6.262: Discrete Stochastic Processes 3/2/11

Lecture 9: Markov rewards and dynamic prog.

Outline:

- Review plus of eigenvalues and eigenvectors
- Rewards for Markov chains
- Expected first-passage-times
- Aggregate rewards with a final reward
- Dynamic programming
- Dynamic programming algorithm

The determinant of an M by M matrix is given by

$$\det A = \sum_{\mu} \pm \prod_{i=1}^{\mathsf{M}} A_{i,\mu(i)} \qquad (1)$$

where the sum is over all permutations μ of the integers $1, \dots, \mathsf{M}$. If $[A_{\mathcal{T}}]$ is t by t, with

$$[A] = \left[\begin{array}{c|c} [A_{\mathcal{T}}] & [A_{\mathcal{TR}}] \\ \hline [0] & [A_{\mathcal{R}}] \end{array}\right] \qquad \det[A] = \det[A_{\mathcal{T}}]\,\det[A_{\mathcal{R}}]$$

The reason for this is that for the product in (1) to be nonzero, $\mu(i) > t$ whenever $i > t$, and thus $\mu(i) \le t$ when $i \le t$. Thus the permutations can be factored into those over 1 to t and those over $t{+}1$ to M.

$$\det[P - \lambda I] = \det[P_{\mathcal{T}} - \lambda I_t]\,\det[P_{\mathcal{R}} - \lambda I_r]$$

$$\det[P - \lambda I] = \det[P_{\mathcal{T}} - \lambda I_t]\ \det[P_{\mathcal{R}} - \lambda I_r]$$

The eigenvalues of $[P]$ are the t eigenvalues of $[P_{\mathcal{T}}]$ and the r eigenvalues of $[P_{\mathcal{R}}]$.

If $\vec{\pi}$ is a left eigenvector of $[P_{\mathcal{R}}]$, then $(0, \dots, 0, \pi_1, \dots, \pi_r)$ is a left eigenvector of $[P]$, i.e.,

$$(\,\vec{0} \mid \vec{\pi}\,) \left[\begin{array}{c|c} [P_{\mathcal{T}}] & [P_{\mathcal{TR}}] \\ \hline [0] & [P_{\mathcal{R}}] \end{array}\right] = \lambda\,(\,\vec{0} \mid \vec{\pi}\,)$$

The left eigenvectors of $[P_{\mathcal{T}}]$ are more complicated but not very interesting.

Next, assume

$$[P] = \left[\begin{array}{c|c|c} [P_{\mathcal{T}}] & [P_{\mathcal{TR}}] & [P_{\mathcal{TR}'}] \\ \hline [0] & [P_{\mathcal{R}}] & [0] \\ \hline [0] & [0] & [P_{\mathcal{R}'}] \end{array}\right]$$

In the same way as before,

$$\det[P - \lambda I] = \det[P_{\mathcal{T}} - \lambda I_t]\,\det[P_{\mathcal{R}} - \lambda I_r]\,\det[P_{\mathcal{R}'} - \lambda I_{r'}]$$

The eigenvalues of $[P]$ are comprised of the t from $[P_{\mathcal{T}}]$, the r from $[P_{\mathcal{R}}]$, and the r' from $[P_{\mathcal{R}'}]$.

If $\vec{\pi}$ is a left eigenvector of $[P_{\mathcal{R}}]$, then $(\vec{0}, \vec{\pi}, \vec{0})$ is a left eigenvector of $[P]$. If $\vec{\pi}$ is a left eigenvector of $[P_{\mathcal{R}'}]$, then $(\vec{0}, \vec{0}, \vec{\pi})$ is a left eigenvector of $[P]$.

Rewards for Markov chains

Suppose that each state i of a Markov chain is associated with a given reward, r_i.

Letting the rv X_n be the state at time n, the (random) reward at time n is the rv $R(X_n)$ that maps $X_n = i$ into r_i for each i.

We will be interested only in expected rewards, so that, for example, the expected reward at time n, given that $X_0 = i$ is $\mathsf{E}[R(X_n)|X_0 = i] = \sum_j r_j P_{ij}^n$.

