8 Dynamic Programming

We are going to discuss multiperiod models that are more general than the CAPM, the APT, and the arbitrage results that we have already studied. In order to study these models, we need to understand a technique called dynamic programming. Dynamic programming is used to solve problems that involve optimization over time. It has been used extensively by economists in all ¯elds. Our optimization problem involves choosing consumption through time to maximize expected utility, so we will use dynamic programming to solve it.

You can read Kreps' appendix (distributed in class) to gain some intuition for the dynamic programming technique. Kreps gives several references to more advanced texts on dynamic methods. If you are interested in working with these types of models, you will probably want to consult a more complete text. There are several alternatives to dynamic programming for solving multiperiod problems. For example, there is a technique know as optimal control that is sort of a continuous-time counterpart to dynamic programming.

Dynamic programming problems can always be categorized as either ¯nite or in¯nite horizon problems. While the techniques for solving these two types of problems are somewhat di®erent, the inferences obtained from the two types are usually the same. For simplicity, we will discuss the ¯nite horizon case ¯rst. After understanding ¯nite horizon problems, we will examine the implications of allowing the horizon to go to in¯nity.

Dynamic programming works well for problems in which agents make their decisions based on just a few variables that we will call state variables. In statistical terms, it works well when just a few state variables are su±cient statistics for predicting the future. It is fairly common in dynamic programming models to assume that all state variables are current values, such as current wealth or current prices. When only current state variables are assumed to matter, we say that the model is a Markovian

model. A Markov process is a stochastic process that satisfies,

$$\text{Markov Process}: \quad f(x_t | x_{t-1}; x_{t-2}; \ldots) = f(x_t | x_{t-1}): \tag{137}$$

8.1 Dynamic Programming with a Finite Horizon

We need to set up some notation before proceding with our discussion of dynamic programming. Let $I = \{1; 2; 3; \ldots\}$ be the set of possible future states and let A be a finite set of feasible actions that you can take. Define R(i,a) as the expected current reward when the state is $i \in I$ and the action chosen is $a \in A$. Define the value function, $V_n(i)$, as the maximum attainable sum of expected current and future rewards when n periods remain and the current state is $i \in I$.

In finite horizon problems, we always start with optimization in the last period and then work backwards to get to the present decision. We begin by thinking about what the value function will be when we have just one period left,

$$V_1(i) = \max_{a \in A} R(i; a); : \tag{138}$$

The optimal policy in the last period is to just maximize your reward given the state. Now let $p_{ij}(a)$ equal the probability that state j occurs next period given that state i describes today and that you choose action a. We can express your value function with two periods left as a function of your final value function,

$$V_2(i) = \max_{a \in A} \{R(i; a) + \sum_j p_{ij} V_1(j)\} \tag{139}$$

If we define $a_2(i)$ as your optimal policy when the current state is i and you have two periods to go, then we want to find a function, $a_2(i)$ that solves (139). We can do this by first finding the optimal policy with one period left, $a_1(i)$, by solving (138). Second, we plug our values for $a_1(i)$ into (139) and in a third step we solve (139). This is what

dynamic programming is all about. We always solve multiperiod problems by thinking about what will be optimal at the end of the problem and then working backwards to determine what is optimal at the beginning of the problem.

Of course, we can generalize this to more than two periods. In general, we take the value function with n periods remaining to be the maximum of the reward function de¯ned over future states and actions,

$$V_n(i) = \max_{a2A} R(i_n; i_{n\,i\,1}; : : : ; i_1; a_n; a_{n\,i\,1}; : : : ; a_1); \tag{140}$$

where $a_¿$ is the action chosen with ¿ periods left and $i_¿$ is the state of the world with ¿ periods left. Because problems like this are usually intractable, we often assume that the reward function is additively separable,

$$V_n(i) = \max_{a2A} \sum_{t=1}^{x} R_t(i; a); \tag{141}$$

where $R_t(i; a)$ indicates the reward function at period t. Additive separability is a fairly strong assumption that may not always be warranted. Some recent work on habit persistance and alternative utility formulations has challenged the assumption that utility is additively separable.[9]

Using a result known as the princple of optimality, the value function can be restated by the Bellman equation:

$$V_n(i) = \max_{a2A} f R(i; a) + \sum_{j}^{x} p_{ij} V_{n\,i\,1}(j) g: \tag{142}$$

