Lecture Note 1

1 Markov decision processes

In this class we will study discrete-time stochastic systems. We can describe the evolution (dynamics) of these systems by the following equation, which we call the system equation:

$$x_{t+1} = f(x_t, a_t, w_t), \tag{1}$$

where $x_t \in \mathcal{S}$, $a_t \in \mathcal{A}_{x_t}$ and $w_t \in \mathcal{W}$ denote the system state, decision and random disturbance at time t, respectively. In words, the state of the system at time $t+1$ is a function f of the state, the decision and a random disturbance at time t. An important assumption of this class of models is that, conditioned on the current state x_t, the distribution of future states $x_{t+1}, x_{t+2}, \ldots$ is independent of the past states $x_{t-1}, x_{t-2}, \ldots$. This is the *Markov property*, which rise to the name *Markov decision processes*.

An alternative representation of the system dynamics is given through *transition probability matrices*: for each state-action pair (x, a), we let $P_a(x, y)$ denote the probability that the next state is y, given that the current state is x and the current action is a.

We are concerned with the problem of how to make decisions over time. In other words, we would like to pick an action $a_t \in \mathcal{A}_{x_t}$ at each time t. In real-world problems, this is typically done with some objective in mind, such as minimizing costs, maximizing profits or rewards, or reaching a goal. Let $u(x, t)$ take values in $\mathcal{A}_x$, for each x. Then we can think of u as a decision rule that prescribes an action from the set of available actions $\mathcal{A}_x$ based on the current time stage t and current state x. We call u a *policy*.

In this course, we will assess the quality of each policy based on costs that are accumulated additively over time. More specifically, we assume that at each time stage t a cost $g_{a_t}(x_t)$ is incurred. In the next section, we describe some of the optimality criteria that will be used in this class when choosing a policy.

Based on the previous discussion, we characterize a Markov decision process by a tuple $(\mathcal{S}, \mathcal{A}_\cdot, P_\cdot(\cdot,\cdot), g_\cdot(\cdot))$, consisting of a state space, a set of actions associated with each space, transition probabilities and costs associated with each state-action pair. For simplicity, we will assume throughout the course that $\mathcal{S}$ and $\mathcal{A}_x$ are finite. Most results extend to the case of countably or uncountably infinite state and action spaces under certain technical assumptions.

2 Optimality Criteria

In the previous section we described Markov decision processes, and introduced the notion that decisions are made based on certain costs that must be minimized. We have established that, at each time stage t, a cost $g_{a_t}(x_t)$ is incurred. In any given problem, we must define how costs at different time stages should be combined. Some optimality criterions that will be used in the course are the following:

1. Finite-horizon total cost:

$$\mathrm{E}\left[\sum_{t=0}^{T-1} g_{a_t}(x_t) \,\middle|\, x_0 = x\right] \tag{2}$$

2. Average cost:

$$\limsup_{T\to\infty} \mathrm{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} g_{a_t}(x_t)\,\middle|\, x_0 = x\right] \tag{3}$$

3. Infinite-horizon discounted cost:

$$\mathrm{E}\left[\sum_{t=0}^{\infty} \alpha^t g_{a_t}(x_t)\,\middle|\, x_0 = x\right], \tag{4}$$

where $\alpha \in (0,1)$ is a discount factor expressing temporal preferences. The presence of a discount factor is most intuitive in problems involving cash flows, where the value of the same nominal amount of money at a later time stage is not the same as its value at a earlier time stage, since money at the earlier stage can be invested at a risk-free interest rate and is therefore equivalent to a larger nominal amount at a later stage. However, discounted costs also offer good approximations to the other optimality criteria. In particular, it can be shown that, when the state and action spaces are finite, there is a large enough $\bar{\alpha} < 1$ such that, for all $\alpha \geq \bar{\alpha}$, optimal policies for the discounted-cost problem are also optimal for the average-cost problem. However, the discounted-cost criterion tends to lead to simplified analysis and algorithms.

Most of the focus of this class will be on discounted-cost problems.

3 Examples

The Markov decision processes has a broad range of applications. We introduce some interesting applications in the following.

