INDR 450/550

Spring 2022

Lecture 24: Dynamic Programming (2)

May 18, 2022

Fikri Karaesmen

Announcements

- Class Exercise at the end of lecture today. If you are participating online, please upload your document under Course Contents/Class Exercises
- Project short description due tonight
- HW 4 due date May 23 (model reduction, trees, forests etc.)

The Problem

- Introduction to Stochastic Dynamic Programming in the context of Capacity Allocation with Multi-class Demand
- Expected revenue for a given policy
- Optimal policy

Capacity Allocation in Revenue Management

- Q units of inventory are available and they have to be sold before a strict due-date. This is appropriate for airlines, trains, hotels, concerts, theater etc.
- There are multiple classes of customers who are willing to pay different prices
- Service industry has found ways of segmenting such classes
  - Student discounts
  - Early reservation discounts
  - Little changes in service (extra luggage, flexiibility in cancelation etc.)
- This generates the following capacity control problem when a demand from a lower class shows up, should we sell the item or reject the demand to keep the item for a potential future customer that might pay a higher price.

Some formality: stochastic DPs

- State: inventory level x at time t.
- Action $a(x,t) = (a_1, a_2)$ where $a_i \in \{\mathrm{A,R}\}$.
  - Example: a(5,1) = (A,A) -> accept demand from Class 1 and Class 2 if we have 5 items with one period remaining until the deadline.
- Policy: **μ**: A complete set of actions for all x and for all t.
  - Example for all x and for all t accept all demands from class 1 and reject all demands from class 2: $a(x,t) = (A, R)$ for all x and for all t .
- Optimal policy: **μ*** a policy that maximizes the expected profit

Policy Evaluation

- We can now compute $w_1(x)$:

$$w_1(x) = q_1(p_1 + w_0(x-1)) + q_2 w_0(x) + q_3 w_0(x) \text{ for all } x \geq 1$$

This enables us to compute $w_1(x)$ for all x. Then, we can compute:

$$w_2(x) = q_1(p_1 + w_1(x-1)) + q_2 w_1(x) + q_3 w_1(x) \text{ for all } x \geq 1$$

We can therefore recursively compute:

$$w_T(x) = q_1(p_1 + w_{T-1}(x-1)) + q_2 w_{T-1}(x) + q_3 w_{T-1}(x) \text{ for all } x \geq 1$$

Policy Evaluation to Optimization

- We can now therefore compute the expected revenue from any policy.
- But this is not a tool for finding the optimal policy, there are too many policies to evaluate even for this simple example.

- Bellman's principle of dynamic programming is about the following insight: we can find the optimal policy for the entire problem (T periods) by combining the optimal policies for subproblems that find the optimal policies starting from start from t=1,2,..T-1.
  - if you find yourself in state x with t periods remaining it does not matter how you got there, the best that you can do is to maximize the expected reward over the remaining horizon.

Policy Evaluation to Optimization

Let $v_1(x)$ be the maximum expected revenue with one period remaining and x items available , we can write

$$
\begin{aligned}
v_1(x) \quad = \quad & q_1 \max\{(p_1 + v_0(x-1)), v_0(x)\} \\
& + q_2 \max\{(p_2 + v_0(x-1)), v_0(x)\} \\
& + q_3 v_0(x)
\end{aligned}
$$

Recall that $v_0(x) = 0$ for all $x \geq 0$. We can therefore extract the optimal action with 1 period to go: $a(x,1) = (A, A)$. It is optimal to sell to both classes with 1 period to go.

Policy Optimization

- But now we can do the same for $v_2(x)$.
  - For $0 \leq x \leq Q$ We have:

$$
\begin{aligned}
v_2(x) \quad = \quad & q_{12} \max\{p_1 + v_1(x-1), v_1(x)\} + q_{22} \max\{p_2 + v_1(x-1), v_1(x)\} + \ldots \\
& + q_{k2} \max\{p_k + v_1(x-1), v_1(x)\} + q_{02} v_1(x)
\end{aligned}
$$

- Going backwards, for t periods remaining, we have for $0 \leq x \leq Q$:

$$
\begin{aligned}
v_t(x) \quad = \quad & q_{1t} \max\{p_1 + v_{t-1}(x-1), v_{t-1}(x)\} + q_{2t} \max\{p_2 + v_{t-1}(x-1), v_{t-1}(x)\} + \ldots \\
& + q_{kt} \max\{p_k + v_{t-1}(x-1), v_{t-1}(x)\} + q_{0t} v_{t-1}(x)
\end{aligned}
$$

- This can be computed if $v_{t-1}(x)$ has already been computed for all x.

Policy Optimization

- To extract the optimal actions a_1 and a_2 at time t for state x we note that:

$$a_1 = A \text{ if } p_1 + v_{t-1}(x-1) \geq v_{t-1}(x), \text{ and } a_1 = R; \text{otherwise}$$

$$a_2 = A \text{ if } p_2 + v_{t-1}(x-1) \geq v_{t-1}(x), \text{ and } a_2 = R; \text{otherwise}$$

- Note that since $p_1 > p_2$, if it is optimal to sell to class 2 at t and x then it is also optimal to sell to class 1.

