Decision Analysis B

Dr Radu Tatar

Finite-stage dynamic programming

This is an alternative to using diagram trees.

The problem we are concerned involves a process $\{x_m\}_{m\geq 0}$, whose state is observed at the beginning of each discrete time period $m \geq 0$.

Definition. Let X_n be a discrete time random process. We say X_n is a discrete time Markov chain if the probability of any particular future behavior of the process, when its present state is known exactly, is not altered by additional knowledge concerning its past behavior.

After the observation of the state x_m at time m, an action a_m must be chosen.

Based on the state $x_m = x$, the action $a_m = a$, and possibly the current stage (or say the current time) m, an (expected one-step) reward $\boxed{R_m(x,a)}$ is earned between stage m and stage $m+1$, and the probability distribution for the next state x_{m+1} of the process is determined, denoted as

$$\boxed{P(x_{m+1} = i_{m+1}|x_m = x, a_m = a) = p_m(i_{m+1}|x,a).}$$

It is assumed that we deal with a Markov process which satisfies

$$P(x_{m+1} = i_{m+1}|x_m = x, a_m = a, x_{m-1}, a_{m-1}, \dots, x_0, a_0) = P(x_{m+1} = i_{m+1}|x_m = x, a_m = a).$$

A policy specifies how to select actions at each stage (or step, time) given the current state and current time. So it is given by a sequence of functions $\pi = \{\pi_0, \pi_1, \dots\}$: $\pi_m(x)$ represents the action that should be used when $x_m = x$.

The problem of interest is to choose a policy that maximizes the expected value of the sum of the rewards earned over a given finite time horizon of length N, i.e., the time horizon is $\{0, 1, \dots, N-1\}$. This can be formulated as:

$$W_N(x_0, \pi) := E^{\pi}_{x_0}\left[\sum_{t=0}^{N-1} R_t(x_t, a_t)\right] \to \max_{\pi}.$$

Of course, for N-stage problems, we only care about the first N components in a policy π.

A policy π^* is called optimal for the N-stage problem if

$$\boxed{W_N(x_0, \pi^*) = \max_{\pi} W_N(x_0, \pi) =: V_N(x_0)}$$

for all the possible values of x_0.

The N-stage problem can be solved recursively as follows:

$$\boxed{V_N(x) = \max_a \{R_0(x,a) + E[V_{N-1}(x_1)|x_0 = x, a_0 = a]\},}$$

where the maximum is taken over the set of admissible actions at time 0, with the state $x_0 = x$. R_0 is the reward obtained after the first step based on initial state x_0 and initial decission a_0. Here $V_N(x)$ is the optimal expected value of the total rewards when there are N stages to go, and the state at the current time is x. (The current time is 0 if there are N stages to go.)

Similarly, $V_{N-1}(x)$ is the optimal expected value of the total rewards when there are $N-1$ stages to go, and the state at the current time is x. (The current time is 1 if there are N stages to go.)

The maximizer is recorded as $\alpha^*_N(x)$ (if there are multiple maximizers, take any one). This gives a function α^*_N if we consider all possible values of the state x.

The next step is:

$$\boxed{V_{N-1}(x) = \max_a \{R_1(x,a) + E[V_{N-2}(x_2)|x_1 = x, a_1 = a]\},}$$

where the maximum is taken over the set of admissible actions at time 1, with the state $x_1 = x$. The maximizer is recorded as $\alpha^*_{N-1}(x)$.

So on so forth.

The terminal condition is

$$\boxed{V_1(x) = \max_a R_{N-1}(x,a),}$$

where the maximum is taken over the set of admissible actions at time 0, with the state $x_{N-1} = x$.

In practice, of course, one should start from the terminal condition, and obtain successively V_1, V_2, ..., until we obtain the desired function V_N. This is called the backward iteration, or the dynamic programming algorithm. An optimal policy is given by the sequence of functions $\boxed{\{\alpha^*_N, \alpha^*_{N-1}, \dots, \alpha^*_1\}}$.

The interpretation of this policy is as follows: if the current time is m, and $x_m = x$, then use action $\alpha^*_{N-m}(x)$.

Remark 1 *(a) The dynamic programming algorithm can be written as*

$$\boxed{V_m(x) = \max_a \{R_{N-m}(x,a) + E[V_{m-1}(x_{N-m+1})|x_{N-m} = x, a_{N-m} = a]\},\ m = 1, 2, \dots, N}$$

with

$$\boxed{V_0(x) \equiv 0}.$$

The first equations are called the optimality equation, also known as the Bellman equation, or dynamic programming equation.

(b) In many problems, R_m does not depend on m, and in that case we write it as R. The same applies to the notation $p(\cdot|x,a)$. If α^*_m does not depend on m, then the policy is called stationary or simple.

