Dynamic Programming - I

Judd Chapter 12

David Childers (thanks to Y. Kryukov, K. Judd, and U. Doraszelski)

CMU, Tepper School of Business

April 10 - 12, 2023

Agenda

- Dynamic problems features:
  - Time: Discrete ($\sum_{t=0}^{T} \beta^t$) or continuous ($\int_0^T e^{-\rho t}$)
  - Time horizon: finite ($T \in \mathbb{N}$) or infinite ($T = \infty$)
  - State: discrete or continuous
  - State transition: deterministic or stochastic
- DP gives us Bellman eq's for any combination of these

1. now: **Theory: Finite and infinite horizon, with cont. state**
2. **Computation with discrete states**
   1. **Value & Policy iteration**
   2. **Stochastic vs. deterministic transitions**
3. Computation with continuous state
   1. Value function iteration
   2. Projection
4. Continuous time

Continuous vs. discrete time

- Assume continuous state, deterministic transitions
- A problem in continuous time:

$$\begin{aligned} \max \quad & \int_0^\infty e^{-\rho t}\pi(x\,(t)\,, u\,(t))dt \\ \text{subject to:} \quad & \dot{x} \equiv x'(t) = f(x(t), u(t)) \\ & x(0) = \bar{x}_0 \end{aligned}$$

  - $u =$ control / policy, $x =$ state
- A problem in discrete time (today):

$$\begin{aligned} \max_c \quad & \sum_{t=0}^\infty \beta^t \pi(x_t, u_t) \\ \text{subject to:} \quad & f\{x_t, x_{t+1}, u_t\} = 0 \\ & x_0 = \bar{x}_0 \end{aligned}$$

  - Different problems with different solutions!

Deriving Bellman's eq.: finite horizon

$$\begin{array}{rl} \max_{u,x\in\mathbb{R}^T} & \sum_{t=0}^{T}\beta^t\pi(x_t,u_t)+\beta^{T+1}W(x_{T+1}) \\ \text{subject to:} & f\{x_t,x_{t+1},u_t\}=0\ \forall t, \qquad x_0=\bar{x}_0 \end{array}$$

- Can re-write objective as

$$\max_{\substack{u_0,..,u_{T-1}\\x_1,..,x_T}}\left\{\sum_{t=0}^{T-1}\beta^t\pi(x_t,u_t)+\beta^T\max_{u_T,x_{T+1}}\left[\begin{array}{c}\pi(x_T,u_T)\\+\beta W(x_{T+1})\end{array}\right]\right\}$$

- Label the last term as the *Value function* $V(x_T,T)$:

$$\begin{aligned} V(x_T,T) \;&:=\; \max_{u_T,x_{T+1}}\pi(x_T,u_T)+\beta W(x_{T+1}) \\ \text{s.t.:} \quad & f\{x_T,x_{T+1},u_T\}=0, \end{aligned}$$

x_T is a parameter here (just like x_0 in the full model)

Next transformation

Using $V(\cdot, T)$, rewrite the objective further:

$$\max_{u_0,..,u_{T-2}} \left\{ \sum_{t=0}^{T-2} \beta^t \pi(x_t, u_t) + \beta^{T-1} \max_{u_{T-1}, x_T} \left[\begin{array}{c} \pi(x_{T-1}, u_{T-1}) \\ +\beta V(x_T, T) \end{array} \right] \right\}$$

and again define

$$\begin{aligned} V(x_{T-1}, T-1) \quad &: \quad = \max_{u_{T-1}, x_T} \pi(x_{T-1}, u_{T-1}) + \beta V(x_T, T) \\ \text{s.t.} \quad &: \quad f\{x_{T-1}, x_T, u_{T-1}\} = 0 \end{aligned}$$

State variable x_{T-1} is a *sufficient statistic*
for whatever happened before period $T-1$.

