Dynamic Programming

An Introduction

Rong Li

School of Finance
Renmin University of China

July 2, 2022

What you can learn

- A basic introduction about dynamic programming in discrete time
- Indirect utility
- Dynamic optimization: A cake-eating example
- Some extensions of the cake-eating problem
- General formulation and the Contraction Mapping Theorem

Indirect Utility: Consumers

Consumer choice theory focuses on households that solve

$$V(I,p) = max_c u(c)$$
$$\text{subject to } pc = I$$

where, c is a vector of consumption goods, p is a vector of prices and I is income.
$V(I,p)$: an indirect utility function, the maximized level of utility from current state (I,p).

Key logic: someone in this state can be predicted to attain this level of utility. One does not need to know what that person will do with his income as long as he will act optimally.

Indirect Utility: Consumers

First Order Condition:

$$\frac{u_j(c)}{p_j} = \lambda \text{ for all j}$$

where, λ is the multiplier on the budget constraint and $u_j(c)$ is the marginal utility from good j.

What happens if we give the consumer a bit more income? Welfare goes up by $V_I(I, p) > 0$

Can the researcher predict what will happen with a little more income? Not really since the optimizing consumer is indifferent with respect to how this is spent:

$$\frac{u_j(c)}{p_j} = V_I(I, p) \text{ for all j}$$

Indirect Utility: Firms

Suppose a firm must choose how many workers to hire at a wage w given its stock of capital k and product price p.

$$\Pi(w, p, k) = max_l f(l, k)p - wl$$

$\Pi(w, p, k)$ summarizes the value of the firm given factor prices, the product price p, and the stock of capital k.

Given $\Pi(w, p, k)$, we can directly compute the marginal value of allowing the firm some additional capital as $\Pi_k(w, p, k) = pf_k(l, k)$ without knowing how the firm will adjust its labor input in response to the additional capital.

Dynamic Optimization: A Cake-Eating Example

Suppose you are presented with a cake of size W_1. At each point of time, $t = 1, 2, 3, ..., T$, you can eat some of the cake but must save the rest.

$$\sum_{t=1}^{T} \beta^{t-1} u(c_t)$$

where $0 \leq \beta \leq 1$ is called the discount factor. We assume that $u()$ is real valued, differentiable, strictly increasing, strictly concave and $lim_{c \to 0} u'(c) \to \infty$.

For now, we assume that the cake does not depreciate or grow. The evolution of the cake over time:

$$W_{t+1} = W_t - c_t \text{ for } t = 1, 2, ..., T$$

How would you find the optimal path of consumption?

A Cake-Eating Example: Direct Attack

One approach is to solve the constrained optimization problem directly.

$$max_{\{c_t\}_1^T,\{W_t\}_1^T} \sum_{t=1}^{T} \beta^{t-1} u(c_t)$$

$$\text{s.t. } W_{t+1} = W_t - c_t \text{ for } t = 1, 2, ..., T$$

Also there are nonnegativity constraints on consuming the cake given by $c_t \geq 0$ and $W_t \geq 0$. W_1 is given.

The flow constraints for each t could be combined, yielding:

$$\sum_{t=1}^{T} c_t + W_{T+1} = W_1 \text{ and } c_t \geq 0 \text{ and } W_{T+1} \geq 0$$

How would you find the optimal path of consumption?

A Cake-Eating Example: Direct Attack

Letting λ be the multiplier on the constraint and ϕ be the multiplier on the nonnegativity constraint on W_{T+1}. Largrange:

$$L = \sum_{t=1}^{T} \beta^{t-1} u(c_t) + \lambda(W_1 - \sum_{t=1}^{T} c_t - W_{T+1}) + \phi W_{T+1}$$

The first order conditions:

$$\beta^{t-1} u'(c_t) = \lambda \text{ for } t = 1, 2, ..., T$$

$$\text{and } \lambda = \phi$$

The nonnegativity constraints on c_t are ignored, as we assume $lim_{c \to 0} u'(c) \to \infty$.

Combining the equations:

$$u'(c_t) = \beta u'(c_{t+1})$$

This is a necessary condition of optimality for any t, called Euler equation.

A Cake-Eating Example: Direct Attack

Euler equation:

$$u'(c_t) = \beta u'(c_{t+1})$$

As long as the problem is finite, the Euler equation holds across all adjacent periods implies that any finite deviations from a candidate solution that satisfies the Euler equations will not increase utility.

