update. For the cooling rate used by Marimon et al. (1990), for example, the convergence is unreasonably slow, taking millions of periods. In contrast, it is usually observed in experimental studies that the human subject tend to learn much faster than corresponding numerical learning algorithms. In our setup, the cooling rate is introduced as a free parameter to affect the learning speed. This could bring forth the possibilities of 'estimating' or 'calibrating' the cooling rate to confront the behavioral observations more successfully.

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., 1998. "Learning by imitation in the Kiyotaki-Wright model of money." Bilkent University Discussion Paper.

Beltrametti, L., R. Fiorentini, L. Marengo and R. Tamborini, 1997. "A learning-to-forecast experiment on the foreign exchange market with a classifier system." *Journal of Economic Dynamics and Control* 21: 1543–75.

Holland, J.H., 1975. "Adaptation in natural and artificial Systems." University of Michigan Press, Ann Arbor.

Kirman, A.P. and N.J. Vriend, 1996. "Evolving market structure: a model of price dispersion and loyalty." Paper presented at the Econometric Society European Meeting, İstanbul.

Lettau, M. and H. Uhlig, 1995. "Rules of thumb and dynamic programming." CentER, Tilburg University, manuscript.

Marimon, R., E. McGrattan and T.J. Sargent, 1990. "Money as a medium of exchange in an economy with artificially intelligent agents." *Journal of Economic Dynamics and Control* 14: 329–73.

Sargent, T.J., 1993. "Bounded rationality in macroeconomics." Oxford University Press, Oxford.

Stokey, N.L., R.E. Lucas, Jr. with E.C. Prescott, 1989. "Recursive methods in economic dynamics." Harvard University Press, Cambridge.

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	49.80	44.66	46.62	41.80	38.62
20	48.67	49.85	44.71	46.67	41.85	38.67
Value	50.23	51.41	46.27	48.23	43.41	40.23

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

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
Value	50.23	51.41	46.27	48.23	43.41	40.23

Table 2: Under slow cooling ($l = 20$), the strengths at the end of period n for a single run. Bottom row displays the augmented value function.

n	S_{22}	S_{21}	S_{20}	S_{11}	S_{10}	S_{00}	$R_{2.}$	$R_{1.}$
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
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
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
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
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
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
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
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
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
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
Value	50.23	51.41	46.27	48.23	43.41	40.23		

Table 3: 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, $m = 0.05$ and $l = 20$. Numbers in the parenthesis are the standard deviations

$l \setminus m$	0.03	0.05	0.1	0.15	0.2	0.3
2	4741 (2343)	2578 (1471)	1619 (1089)	1263 (1013)	890 (549)	713.1 (529)
5	903 (451)	652 (370)	351 (232)	255 (113)	197 (101)	144.6 (62)
10	402 (221)	334 (130)	214 (108)	158 (68)	146 (69)	98.1 (40)
20	192 (58)	174 (61)	146 (54)	126 (50)	102 (48)	86.5 (30)
40	135 (41)	136 (40)	122 (40)	105 (32)	96 (30)	79.3 (26)

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