IE 3186: Approximate Dynamic Programming **Fall 2018**

Lecture 1: Overview of MDP Models

Lecturer: Daniel Jiang *Scribes: Jing Yang*

References:

D.P. Bertsekas. *Dynamic Programming and Optimal Control: Approximate Dynamic Programming*, Vol. 2, 4th ed, Athena Scientific, Belmont MA, 2012. (§1.1)

W.B. Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed, Wiley & Sons, 2007. (Chapter 3)

1.1 Finite-horizon MDP Models

The core components of a finite-horizon MDP are:

(1) a discrete-time dynamical system

$$x_{k+1} = f_k(x_k, u_k, w_{k+1}),\ k = 0, 1, \ldots, N-1,$$

where

- k indexes time up to a finite time horizon N;
- $x_k \in X_k$ is a "state";
- $u_k \in U_k(x_k)$ is a "control," "action," or "decision";
- w_{k+1} is "noise" or "disturbance" independent from x_k, u_k and across time.

(2) an additive cost function g_k such that the total cost is written as:

$$g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_{k+1}),$$

where the first part is the cost landing in a state x_N in the last period. Our goal is to optimize:

$$\mathbf{E}\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_{k+1})\right].$$

Remark 1.1. *It is also common to specify a system using transition probabilities:* $p^a_{ss'} = \mathbf{P}(f_k(s, a, w) = s')$.

Example 1.2. *(Inventory control)*

- *ordering a quantity of some products to meet demand and minimize cost;*
- x_k *= inventory in stock at time* k*;*
- u_k *= how much to order at time* k*;*
- w_{k+1} *= demand at time* $k+1$*, satisfied using* x_k*,* u_k*;*
- *transition:* $x_{k+1} = x_k + u_k - w_{k+1}$ *(backlog* $\Leftrightarrow x_k < 0$*).*

Optimization problem:

$$\min_{u_0, u_1, \ldots, u_{N-1}} R(x_N) + \sum_{k=0}^{N-1} (cu_k + r(x_k)),$$

where

- $R(x_N)$*: the disposal cost in the last time period;*
- cu_k*: the order cost;*
- $r(x_k)$*: holding cost* ($x_k > 0$)*/backlog cost* ($x_k < 0$)*.*

In general, we have two different ways to minimize the above problem:

(a) open-loop minimization: choose $u_0, \ldots, u_{N-1}$ *all at time 0.*

(b) close-loop minimization: choose u_k *at time* k*.*

Close-loop minimization is what we focus on, under which our goal becomes minimizing over policies π*.*

1.2 Optimal Policies

Mathematically, we are searching for a sequence of functions $\mu_0, \mu_1, \ldots, \mu_{N-1}$ such that $\mu_k(x_k) \in \mathcal{U}_k(x_k)$ (deterministic maps from states to actions).

Definition 1.3. *An admissible policy* $\pi = (\mu_0, \ldots, \mu_{N-1})$ *is a sequence of functions mapping from states* x_k *to actions in* $\mathcal{U}_k(x_k)$*.*

Under π, the system evolves like

$$x_{k+1} = f_k(x_k, \mu_k(x_k), w_{k+1}).$$

Definition 1.4. *Value function (or "cost to go" function) is:*

$$J_\pi(x_0) = \mathbf{E}\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_{k+1})\right].$$

The optimal policy minimizes $J_\pi(x_0)$, that is

$$J_{\pi^*}(x_0) = \min_{\pi\in\Pi} J_\pi(x_0) = J^*(x_0) \quad \forall x_0,$$

where Π is the set of all possible admissible policies.

Remark 1.5. *We've implicitly only considered Markov & deterministic policies. There are also:*

(1) history dependent (x_k depends on $x_{k-1}, x_{k-2}, \ldots$)

(2) stochastic (returns a distribution over $\mathcal{U}_k$).

By well-known results (Section 4.4, Puterman), we know that there exists an optimal policy that is Markov & deterministic. The proof ideas are as follows.

(1) For the history dependent case, costs/transitions depend only on x_k (Theorem 4.4.2 (a), Puterman).

(2) For the stochastic case, we can shift the distribution toward deterministic policy without increasing cost (Lemma 4.3.1, Puterman).

Principle of Optimality: let $\pi^* = \{\mu_0^*, \ldots, \mu_{N-1}^*\}$ be an optimal policy. Suppose some state x_i at time i is visited by π^* with probability > 0. Consider

$$\min_{\text{policies}} \mathbf{E}\left[g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_{k+1})\right], \qquad (\star)$$

the tail policy $\{\mu_i^*, \ldots, \mu_{N-1}^*\}$ is optimal for $(\star)$.
The intuition behind: if it is not optimal, then π^* cannot be optimal for the original problem.

Algorithm 1 Backward Dynamic Programming Algorithm

Let $J_N(x_N) = g_N(x_N) \quad \forall x_N \in X_N$
for $k < N$ **do**
$J_k(x_N) = \min_{u_k \in U_k(x_k)} \mathbf{E}[g_k(x_k, u_k, w_{k+1}) + J_{k+1}(f_k(x_k, u_k, w_{k+1}))]$
end for

Then $J_0^*(x_0) = J_0(x_0)$ and if u_k^* minimizes RHS, then $\mu_k^*(x_k) = u_k^*$ is the optimal policy.

Proof. Let

$$J_k^*(x_k) = \min_{\pi^k} \mathbf{E}\left[g_N(x_N) + \sum_{i=k}^{N-1} g_i(x_i, \mu_i(x_i), w_{i+1})\right],$$

where $\pi^k = (\mu_k, \ldots, \mu_{N-1})$. Our goal is to show that $J_k^* = J_k$, then $J_0^* = J^* = J_0$.