Example 0.0.0.1

Flight Centre is a global travel agency with business covering most of the attractions in the world. Now they are planning a 3-day event to sell a recent project "RockiesRock", which is a 7-day tour in Banff, Canada. For each day, the volume of sales could be high or low and the daily profits made are £80, 000 and £40, 000 respectively. In the base case (i.e. no promotion), a day with high sales will be followed by a day with high sales with probability 0.60; whereas a day with low sales will become high sales on the next day with probability 0.30. Flight Centre considers also two ways of promotions in the afternoon, after it becomes clear whether the sales are high or low:
(1) Offer a 20%-off discount to the next 10 customers;
(2) Place a TV advertisement on BBC.

Option 1 will increase the probability of getting two consecutive high sales days to 0.70 and there will be a 60% chance for a low sales day to change to a high sales day. However, the profit will be reduced by 10%. Option 2 costs a £5, 000 fee charged by BBC but the next day sales will be high with probability 0.70.

Apply the dynamic programming method to generate the optimal solution for Flight Centre for this 3-day event.

<u>Solution.</u> The state of the process corresponds to the volume of sales, and we put the state space as $S = \{H, L\}$; the admissible action space at any state $s \in S$ is $A = \{0, 1, 2\}$, where 0 means "no promotion" and 1(2) is the corresponding way of promotion.

Table of the (expected one-step) rewards $R(s, a)$ (in £1000):

a / s	0	1	2
H	80	72	75
L	40	36	35

Transition probabilities $p(y|s, a)$ are given by the following tables

$a = 0$:

From To y / s	H	L
H	0.6	0.4
L	0.3	0.7

$a = 1$:

From To y / s	H	L
H	0.7	0.3
L	0.6	0.4

$a = 2$:

From To y / s	H	L
H	0.7	0.3
L	0.7	0.3

The number of periods is $N = 3$.

We will solve the optimality equation

$$V_m(s) = \max_{a \in A} \left\{ R(s, a) + \sum_{y \in S} p(y|s, a) V_{m-1}(y) \right\}, \ m = 1, 2, 3$$

with

$$V_0(s) \equiv 0.$$

Computing $V_1(s) = \max_{a\in\{0,1,2\}} R(s,a)$ corresponding to Day 2 (the state is the volume of sales in the morning):

a	0	1	2	Optimal	Maximal
s				decision	reward $V_1(s)$
H	80	72	75	$\alpha_1^*(H) = 0$	80
L	40	36	35	$\alpha_1^*(L) = 0$	40

Computing $V_2(s) = \max_{a\in\{0,1,2\}}\{R(s,a) + \sum_{y\in S} p(y|s,a)V_1(y)\}$ corresponding to Day 1 (the state is the volume of sales in the morning):

a s	0	1	2	Optimal decision and maximal reward $V_2(s)$
H	$R(H,0)+$ $p(H\|H,0)V_1(H)$ $+p(L\|H,0)V_1(L)$ $= 80 + 0.6 \cdot 80$ $+0.4 \cdot 40 = 144$	$R(H,1)+$ $p(H\|H,1)V_1(H)$ $+p(L\|H,1)V_1(L)$ $= 72 + 0.7 \cdot 80$ $+0.3 \cdot 40 = 140$	$R(H,2)+$ $p(H\|H,2)V_1(H)$ $+p(L\|H,2)V_1(L)$ $= 75 + 0.7 \cdot 80$ $+0.3 \cdot 40 = 143$	$\alpha_2^*(H) = 0$ 144
L	$R(L,0)+$ $p(H\|L,0)V_1(H)$ $+p(L\|L,0)V_1(L)$ $= 40 + 0.3 \cdot 80$ $+0.7 \cdot 40 = 92$	$R(L,1)+$ $p(H\|L,1)V_1(H)$ $+p(L\|L,1)V_1(L)$ $= 36 + 0.6 \cdot 80$ $+0.4 \cdot 40 = 100$	$R(L,2)+$ $p(H\|L,2)V_1(H)$ $+p(L\|L,2)V_1(L)$ $= 35 + 0.7 \cdot 80$ $+0.3 \cdot 40 = 103$	$\alpha_2^*(L) = 2$ 103

Computing $V_3(s) = \max_{a\in\{0,1,2\}}\{R(s,a) + \sum_{y\in S} p(y|s,a)V_2(y)\}$ corresponding to Day 0 (the state is the volume of sales in the morning):