The expected aggregate reward over the n steps from m to $m+n-1$, conditional on $X_m = i$ is then

$$\begin{aligned} v_i(n) &= \mathsf{E}\Big[R(X_m) + \cdots + R(X_{m+n-1})|X_m = i\Big] \\ &= r_i + \sum_j P_{ij} r_j + \cdots + \sum_j P_{ij}^{n-1} r_j \end{aligned}$$

$$v_i(n) = r_i + \sum_j P_{ij} r_j + \cdots + \sum_j P_{ij}^{n-1} r_j$$

If the Markov chain is an ergodic unichain, then successive terms of this expression tend to a steady state gain per step,

$$g = \sum_j \pi_j r_j,$$

which is independent of the starting state. Thus $v_i(n)$ can be viewed as a transient in i plus ng.

The transient is important, and is particularly important if $g = 0$.

Expected first-passage-time

Suppose, for some arbitrary unichain, that we want to find the expected number of steps, starting from a given state i until some given recurrent state, say 1, is first entered. Assume $i \neq 1$.

This can be viewed as a reward problem by assigning one unit of reward to each successive state until state 1 is entered.

Modify the Markov chain by changing the transition probabilities from state 1 to $P_{11} = 1$. We set $r_1 = 0$, so the reward stops when state 1 is entered.

For each sample path starting from state $i \neq 1$, the probability of the initial segment until 1 is entered is unchanged, so the expected first-passage-time is unchanged.

The modified Markov chain is now an ergodic unichain with a single recurrent state, i.e., state 1 is a trapping state.

Let $r_i = 1$ for $i \neq 1$ and let $r_1 = 0$.

Thus if state 1 is first entered at time ℓ, then the aggregate reward, from 0 to n, is ℓ for all $n \geq \ell$.

The expected first passage time, starting in state i, is $v_i = \lim_{n\to\infty} v_i(n)$.

There is a sneaky way to calculate this for all i.

For each $i \neq 1$, assume that $X_0 = i$. There is then a unit reward at time 0. In addition, given that $X_1 = j$, the remaining expected reward is v_j. Thus $v_i = 1 + \sum_j P_{ij} v_j$ for $i \neq 1$, with $v_1 = 0$.

The expected first-passage-time to state 1 from state $i \neq 1$ is then

$$v_i = 1 + \sum_j P_{ij} v_j \qquad \text{with } v_1 = 0$$

This can be expressed in vector form as

$$\vec{v} = \vec{r} + [P]\vec{v} \qquad \text{where } \vec{r} = (0, 1, 1, \dots, 1),$$

and $v_1 = 0$ and $P_{11} = 1$.

Note that if $\vec{v}$ satisfies $\vec{v} = \vec{r} + [P]\vec{v}$, then $\vec{v} + \alpha\vec{e}$ also satisfies it, so that $v_1 = 0$ is necessary to resolve the ambiguity.

Also, since $[P]$ has 1 as a simple eigenvalue, this equation has a unique solution with $v_1 = 0$.

Aggregate rewards with a final reward

There are many situations in which we are interested in the aggregate reward over n steps, say time m to $m+n-1$, followed by a special final reward u_j for $X_{m+n} = j$.

The flexibility of assigning such a final reward will be particularly valuable in dynamic programming.

The aggregate expected reward, including this final reward, is then

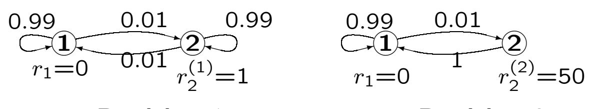
$$v_i(n, \vec{u}) = r_i + \sum_j P_{ij} r_j + \cdots + \sum_j P_{ij}^{n-1} r_j + \sum_j P_{ij}^{n} u_j$$

In vector form,

$$\vec{v}(n, \vec{u}) = \vec{r} + [P]\vec{r} + \cdots + [P^{n-1}]\vec{r} + [P^n]\vec{u}$$

Dynamic programming

Consider a discrete-time situation with a finite set of states, $1, 2, \dots, \mathsf{M}$, where at each time ℓ, a decision maker can observe the state, say $X_\ell = j$ and choose one of a finite set of alternatives. Each alternative k consists of a current reward $r_j^{(k)}$ and a set of of transition probabilities $\{P_{j\ell}^{(k)};\ 1 \le \ell \le \mathsf{M}\}$ for going to the next state.