The principle of optimality basically states that if a particular strategy is optimal for each point in time at that point in time and if an optimal strategy is going to be followed

[9]See Epstien, L. and S. Zin (1991) \Substitution, Risk Aversion, and the Temporal Behavior of Consumption and Asset Returns: An Empirical Analysis," Journal of Political Economy, 99, p. 263-86.

for all future points in time then the particular strategy is optimal. Kreps motivates this with a little math. He says we can easily convert the problem

$$\max_{x;y} f(x;y) \tag{143}$$

into the equivalent problem

$$\max_{x}[\max_{y} f(x;y)]: \tag{144}$$

This mathematical operation is essentially what we have done above in converting the value equation, (141), into the Bellman equation, (142). We use the Bellman equation to learn about the optimal policy function, $a_n(i)$. Once again, the optimal policy function is a rule that describes the optimal choice of action when the state is i and there are n periods left.

There are three principal ways to use Bellman equations:

1. We can sometimes use Bellman equations to obtain explicit analytic solutions for $V_n(i)$ and $a_n(i)$.

2. We often use Bellman equations to characterize the properties of $V_n(i)$ and $a_n(i)$.

3. We can sometimes solve for $V_n(i)$ and $a_n(i)$ numerically. We can always do this in principle, but some problems become too large to be tractable.

The Bellman equation is a fundamental building block in a dynamic programming model. Deriving the appropriate Bellman equation is usually the ¯rst step in analyzing a dynamic, ¯nite horizon model.

Once we have a Bellman equation, we typically look at the ¯rst order conditions that solve the equation's maximization problem. These ¯rst order conditions are often rich enough to provide us with the elements of an interesting model. We also use a condition called an envelope condition at times. Envelope conditions are derived by applying the envelope theorem. The envelope theorem can be understood as follows.

Suppose that we want to maximize $f(x;a)$ over x. We can think of a as being a state variable and x as being a choice variable. For every value of a in this problem there will be a maximizing value of x. In what Varian calls "sufficiently regular" cases, we can think of defining a function, $x(a)$ that gives the optimal x value for each value of a. We can also think of the value function in these terms as $V(a) = f(x(a);a)$. If we take the derivative of the value function with respect to the state variable, we obtain

$$\frac{\partial V(a)}{\partial a} = \frac{\partial f(x(a);a)}{\partial x}\frac{\partial x(a)}{\partial a} + \frac{\partial f(x(a);a)}{\partial a}. \tag{145}$$

But we know that $x(a)$ is the value of x that maximizes f, so

$$\frac{\partial f(x(a);a)}{\partial x}\frac{\partial x(a)}{\partial a} = 0; \tag{146}$$

and

$$\frac{\partial V(a)}{\partial a} = \frac{\partial f(x(a);a)}{\partial a}\Big|_{x=x(a)}. \tag{147}$$

This is a very simple statement of the envelope theorem. In the dynamic programming context, if we take the derivative of the value function with respect to the state variables and if we hold the choice variables (the actions) at their optimal levels, then we can consider the derivatives of the value function with respect to the choice variables to be equal to zero.

8.2 Example: The Gambler's Problem

Let's work through a simple example to illustrate the method. Suppose that in each of T periods a gambler can bet up to his entire wealth. With probability p the gambler wins and the size of his reward is equal to the size of his bet. With probability $(1 - p)$ the gambler loses the amount of his bet. The gambler's objective is to maximize $E[\ln(\text{final wealth})]$. Let x equal the gambler's current wealth and $\theta \in [0;1]$ equal the

fraction that he chooses to bet this period. So in this problem, x is the state variable and ® is the action or the choice variable. The gambler's Bellman equation is

$$V_n(x) = \max_{®2[0;1]} fpV_{n¡1}(x + ®x) + (1 ¡ p)V_{n¡1}(x ¡ ®x)g \tag{148}$$

This Bellman equation is very simple because there is no current reward. Since the gambler has log utility, we also know that

$$V_0 = \ln(x): \tag{149}$$

We will show that when $p \cdot \frac{1}{2}$, the optimal policy function is $®_n(x) = 0\ 8n; x$.