Bellman's principle – finite horizon

Original problem:

$$\max_{u,x\in\mathbb{R}^T} \sum_{t=0}^{T} \beta^t \pi(x_t, u_t) + \beta^{T+1} W(x_{T+1})$$

is equivalent to *Bellman equations* for $t = 1, ..., T$:

$$V(x_t, t) = \max_{u_t, x_{t+1}} \pi(x_t, u_t) + \beta V(x_{t+1}, t+1) \qquad (*)$$
$$\text{s.t. } f\{x_t, x_{t+1}, u_t\} = 0,$$

- $V(\cdot,\cdot) =$ (max of) net present value of remaining payoffs:

$$V(x_s, s) = \max_{u,x} \sum_{t=s}^{T} \beta^{t-s} \pi(x_t, u_t) + \beta^{T+1-s} W(x_{T+1})$$

Bellman principle – interpretation

- Instead of solving for $(u_0, x_1, u_1, ..., u_T, x_{T+1})$,
- Solve a sequence of problems for $t = T, T-1, ..., 1$
  - Each problem has an arbitrary parameter x_t
  - Solution (*Policy function*): $U_t(x_t, t)$
  - Maximized objective (*Value function*): $V(x_t, t)$
- Can compute time path for the original problem:
  - x_0 known, $u_0 = U(x_0, 0)$,
  - x_1 solves $f\{x_0, x_1, u_0\} = 0$, $u_1 = U(x_1, 1)$, and so on
- Or we can analyze policy and value functions

Infinite horizon model

- Explicit model:

$$\begin{aligned} V(x_0, 0) &= \max_{u_0, x_1, u_1, \ldots} \sum_{t=0}^{\infty} \beta^t \pi(x_t, u_t) \\ \text{s.t.} &: f\{x_t, x_{t+1}, u_t\} = 0 \ \forall t, x_0 = \bar{x}_0 \end{aligned}$$

- Isolate the first term in the objective:

$$\max_{u_0, x_1} \left\{ \pi(x_0, u_0) + \beta \max_{u_1, x_2, u_2, \ldots} \sum_{t=1}^{\infty} \beta^{t-1} \pi(x_t, u_t) \right\}$$

- Define the value function:

$$V(x_1, 1) = \max_{u_1, x_2, u_2, \ldots} \sum_{t=1}^{\infty} \beta^{t-1} \pi(x_t, u_t)$$

- Compare $V(x_1, 1)$ to explicit model (Hint: define $\tau = t - 1$)

Bellman principle – infinite horizon

- Time does not affect Value function, only the state does
- **Bellman** equations for infinite time horizon:

$$\begin{aligned} V(x) &= \max_{u,x^+\in\mathbb{R}} \pi(x,u) + \beta V(x^+) \qquad \textbf{(**)} \\ \text{s.t.} &: \quad f\{x,x^+,u\} = 0, \end{aligned}$$

  $x =$ state in current period, $x^+ =$ in the next period
- Unlike finite time, there is only one value function $V(x)$
- **Policy function** (decision rule):

$$U(x) = \arg\max \ldots$$

Richard Bellman, 1962

Practical solution to Bellman

- Policy function is defined by:

$$U(x) = \arg\max_u \pi(x,u) + \beta V\left(x^+(x,u)\right)$$

  - Note; $x^+(u,x)$ solves the constraint for x^+

- Take FOC (w.r.t. u):

$$\pi_u(x,U(x)) + \beta V'(x^+(x,U(x)))x_u^+(x,U(x)) = 0 \qquad (***)$$

- If $U(x)$ solves (***), we can replace (**) with:

$$V(x) = \pi(x,U(x)) + \beta V\left(x^+(x,U(x))\right) \qquad (****)$$

- (***)-(****) are a system of two functional equations, which we solve for two unknown functions: $V(x)$ and $U(x)$

More on Bellman's solution

- $V'(x^+(x, U(x)))$ in (***) nests unknown functions
  - To avoid it, use Envelope theorem for (**):

$$\begin{aligned} V'(x) &= \pi_x(x, U(x)) + \beta V'(x^+(x, U(x)))x_x^+(x, U(x)) \\ \Rightarrow \beta V'(x^+(x, U(x))) &= \left[V'(x) - \pi_x(x, U(x))\right] / x_x^+(x, U(x)) \end{aligned}$$