Is this enough? Not quite. Imagine a candidate solution that satisfies the Euler equations but has the property that $W_T > c_T$ so that there is cake left over.

The Euler equation is necessary but not sufficient. Need $W_{T+1} = 0$. This constraint is binding because $\lambda = \phi > 0$.

A Cake-Eating Example: Direct Attack

The nonnegativity constraint serves two important purposes.

First, in the absence of $W_{T+1} \geq 0$, the agent would want to set $W_{T+1} = -\infty$.

Second, the fact that the constraint is binding in the optimal solution guarantees that cake does not remain after period T.

The problem is pinned down by an initial condition W_1 is given and a terminal condition $W_{T+1} = 0$ and the set of T-1 Euler equations.

A Cake-Eating Example: Direct Attack

The solution to this problem be denoted by $V_T(W_1)$, where T is the horizon of the problem and W_1 is the initial size of the cake.

$V_T(W_1)$ represents the maximal utility flow from a T-period problem given a size W_1 cake. We call this a value function.

A Cake-Eating Example: Dynamic Programming Approach

Suppose we change the problem slightly: add a period 0 and give an initial cake of size W_0.

By adding a period 0 to our original problem, we can take advantage of the information provided in $V_T(W_1)$.

$$max_{c_0} u(c_0) + \beta V_T(W_1)$$
$$\text{where}$$
$$W_1 = W_0 - c_0, \; W_0 \text{ given}$$

Instead of choosing a sequence of consumption, we just find c_0. Then W_1 is determined and the value of the problem is given by $V_T(W_1)$. This is a recursive formulation.

A Cake-Eating Example: Dynamic Programming Approach

For the purposes of the dynamic programming problem, it does not matter how the cake will be consumed after the initial period.

All that is important is the agent will be acting optimally and thus generating utility given by $V_T(W_1)$.

This is the principle of optimality, due to Richard Bellman.

A Cake-Eating Example: Dynamic Programming Approach

The first order condition is given by

$$u'(c_0) = \beta V_T'(W_1)$$

The marginal gain from reducing consumption a little in period 0 is summarized by the derivative of the value function.

We know $V_T'(W_1) = u'(c_1) = \beta^t u'(c_t + 1)$, for $t = 1, 2, ..., T - 1$
Yields, $u'(c_t) = \beta u'(c_{t+1})$, for $t = 0, 1, 2, ..., T - 1$. A familiar Euler equation.

Some extensions of the cake-eating problem: Infinite Horizon

$$max_{\{c_t\}_1^\infty, \{W_t\}_2^\infty} \sum_{t=1}^{\infty} \beta^t u(c_t)$$

$$\text{s.t. } W_{t+1} = W_t - c_t \text{ for } t = 1, 2, \ldots$$

In specifying this as a dynamic programming problem, we write

$$V(W) = max_{c \in [0,W]} u(c) + \beta V(W - c) \text{ for all W}$$

$V(W)$ is the value of the infinite horizon problem starting with a cake of size W.

Some extensions of the cake-eating problem: Infinite Horizon

$$V(W) = max_{c \in [0,W]} u(c) + \beta V(W - c) \text{ for all W}$$

The STATE VARIABLE is the size of the cake W.

The CONTROL VARIABLE is the variable being chosen.

The dependence of the state tomorrow on the state today and the control toay, given by $W' = W - c$ is called the TRANSITION EQUATION.

Some extensions of the cake-eating problem: Infinite Horizon

Alternatively, we can specify the problem so that instead of choosing today's consumption we choose tomorrow's state:

$$V(W) = max_{W' \in [0,W]} u(W - W') + \beta V(W') \text{ for all W}$$

This expression is known as a functional equation, called BELLMAN EQUATION.

The unknown in the Bellman equation is the value function itself: the idea is to find a function V(W) that satisfies this condition for all W.

We can express all relations without an indication of time. This is the essence of stationarity.

Some extensions of the cake-eating problem: Infinite Horizon

The next part of this lecture addresses the question of whether there exists a value function that satisfies the Bellman equation. For now, we assume that a solution exists.