We can compute both the expected optimal profit and the corresponding optimal policy from the same recursion.

Policy Optimization

- Equivalently we can write the optimality conditions as:

$$a_1 = A \text{ if } p_1 \geq v_{t-1}(x) - v_{t-1}(x-1), \text{ and } a_1 = R; \text{otherwise}$$

$$a_2 = A \text{ if } p_2 + \geq v_{t-1}(x) - v_{t-1}(x-1), \text{ and } a_2 = R; \text{otherwise}$$

- We note that $\Delta(x) = v_{t-1}(x) - v_{t-1}(x-1)$ is a critical quantity. It corresponds to the marginal value of a seat.
- In the context of capacity allocation $\Delta(x)$ is known as the bid price at time t of seat x.
- It is optimal to sell to a class i customer only if $p_i >$ current bid price.

Policy Optimization

- Obtain $v_t(x)$ by recursion

A	B	C	D	E	F	G	H	I	J	K	L
q1	**q2**	**p1**	**p2**	**q3**							
0.2	0.7	500	100	0.1							
					186.88	289.296	398.4704	454.8293	493.0157		
v(x,t)											
x↓ t→	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	0	0	0	0	0	0	0	0	0	0	0
1	0	170	236	288.8	331.04	364.832	391.8656	413.4925	430.794	444.6352	455.7081
2	0	170	340	419.2	493.12	560.704	621.5296	675.5968	723.1759	764.6995	800.6867
3	0	170	340	510	598.28	677.248	753.9392	827.4573	897.0852	962.3033	1022.783
4	0	170	340	510	680	776.452	857.1684	936.5226	1014.71	1091.185	1165.408
5	0	170	340	510	680	850	953.8068	1036.832	1116.77	1196.358	1275.323
6	0	170	340	510	680	850	1020	1130.426	1216.192	1296.712	1376.642
7	0	170	340	510	680	850	1020	1190	1306.384	1395.211	1476.562
8	0	170	340	510	680	850	1020	1190	1360	1481.745	1573.864
9	0	170	340	510	680	850	1020	1190	1360	1530	1656.571
10	0	170	340	510	680	850	1020	1190	1360	1530	1700
11	1	170.1	340.01	510.001	680.0001	850	1020	1190	1360	1530	1700
12	2	171.1	340.2	510.029	680.0038	850.0005	1020	1190	1360	1530	1700
13	3	172.1	341.2	510.3	680.0561	850.009	1020.001	1190	1360	1530	1700
14	4	173.1	342.2	511.3	680.4	850.0905	1020.017	1190.003	1360	1530	1700
15	5	174.1	343.2	512.3	681.4	850.5	1020.131	1190.029	1360.005	1530.001	1700
16	6	175.1	344.2	513.3	682.4	851.5	1020.6	1190.178	1360.044	1530.009	1700.002
17	7	176.1	345.2	514.3	683.4	852.5	1021.6	1190.7	1360.23	1530.062	1700.015
18	8	177.1	346.2	515.3	684.4	853.5	1022.6	1191.7	1360.8	1530.287	1700.085
19	9	178.1	347.2	516.3	685.4	854.5	1023.6	1192.7	1361.8	1530.9	1700.349
20	10	179.1	348.2	517.3	686.4	855.5	1024.6	1193.7	1362.8	1531.9	1701

What is the expected optimal profit (when using the optimal admission policy) if there are 5 seats remaining and 10 periods until the time of flight?

Policy Optimization

- Extract the optimal policy from $v_t(x)$

q1	q2	p1	p2	q3				
0.2	0.7	500	100	0.1				
					186.88	289.296	398.4704	454.8293

v(x,t)								
x↓ t→	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	170	236	288.8	331.04	364.832	391.8656	413.4925
2	0	170	340	419.2	493.12	560.704	621.5296	675.5968
3	0	170	340	510	598.28	677.248	753.9392	827.4573
4	0	170	340	510	680	776.452	857.1684	936.5226
5	0	170	340	510	680	850	953.8068	1036.832
6	0	170	340	510	680	850	1020	1130.426
7	0	170	340	510	680	850	1020	1190
8	0	170	340	510	680	850	1020	1190
9	0	170	340	510	680	850	1020	1190
10	0	170	340	510	680	850	1020	1190

a(x,t):	admit decision for class 2	1 corresponds to admit, 0 corresponds to reject								
x↓ t→	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	1	1	0	0	0	0	0	0	0
3	0	1	1	1	1	0	0	0	0	0
4	0	1	1	1	1	1	1	0	0	0
5	0	1	1	1	1	1	1	1	0	0
6	0	1	1	1	1	1	1	1	1	1
7	0	1	1	1	1	1	1	1	1	1
8	0	1	1	1	1	1	1	1	1	1
9	0	1	1	1	1	1	1	1	1	1
10	0	1	1	1	1	1	1	1	1	1

If we have two periods remaining and 1 seat available, it is optimal to reject a sale from class 2
$p_2 + v_1(0) = 100 + 0 \leq v_1(2) = 170$.