Queueing Networks

Consider the queueing network in Figure 1. The network consists of three servers and two different external jobs, fixed routes $1, 2, 3$ and $4, 5, 6, 7, 8$, forming a total of 8 queues of jobs at distinct processing stage. We assume the service times are distributed according to geometric random variables: When a server i devotes a time step to serving a unit from queue j, there is a probability μ_{ij} that it will finish processing the unit in that time step, independent of the past work done on the unit. Upon completion of that processing step, the unit is moved to the next queue in its route, or out of the system if all processing steps have been completed. New units arrive at the system in queues $j = 1, 4$ with probability λ_j in any time step, independent of the previous arrivals.

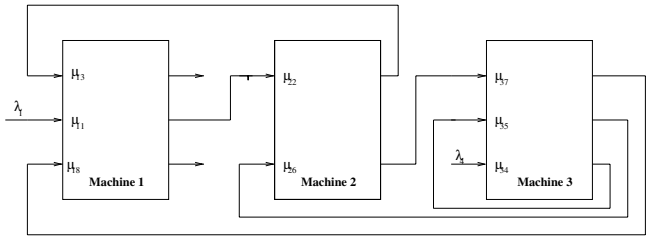

Figure 1: A queueing system

A common choice for the state of this system is an 8-dimensional vector containing the queue lengths. Since each server serves multiple queues, in each time step it is necessary to decide which queue each of the different servers is going to serve. A decision of this type may be coded as an 8-dimensional vector a indicating which queues are being served, satisfying the constraint that no more than one queue associated with each server is being served, i.e., $a_i \in \{0, 1\}$, and $a_1 + a_3 + a_8 \leq 1$, $a_2 + a_6 \leq 1$, $a_4 + a_5 + a_7 \leq 1$. We can impose additional constraints on the choices of a as desired, for instance considering only non-idling policies.

Policies are described by a mapping u returning an allocation of server effort a as a function of system x. We represent the evolution of the queue lengths in terms of transition probabilities - the conditional probabilities for the next state $x(t+1)$ given that the current state is $x(t)$ and the current action is $a(t)$. For instance

$$
\begin{aligned}
&Prob(x_1(t+1) = x_1(t) + 1 \,|\, x(t), a(t)) = \lambda_1, \\
&Prob(x_3(t+1) = x_3(t) + 1, x_2(t+1) = x_2(t) - 1 \,|\, (x(t), a(t)) = \mu_{22} I(x_2(t) > 0, a_2(t)) = 1), \\
&Prob(x_3(t+1) = x_3(t) - 1 \,|\, (x(t), a(t)) = \mu_{13} I(x_3(t) > 0, a_3(t)) = 1),
\end{aligned}
$$

corresponding to an arrival to queue 1, a departure from queue 2 and an arrival to queue 3, and a departure from queue 3. $I(\cdot)$ is the indication function. Transition probabilities related to other events are defined similarly.

We may consider costs of the form $g(x) = \sum_i x_i$, the total number of unfinished units in the system. For instance, this is a reasonably common choice of cost for manufacturing systems, which are often modelled as queueing networks.

Tetris

Tetris is a computer game whose essence rule is to fit a sequence of geometrically different pieces, which fall from the top of the screen stochastically, together to complete the contiguous rows of blocks. Pieces arrive sequentially and the geometric shape of the pieces are independently distributed. A falling piece can be rotated and moved horizontally into a desired position. Note that the rotation and move of falling pieces must be scheduled and executed before it reaches the remaining pile of pieces at the button of the screen. Once a piece reaches the remaining pile, the piece must resite there and cannot be rotated or moved.

To put the Tetris game into the framework of Markov decision processes, one could define the state to correspond to the current configuration and current falling piece. The decision in each time stage is where to place the current falling piece. Transitions to the next board configuration follow deterministically from the current state and action; transitions to the next falling piece are given by its distribution, which could be, for instance, uniform over all piece types. Finally, we associate a reward with each state-action pair, corresponding to the points achieved by the number of rows eliminated.

Portfolio Allocation

Portfolio allocation deals with the question of how to invest a certain amount of wealth among a collection of assets. One could define the state as the wealth of each time period. More specifically, let x_0 denote the initial wealth and x_t as the accumulated wealth at time period t. Assume there are n risky assets, which correspond to random rate of return $e_1, \ldots, e_n$. Investors distribute fractions $a = (a^1, \ldots, a^n)$ of their wealth among the n assets, and consume the remaining fraction $1 - \sum_{i=1}^{n} a^n$. The evolution of wealth x_t is given

by

$$x_{t+1} \quad = \quad \sum_{i=1}^{n} a_t^i e_i x_t.$$

Therefore, transition probabilities can be derived from the distribution of the rate of return of each risky assets. We associate with each state-action pair (x, a) a reward $g_a(x) = x(1 - \sum_{i=1}^{n} a^i$, corresponding to the amount of wealth consumed.