  - In many models, $x_x^+ = 1$
- Explicit problem solves for infinite sequence $u_1, u_0, \ldots$; DP solves for functions $V(x)$ and $U(x)$
- If we want to construct timepath of u_t's:
  1. $u_0 = U(x_0)$, x_1 solves $f\{x_0, x_1, u_0\} = 0$,
  2. $u_1 = U(x_1)$, and so on

Dynamic Programming theory

- Can often work with bounded support of $V(\cdot)$:
  - Neigborhood of the steady state x^*
  - Time path: $x \in [x_0, x^*]$
  - or just $x \in \left[-10^6, +10^6\right]$
- Functions of such x are a vector space;
  defining a norm makes it a metric space.
- Maximization problem in (**) is an operator on that space:

$$(T(V))(x) \equiv \max_{u, x^+ \in \mathbb{R}} \pi(x,u) + \beta V(x^+)$$

- Bellman equation is thus a fixed-point problem:

$$V = T(V)$$

- If T is a contraction, a unique solution exists

DP Theory: Contraction mapping

Contraction mapping: T such that for some $\beta \in (0,1)$:

$$\|T(V) - T(W)\| \leq \beta \|V - W\|, \qquad \forall V, W$$

Blackwell's sufficient conditions for contraction:

1. $B(X)$ is a set of bounded functions $V : X \subseteq \mathbb{R}^n \to \mathbb{R}$
2. *Monotonicity*: $V, W \in B(X)$ such that $V(x) \leq W(x)\ \forall x \Rightarrow$
$$(T(V))(x) \leq (T(W))(x), \forall x \in X$$
3. *Discounting*: there exists $\beta \in (0,1)$ such that
$$(T(V+a))(x) \leq (T(V))(x) + \beta a$$
$\forall V \in B(X)$, $\forall a \geq 0$, $\forall x$

Bellman operator is a contraction if:

- $x \in X$ – compact set (e.g. neighborhood of the steady state)
- π is bounded & $\beta < 1$ (typically satisfied in Economics)

Extension A: stochastic transition

- Explicit model (infinite time):

$$\max_{u} \mathbf{E}_x \left[\sum_{t=0}^{\infty} \beta^t \pi(x_t, u_t) \right]$$

$$\text{where} \quad : \quad x_{t+1} \sim F\left(\cdot | x_t, u_t\right) \ \forall t, x_0 = \bar{x}_0$$

  - x_{t+1} is a Markov process conditioned on u_t

- Bellman equation:

$$V(x) = \max_{u, x^+ \in \mathbb{R}} \pi(x, u) + \beta \mathbf{E}_{x^+} \left[V(x^+) | x, u \right],$$

$$\mathbf{E}_{x^+} \left[V(x^+) | x, u \right] = \int_X V\left(x^+\right) dF\left(x^+ | x, u\right)$$

- FOC:

$$\pi_u(x, U(x)) + \beta \int_X V\left(x^+\right) dF_u\left(x^+ | x, U(x)\right) = 0$$

Extension B: stochastic payoff

- Explicit model (determinstic state transitions):

$$\begin{aligned} & \max_u \mathbf{E}_\varepsilon \left[\sum_{t=0}^{\infty} \beta^t \pi(x_t, u_t, \varepsilon_t) \right] \\ \text{s.t.} \quad : \quad & f\{x_t, x_{t+1}, u_t\} = 0 \ \forall t \\ \varepsilon_t \quad \sim \quad & G(\cdot), \text{ i.i.d.} \end{aligned}$$

  - Agent learns ε_t only in period t (but not before)
- Full-information Bellman eq.:

$$\begin{aligned} \tilde{V}(x, \varepsilon) &= \max_u \pi(x, u, \varepsilon) + \beta V\left(x^+(x, u)\right) \\ \tilde{U}(x, \varepsilon) &= \arg\max \ldots \end{aligned}$$

