DYNAMIC PROGRAMMING: AN OVERVIEW

Kenneth L. Judd
Hoover Institution

May 10, 2010

DYNAMIC PROGRAMMING:

DEFINITIONS AND EXAMPLES

Discrete-Time Dynamic Programming

- Objective:

$$E\left\{\sum_{t=1}^{T}\pi(x_t,\,u_t,\,t)+W(x_{T+1})\right\},$$

  - X: set of states
  - $\mathcal{D}$: the set of controls
  - $\pi(x,\,u,\,t)$ payoffs in period t, for $x\in X$ at the beginning of period t, and control $u\in\mathcal{D}$ is applied in period t.
  - $D(x,t)\subseteq\mathcal{D}$: controls which are feasible in state x at time t.
  - $F(A;x,u,t)$: probability that $x_{t+1}\in A\subset X$ conditional on time t control and state

- Value function

$$V(x,t)\equiv\sup_{\mathcal{U}(x,t)}E\left\{\sum_{s=t}^{T}\pi(x_s,\,u_s,\,s)+W(x_{T+1})|x_t=x\right\}.$$

- Bellman equation

$$V(x,t)=\sup_{u\in D(x,t)}\pi(x,\,u,\,t)+E\left\{V(x_{t+1},\,t+1)|x_t=x,u_t=u\right\}$$

- Existence: boundedness of π is sufficient
- Notational convenience: drop $u\in D(x,t)$ constraints and encode them in payoff function.

Autonomous, Infinite-Horizon Problem:

- Objective:

$$\max_{u_t} E\left\{\sum_{t=1}^{\infty} \beta^t \pi(x_t,\, u_t)\right\}$$

  - X: set of states
  - $\mathcal{D}$: the set of controls
  - $D(x) \subseteq \mathcal{D}$: controls which are feasible in state x.
  - $\pi(x,\, u)$ payoff in period t if $x \in X$ at the beginning of period t, and control $u \in \mathcal{D}$ is applied in period t.
  - $F(A; x, u)$: probability that $x^+ \in A \subset X$ conditional on current control u and current state x.

- Value function definition: if $\mathcal{U}(x)$ is set of all feasible strategies starting at x.

$$V(x) \equiv \sup_{\mathcal{U}(x)} E\left\{\sum_{t=0}^{\infty} \beta^t \pi(x_t,\, u_t) \,\middle|\, x_0 = x\right\},$$

- Bellman equation for $V(x)$

$$V(x) = \sup_{u} \ \pi(x,\, u) + \beta\, E\ \left\{V(x^{+})|x, u\right\} \equiv (TV)(x),$$

- Optimal policy function, $U(x)$, if it exists, is defined by

$$U(x) \ \in \ \arg\max_{u} \ \pi(x,\, u) + \beta\, E \left\{V(x^{+})|x, u\right\}$$

- Standard existence theorem:

Theorem 1 *If X is compact, $\beta < 1$, and π is bounded above and below, then the map*

$$TV = \sup_{u} \ \pi(x, u) + \beta E \left\{V(x^{+}) \mid x, u\right\}$$

is monotone in V, is a contraction mapping with modulus β in the space of bounded functions, and has a unique fixed point.

Applications

- Economics
  - Life-cycle decisions on labor, consumption, education
  - Business investment
  - Portfolio problems
  - Economic policy
- Operations Research
  - Scheduling, queueing
  - Inventory management
- Climate change
  - Business response to climate policies
  - Optimal policy response to climate change

Simple Deterministic Growth Example

- Problem:

$$
\begin{aligned}
V(k_0) = \max_{c_t} \textstyle\sum_{t=0}^{\infty} \beta^t u(c_t), \\
k_{t+1} = F(k_t) - c_t \\
k_0 \text{ given}
\end{aligned}
$$

- Bellman equation

$$V(k) = \max_{c} \; u(c) + \beta V(F(k) - c).$$

- First-order condition

$$0 = u'(c) - \beta V'(F(k) - c)$$

- Solution is a policy function $C(k)$ and a value function $V(k)$ satisfying

$$V(k) = u(C(k)) + \beta V(F(k) - C(k)) \tag{1}$$

$$0 = u'(C(k)) - \beta V'(F(k) - C(k)) \tag{2}$$

  – Eqn.(2) defines value function for *any* policy function

  – Eqn (1) defines policy function in terms of the value function.

