MS-E2148 Dynamic optimization

Lecture 10

Contents

- Stationary, discounted problems
- DP algorithm and infinite horizon
- Bellman equation and solving it numerically

- Material from books:
  - D. Bertsekas: Dynamic Programming and Optimal Control, Vol. 2, Athena Scientific 2001
  - M.J. Miranda & P.L. Fackler: Applied Computational Economics and Finance, MIT Press 2002

Stationary problems and discounting

- Discrete time problem where we minimize

$$E_{w_k}\left\{\alpha^N J(x_N)+\sum_{k=0}^{N-1}\alpha^k g(x_k,\mu_k(x_k),w_k)\right\}, \qquad k=0,...,N-1 \tag{1}$$

  so that

$$x_{k+1}=f(x_k,u_k,w_k)$$

  and $J_N(x)=\alpha^N J(x)$

- The problem is *stationary*: $g_k = g$ and $f_k = f$ for all k
- The problem is also *discounted*: the factor $\alpha \in (0,1)$ weights more the instant gains/losses compare to those far in the future

Stationary problems and discounting

- The minimum is $J^*(x) = \min_\pi J_\pi(x)$ and the optimal control π can be solved using DP algorithm
- If we are at stage with k stages *remaining*, DP algorithm gives the expected cost-to-go

$$J_{N-k}(x) = \min_u E\left\{\alpha^{N-k} g(x,u,w) + J_{N-k+1}(f(x,u,w))\right\} \tag{2}$$

  as a function of the state x
- (2) is the cost-to-go for the subproblem of length k

Stationary problems and discounting

- Let us formulate the cost-to-go in another form using the function
$$V_k(x) = \frac{J_{N-k}(x)}{\alpha^{N-k}}$$
- Thus, $V_N(x)$ is the cost-to-go $J_0(x)$ for the problem of length N
- The cost-to-go (2) in DP algorithm can be written

$$V_{k+1}(x) = \min_u E\Big\{g(x,u,w) + \alpha V_k(f(x,u,w))\Big\}, \quad k = 0, ..., N-1 \tag{3}$$

where $V_0(x) = J_N(x)$

Stationary problems and discounting

- Note that (3) is "forward DP": if we have computed the optimal cost for stage $N-1$, V_{N-1}, we get V_N in one iteration
- E.g.: if we know the final cost J_N, we can compute $V_0 = J_N$, and

$$V_1(x) = \min_u E\Big\{g(x,u,w) + \alpha V_0(f(x,u,w))\Big\}$$
$$V_2(x) = \min_u E\Big\{g(x,u,w) + \alpha V_1(f(x,u,w))\Big\}$$

  and so on
- The property is due to stationarity: at each stage g and f are the same

Stationary problems and discounting

Bellman equation

- Equation (3) can be used in solving the *infinite horizon* problem iteratively:
- For each computation of (3), we increase the length of the problem by one stage – we can convert the finite length problem into infinite length by taking the limit $k \to \infty$:

$$J^*(x) = \min_u E_w \Big\{ g(x, u, w) + \alpha J^*(f(x, u, w)) \Big\} \qquad (4)$$

- Equation (4) is called the *Bellman equation*
- The total cost of the stationary, discounted, ∞ horizon problem is $J(x_0) = \lim_{N \to \infty} \sum_{k=0}^{N-1} \alpha^k g(x_k, u_k)$

The reasoning in the Bellman equation

- Let us define a function $(TJ)(x)$ that we get for each function $J(x)$ by DP iteration:

$$(TJ)(x) = \min_u E_w \Big\{ g(x, u, w) + \alpha J(f(x, u, w)) \Big\} \quad (5)$$

- TJ is the optimal cost for the one-stage problem where the current cost is g and the final cost αJ
- T can be seen as a mapping that transforms J into a new function like one step of DP: $J_{k+1} = TJ_k$
- This is called fixed-point iteration that converges at rate α to the unique solution of $J = TJ$, J^*
- The convergence required that the stage costs are bounded $|g(x, u)| \leq M$ for all x, u; this is satisfied is the state and control sets are finite

The reasoning in the Bellman equation

- Bellman equation says that J^* is a fixed-point of mapping T
- Bellman equation is a *functional equation*
- It is solved by a stationary control law

$$\pi = \{\mu, \mu, ...\}$$

  Let us also denote $J_\mu(x) = J^*(x)$ as the optimal stationary control law

- Sometimes, the recursion of DP algorithm is called Bellman equation for finite length problem

Bellman equation: example

- Let us maximize the infinite horizon discounted utility:

$$\sum_{k=0}^{\infty} \alpha^k \ln(u_k), \qquad \alpha \in (0,1) \tag{6}$$

  when the system is deterministic $x_{k+1} = \theta(x_k - u_k)$, $\theta > 0$, and the controls are bounded $u_k \in [0, x_k]$