a s	0	1	2	Optimal decision and maximal reward $V_3(s)$
H	$R(H,0)+$ $p(H\|H,0)V_2(H)$ $+p(L\|H,0)V_2(L)$ $= 80 + 0.6 \cdot 144+$ $0.4 \cdot 103 = 207.6$	$R(H,1)+$ $p(H\|H,1)V_2(H)$ $+p(L\|H,1)V_2(L)$ $= 72 + 0.7 \cdot 144+$ $0.3 \cdot 103 = 203.7$	$R(H,2)+$ $p(H\|H,2)V_2(H)$ $+p(L\|H,2)V_2(L)$ $= 75 + 0.7 \cdot 144+$ $0.3 \cdot 103 = 206.7$	$\alpha_3^*(H) = 0$ 207.6
L	$R(L,0)+$ $p(H\|L,0)V_2(H)$ $+p(L\|L,0)V_2(L)$ $= 40 + 0.3 \cdot 144+$ $0.7 \cdot 103 = 155.3$	$R(L,1)+$ $p(H\|L,1)V_2(H)$ $+p(L\|L,1)V_2(L)$ $= 36 + 0.6 \cdot 144+$ $0.4 \cdot 103 = 163.6$	$R(L,2)+$ $p(H\|L,2)V_2(H)$ $+p(L\|L,2)V_2(L)$ $= 35 + 0.7 \cdot 144+$ $0.3 \cdot 103 = 166.7$	$\alpha_3^*(L) = 2$ 166.7

The conclusion is as follows. On the first and second day, place a TV advertisement only if the sales are low. (Otherwise, no promotion.) On the third day, no promotion is needed. As a result, the total expected reward from this 3-day event equals £207,600 (£166,700) if the sales in the first day were high (low).

Example 0.0.0.2 (Gambling model)

At each play of the game, a gambler can bet any nonnegative amount up to his present fortune and will either win or lose that amount with probability $p \in (0,1)$ and $1-p$, respectively. If he wins the bet, he receives the amount he betted as a net profit (i.e., his fortune after a win becomes his fortunate before the bet plus the amount he betted). If he loses the bet, he loses the amount he betted.The gambler is allowed to make n bets, and his objective is to maximize the expectation of the logarithm of his final fortune. What strategy achieves this end when $p > 1/2$? What is the optimal value of this problem if the gambler has initial fortune $x > 0$?

Solution. We formulate this as an n-stage DP problem. The state x_m at time $0 \leq m \leq n-1$ is the fortune at the hand of the gambler, and the action $a_m \in [0,1]$ is the proportion of the fortune in hand to put in bet. Then $x_{m+1} = x_m + a_m x_m$ with probability p, and $x_{m+1} = x_m - a_m x_m$ with probability $1-p$. The optimality equation is

$$\begin{aligned} V_1(x) &= \max_{a\in[0,1]} \{p \ln(x(1+a)) + (1-p)\ln(x(1-a))\}; \\ V_{m+1}(x) &= \max_{a\in[0,1]} \{pV_m(x(1+a)) + (1-p)V_m(x(1-a))\}, \ 1 \leq m \leq n-1. \end{aligned}$$

Analyze the optimality equation as follows.

$$\begin{aligned} V_1(x) &= \max_{a\in[0,1]} \{p\ln(1+a) + (1-p)\ln(1-a) + \ln x\} \\ &= \max_{a\in[0,1]} \{p\ln(1+a) + (1-p)\ln(1-a)\} + \ln x \end{aligned}$$

Clearly, $a = 1$ is not a maximizer. So search for a maximizer over $a \in [0,1)$. It is possible to check that $p\ln(1+a) + (1-p)\ln(1-a)$ has nonpositive secondary derivative, and so the maximizer can be found by solving $\frac{d}{da}\{p\ln(1+a) + (1-p)\ln(1-a)\} = 0$, i.e., The first derivative is

$$\frac{p}{1+a} - \frac{1-p}{1-a}$$

The second derivative is

$$-\frac{p}{(1+a)^2} - \frac{1-p}{(1-a)^2}$$

which is always negative as $p > 0, 1-p > 0$ so the function has a maximum.

To find the maximum we set the first derivative to zero $p(1-a)-(1-p)(1+a) = 0$ and so $a = p-(1-p) = 2p-1$. Therefore, we put $\alpha_1^*(x) = 2p-1$ and $V_1(x) = C + \ln x$ with $C = p\ln(2p) + (1-p)\ln(2(1-p))$.

Now

$$\begin{aligned} V_2(x) &= \max_{a\in[0,1]} \{pV_1(x(1+a)) + (1-p)V_1(x(1-a))\} \\ &= \max_{a\in[0,1]} \{p(\ln(x(1+a)) + C) + (1-p)(\ln(x(1-a)) + C)\} \\ &= \max_{a\in[0,1]} \{p\ln(1+a) + (1-p)\ln(1-a)\} + \ln x + C = V_1(x) + C \end{aligned}$$

and $\alpha_2^*(x) = 2p-1$, and an inductive argument shows $V_m(x) = V_1(x) + (m-1)C = \ln x + mC$ and $\alpha_m^*(x) = 2p-1$ for all $m = 1, 2, \ldots, n$.

To sum up, the gambler should always do the following: at the beginning of each stage, if he has x fortune in hand, then he should bet $(2p-1)x$. If at the beginning he has x fortune, then the optimal value of this problem is $\ln x + nC$. □