MS-E2148 Dynamic optimization

Lecture 9

Contents

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

- Material from:
  - D. Bertsekas: Dynamic Programming and Optimal Control, Vol. 2, Athena Scientific 2001

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 compared to those 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* (e.g., a stage $N-1$, with $k=1$ remaining), DP algorithm gives the expected cost-to-go

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

  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 $V_0(x) = J_N(x)/\alpha^N$

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 $V_0(x) = J_N(x)/\alpha^N$, 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 of the same form.

- 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, $N \to \infty$:

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

- Equation (4) is called the infinite horizon *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:

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

- TJ is the optimal cost for the one-stage problem where the current cost is g and the final cost-to-go 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 requires 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 bounded.

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 $J_\pi(x) = J^*(x)$ as the optimal stationary cost value.

- Sometimes, the recursion of the finite stage 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:

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

  $\Leftrightarrow$ 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 define the notation for the numerical methods
- We assume a finite-state discrete-time dynamic system whereby, at state i, the use of a control u specifies the state transition probability $P_{ij}(u)$ to the next state j.
- We call P the **state transition matrix.**
- Here the state i is an element of a finite state space, and the control u is constrained to take values in a given finite constraint set $U(i) \in R^m$, which may depend on the current state i.
- We can suppress w from the Bellman equation.
- $v \in R^n$ a vector of the value function; $v(i) =$ is the value at state i
- Each u corresponds to a vector of utility $g(u) \in R^n$, and thus

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

Numerical methods

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

$$v = \max_{u}\{g(u) + \alpha P(u)v\} \tag{7}$$

  where "max" means vector operation that maximizes each row separately
- This equation can be solved recursively

Numerical methods

- Let t be the iteration number
- For an infinite length problem, we can use the **value iteration**:

0. Set the convergence tolerance τ, and initial guess for the value function $v^0(i)$, for all states i. Let $t = 1$. Solve the function
$$\mu^0 = \arg\max_u \{g(u) + \alpha P(u) v^0\}$$
1. Iteration step: update the value function
$$v^t = g(\mu^{t-1}) + \alpha P(\mu^{t-1}) v^{t-1}$$
2. Stopping condition: if $\|v^t - v^{t-1}\| < \tau$, set
$$\mu^t = \arg\max_u \{g(u) + \alpha P(u) v^t\}$$
and stop; otherwise $t = t + 1$, return to step 1.

Numerical methods

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

0. Initial guess for the optimal policy $\mu^0(i)$, for all states i. Let $t = 1$. Solve the value function

$$v^0 = \left(I - \alpha P(\mu^0)\right)^{-1} g(\mu^0)$$

1. Calculate a new value for the optimal policy

$$\mu^t = \arg\max_u \left\{g(u) + \alpha P(u) v^{t-1}\right\}$$

2. Solve the value function

$$v^t = \left(I - \alpha P(\mu^t)\right)^{-1} g(\mu^t)$$

3. Stopping condition: if the value function does not improve ($v^t - v^{t-1} = 0$) stop; otherwise $t = t + 1$, and return to step 1.

- If $\alpha < 1$, the value function exists and is unique for the ∞ horizon problem
- If there is an absorbing state *a*, i.e., a state *a* for which it holds $P_{aa} = 1$, and is P_{aj}, for all states $j \neq a$, then the value function exists even when $\alpha = 1$.
- 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

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

- We assume to know the transition and cost functions in advance. We could as well assume they are unknown and must be learned through trial and error (*reinforcement learning*).
- Reinforcement learning can be combined with function approximation (*deep reinforcement learning*). One solution is to use an artificial neural network as a function approximator. It may speed up learning in finite problems, due to the fact that the algorithm can generalize earlier experiences to previously unseen states.
- Applications of reinforcement learning include AlphaGo (first computer program to beat a human professional player in the game of Go) and autonomous driving.

Summary

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