- Bellman:

$$V(x) = \max_u \left\{ \ln(u) + \alpha V(\theta(x - u)) \right\}$$

  we are balancing the instant utility and discounted future utility

Bellman equation: example

- Implicit equation is not good to solve without a trial... let us guess$^{*)}$ that $V(x) = a + b\ln(x)$ for some a, b:

$$a + b\ln(x) = \max_u \left\{ \ln(u) + \alpha a + \alpha b\ln(\theta(x-u)) \right\}$$

- The first-order condition for the right-hand side maximization:

$$\frac{1}{u^*} - \frac{\alpha b}{x - u^*} = 0$$

from which $u^* = \frac{x}{\alpha b + 1}$

- (* In general, the functional equation of the form $f(xy) = f(x) + f(y)$ is satisfied by the logarithm function)

Bellman equation: example

- With the trial, the Bellman is:

$$a + b\ln(x) = \alpha a + \ln\left(\frac{x}{\alpha b + 1}\right) + \alpha b \ln\left(\theta\left(x - \frac{x}{\alpha b + 1}\right)\right)$$

from which we get the constants by comparing the multipliers $b = 1/(1-\alpha)$; and thus the solution $u^* = (1-\alpha)x$

Numerical methods

- Let us denote the vector notation for the numerical methods
- $v \in R^n$ a vector of the value function; $v_i =$ is the value at state i
- $u_i \in U$ is the control at state i
- Each $u \in U^n$ corresponds to a vector of utility $g(u) \in R^n$, and thus

$$g_i(u) = \text{ is the utility at state } i \text{ with control } u_i$$

- Stochastics is described with a state transition matrix $P \in R^{n \times n}$:

$$P_{ij}(u) = \text{ probability to move from state } i \text{ to state } j \text{ with } u_i$$

Numerical methods

- (Note that there actually are *m* matrices; one for each control)
- With this notation, we get the vector-valued recursion (Bellman/DP):

$$v_t = \max_u \{g(u) + \alpha P(u) v_{t+1}\} \tag{7}$$

where "max" means vector operation that maximizes each row separately

Numerical methods

- For finite length problem, we can use the **backward recursion**:

0. Initial step: set T, g, P, α and v_{T+1} and set $t \leftarrow T$
1. Recursion step: when we have v_{t+1}, compute v_t and u_t:

$$\begin{aligned} v_t &\leftarrow \max_u \{g(u) + \alpha P(u) v_{t+1}\} \\ u_t &\leftarrow \arg\max_u \{g(u) + \alpha P(u) v_{t+1}\} \end{aligned}$$

2. Stopping condition: if $t = 1$ stop; otherwise, set $t \leftarrow t - 1$ and return to step 1

Numerical methods

- For infinite length problem, we can use the **value iteration**:

0. Initial step: set g, P, α and convergence tolerance τ and initial guess to the value function v
1. Iteration step: update the value function

$$v \leftarrow \max_u \{g(u) + \alpha P(u) v\}$$

2. Stopping condition: if $\|\Delta v\| < \tau$, set

$$u \leftarrow \arg\max_u \{g(u) + \alpha P(u) v\}$$

and stop; otherwise, return to step 1.

Numerical methods

- For infinite length problem, we can also use the **policy iteration**: (Howard 1960)

0. Initial step: set g, P, α and initial guess for value function v
1. Iteration step: update first the control

$$u \leftarrow \arg\max_u \{g(u) + \alpha P(u) v\}$$

   and then the value function

$$v \leftarrow (I - \alpha P(u))^{-1} g(u)$$

2. Stopping condition: if the value function does not improve ($\Delta v = 0$) stop; otherwise, return to step 1.

Numerical methods

Observations

- If $\alpha < 1$, the value function exists and is unique for the ∞ horizon problem

- Policy iteration gives the exact solution when the number of states and controls is finite; it is based on the Newton's method where the root of Bellman equation is solved

$$v - \max_u \{g(u) + \alpha P(u)v\} = 0$$

Numerical methods

Observations

- The computational complexity is proportional to the dimensions of the state and control variables:
- Let us assume that the state is vector of length n and each component can have ℓ values; then the computer has to go through n^ℓ states
- Same holds for the dimensions of control which is of length m and each component has ℓ_2 values
- To solve the whole problem, we have to do $n^{\ell_1} \cdot m^{\ell_2}$ operations on each iteration

  $\Rightarrow$ the *curse of dimensionality* (R. Bellman 1957)

Summary

- Discrete time stationary, discounted problems
- Bellman equation
- Numerical methods to solve Bellman equation: backward recursion, value iteration, policy iteration