4 Solving Finite-Horizon Problems

Finding a policy that minimizes the finite-horizon cost corresponds to solving the following optimization problem:

$$\min_{u(\cdot,\cdot)} \mathrm{E}\left[\sum_{t=0}^{T-1} g_{u(x_t,t)}(x_t)|x_0 = x\right] \tag{5}$$

A naive approach to solving (5) is to enumerate all possible policies $u(x, t)$, evaluate the corresponding expected cost, and choose the policy that maximizes it. However, note that the number of policies grows exponentially on the number of states and time stages. A central idea in dynamic programming is that the computation required to find an optimal policy can be greatly reduced by noting that (5) can be rewritten as follows:

$$\min_{a\in\mathcal{A}_x}\left\{g_a(x) + \sum_{y\in\mathcal{S}} P_a(x,y) \min_{u(\cdot,\cdot)} \mathrm{E}\left[\sum_{t=1}^{T-1} g_{u(x_t,t)}(x_t)|x_1 = y\right]\right\}. \tag{6}$$

Define $J^*(x, t_0)$ as follows:

$$J^*(x,t_0) = \min_{u(\cdot,\cdot)} \mathrm{E}\left[\sum_{t=t_0}^{T-1} g_{u(x_t,t)}(x_t)|x_1 = y\right].$$

It is clear from (6) that, if we know $J^*(\cdot, t_0 + 1)$, we can easily find $J^*(x, t_0)$ by solving

$$J^*(x,t_0) = \min_{a\in\mathcal{A}_x}\left\{g_a(x) + \sum_{y\in\mathcal{S}} P_a(x,y)J^*(y,t_0+1)\right\}. \tag{7}$$

Moreover, (6) suggests that an optimal action at state x and time t_0 is simply one that minimizes the right-hand side in (7). It is easy to verify that this is the case by using backwards induction.

We call $J^*(x, t)$ the *cost-to-go function*. It can be found recursively by noting that

$$J^*(x, T-1) = \min_a g_a(x)$$

and $J^*(x, t), t = 0, \ldots, T-2$, can be computed via (7).

Note that finding $J^*(x, t)$ for all $x \in \mathcal{S}$ and $t = 0, \ldots, T-1$ requires a number of computations that grow linearly in the number of states and time stages, even though there are exponentially many policies.

5 Introduction to Discounted-Cost Problems

Based on the discussion for the finite-horizon problem, we may conjecture that an optimal decision for the infinite-horizon, discounted-cost problem may be found as follows:

1. Find (somehow) for every x and t_0,

$$J^*(x,t_0) = \min_{u(\cdot,\cdot)} \mathrm{E}\left[\sum_{t=t_0}^{\infty} \alpha^{t-t_0} g_{u(x_t,t)}(x_t) | x_{t_0} = x\right] \tag{8}$$

2. The optimal action for state x at time t_0 is given by

$$u^*(x,t_0) = \mathrm{argmin}_{\mathrm{a}\in\mathcal{A}_\mathrm{x}} \left\{ \mathrm{g_a(x)} + \alpha \sum_{\mathrm{y}\in\mathcal{S}} \mathrm{P_a(x,y) J^*(y, t_0+1)} \right\}. \tag{9}$$

We may also conjecture that, as in the finite-horizon case, $J^*(x,t)$ satisfies a recursive relation of the form

$$J^*(x,t) = \min_{a\in\mathcal{A}_x} \left\{ g_a(x) + \alpha \sum_{y\in\mathcal{S}} P_a(x,y) J^*(y,t+1) \right\}.$$

The first thing to note in the infinite-horizon case is that, based on expression (8), we have $J^*(x,t) = J^*(x,t') = J^*(x)$ for all t and t'. Indeed, note that, for every u,

$$\begin{aligned}
\mathrm{E}\left[\sum_{t=t_0}^{\infty} \alpha^{t-t_0} g_{u(x_t,t)}(x_t) | x_{t_0} = x\right] &= \sum_{t=t_0}^{\infty} \alpha^{t-t_0} Prob_u(x_t = y | x_{t_0} = x) g_{u(y)}(y) \\
&= \sum_{t=t_0}^{\infty} \alpha^{t-t_0} Prob_u(x_{t-t_0} = y | x_0 = x) g_{u(y)}(y) \\
&= \sum_{t=0}^{\infty} \alpha^{t} Prob_u(x_t = y | x_0 = x) g_{u(y)}(y).
\end{aligned}$$

Intuitively, since transition probabilities $P_u(x,y)$ do not depend on time, infinite-horizon problems look the same regardless of the value of the initial time state t, as long as the initial state is the same.

