Dynamic Programming

Zi Wang
HKBU

Spring 2025

Cake-Eating Problem

- Infinite time: $t = 0, 1, \ldots,$ $W_0 > 0$ is given

$$V(W_0) \equiv \max_{(c_t)_{t=0}^{\infty}, (W_t)_{t=1}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(c_t) \tag{1}$$
$$\text{s.t. } W_{t+1} = W_t - c_t, \quad W_{t+1} \geq 0, \quad t = 0, 1, \ldots$$

- Dynamic programming: Bellman equation

$$V(W_t) = \max_{c_t \in [0, W_t]} u(c_t) + \beta V(W_{t+1}), \quad \text{s.t. } W_{t+1} = W_t - c_t \tag{2}$$

  - State variable: W_t
  - Control variable: c_t

- Rewrite the Bellman equation:

$$V(W_t) = \max_{W_{t+1} \in [0, W_t]} u(W_t - W_{t+1}) + \beta V(W_{t+1}) \tag{3}$$

Value Function Iteration

- Grid $[0, W_0]$ as $w_j = \frac{j}{J} W_0$ for $j = 1, \ldots, J$
- Initial guess $v_j^{(0)} = u\left(w_j\right)$
- Update $v_j^{(1)} = \max_{k=1,\ldots,j} u\left(w_j - w_k\right) + \beta v_k^{(0)}$
- Iterate until $v_j^{(t)} = v_j^{(t-1)}$ for all $j = 1, \ldots, J$

Policy Function Iteration

- Policy function: $c\left(W_t\right) \equiv \arg\max_{c_t \in [0, W_t]} u\left(c_t\right) + \beta V\left(W_t - c_t\left(W_t\right)\right)$
- Equivalently: $g\left(W_t\right) \equiv \arg\max_{W_{t+1} \in [0, W_t]} u\left(W_t - W_{t+1}\right) + \beta V\left(W_{t+1}\right)$
- Policy function iteration
  - Grid $[0, W_0]$ as $w_j = \frac{j}{J} W_0$ for $j = 1, \ldots, J$
  - Initial guess $g_j^{(0)} = w_k$, $k = 1, \ldots, j-1$
  - Solve v_j by iterating $v_j^{(t)} = u\left(w_j - w_k\right) + \beta v_k^{(t-1)}$ (inner loop)
  - Update $g_j^{(1)} = w_k$, where $k = \arg\max_{s=1,\ldots,j-1} u\left(w_j - w_s\right) + \beta v_s$
  - Iterate until $g_j^{(t)} = g_j^{(t-1)}$ for all $j = 1, \ldots, J$

Stochastic Cake-Eating Problem

- Random utility shifter: $z_t \in \mathcal{Z} \equiv \{z_1, \ldots, z_S\}$
- Bellman equation:

$$V(W_t, z_t) = \max_{W_{t+1} \in [0, W_t]} z_t u(W_t - W_{t+1}) + \beta E_t V(W_{t+1}, z_{t+1}) \tag{4}$$

- Example:
  - $\mathcal{Z} \equiv \{z_1, z_2\}$ with $z_2 > z_1 > 0$
  - Markov shifter with a transition matrix: $\Pi = \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix}$
  - Bellman equation:

$$V(W, z_s) = \max_{W' \in [0, W]} z_s u\left(W - W'\right) + \beta \left[\pi_{ss} V\left(W', z_s\right) + \pi_{s,-s} V\left(W', z_{-s}\right)\right] \tag{5}$$

Value Function Iteration

- Grid $[0, W_0]$ as $w_j = \frac{j}{J} W_0$ for $j = 1, \dots, J$

- Initial guess $v_{j,s}^{(0)} = z_s u\left(w_j\right)$

- Update by:

$$\begin{aligned} v_{j,1}^{(1)} &= \max_{k=1,\dots,j} z_1 u\left(w_j - w_k\right) + \beta \left[\pi_{11} v_{k,1}^{(0)} + \pi_{12} v_{k,2}^{(0)}\right] \\ v_{j,2}^{(1)} &= \max_{k=1,\dots,j} z_2 u\left(w_j - w_k\right) + \beta \left[\pi_{21} v_{k,1}^{(0)} + \pi_{22} v_{k,2}^{(0)}\right] \end{aligned} \tag{6}$$

- Iterate until $v_j^{(t)} = v_j^{(t-1)}$ for all $j = 1, \dots, J$

Dynamic Programming: General Form

- Maliar et al. (2021): Optimization problem
  - Exogenous state $m_{t+1} \in \mathbb{R}^{n_m}$ follows a Markov process driven by an i.i.d. innovation process $\epsilon_t \in \mathbb{R}^{n_m}$ with a transition function M: $m_{t+1} = M(m_t, \epsilon_t)$
  - Endogenous state $s_{t+1} \in \mathbb{R}^{n_s}$ is driven by the exogenous state m_t and controlled by a choice $x_t \in \mathbb{R}^{n_x}$ according to a transition function S: $s_{t+1} = S(m_t, s_t, x_t, m_{t+1})$
  - The choice x_t satisfies the constraint: $x_t \in X(m_t, s_t)$
  - The state (m_t, s_t) and choice x_t determine the period reward $r(m_t, s_t, x_t)$
  - The agent maximizes discounted lifetime reward: $\max_{\{x_t, s_{t+1}\}_{t=0}^{\infty}} E_0\left[\sum_{t=0}^{\infty} \beta^t r(m_t, s_t, x_t)\right]$
- Bellman equation: Value function $V : \mathbb{R}^{n_m} \times \mathbb{R}^{n_s} \to \mathbb{R}$

$$V(m, s) = \max_{x \in X(m,s), s'=S(m,s,x,m'), m'=M(m,\epsilon)} \max \left\{ r(m, s, x) + \beta E_\epsilon \left[V(m', s') \right] \right\} \tag{7}$$

Dynamic Programming: Curse of Dimensionality

- When the number of state variables (exogenous+endogenous) increases, the dimensionality of value function increases exponentially
- Infeasible to assign grids in high-dimensional state space
- Approximate $V(.,.)$ by polynomials:
  - cannot handle discrete states
  - the number of parameters increase exponentially
- Deep learning:
  - multi-layer neural networks to approximate high-dimensional functions
  - Maliar et al. (2021) Deep learning for solving dynamic economic models