Math 525: Lecture 21

April 03, 2018

So far, we have worked primarily with (stationary) Markov chains whose transition matrices are "constant". In this lecture, we explore the following question: what if we could "control" the transition matrix? In this context, we will have a transition matrix $P(\pi)$ that depends on some quantity π which we, the "controller", get to choose.

1 Markov decision processes

For this lecture, our setting is as follows:

- $S = \{1, \ldots, m\}$ is a finite state space.
- To each state i in S is associated a nonempty countable set $\mathcal{A}_i$ which we can intuitively think of as all the "actions" available at state i.

Definition 1.1. A *stationary policy* π_0 is a function whose domain is S and which satisfies $\pi_0(i) \in \mathcal{A}_i$ for all i. The set of all stationary policies is denoted Π_0

Definition 1.2. A *randomized policy* $(\pi_n)_{n\geq 0}$ is a sequence in which each $\pi_n(i)$ is a random variable satisfying

- $\pi_n(i)$ takes values in $\mathcal{A}_i$ a.s. and
- $\{\pi_n(i) = a\} \in \mathcal{F}_n$ for each a in $\mathcal{A}_i$.

The set of all randomized policies is denoted Π.

For each state i in S and action a in $\mathcal{A}_i$, let $p_i(a)$ denote a nonnegative column vector satisfying $p_i(a)^\intercal e = 1$. Given a randomized policy π, let $(X_n^\pi)_{n\geq 0}$ denote a Markov chain satisfying

$$\mathbb{P}(X_{n+1}^\pi = j \mid X_n^\pi = i) = p_i(\pi_n(i))^\intercal e_j.$$

That is, the transition matrix at time n is

$$P(\pi_n) = \begin{pmatrix} p_1(\pi_n(1))^\intercal \\ p_2(\pi_n(2))^\intercal \\ \vdots \\ p_m(\pi_n(m))^\intercal \end{pmatrix}.$$

Now, let $c : S \to \mathbb{R}$, $0 \le d < 1$, and

$$J(i,\pi) = \mathbb{E}\left[\sum_{n\ge 0} d^n c(X_n^\pi) \middle| X_0^\pi = i\right]. \tag{1}$$

We can think of

- $c(X_n^\pi)$ as the cost incurred at time n and
- d^n as a discount factor which attempts to capture the fact that costs incurred in the "future" are not as bad as costs incurred "today".

Our objective is to pick π so as to minimize $J(i,\pi)$. That is, we are interested in the quantity

$$\boxed{v(i) = \inf_{\pi\in\Pi} J(i,\pi)} \tag{2}$$

We call (2) a *Markov decision process* (MDP).

Proposition 1.3. *$v(i)$ is bounded for each i.*

Proof. This is a trivial consequence of the discount factor being strictly less than one:

$$|v(i)| \le \sum_{n\ge 0} d^n \max_j |c(j)| = \frac{1}{1-d} \max_j |c(j)|. \qquad \square$$

Remark 1.4. We glossed over defining $\mathcal{F}_n$ earlier, so we return to that now. Given a particular randomized policy π, we define

$$\mathcal{F}_n = \sigma(X_0^\pi, \ldots, X_n^\pi).$$

This definition seems, at first glance, circular... it seems as though π_n depends on $\mathcal{F}_n$ and vice versa. However, if we look a bit closer at the definition of X_n^π, we note that it only depends on the $\pi_0, \ldots, \pi_{n-1}$. In light of this, we can write $X_n^\pi \equiv X_n^{\pi_0,\ldots,\pi_{n-1}}$. Therefore,

$$\mathcal{F}_n = \sigma(X_0, X_1^{\pi_0}, X_2^{\pi_0,\pi_1}, \ldots, X_n^{\pi_0,\pi_1,\ldots,\pi_{n-1}})$$

and hence π is well-defined.