- Integrated Bellman:

$$V(x) = \mathbf{E}_\varepsilon \tilde{V}(x, \varepsilon)$$

  - "$U(x)$" is a distribution, driven by ε

Stochastic payoff example

- Stochastic payoff framework is often used to model discrete choice:
  - $u \in \{u_1, ..., u_m\}$, e.g. exit/enter, or work/school/unempl.
  - $\varepsilon_t = \{\varepsilon_{t1}, ..., \varepsilon_{tm}\}$, $\varepsilon_{tj} \sim$ i.i.d. Type I Extreme Value, multiplied by σ (the variance parameter)
  - Choice-specific shocks: $\pi(x, u_j, \varepsilon) = \pi_j(x) + \varepsilon_j$;
  - let $\delta_j = \pi_j(x) + \beta V\left(x^+\left(x, u_j\right)\right)$
- Then we have closed-form expressions:

$$V(x) = \sigma \log\left\{\sum_{j=1}^m \exp\left(\delta_j/\sigma\right)\right\}$$

$$\Pr\left\{U(x) = u_j\right\} = \frac{\exp\left(\delta_j/\sigma\right)}{\sum_{l=1}^m \exp\left(\delta_l/\sigma\right)}$$

Numeric solution: discrete states

- Discrete states are easier to understand
  - Continuous states (next class) might be faster to solve
- States $X = \{x_i\}_{i=1}^{n}$ - a vector of possible values
  - Value function becomes a vector: $V(\cdot) = \{V_i = V(x_i)\}_{i=1}^{n}$
- Stochastic transition – first-order Markov process:
  - Let current state be x_i and control be u
  - Then, $q_{ij}(u) =$ probability of next-period state being x_j
  - It is common to limit transitions to 2-3 neighboring states
- Deterministic transitions will be covered separately
  - $q_{ij}(u) \in \{0, 1\}$, so we cannot differentiate it
  - Policy u becomes discrete as well

Finite horizon

- Finite-horizon case $(t = 0, .., T)$:

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

  - with terminal condition $V_i^{T+1} \equiv W(x_i)$
- Recursive system of $n\,(T+1)$ equations in $n\,(T+1)$ unknowns:
  - Have $V_i^{T+1} \equiv W(x_i)$, maximization problem gives $\left\{V_i^T\right\}_{i=1}^n$
  - Continue using V_i^{t+1}'s to compute V_i^t's
  - Stop when $t = 0$ is reached

Discrete states + infinite time horizon

- No more time-dependence: $V_i = V(x_i)$, $i = 1, \ldots, n$.
- The Bellman equation is:

$$V_i = \max_u \pi(x_i, u) + \beta \sum_{j=1}^n q_{ij}(u) V_j. \tag{1}$$

$$U_i^* = \arg\max ...$$

  - Can have constraints on u (e.g. $0 \leq c \leq k + f(k)$)
- System of $2n$ nonlinear equations in $2n$ unknowns:

$$\begin{aligned} V_i &= \pi(x_i, U_i^*) + \beta \sum_{j=1}^n q_{ij}(U_i^*) V_j \\ 0 &= \pi_u(x_i, U_i^*) + \beta \sum_{j=1}^n q'_{ij}(U_i^*) V_j \end{aligned}$$

- Can solve using any method, but the most common way is fixed point iteration (next slide)

Value function iteration

- Define the operator T pointwise by
$$(TV)_i = \max_u \pi(x_i, u) + \beta \sum_{j=1}^{n} q_{ij}(u) V_j, \quad i = 1, \ldots, n.$$
    - Initialization: Choose initial guess V^0 and stopping criterion ϵ.
    - Step 1: Compute $V^{k+1} = TV^k$.
    - Step 2: If $||V^{k+1} - V^k|| < \epsilon$, stop; otherwise, go to step 1.
- The sequence $\{V^k\}_{k=0}^{\infty}$ converges linearly at rate β to V^* and:
$$||V^k - V^*|| \le \frac{||V^{k+1} - V^k||}{1-\beta}.$$
- To ensure $||V^{k+1} - V^*|| < \epsilon$, stop if $||V^{k+1} - V^k|| \le \epsilon(1-\beta)$.
- Maximization is the slowest part.
    - Good V^0: monotonicity, concavity as in the solution
    - Good guess for u_i: solution from previous iteration

Policy function iteration – idea

- Let $U = \{U_i \equiv U(x_i)\}_{i=1}^n$ be a guess of the policy function
- Value iteration follows chosen policy for one period
- Even if U is close to optimal at each iteration,
  it can take many iterations to compute the optimal V.