Decision 1 Decision 2

For this example, decision 2 seeks instant gratification, whereas decision 1 seeks long term gratification.

Assume that this process of random transitions combined with decisions based on the current state starts at time m in some given state and contines until time $m + n - 1$.

After the nth decision, made at time $m+n-1$, there is a final transition based on that decision.

At time $m + n$, there is a final reward, $(u_1, \dots, u_{\mathsf{M}})$ based on the final state.

The objective of dynamic programming is both to determine the optimal decision at each time and to determine the expected reward for each starting state and for each number n of steps.

As one might suspect, it is best to start with a single step ($n = 1$) and then proceed to successively more steps.

Surprisingly, this is best thought of as starting at the end and working back to the beginning.

The algorithm to follow is due to Richard Bellman. Its simplicity is a good example of looking at an important problem at the right time. Given the formulation, anyone could develop the algorithm.

The dynamic programming algorithm

As suggested, we first consider the optimal expected aggregate reward over a single time period.

That is, starting at an arbitrary time m in a given state i, we make a decision, say decision k at time m.

This provides a reward $r_i^{(k)}$ at time m. Then the selected transition probabilities, P_{ij}^k lead to a final expected reward $\sum_j u_j P_{ij}^k$ at time $m+1$.

The decision k is chosen to maximize the corresponding aggregate reward, i.e.,

$$v_i^*(1) = \max_k \left(r_i(k) + \sum_j u_j P_{ij}^k \right)$$

Next consider $v_i^*(2, \vec{u})$, i.e., the maximal expected aggregate reward starting at $X_m = i$ with decisions made at times m and $m+1$ and a final reward at time $m+2$.

The key to dynamic programming is that an optimal decision at time $m+1$ can be selected based only on the state j at time $m+1$.

This decision (given $X_{m+1} = j$) is optimal independent of the decision at time m.

That is, whatever decision is made at time m, the maximal expected reward at times $m+1$ and $m+2$, given $X_{m+1} = j$, is $\max_k \left(r_j^{(k)} + \sum_\ell P_{j\ell}^{(k)} u_\ell \right)$. This is $v_j^*(1, \vec{u})$, as just found.

We have just seen that

$$v_j^*(1, \vec{u}) = \max_k \left(r_j^{(k)} + \sum_\ell P_{j\ell}^{(k)} u_\ell \right)$$

is the maximum expected aggregate reward over times $m+1$ and $m+2$, conditional $X_{m+1} = j$.

Thus the maximum expected aggregate reward over $m, m+1, m+2$, conditional on $X_m = i$, is

$$v_i^*(2, \vec{u}) = \max_k \left(r_i^{(k)} + \sum_j P_{ij}^{(k)} v_j^*(1, \vec{u}) \right)$$

This same procedure can be used to find the optimal policy and optimal expected reward for $n = 3$,

$$v_i^*(3, \vec{u}) = \max_k \left(r_i^{(k)} + \sum_j P_{ij}^{(k)} v_j^*(2, \vec{u}) \right)$$

This solution shows how to choose the optimal decision at time m and finds the optimal aggregate expected reward, but is based on first finding the optimal solution for $n = 2$. In general,

$$v_i^*(n, \vec{u}) = \max_k \left(r_i^{(k)} + \sum_j P_{ij}^{(k)} v_j^*(n-1, \vec{u}) \right)$$

For any given n, then, the algorithm calculates $v_j^*(m, \vec{u})$ for all states j and all $m \le n$, starting at $m = 1$.

This is the dynamic programming algorithm.