The first order condition is $u'(c) = \beta V'(W')$

The Benveniste-Scheinkman (BS) condition is $V'(W) = u'(c)$

Combine the two, get Euler equation, $u'(c) = \beta u'(c')$

The link from consumption and the next period's cake to the size of the cake is called policy function:

$$c = \phi(W), W' = \psi(W) \text{ for all W}$$

Some extensions of the cake-eating problem: Taste Shocks

A convenient feature of the dynamic programming problem is the ease with which uncertainty can be introduced.

Suppose that utility over consumption is given by, $\varepsilon u(c)$

where ε is a random variable with $\varepsilon \in \{\varepsilon_h, \varepsilon_l\}$ and both of them are greater than zero.

We assume that the taste shock follows a first-order Markov process, which means that the probability that a particular realization of ε occurs in the current period depends ONLY on the value of ε in the previous period.

Denote, $\pi_{lh} = Prob(\varepsilon' = \varepsilon_h | \varepsilon = \varepsilon_l)$. And Π with elements of π_{ij} is called transition matrix.

Some extensions of the cake-eating problem: Taste Shocks

The Bellman equation is written:

$$V(W,\varepsilon) = max_{W'\in[0,W]}\varepsilon u(W-W') + \beta E_{\varepsilon'|\varepsilon} V(W',\varepsilon') \text{ for all } (W,\varepsilon)$$

The first order condition is: $\varepsilon u'(W-W') = \beta E_{\varepsilon'|\varepsilon} V_1(W',\varepsilon')$ for all (W,ε)

The Benveniste-Scheinkman (BS) condition is $V_1(W,\varepsilon) = \varepsilon u'(W-W')$

Combine the two, get Euler equation, $\varepsilon u'(W-W') = \beta E_{\varepsilon'|\varepsilon}[\varepsilon' u'(W'-W'')]$

Some extensions of the cake-eating problem: Discrete Choice

To illustrate the flexibility of the dynamic programming approach, we show a discrete choice problem.

Suppose that the cake must be eaten in one period as a whole. And we modify the transition equation to allow the cake to depreciate at rate ρ. An optimal stopping problem.

Given the current taste shock, ε, then

$$V^E(W,\varepsilon) = \varepsilon u(W)$$

and

$$V^N(W,\varepsilon) = \beta E_{\varepsilon'|\varepsilon} V(\rho W, \varepsilon')$$

where

$$V(W,\varepsilon) = max(V^E(W,\varepsilon), V^N(W,\varepsilon)) \text{ for all } (W,\varepsilon)$$

Some extensions of the cake-eating problem: Discrete Choice

$$V^E(W,\varepsilon) = \varepsilon u(W)$$

and

$$V^N(W,\varepsilon) = \beta E_{\varepsilon'|\varepsilon} V(\rho W, \varepsilon')$$

where

$$V(W,\varepsilon) = max(V^E(W,\varepsilon), V^N(W,\varepsilon)) \text{ for all } (W,\varepsilon)$$

$\varepsilon u(W)$ is the direct utility flow from eating the cake. Once the cake is eaten, the problem has ended. So $V^E(W,\varepsilon)$ is just a one-period return.

If the agent waits, then in the next period the cake is of size ρW.

The agent needs to make decision between these two discrete choices.

General Formulation: Nonstochastic Case

A payoff function for period t given by $\hat{\sigma}(s_t, c_t)$

where, s_t is the state vector and c_t is the control vector.

The transition equation is $s_{t+1} = \tau(s_t, c_t)$.

NOTE: The state vector completely summarizes all of the information from the past that is needed to make a forward-looking decision.

We assume $c \in C$ and $s \in S$ and $\hat{\sigma}(s, c)$ is bounded for $(s, c) \in S \times C$.

Two properties: stationarity (not depending on time) and discounting $(0 < \beta < 1)$.

General Formulation: Nonstochastic Case

The Bellman equation:

$$V(s) = max_{c \in C(s)} \hat{\sigma}(s, c) + \beta V(s') \text{ for all } s \in S.$$

where, $s' = \tau(s, c)$.

Alternatively,

$$V(s) = max_{s \in \Gamma(s)} \sigma(s, s') + \beta V(s') \text{ for all } s \in S.$$

We need to find the value function that satisfies the Bellman equation.

