Intro to Dynamic Programming

Xiaoyang Li

Friday 20th October, 2023

1 Neoclassical growth model

Suppose we are in a the standard neoclassical growth model and households solve

$$\begin{aligned}
&\max_{\{c_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t \ln c_t \\
&s.t.\ c_t + k_{t+1} = z k_t^{\alpha} + (1-\delta)\, k_t \ \forall t \text{ “law of motion”} \\
&c \geq 0 \\
&k_{t+1} \geq 0 \\
&k_0 \text{ given}
\end{aligned}$$

1.1 Neoclassical growth as a Bellman equation

The problem above is written from the viewpoint of someone looking forward from period 0. What if we wanted to write it in terms of someone considering a decision at any arbitrary period t, taking into account their preferences in the next period t'?

As a starting period, we know that in this period we get to consume $\ln(c)$. What about next period? Here is where the magic of the recursive formulation comes in: when we think about next period, we not only think about our next period utility, but our lifetime utility!

Instead of summing up all our period utilities, we are nesting our period utilities. The difference between the two setups is that in the first, we make one single decision in period 0 about how to allocate consumption and savings in every period t, whereas in the second it's the reverse: we make infinitely many decisions in every period t.

In some sense, dynamic programming is the ultimate procrastinator's way of making decisions. How am I planning to consume and save? Well, I choose some consumption today and defer other maximizing decisions to tomorrow, knowing that eventually I will maximize my lifetime utility. Every day I wake up, I look at what assets I own call k, and given that I want to maximize my lifetime utility, I choose my consumption c and

the amount of assets that I want to leave for tomorrow call k'. Then I just defer all my other planmaking until tomorrow. Although I'm trying to maximize my lifetime utility, since it's defined recursively to distinguish it we call it our "value" V. V takes in k because it's the only thing I need to decide what to consume c and what to save k'.

$$\begin{aligned} V(k) &= \max_{c,k'} \left[\ln(c) + \beta V(k')\right] \\ \text{such that } c_t &\geq 0,\ k' \geq 0 \end{aligned}$$

Next, let's write our recursive problem as just a function of the state variable k. In the law of motion, we can solve for c in terms of k:

$$\begin{aligned} c_t &= z k_t^{\alpha} + (1-\delta) k_t - k_{t+1} \\ \text{or } c &= z k^{\alpha} + (1-\delta) k - k' \end{aligned}$$

Subbing in now, we have

$$V(k) = \max_{k'} \left[\ln\left(z k^{\alpha} + (1-\delta) k - k'\right) + \beta V(k')\right] \tag{1}$$

where note that the boundaries for k' are

$$k' \in [0, z k^{\alpha} + (1-\delta) k]$$

This comes from the resource constraint, imposing the condition $c \geq 0$.

1.2 FOC and Envelope Condition

The optimality conditions always comes from your choice and state variables (see the document "Bellman for kids"!). In this case, our choice variable is k' and our state variable is k.

At any arbitrary period, given k, we can control how much k' we have next period (by consuming more or less). That's why intuitively, we are choosing k' so that

$$\text{FOC: } V_{k'}(k) = \frac{\partial V}{\partial k'} = -\frac{1}{z k^{\alpha} + (1-\delta) k - k'} + \beta V_{k'}(k') = 0 \tag{2}$$

Therefore,

$$\frac{1}{z k^{\alpha} + (1-\delta) k - k'} = \beta V_{k'}(k') \tag{3}$$

Note that the quantity on the left hand side is

$$\frac{\partial u}{\partial k'} = \frac{\partial \text{Utility Today}}{\partial k'}$$

We are close. But what is $V_{k'}(k')$?

1.2.1 The Envelope Condition

In one sentence: the envelope condition is the idea that we can ignore the chain rule when we take the derivative of next period V with respect to the state variable k. Essentially, we ignore the fact that the choice variable k' is a function of k. Why? k' is really $\arg\max k'$, and has already been optimized.

Back to the question of what we do with $V_{k'}(k')$. Let's look at $V_k(k)$ instead; it may be more tractable. Then we can just iterate forward to get $V_{k'}(k')$ – keep reading!

Look at equation 1. To get equation 1, we plugged in the budget constraint $c = zk^{\alpha} + (1-\delta)k - k'$. Notice that k' is in fact a function of k and we may think we need to use the chain rule on k' when we differentiate V. But no! This is where the Envelope condition comes in: we don't need the chain rule because k' has already been maximized in the FOC (equation 2)! Indeed, we can think of the k' in the function as really the $\arg\max k'$. Hence, we get

$$V_k(k) = \frac{\partial V}{\partial k} = \frac{\alpha z k^{\alpha-1} + (1-\delta)}{zk^{\alpha} + (1-\delta)k - k'} \tag{4}$$

Notice I ignored the $\beta V(k')$ in equation 1. We just used the Envelope condition folks.

1.3 Write the Euler Equation

What is the Euler equation? The Euler equation is an intertemporal condition that relates the marginal utility of consuming today with the marginal utility of saving that unit and consuming tomorrow. How do we get there? Look at equation 4. We can iterate forward one period to get

$$V_{k'}(k') = \frac{\alpha z k'^{\alpha-1} + (1-\delta)}{zk'^{\alpha} + (1-\delta)k' - k''}$$

Aha! We can now complete equation 3 by filling in for $V_{k'}(k')$. Let's plug in to get

$$\begin{aligned} \frac{1}{zk^{\alpha} + (1-\delta)k - k'} &= \beta \frac{\alpha z k'^{\alpha-1} + (1-\delta)}{zk'^{\alpha} + (1-\delta)k' - k''} \\ \frac{\partial \text{Utility Today}}{\partial k'} &= \text{discount} * \frac{\partial \text{Utility Tomorrow}}{\partial k'} \end{aligned}$$

This is the intertemporal Euler equation.

2 Resources

Alex Albright's Introduction to Bellmans
https://thelittledataset.com/2017/03/21/a-bellman-equation-about-nothing/