Deterministic Dynamic Programming IV: Practical and Computational Aspects

Timothy Kam

Outline

1. Algorithm and Implementation Issues
2. Implementable Algorithm(s)

Algorithm and Implementation I

Previously ...

- We had taken the description of the optimal solution of the RCK problem as far as possible.
- In this example, we were able to describe the properties of v and π with features that increase with additional regularity assumptions on primitives, U and f.
- To actually *solve* for the optimal outcomes, we need to resort to numerical approximation and computation.
- First we need to set up a strategy for approximation and computation — i.e. develop an algorithm.

Algorithm and Implementation II

Let $\epsilon > 0$. Algorithm:

1. Pick some initial guess for $v_n : X \to \mathbb{R}$, where, $n = 0$.
2. Solve and evaluate the Bellman operator:

$$v_{n+1}(k) = \max_{k' \in \Gamma(k)} \{U[f(k) - k'] + \beta v_n(k')\},$$

   for every $k \in X$. Also store:

$$\pi_n(k) = \text{arg} \max_{k' \in \Gamma(k)} \{U[f(k) - k'] + \beta v_n(k')\}.$$

3. Calculate distance between consecutive updates: $d(v_{n+1}, v_n)$.
4. While $d(v_{n+1}, v_n) \geq \epsilon$, repeat Steps 2-4.

Algorithm and Implementation III

Implementation issues:

- How to represent state space $X = [\underline{k}, \overline{k}]$ (infinite set) on computer (finite storage)?
- How to represent v_n, $n = 0, 1, ...$, each a function, an element of an infinite dimensional space?
- How to compute the "$\max$" operator in Step 2?
- How to represent and store π_n, also a function?
- What is the appropriate metric d?

Implementable Algorithm(s) I

Let $\epsilon > 0$. Algorithm:

1. Discretized state space $\hat{X} = [\underline{k} < \cdots < \overline{k}]$ on computer.
2. Pick some initial approximate guess for $\hat{v}_n : \hat{X} \to \mathbb{R}$, where, $n = 0$.
3. Solve and evaluate the Bellman operator (approximate maximization):

   $$\hat{v}_{n+1}(k) = \hat{T}(\hat{v}_n)(k) = \max_{k' \in \Gamma(k)} \{U[f(k) - k'] + \beta \hat{v}_n(k')\},$$

   for every $k \in \hat{X}$. Also store:

   $$\hat{\pi}_n(k) = \text{arg} \max_{k' \in \Gamma(k)} \{U[f(k) - k'] + \beta \hat{v}_n(k')\}.$$

4. Calculate distance between consecutive updates: $d(\hat{v}_{n+1}, \hat{v}_n)$.
5. While $d(\hat{v}_{n+1}, \hat{v}_n) \geq \epsilon$, repeat Steps 2-4.

Implementable Algorithm(s) II

$v = T(v)$ is true, but unknown and nonanalytic value function.

Two other sources of errors, on top of $\epsilon > 0$:

- Step 2: Given $\hat{X}$, error from function approximation, $d(\hat{v}, v)$; and
- Step 3: (Aggregate) error arising from approximate maximization scheme: $d(\hat{T}(v), T(v))$.

Implementable Algorithm(s) III

We will consider two ways to tackle Steps 2 and 3:

- Method A:
  - Step 2A: Represent as finite $\hat{v}_n = [\hat{v}_n(\underline{k}) < \cdots < \hat{v}_n(\overline{k})]$, each $n = 0, 1, ...$
  - Step 3A: Given Step 2A, and $\hat{X}$ a finite set, $\max$ operator in $\hat{T}$ is just a "Table-lookup" maximization problem.

Implementable Algorithm(s) IV

- Method B:
  - Step 2B: Represent infinite $\hat{v}_n$ with known families of analytic functions. But store finite list $[\hat{v}_n(\underline{k}) < \cdots < \hat{v}_n(\overline{k})]$, each $n = 0, 1, ...$
  - Step 3B: Given Step 2A, and $\hat{X}$ a finite set, we can always interpolate for points not contained in $[\hat{v}_n(\underline{k}) < \cdots < \hat{v}_n(\overline{k})]$. Apply gradient/non-gradient based optimization algorithms to evaluate $\max$ operator in $\hat{T}$.

Implementable Algorithm(s) V

- Method A is known as the discretization method. Requires a lot of points in $\hat{X}$ to get an accurate approximate solution. Costly if X is multidimensional state space!
- Method B:
  - often implemented using projection methods on bounded and continuous function spaces.
  - Optimization doable in a continuous way since the (approximate) objective function is now continuous.
  - Requires less points in $\hat{X}$ to get accurate approximate solutions.
  - In practice, curse of dimensionality still a problem.