Base case: $k = N$, $J_N^*(x_N) := g_N(x_N)$. By definition, $J_N^* = J_N$.

Induction step: assume for some k, $J_{k+1}^*(x_{k+1}) = J_{k+1}(x_{k+1})$ for all x_{k+1}. Note that $\pi^k = (\mu_k, \pi^{k+1})$, then we have

$$\begin{aligned}
J_k^*(x_k) &= \min_{(\mu_k, \pi^{k+1})} \mathbf{E}\left[g_k(x_k, \mu_k(x_k), w_{k+1}) + g_N(x_N) + \sum_{i=k+1}^{N-1} g_i(x_i, \mu_i(x_i), w_{i+1})\right] \\
&= \min_{(\mu_k, \pi^{k+1})} \mathbf{E}\left[g_k(\ldots) + \mathbf{E}[g_N(\ldots) + \sum_{i=k+1}^{N-1} g_i(\ldots)|x_{k+1}]\right] \\
&= \min_{\mu_k} \mathbf{E}\left[g_N(\ldots) + \min_{\pi^{k+1}} \mathbf{E}[g_N(\ldots) + \sum_{i=k+1}^{N-1} g_i(\ldots)|x_{k+1}]\right] \\
&= \min_{\mu_k} \mathbf{E}[g_k(\ldots) + J_{k+1}^*(x_{k+1})] \\
&= \min_{u_k \in U_k(x_k)} \mathbf{E}[g_k(x_k, u_k, w_{k+1}) + J_{k+1}(f_k(x_k, u_k, w_{k+1}))] \\
&= J_k(x_k),
\end{aligned}$$

where we used the principle of optimality in the third line. ■

Example 1.6. *(Comparison between decision tree) Suppose $|X_k| \leq n$, $|\mathcal{U}_k| \leq m$. Then, the backward DP needs to compute the expressions like $g_k(\ldots) + J_{k+1}(\ldots)$ $\underline{Nmn}$ times. However, a decision tree approach requires m^N computations.*

Example 1.7. *(Parking Problem) Let $0, 1, \ldots, N-1$ denote N parking spaces with cost c_k, and N denote a parking garage with cost C, which is more expensive. Once*

you are in front of a spot, it is available with probability p and taken with probability $1-p$ (i.i.d).

Model:

- *State space: $\{A, T, D\}$, where state D denotes an artificial state where the decision maker is done.*
- *Action space: $\{park,\ next\}$ at state A, $\{next\}$ at state T and $\{stay\}$ at state D.*
- *Cost function: $g_k(A, park) = c_k$, $g_k(\cdot, next) = 0$, $g_k(\cdot, stay) = 0$, $g_k(\cdot) = C$.*
- *Transitions:*

$$x_{k+1} = f_k(x_k, u_k, w_{k+1}) = \begin{cases} w_{k+1} & \text{if } u_k = next, \\ D & \text{if } u_k = park, \\ D & \text{if } u_k = stay. \end{cases}$$

- *Disturbance:*

$$w_{k+1} = \begin{cases} A & w.p.\ p, \\ T & w.p.\ 1-p. \end{cases}$$

$J_k^*(D) = 0, \quad \forall k.$
$J_k^*(A) = \min\{c_k + J_{k+1}^*(D), pJ_{k+1}^*(A) + (1-p)J_{k+1}^*(T)\}.$
$J_k^*(T) = pJ_{k+1}^*(A) + (1-p)J_{k+1}^*(T).$

1.3 Stationary Case

Consider case there costs are identical across time, but discounted:

$$g_k(x_k, u_k, w_{k+1}) = \gamma^k g(x_k, u_k, w_{k+1}),$$

$$g_N(x_N) = \gamma^N g(x_N).$$

Also, $u_k(x) = u(x) \quad \forall x$, $f_k = f$, $X_k = X$ for all k, and w_{k+1} is i.i.d.

Bellman operator notation: let μ be a stationary policy ($\mu(x) \in \mathcal{U}(x)$), that is, $\pi = (\mu, \mu, \ldots, \mu)$. Let $J : X \to \mathbb{R}$. Define the map which takes J and returns another value function ($T_\mu J$) as

$$(T_\mu J)(x) = \mathbf{E}[g(x, \mu(x), w) + \gamma J(f(x, \mu(x), w))],$$

which is the "evaluation Bellman operator for policy μ." $(T_\mu J)$ answers the question "if J is the cost-to-go, what is the expected cost of one step of μ?". Accordingly, we can also define the optimizing version of T_μ:

$$(TJ)(x) = \min_{\mu \in \mathcal{U}} \mathbf{E}[g(x, \mu(x), w) + \gamma J(f(x, \mu(x), w))]$$

T is called the "Bellman operator," which answers the question "if J is the cost to go, what is the best I can achieve?"

Example 1.8 (Backward DP using Shorthand)**.** *At the last time period, we have* $J_{N-1}(x) = (Tg)(x)$*. Similarly,* $J_{N-2}(x) = (TJ_{N-1})(x) = (TTg)(x) = (T^2 g)(x) = J^*_{N-2}$*. Recursively applying these relationships, we can write* $J_0(x) = (T^N g)(x) = J^*_0(x) = J^*(x)$*. We also have that* $T^*_{\mu_k} J^*_{k+1} = T J^*_{k+1}$ *for all* k*.*

Example 1.9 (Shorthand for Evaluation)**.** *Let* $\pi = (\mu_0, \ldots, \mu_{N-1})$*. We can write the value of policy* π *by iterating the evaluation version of the Bellman operator:*

$$J_\pi(x) = (T_{\mu_0}, T_{\mu_1}, \ldots, T_{\mu_{N-1}} g)(x).$$