Math 416 Lecture 28

Infinite Horizon discounted reward problem

From the last lecture: The value function of policy $u$ for the infinite horizon problem with discount factor $\alpha$ and initial state $i$ is

$$W(i, u) = E[\sum_{n=0}^{\infty} \alpha^n r(X_n, u(X_n)) | X_0 = i].$$

The optimal value function is $W(i) = \max_u W(i, u)$. $u^*$ is an optimal policy iff each initial state $i$, $W(i, u^*) = W(i)$.

Recurrent equation for $W(i, u)$.

$$W(i, u) = r(i, u(i)) + \alpha \sum_j T_{u(i)}(j) W(j, u).$$

Dynamic programming equation for $W(i)$.

$$W(i) = \max_{a \in A} \{ r(i, a) + \alpha \sum_j T_{a}(j) W(j) \}.$$ Let $u^*$ be the policy such that $u^*(i) = a$ which gives the max value for $W(i)$. Then $u^*$ is an optimal and $W(i, u^*) = W(i)$.

Approximation procedure for $W(i)$: Suppose the states are $S = \{ i_0, i_1, i_2, \ldots \}$. Let $\overline{w}$ be the column vector of values $W(i)$:

$$\overline{w} = [w(0), w(1), w(2), \ldots]^T = [W(i_0), W(i_1), W(i_2), \ldots]^T$$

(superscript $T$ means transpose). Let $\overline{w}_0 = [0, 0, 0, \ldots]^T$. Define $\overline{w}_{n+1}$ by:

$$\overline{w}_{n+1} = \max_{a \in A} \{ r(i, a) + \alpha \sum_j T_{a}(j) \overline{w}_{n}(j) \}.$$ This sequence converges to $W(i)$ (Lecture 18).

$$\overline{w}_{n+1}$$ determines a maximizing action $u(i) = a$. Continue the approximation procedure until this action stabilizes on the same action from one approximation to the next, i.e., for all states $i$, the policy $u(i)$ for $\overline{w}_{n}(i)$ is the same as that for $\overline{w}_{n+1}(i)$. Then check to see if the policy $u(i)$ is an optimal policy. $u(i)$ is optimal iff $W(i, u) = W(i)$ iff $W(i, u)$ satisfies the dynamic programming equation for $W(i)$ iff $W(i, u) = \max_{a \in A} \{ r(i, a) + \alpha \sum_j T_{a}(j) W(j, u) \}$. If not, continue the approximation procedure. Once the dynamic programming equation does hold for all states $i$, then $u(i)$ is the optimal policy $u^*$ and $W(i) = W(i, u)$ is the exact value of $W(i)$.

Now compute $W(i, u)$ from $u$ via matrix algebra.

Suppose the state space is $S = \{ 0, 1, 2, \ldots \}$.

Lemma. Suppose $u$ is the action policy.

Let $T_u(i, j)$ be the transition matrix for going from state $i$ to state $j$ when policy $u$ is followed. Let $r_u = [r(0, u(0)), r(1, u(1)), r(2, u(2)), \ldots]^T$.

Let $X = \{ x_0, x_1, x_2, \ldots \}^T = [W(0, u), W(1, u), W(2, u), \ldots]^T$.

Then $X$ is the solution to the matrix equation

$$(I - \alpha T_u)X = r_u.$$ Proof. Since $X(i) = W(i, u)$, the equation

$$W(i, u) = r(i, u(i)) + \alpha \sum_j T_{u(i)}(j) W(j, u)$$ for all $i$ becomes,

$$X(i) = r(i, u(i)) + \alpha \sum_j T_{u(i)}(j) X(j)$$ and hence

$$[X(0), X(1), X(2), \ldots]^T = \{ r(0, u(0)), r(1, u(1)), r(2, u(2)), \ldots \}^T.$$ Note that $\sum_j T_{u(i)}(j) X(j)$ is the matrix product $T_u \cdot X$.

Hence $X = r_u + \alpha T_u \cdot X$. Hence $(I - \alpha T_u)X = r_u$.

A Markov decision problem has two states $i=1, 2$ and two actions $a = c, d$. The discount is $\alpha = 0.9$. The rewards $r(i, a)$ are $r(1, c) = 5, r(2, c) = 2, r(1, d) = 4, r(2, d) = 3$.

The transition matrices for actions $a = c, d$ are

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/2</td>
<td>1/2</td>
</tr>
<tr>
<td>2</td>
<td>2/3</td>
<td>1/3</td>
</tr>
</tbody>
</table>

$$W_n(1) = \max_{a \in \{c, d\}} \{ r(i, a) + \alpha \sum_j T_{a}(j) W_n(j) \}.$$ Let $r(i, c) + 0.9 \sum_j T_{c}(j) W_n(j)$, $r(i, d) + 0.9 \sum_j T_{d}(j) W_n(j)$.

$$W_n(1) = \max\{ 5 + 0.9[\frac{1}{2} W_n(1) + \frac{1}{2} W_n(2)], 4 + 0.9[\frac{1}{2} W_n(1) + \frac{1}{2} W_n(2)] \}$$

$$W_n(2) = \max\{ 2 + 0.9[\frac{3}{5} W_n(1) + \frac{2}{5} W_n(2)], 3 + 0.9[\frac{3}{5} W_n(1) + \frac{2}{5} W_n(2)] \}$$

$$W_0(1) = 0, W_0(2) = 0$$

$$W_1(1) = \max\{ 5 + 0.9[0 + 0], 4 + 0.9[0 + 0] \} = \max\{ 5, 4 \} = 5$$

with maximizing action $u(1) = c$.