- Once U is computed, update of V is linear
- Define transition matrix $Q(U) : [Q(U)]_{ij} = q_{ij}(U_i)$,
  and payoff vector $\Pi(U) : [\Pi(U)]_i = \pi(x_i, U_i)$
- Following policy U indefinitely leads to value f-n V^U
  that solves a system of linear equations:

$$
\begin{aligned}
V^U &= \Pi(U) + \beta Q(U) V^U \\
V^U &= (I - \beta Q(U))^{-1} \Pi(U).
\end{aligned} \tag{2}
$$

Policy function iteration – implementation

- Initialization: Choose initial guess V^0 and stopping criterion ϵ. (Or: Choose U^0 instead of V^0 and go to step 2.)

1. Compute U^{k+1} by solving (1) with V^k in r.h.s.
2. Plug U^{k+1} into (2), solve for V^{k+1}.
3. If $||V^{k+1} - V^k|| < \epsilon$, stop; otherwise, go to step 1.

- Setting up (2) for Step 2 can be a lot of code
- (2) computes the value of following policy U^{k+1} indefinitely
- Modified policy iteration: follow U^{k+1} for m periods:
  - 2.a: Set $W^0 = V^k$.
  - 2.b: Compute $W^{j+1} = \pi\left(U^{k+1}\right) + \beta Q\left(U^{k+1}\right) W^j$ for $j = 0, \ldots, m-1$
  - 2.c: Set $V^{k+1} = W^m$.

Extension: deterministic state transition

- Formally, $q_{ij}(u) \in \{0,1\}$
  - Not differentiable $\Rightarrow$ no more FOC
  - Instead, we use deterministic l.o.m: $x^+ = f(x_i, u)$
- Practically, we have to have $x^+ \in X = \{x_i\}_{i=1}^n$, so u is limited to values that ensure this
  - If $x = x_i$, then for any $x^+ = x_j$, we have $u_{ij} = f^{-1}(x_i, x_j)$
  - This lets us define to $\pi_{ij} \equiv \pi(x_i, u_{ij})$
- Fixed point methods iterate on:

$$\begin{aligned} V_i^{k+1} &= \max_j \pi_{ij} + \beta V_j^k \\ U_i &= \arg\max ... \end{aligned}$$

  - Very simple to implement, but there is a cost

Problems with deterministic state transition

- Sawtooth policy function & waves in value function
  - Discrete u creates errors in V, which accumulate until a large correction
- Problems with convergence, or lack of solution altogether
  - You are playing a game against your future self, and restrict this game to pure strategies.
  - Use dampening to try get V to converge
- If you suspect sawtooth pattern or discretization effects:
  - Switch to continuous state (next topic)
  - Try changing number of states, see if shape of policy function changes: Finer grid should lead to smaller teeth.

Example - optimal growth

- Discrete time:

$$\begin{array}{rc} \max_{\{c_t\}} & \sum_{t=0}^{\infty} \beta^t u\left(c_t\right) \\ \text{subject to:} & k_{t+1} - k_t = f\left(k_t\right) - c_t \end{array}$$

- ... and state: $k \in K = \{k_1, k_2, \ldots, k_n\}$
- Constraint implies that: $c_t = k_t + f\left(k_t\right) - k_{t+1}$, so

$$\max_{\{k_t\}} \sum_{t=0}^{\infty} \beta^t u\left(k_t + f\left(k_t\right) - k_{t+1}\right)$$

  - I.e. we limit $c\left(k\right)$ to a discrete set of values
- Formulate Bellman equation:

$$V(k) = \max_{k^+ \in K} u(k + f(k) - k^+) + \beta V(k^+)$$

Example - value iteration

- Index states by $i = 1, ..., n$
- Our simplification implies deterministic state transition: $q_{ij} \in \{0, 1\}$
  - This indeed simplifies math & coding
  - But can lead to unreasonable solutions
- Start with a guess $\left\{V_i^0\right\}_{i=1}^n$
- Step 1: for each i, solve:

$$V_i^{k+1} = \max_j u(k_i + f(k_i) - k_j) + \beta V_j^k \tag{3}$$

- Step 2: check if $\max_i \left| V_i^k - V_i^{k+1} \right| < \varepsilon$; stop if yes.