With one gamble left, the gambler has the value function,

$$V_1(x) = \max_{®2[0;1]} fp \ln(x + ®x) + (1 ¡ p) \ln(x ¡ ®x)g \tag{150}$$

We can solve for the ¯rst order condition for this problem,

$$\frac{@}{@®} = \frac{px}{x + ®x} ¡ \frac{(1 ¡ p)x}{x ¡ ®x} \tag{151}$$

which implies that the optimal value for ® is

$$® = 2p ¡ 1 \tag{152}$$

When $p \cdot \frac{1}{2}$, the optimal fraction of wealth to gamble is less than or equal to zero. Thus, for $p \cdot \frac{1}{2}$, it is never optimal for the gambler to gamble in the last period.

We can prove that this is true for any n. We will use a proof of induction, which contains the following steps:

1. Basis step - prove the hypothesis for n = 1, the starting point.

2. Induction hypothesis step - suppose that the hypothesis is true for n ¡ 1.

3. Induction step - under the supposition, prove the hypothesis for n.

If we follow these steps then our proof by induction will be complete.

We showed above that $V_1(x) = \ln(x)$ because it will never be optimal for the gambler to gamble in the last period. We will now hypothesize that $V_{n-1}(x) = \ln(x)$. We need to show that $V_n(x) = \ln(x)$ under this supposition. This must be true since we can express our value function as,

$$V_n(x) = \max_{\alpha \in [0,1]} \{p \ln(x + \alpha x) + (1 - p) \ln(x - \alpha x)\}; \tag{153}$$

$$= \max_{\alpha \in [0,1]} \ln(x) + \{p \ln(1 + \alpha) + (1 - p) \ln(1 - \alpha)\}; \tag{154}$$

and because the maximum value of the term in brackets is zero. Thus, the value function with n periods to go is

$$V_n(x) = \ln(x) \tag{155}$$

The problem in every period is the same and our proof by induction is finished.

What about the case when $p > \frac{1}{2}$? When $p > \frac{1}{2}$ the optimal α with one period remaining will still be given by $2p - 1$. If we substitute this value of α into the value function with one period left, we obtain

$$V_1(x) = p \ln[x + (2p - 1)x] + (1 - p) \ln[x - (2p - 1)x] \tag{156}$$

$$= \ln[x] + \ln[2] + p \ln[p] + (1 - p) \ln[1 - p] = \ln[x] + C: \tag{157}$$

We will leave as a homework problem the task of deriving the value function in this problem and showing that the optimal policy function is always

$$a_n(x) = 2p - 1 \tag{158}$$

for each period, n. Note that this policy rule is a stationary rule - it does not change with time. This is a nice property for optimal policy functions to have.

8.3 Dynamic Programming with an In¯nite Horizon

The models that we will examine all have ¯nite horizons. It is useful, however, to know what happens to dynamic programming when the horizon is not assumed to be ¯nite. There are two big changes that occur when going from ¯nite problems to in¯nite problems.

First, in ¯nite problems, we can always start at the last period and then work forward to derive our answer. In in¯nite horizon problems, there is no last period to begin at. Thus, we cannot usually just write down a Bellman equation and derive an optimal policy. Rather, we have to conjecture a form for the value function and an optimal policy rule and then we have to determine whether these conjectures are correct. We usually make conjectures that seem reasonable - people often \tweak" results that others have found in the past. We validate our conjectures by showing that it is not possible to improve upon our policy functions. Something very similar to the induction proof outlined above is implemented for this purpose.

Second, in in¯nite horizon problems we usually need some sort of convergence result that is commonly referred to as a transversality condition. What do transversality conditions look like? They can look something like

$$\lim_{t \to 1} E_t \, {}^{-t} \frac{@V_t(x)}{@x} x = 0 \tag{159}$$

Intuitively, they can involve restrictions like the restriction that the discounted terminal value of a stock goes to zero as the time to liquidation goes to in¯nity. If you want to know more about in¯nite horizon methods you can consult one of the references in Kreps.

8.4 Homework Problems

1. Solve the gambler's problem for the case when $p > \frac{1}{2}$. Derive the value function at each point in time, V_n. Show by induction that the optimal policy is always given by (158).

2. Do problem number 1 in Kreps' appendix.