$$W_1(2) = \max\{ 2 + 0.9[0 + 0], 3 + 0.9[0 + 0] \} = \max\{ 2, 3 \} = 3$$

with maximizing action $u(2) = d$.

$$W_2(1) = \max\{ 5 + 0.9[\frac{3}{5} + \frac{2}{5}], 4 + 0.9[\frac{3}{5} + \frac{2}{5}] \}$$

$$= \max\{ 5 + 0.9[ 4], 4 + 0.9[ \frac{14}{15} ] \} = \max\{ 8.6, 7.15 \} = 8.6$$

with maximizing action $u(1) = c$. (Calculate fractions directly.

$$W_2(2) = \max\{ 2 + 0.9[\frac{3}{5} + \frac{2}{5}], 3 + 0.9[\frac{3}{5} + \frac{2}{5}] \} = \max\{ 2 + 0.9[\frac{14}{15}], 3 + 0.9[\frac{14}{15}] \} = \max\{ 5.9, 6.3 \} = 6.3$$

with maximizing action $u(2) = d$.

For both $W_1$ and $W_2$, $u(1), u(2)$ are the same. Hence we test to see if $u(1) = c, u(2) = d$ is an optimal policy. For this $u$, $T_u = \begin{bmatrix} 1/2 & 1/2 \\ 1/3 & 2/3 \end{bmatrix}$.

The first row of $T_{u(i,j)}$ is from $T_c$ since $u(1) = c$ and the second row is from $T_d$ since $u(2) = d$. $r_u = [r(1, u(1)), r(2, u(2))]^T$ $= [r(1, c), r(2, d)]^T = [5, 3]^T$.

Thus the equation

$$(I - \alpha T_u)X = r_u$$ becomes

$$\begin{bmatrix} 1 - 0.9 \frac{1}{2} & -0.9 \frac{1}{2} \\ -0.9 \frac{1}{2} & 1 - 0.9 \frac{3}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 5 \\ 3 \end{bmatrix}$$

which is the system

$$\begin{bmatrix} \frac{1}{20} x - \frac{9}{20} y = 5 \\ \frac{3}{10} x + \frac{4}{10} y = 3 \end{bmatrix}$$ with solution

$$x = \frac{670}{17} = W(1, u), y = \frac{630}{17} = W(2, u).$$ Now check if $W(i, u)$ satisfies the dynamic programming equation for $W(i)$: $W(i, u) = \max_{a \in A} \{ r(i, a) + \alpha \sum_j T_{a}(i, j) W(j, u) \}$. If so, then $W(i) = W(i, u)$ and $u$ is optimal.

For $i = 1$ we get $W(1, u) = \max_{a \in A} \{ 5 + 0.9[\frac{1}{2} + \frac{1}{2}], 0.9 \frac{14}{15} \frac{17}{17} \} = \{ 7.6, 7.3 \}$ iff true.

Likewise for $i = 2$. 

$$\begin{bmatrix} 1/2 & 1/2 \\ 2/3 & 1/3 \end{bmatrix}$$

$w(1)$

$\begin{bmatrix} 1/4 & 3/4 \\ 1/3 & 2/3 \end{bmatrix}$

$w(2)$

$\begin{bmatrix} 5 \end{bmatrix}$

$\begin{bmatrix} 3 \end{bmatrix}$

$\begin{bmatrix} 8.6 \end{bmatrix}$

$\begin{bmatrix} 7.15 \end{bmatrix}$

$\begin{bmatrix} 8.6 \end{bmatrix}$

$\begin{bmatrix} 7.15 \end{bmatrix}$

$\begin{bmatrix} 5.9 \end{bmatrix}$

$\begin{bmatrix} 6.3 \end{bmatrix}$

$\begin{bmatrix} 6.3 \end{bmatrix}$
Optimal stopping times

Suppose you have invested in a stock. When should you sell it? To make money, you must sell if for more than the amount originally paid. When the stock is selling at a high price, should you sell or wait for an even higher price? For the case of the unpredictable stock market the problem is intractable. But we can answer these questions for Markov chains with known transition probabilities.

Let \( S = \{0, 1, 2, \ldots \} \) be the set of states for a Markov chain; one state for each node of its diagram.

Let \( T(i,j) \) be the probability of going from \( i \) to \( j \) assuming you choose to continue.

Let \( r(i) \) be the reward to state \( i \) (the chain has a node for each state) (this reward is collected when you stop at that state).

At each state \( i \), we chose an action \( a \in \{s, c\} \).

Choose \( a = s \) if you wish to stop and collect the reward assigned to the current node. Thus \( r(i, s) = r(i) \).

Choose action \( a = c \) if you wish to continue. In this case you don’t collect the reward and so \( r(i, c) = 0 \).

You continue from state to state according to the transition probabilities until you stop at a state and collect it’s reward.

A solution to a stopping problem is a policy \( u(i) \) which determines, for each state \( i \), whether you should (s) stop (\( u(i) = s \)) and collect the reward \( r(i, s) \) or (c) continue (\( u(i) = c \)).

Let \( W(i) \) be the expected reward if you start at \( i \) and act optimally. Clearly \( W(i) \geq r(i) \). \( W(i) = r(i) \) if you stop at \( i \) instead of continuing on to other states.

Here are some examples.

For each state \( i \) we have to decide if we should stop and collect the reward, or continue. In the picture, the bracketed numbers \( [r(i)] \) are the rewards.

Put the correct action \( u(i) \) in the space between ( )’s.

At the bottom of each node list the expected reward \( W(i) \) if you start at \( i \) and act optimally.

Convention: unlabeled arrows have probability 1.

You must stop sometime, infinite loops are not allowed. Except for the first picture with states 1, 2, 3, state numbers are omitted.