General Stochastic Accumulation

- Multidimensional Problem:

$$
\begin{aligned}
V(k,\theta) = \max_{c_t,\ell_t}\ E & \left\{ \sum_{t=0}^{\infty} \beta^t\ u(c_t,\ell_t,\theta_t) \right\} \\
k_{t+1} &= F(k_t,\ell_t,\theta_t) - c_t \\
\theta_{t+1} &= g(\theta_t,\varepsilon_t) \\
k_0 &= k,\ \ \theta_0 = \theta.
\end{aligned}
$$

- State variables:
  - k: productive capital stocks, endogenous (could include lags, human capital, etc.)
  - θ: productivity and taste states, exogenous
- Intratemporal Choices
  - Consumption and leisure here
  - Could be allocation of time to education, and other activities
- The dynamic programming formulation is

$$V(k,\theta) = \max_{c,\ell}\ u(c,\ell) + \beta E\{V(F(k,\ell,\theta) - c, \theta^+)|\theta\}, \tag{12.1.21}$$

where θ^+ is next period's θ realization

Dynamic Asset Allocation Problem

- Initial wealth W_0; wealth at beginning of time t is a random variable W_t; all assets at time $t = T$, W_T, liquidated and valued at $u\left(W_T\right)$.
- B_t is bond investment at end of time t with safe return $(1 + r)$
- $S_{i,t}$ is investment in stock i with random return $R_{i,t}$, for $1 \leq i \leq n$
- Budget constraint at time t

$$W_t = B_t + \sum_{i=1}^{n} S_{it}$$

- Wealth at time $t + 1$

$$W_{t+1} = (1 + r)B_t + \sum_{i=1}^{n} R_{it} S_{it}$$

- Objective:

$$\max E\left\{u(W_T)\right\}$$

DYNAMIC PROGRAMMING:

STANDARD METHODS

Discrete State Space Problems

- Discretize the state
  - Approximates continuous states
  - Use value function iteration
- Performance;
  - Algorithm always works for finite-horizon problems but .... *slowly*
  - Algorithm only works for infinite-horizon problems if you are very patient
  - Discretize states is impractical for multidimensional problems
- Bellman equation: time t value function is

$$V_i^t = \max_u \left[\pi(x_i, u, t) + \beta \sum_{j=1}^{n} q_{ij}^t(u)\, V_j^{t+1}\right], \ \ i = 1, \cdots, n$$

- Bellman equation can be directly implemented.
  - Called *value function iteration*
  - It is only choice for finite-horizon problems because each period has a different value function.

Policy Iteration (a.k.a. Howard improvement)

- Value function iteration is a slow process
  - The only possible method for finite-horizon problems
  - Slow for infinite-horizon problems since error is

$$\left\| V^k - V^* \right\| \le \frac{1}{1-\beta} \left\| V^{k+1} - V^k \right\|$$

  - Linear convergence at rate β; convergence very slow if β is close to 1.
- Policy iteration is faster

Piecewise Linear Interpolation for Continuous-State Problems

- Bellman equation:

$$V(x) = \max_{u \in D(x)} \pi(u,x) + \beta\, E\{V(x^+)|x,u\} \equiv (TV)(x). \tag{12.7.1}$$

  – Discretization essentially approximates V with a step function

  – Piecewise linear approximation is more natural if true V is continuous; is method taught in kindergarten.