2 Dynamic programming

By the Markov property,

$$\begin{aligned} J(i,\pi) = \mathbb{E}^i\left[c(X_0^\pi) + \sum_{n\ge 1} d^n c(X_n^\pi)\right] &= c(i) + d\mathbb{E}^i\left[\sum_{n\ge 0} d^n c(X_{n+1}^\pi)\right] \\ &= c(i) + d\mathbb{E}^i\left[J(X_1^\pi, (\pi_n)_{n\ge 1})\right] \ge c(i) + d\sum_j (P(\pi_0))_{ij} v(j) \end{aligned}$$

where π_0 is some stationary policy. Taking infimums of both sides of this equality,

$$v(i) \geq \inf_{\pi_0 \in \Pi_0} \left\{ c(i) + d \sum_j (P(\pi_0))_{ij} v(j) \right\}. \tag{3}$$

Now, fix $\epsilon > 0$. For each i, let $\pi^i = (\pi^i_n)_{n \geq 0}$ be a randomized policy which satisifes

$$v(i) \geq J(i, \pi^i) + \epsilon.$$

Let π_0 be an arbitrary stationary policy. Define a new randomized policy $\pi^\epsilon = (\pi^\epsilon_n)_{n \geq 0}$ by

$$\pi^\epsilon_n = \begin{cases} \pi_0 & \text{if } n = 0 \\ \sum_i \mathbf{1}_{\{X_1^{\pi_0} = i\}} \pi^i_{n-1} & \text{if } n > 0. \end{cases}$$

Note that

$$v(i) \leq J(i, \pi^\epsilon) = c(i) + d \sum_j (P(\pi_0))_{ij} J(j, \pi^j) \leq c(i) + d \sum_j (P(\pi_0))_{ij} v(j) - \epsilon.$$

Now, take infimums of both sides to get

$$v(i) \leq \inf_{\pi_0 \in \Pi_0} \left\{ c(i) + d \sum_j (P(\pi_0))_{ij} v(j) \right\} - \epsilon. \tag{4}$$

We can take $\epsilon \downarrow 0$ and combine (3) and (4) to arrive at

$$v(i) = \inf_{\pi_0 \in \Pi_0} \left\{ c(i) + d \sum_j (P(\pi_0))_{ij} v(j) \right\}. \tag{5}$$

The implications of this are amazing! We started out with an objective function (1) that was daunting: minimizing it would require picking a stationary policy for each time n. However, we were able to use the Markov property to reduce this to a "local" problem that only involves minimizing over all stationary policies π_0. In fact, we can simplify (5) even further. First, we need some notation:

$$\text{for } \{y_\alpha\}_\alpha \in \mathbb{R}^n, \ \inf_\alpha y_\alpha \text{ is the vector with entries } \inf_\alpha (y_\alpha)_i.$$

Theorem 2.1 (Dynamic programming)**.** *Let* $v = (v(1), \ldots, v(m))^\intercal$ *and* $c = (c(1), \ldots, c(m))^\intercal$ *where* $v(i)$ *is the quantity defined by (2). Then,*

$$\boxed{\sup_{\pi_0 \in \Pi_0} \{(I - dP(\pi_0))\, v - c\} = 0}$$

Proof. We can rewrite (5) as

$$v = \inf_{\pi_0 \in \Pi} \{c + dP(\pi_0) v\}. \tag{6}$$

Moving some terms around, we obtain the desired result. □

In fact, the situation is much more general than we have let on. We can allow for more general discount factors and costs:

$$J(i, \pi) = \mathbb{E}\left[\sum_{n \geq 0} d(\pi_n, X_n^\pi)^n c(\pi_n, X_n^\pi) \middle| X_0^\pi = i\right].$$

However, in this case, it is no longer necessarily the case that $v(\cdot)$ is bounded. When it is, the corresponding dynamic programming equation is

$$\sup_{\pi_0 \in \Pi_0} \{(I - D(\pi_0) P(\pi_0))\, v - c(\pi_0)\} = 0 \tag{7}$$

where $D(\pi_0) = \operatorname{diag}(d(\pi_0(1), 1), \ldots, d(\pi_0(m), m))$ and $c(\pi_0) = (c(\pi_0(1), 1), \ldots, c(\pi_0(m), m))^\intercal$.

In light of this, the remainder of this lecture is focused on (7). In particular, we would like to know if an arbitrary vector v satisfies (7), is it necessarily equal to the MDP (2)? Moreover, can we use (7) to compute the MDP?