Continuous-State Dynamic Programming[1]

Kenneth L. Judd, Hoover Institution

January 31, 2014

[1] Joint work with Yongyang Cai.

Continuous Methods for Continuous-State Problems

- Basic Bellman equation:

$$V(x) = \max_{u \in D(x)} \pi(u, x) + \beta \, E\{V(x^+)|x, u)\} \equiv (TV)(x).$$

- Discretization essentially approximates V with a step function
  - Approximation theory provides better methods to approximate continuous functions.
- General Task
  - Find good approximation for V
  - Identify parameters

General Parametric Approach: Approximating $V(x)$

- Choose a finite-dimensional parameterization

$$V(x) \doteq \hat{V}(x; a), \ a \in R^m$$

and a finite number of states

$$X = \{x_1, x_2, \cdots, x_n\}$$

- polynomials with coefficients a and collocation points X
  - splines with coefficients a with uniform nodes X
  - rational function with parameters a and nodes X
  - neural network
  - specially designed functional forms
- Objective: find coefficients $a \in R^m$ such that $\hat{V}(x; a)$ "approximately" satisfies the Bellman equation.

General Parametric Approach: Approximating T

For each x_j, $(TV)(x_j)$ is defined by

$$v_j = (TV)(x_j) = \max_{u \in D(x_j)} \pi(u, x_j) + \beta \int \hat{V}(x^+; a) dF(x^+ | x_j, u)$$

and is approximated by $\hat{T}$

$$v_j = (\hat{T} V)(x_j) \doteq (TV)(x_j)$$

- Integration step: for ω_j and x_j for some numerical quadrature formula

$$\begin{aligned} E\{V(x^+; a) | x_j, u)\} &= \int \hat{V}(x^+; a) dF(x^+ | x_j, u) \\ &= \int \hat{V}(g(x_j, u, \varepsilon); a) dF(\varepsilon) \\ &\doteq \sum_{\ell} \omega_\ell \hat{V}(g(x_j, u, \varepsilon_\ell); a) \end{aligned}$$

- Maximization step: for $x_i \in X$, evaluate

$$v_i = (T\hat{V})(x_i)$$

  - Hot starts
  - Concave stopping rules
- Fitting step:
  - Data: $(v_i, x_i),\ i = 1, \cdots, n$
    - Objective: find an $a \in R^m$ such that $\hat{V}(x; a)$ best fits the data
    - Methods: determined by $\hat{V}(x; a)$

Approximating T with Hermite Data

Conventional methods just generate data on $V(x_j)$:

$$v_j = \max_{u \in D(x_j)} \pi(u, x_j) + \beta \int \hat{V}(x^+; a) dF(x^+ | x_j, u)$$

- Envelope theorem:
  - If solution u is interior,

$$v_j' = \pi_x(u, x_j) + \beta \int \hat{V}(x^+; a) dF_x(x^+ | x_j, u)$$

  - If solution u is on boundary

$$v_j' = \mu + \pi_x(u, x_j) + \beta \int \hat{V}(x^+; a) dF_x(x^+ | x_j, u)$$

    where μ is a Kuhn-Tucker multiplier
- Since computing v_j' is cheap, we should include it in data:
  - Data: (v_i, v_i', x_i), $i = 1, \cdots, n$
  - Objective: find an $a \in R^m$ such that $\hat{V}(x; a)$ best fits Hermite data
  - Methods: determined by $\hat{V}(x; a)$

General Parametric Approach: Value Function Iteration

$$
\begin{aligned}
\text{guess } a \quad &\longrightarrow \quad \hat{V}(x;a) \\
&\longrightarrow \quad (v_i, x_i), \;\; i = 1, \cdots, n \\
&\longrightarrow \quad \text{new } a
\end{aligned}
$$

- Comparison with discretization
  - This procedure examines only a finite number of states, x_i:
    - But does *not* assume that the state is always in this finite set.
    - Choices for the x_i are guided by approximation methods
  - Procedure examines only a finite number of ε values for the stochastic shocks
    - But does *not* assume that they are the only ones realized
    - Choices for the ε_i come from quadrature methods
- Synergies
  - Smooth interpolation allows us to use Newton's method for max step.
  - Smooth interpolation allows more efficient quadrature in (12.7.5).
  - Efficient quadrature reduces cost of computing objective in max problem

- Finite-horizon problems
  - Must use value function iteration since $V(x,t)$ depends on time t.
  - Begin with terminal value function, $V(x,T)$
  - Compute approximations for each $V(x,t)$, $t = T-1, T-2$, etc.

Algorithm 12.5: Parametric Dynamic Programming
with Value Function Iteration

Objective:	Solve the Bellman equation, (12.7.1).
Step 0:	Choose functional form for $\hat{V}(x; a)$, and choose the approximation grid, $X = \{x_1, ..., x_n\}$. Make initial guess $\hat{V}(x; a^0)$, and choose stopping criterion $\epsilon > 0$.
Step 1:	Maximization step: Compute $v_j = (T\hat{V}(\cdot; a^i))(x_j)$ for all $x_j \in X$.
Step 2:	Fitting step: Using the appropriate approximation method, compute the $a^{i+1} \in R^m$ such that $\hat{V}(x; a^{i+1})$ approximates the (v_i, x_i) data.
Step 3:	If $\parallel \hat{V}(x; a^i) - \hat{V}(x; a^{i+1}) \parallel < \epsilon$, STOP; else go to step 1.

Convergence

- T is a contraction mapping
- $\hat{T}$ may be neither monotonic nor a contraction
- Shape problems
  - Standard approximation methods do not preserve shape
  - Shape problems may become worse with value function iteration

Solution to shape problems

- Use shape-preserving approximations
  - Piecewise linear preserves shape in one dimension.
  - Multilinear approximation does not preserve shape
  - Shape preserving splines are available for dimensions one and two.
- Impose shape restrictions in fitting
  - Use least squares, not interpolation
  - Add shape constraints to least squares problem
    - Demand correct slopes at some points
    - Demand correct curvature at some points.
  - These methods work well in one dimension, but slow algorithm down considerably for higher dimensions
- Open research question: What is the best combination of smooth functional form and fitting procedure that preserves shape?

Summary

- Discretization methods
- Easy to implement
    - Numerically stable
    - Amenable to many accelerations
    - Poor approximation to continuous problems
- Continuous approximation methods
    - Can exploit smoothness in problems
    - Must work to avoid numerical instabilities
    - Acceleration is less possible