- Performance;

  – Algorithm always works for finite-horizon problems, faster than discetizing the state space, but still....... *slow!*

  – Piecewise linear interpolation is hard for two- and three-dimensional problems; messy and intractable in high dimensions

Linear Programming Approach

- If both states and actions are finite, we can reformulate dynamic programming as a linear programming problem.
- Bellman equatoin is equivalent to the linear program

$$\begin{array}{ll} \min_{V_i} \sum_{i=1}^{n} V_i & \\ s.t. \quad V_i \geq \pi(x_i, u) + \beta \sum_{j=1}^{n} q_{ij}(u) V_j, \ \forall i, u \in \mathcal{D}, \end{array} \tag{12.4.10}$$

- Computational considerations
  - No iteration – nice!
- The LP probem may be huge, but perhaps tractable.

DYNAMIC PROGRAMMING:

COMPUTATIONAL ISSUES AND SOLUTIONS

Mathematical Formulation of DP

- Problem: Given current situation x (the state), what actions a do I take today to maximize payoff?
  - Portfolio problems: stocks versus bonds
  - Life-cycle problems
  - Inventory management
- Canonical mathematical problem: find function $V : \mathbb{R}^k \times \mathbb{N}^m \to R$ expressing expected discounted payoff and solves the fixed-point problem in a Banach space of functions V

$$V(x) = \max_{u \in D(x)} \pi(u, x) + \beta \int V(f\,(x, u, z))d\mu\,(z) \equiv (TV)(x)$$

  - x: state of system; typically x in a bounded subset of $\mathbb{R}^k \times \mathbb{N}^m$
  - $u \in D(x)$: feasible choices when state is x.
  - z: random disturbances
  - f: tomorrow's state given today's state, today's choice, and random shock.
  - $\beta < 1$: discount factor
- V encodes *all* information about the solution

General Parametric Approach: Approximating T

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

$$v_j = (T\hat{V})(x_j) = \max_{u \in D(x_j)} \pi(u, x_j) + \beta \int \hat{V}(x^+; a) dF(x^+|x_j, u) \tag{12.7.5}$$

- In practice, we compute the approximation $\hat{T}$

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

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

$$E\{\hat{V}(x^+; a)|x_j, u)\} = \int \hat{V}(x^+; a) dF(x^+|x_j, u)$$

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

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

  – 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 $\hat{V}(x; a)$

General Parametric Approach: Value Function Iteration

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

- Convergence
  - Useful theory fact: T is a contraction mapping
  - Computational challenge: constructing $\hat{T}$ so that it is monotonic and/or a contraction mapping
    - Not easy
    - Is it necessary?

- Computational Problem I: Approximating $V(x)$
  - Choose a finite-dimensional parameterization:

$$V(x) \doteq \hat{V}(x; a), \;\; a \in R^m \tag{3}$$

  - Choose a finite number of states:

$$X = \{x_1, x_2, \cdots, x_n\}, \tag{4}$$

  - Objective: find coefficients $a \in R^m$ such that $\hat{V}(x; a)$ "approximately" satisfies the Bellman equation for $x \in X$.
  - Standard methods
    * discrete states, step functions,
    * piecewise linear functions
    * ordinary polynomials and splines
  - Can we find better? YES!

- Computational Problem II: Integration step
  - Use some quadrature rule Q to approximate

$$\int V(f(x,u,z))d\mu(z) \cong Q\left(V(f(x,u,z)),\mu(z)\right)$$

  - Standard methods
    * product rules
    * Monte Carlo
  - Can we do better? YES!

- Computational Problem III: Maximization step
- For each x_i on some grid, numerically solve

$$v_i = \max_{u \in D(x)} \pi(u, x_i) + \beta Q\left(V\left(f\left(x_i, u, z\right)\right), \mu\left(z\right)\right) \tag{5}$$

  - Standard methods
    * bisection, Nelder-Mead
    * fmincon
    * use a single processor
  - Can we do better? YES!

- Computational Problem IV: Fitting step
  - Construct data to find an $a \in R^m$ such that $\hat{V}(x; a)$ fits the data
  - Standard methods
    - Piecewise linear interpolation
    - Multilinear interpolation
    - Polynomials and splines: often unstable!
  - Can we do better? YES!

- How do we find better methods?
  - Learn and use methods from approximation, quadrature, optimization, and computer science literatures
  - Construct our own methods!