Note also that, since $J^*(x,t) = J^*(x)$, we can also infer from (9) that the optimal policy $u^*(x,t)$ does not depend on the current stage t, so that $u^*(x,t) = u^*(x)$ for some function $u^*(\cdot)$. We call policies that do not depend on the time stage *stationary*. Finally, J^* must satisfy the following equation:

$$J^*(x) = \min_{a\in\mathcal{A}_x} \left\{ g_a(x) + \alpha \sum_{y\in\mathcal{S}} P_a(x,y) J^*(y) \right\}.$$

This is called *Bellman's equation*.

We will show in the next lecture that the cost-to-go function is the unique solution of Bellman's equation and the stationary policy u^* is optimal.

6 The Dynamic Programming Operators T and T_u

We now introduce some shorthand notation. For every stationary policy u, we let g_u denote the vector with entries $g_{u(x)}(x)$, and P_u denote the matrix with entries $P_{u(x)}(x,y)$. We define the dynamic programming operators T_u and T as follows. For every function $J : \mathcal{S} \mapsto \Re$, we have

$$T_u J = g_u + \alpha P_u J,$$

and

$$TJ = \min_u T_u J.$$

With this new notation, Bellman's equation becomes

$$J^* = TJ^*,$$

and the policy u^* defined in the previous section satisfies

$$T_{u^*} J^* = TJ^*.$$

More generally, for any function J, we call a policy u *greedy* with respect to J if

$$T_u J = TJ.$$

We denote any policy that is greedy with respect to J by u_J.

The following basic properties about the operator T are relevant to much of the analysis in this course.

Lemma 1 (Monotonicity) *Let $J \leq \bar{J}$ be arbitrary. Then $TJ \leq T\bar{J}$.*

Proof Since $P_u \geq 0$, we have for all u

$$\begin{aligned} T_u J &= g_u + \alpha P_u J \\ &\leq g_u + \alpha P_u \bar{J} \\ &= T_u \bar{J}. \end{aligned}$$

Now

$$\begin{aligned} TJ &\leq T_{u_{\bar{J}}} J \\ &\leq T_{u_{\bar{J}}} \bar{J} \\ &= T\bar{J} \end{aligned}$$

□

We let e denote the vector with all entries equal to one.

Lemma 2 (Offset) *For all J and $k \in \Re$, we have $T(J + ke) = TJ + \alpha ke$.*

Proof We have

$$\begin{aligned} T(J + ke) &= \min_u \{g_u + \alpha P_u (J + ke)\} \\ &= \min_u \{g_u + \alpha P_u J + \alpha ke\} \\ &= TJ + \alpha ke. \end{aligned}$$

The second inequality follows from the fact that $P_u e = e$, since $\sum_{y \in \mathcal{S}} P_u(x, y) = 1$. □

Lemma 3 (Maximum-Norm Contraction) *For all J and $\bar{J}$, we have $\|TJ - T\bar{J}\|_\infty \leq \alpha \|J - \bar{J}\|_\infty$.*

Proof First, we have

$$\begin{aligned} J &= \bar{J} + J - \bar{J} \\ &\leq \bar{J} + \|J - \bar{J}\|_\infty e. \end{aligned}$$

. We now have

$$\begin{aligned} TJ - T\bar{J} &\leq T(\bar{J} + \|J - \bar{J}\|_\infty e) - T\bar{J} \\ &= T\bar{J} + \alpha\|J - \bar{J}\|_\infty e - T\bar{J} \\ &= \alpha\|J - \bar{J}\|_\infty e. \end{aligned}$$

The first inequality follows from monotonicity and the second from the offset property of T. Since J and $\bar{J}$ are arbitrary, we conclude by the same reasoning that $T\bar{J} - TJ \leq \alpha\|J - \bar{J}\|_\infty e$. The lemma follows. □