Example - other methods & extension

- **Modified policy iteration**
  - Step 1: same; save $j_i = \arg\max_j \ldots$
  - Step 2a: Set $W_i^0 = V_i^k, \qquad i = 1, ..., n$
  - Step 2b: Compute $W_i^{\tau+1} = u(k_i + f(k_i) - k_{j_i}) + \beta W_{j_i}^{\tau}$, repeat for $\tau = 0, \ldots, n$
  - Step 2c: Set $V_i^{k+1} = W_i^{n+1}, \qquad i = 1, ..., n$
- **Gauss-Seidel**: use V_j^{k+1} in (3) whenever available
- Can use dampening or acceleration
- **Extension**: restoring continuity via stochastic transition
  - We want $c_i = c\left(k_i\right) \in R_+$. But then, $k^+\left(k_i\right)$ might not be on the grid
  - $\Rightarrow$ Modify model: $q_{ij}\left(c_i\right)$ randomizes between grid points around $k^+\left(k_i\right)$, to form approximation to $V\left(k^+\right)$

Note on Gaussian methods

- Value function iteration is **Pre**-Gauss-Jacobi iteration:

$$V_i^{k+1} = \max_{u \in D(x_i)} \pi(x_i, u) + \beta \sum_{j=1}^{n} q_{ij}(u) V_j^k, \quad i = 1, \dots, n.$$

  V_i^k (old value) appears in r.h.s.

- True Gauss-Jacobi – solve for V_i:

$$V_i^{k+1} = \max_{u \in D(x_i)} \frac{\pi(x_i, u) + \beta \sum_{j \neq i} q_{ij}(u) V_j^k}{1 - \beta q_{ii}(u)}, \quad i = 1, \dots, n.$$

- Pre-Gauss-Seidel iteration: use already computed V_j^{k+1}'s

Upwind Gauss-Seidel - idea

Gauss-Seidel method depends on the ordering of states.

- At the steady state, we have a static problem
  - Can solve for V_i and u_i in a single iteration
  - Backward Induction: solve states that transit into the steady one,
  - then states that transit into them, and so on
  - This works best with deterministic state transitions
- Stochastic transition – **Upwind Gauss-Seidel**:
  - Recall: $q_{i,j}(U_i) =$ prob. of j being the next period's state
  - Record optimal policy U_i^{k-1} from previous iteration
  - Re-order the states so that $q_{i,i+1}(U_i^{k-1}) \leq q_{i+1,i}(U_{i+1}^{k-1})$,
  - Then update V by visiting the states in this new order

Upwind Gauss-Seidel - tricks

- "$q_{i,i+1}(U_i^{k-1}) \leq q_{i+1,i}(U_{i+1}^{k-1})$" takes time to compute and sort
- *Simulated* upwind G-S:
  - Simulate the Markov process under U^k, which approximates the limiting distribution
  - Visit the states in the order of decreasing probability
- *Alternating sweep* G-S:
  - Visit the states in increasing order if k is odd, ...
  - ... and decreasing order if k is even.
  - $\Rightarrow$ "Right" order on every second iteration.

Additional References

- Dynamic programming in Economics
  - Classics: Ljungqvist and Sargent *Recursive Macroeconomic Theory*, Stokey, Lucas, and Prescott *Recursive Methods in Economic Dynamics*
  - Modern readable intro: Stachurski *Economic Dynamics: Theory and Computation*
  - Code: *Quantecon*
- Reinforcement Learning: CompSci perspective
  - Classic: Sutton and Barto *Reinforcement Learning: An Introduction*
  - Code: Kochendorfer, Wheeler, Wray *Algorithms for Decision Making*
  - Notes: Hazan, *Computational Control Theory*, Agarwal, Jiang, Kakade, Sun *Reinforcement Learning: Theory and Algorithms*, Silver *Introduction to Reinforcement Learning*