General Formulation: Nonstochastic Case

Theorem (1)

Assume that $\sigma(s, s')$ is real-valued, continuous, and bounded, $0 < \beta < 1$, and that the constraint set, $\Gamma(s)$, is nonempty, compact and continuous. Then there exists a unique value function $V(s)$ that solves the Bellman equation.

An intuitive sketch, we define an operator:

$$T(W)(s) = max_{s \in \Gamma(s)} \sigma(s, s') + \beta W(s') \text{ for all } s \in S.$$

We take a guess on the value function and produce another value function, $T(W)(s)$. The fixed point, $V(s) = T(V)(s)$, of the mapping is a solution.

We need to show the $T(W)$ is a contraction. Need to show monotonicity and discounting.

General Formulation: Nonstochastic Case

Monotonicity: if $W(s) \geq Q(s)$ for all $s \in S$, then $T(W)(s) \geq T(Q)(s)$ for all $s \in S$.

Let $\phi_Q(s)$ be the policy function obtained from

$$max_{s \in \Gamma(s)} \sigma(s, s') + \beta Q(s') \text{ for all } s \in S.$$

Then,

$$T(W)(s) = max_{s \in \Gamma(s)} \sigma(s, s') + \beta W(s') \geq \sigma(s, \phi_Q(s)) + \beta W(\phi_Q(s)) \geq \\ \sigma(s, \phi_Q(s)) + \beta Q(\phi_Q(s)) = T(Q)(s) \text{ for all } s \in S.$$

General Formulation: Nonstochastic Case

Discounting: adding a constant to W leads T(W) to increase by less than this constant.

We have

$$T(W+k)(s) = max_{s \in \Gamma(s)} \sigma(s,s') + \beta[W(s')+k] = T(W)(s) + \beta k < T(W)(s) + k \text{ for all } s \in S.$$

Since we assume that the discount factor is less than 1.

General Formulation: Nonstochastic Case

The fact that T(W) is contraction allows us to take advantage of the contraction mapping theorem.

This theorem implies that there is unique fixed point and this fixed point can be reached by an iteration process using an arbitrary initial condition (value function iteration).

Guess $V_0(s)$ for all $s \in S$, then $V_1 = T(V_0)$. If $V_1 = V_0$ for all $s \in S$, we have the solution. Else, $V_2 = T(V_1)$, and continue iterating until $T(V) = V$.

Assignment: suppose $u(c) = log(c)$, perform the value function iteration of our cake eating problem using any computer language.

General Formulation: Nonstochastic Case

Theorem (2)

Assume that $\sigma(s, s')$ is real-valued, continuous, CONCAVE, and bounded, $0 < \beta < 1$, that S is convex subset of R^k, and that the constraint set, $\Gamma(s)$, is nonempty, compact and continuous. Then the unique solution is stricly concave. Further the policy function is a continuous, single-valued function.

The proof of the theorem relies on showing that strict concavity is preserved by $T(V)$. Given that $\sigma(s, s')$ is concave, then the initial guess can be:

$$V_0(s) = max_{s' \in \Gamma(s)} \sigma(s, s')$$

V_0 is strictly concave. Since $T(V)$ preserves this property, the solution to the Bellman equation is strictly concave.

General Formulation: Stochastic Dynamic Programming

Let $\varepsilon \in \Phi$, represent the vector of shocks and it is a first order Markov process.

The Bellman equation is

$$V(s,\varepsilon) = max_{s \in \Gamma(s,\varepsilon)} \sigma(s, s', \varepsilon) + \beta E_{\varepsilon'|\varepsilon} V(s', \varepsilon') \text{ for all } (s,\varepsilon).$$

Theorem (3)

Assume that $\sigma(s, s', \varepsilon)$ is real-valued, continuous, concave, and bounded, $0 < \beta < 1$, and that the constraint set, $\Gamma(s,\varepsilon)$, is nonempty, compact and convex. Then,

1. *there exists a unique value function $V(s,\varepsilon)$ that solves the above Bellman equation.*
2. *there exists a stationary policy function, $\phi(s,\varepsilon)$.*

With $\beta < 1$, discounting holds and monotonicity holds as before. (An application of Blackwell's theorem.)

Conclusion

This lecture has provided a theoretical structure for the dynamic optimization problems that can be used widely in economic analysis.

Slides can be found at www.rongli.cc

The End