Notes on Dynamic Programming

Fall 2017

Rajeev Kohli

Integer knapsack problem

Knapsack has weight capacity c (integer).
Can be filled using some or all of m items.
Item i has weight w_i and value v_i (integers).
Objective: maximize the knapsack value, subject to capacity constraint.

Consider items in order $i = 1, \ldots, m$ and capacities $j = 1, \ldots, c$.

Let m_{ij} = max. number of units of item i that can fit using capacity j.

$V(i,j)$ = value of filling the knapsack using items $1, \ldots, i$ and capacity j
(knapsack value at stage i and state j).

$$V(i,j) = \max_{x_i \in \{0,\ldots,m_{ij}\}} \left\{ v_i x_i + V(i-1, j-w_i x_{ij}) \right\},\ i = 1, \ldots, m, j = 1, \ldots, c$$

Optimal stopping rule for a game of dice

Roll a fair die at most five times. Stop whenever you want and receive as a reward the number shown on the die at the time you stop. What is the stopping rule that maximizes the expected payoff?

Stage i: i-th toss of a die.

State j: outcome of a toss, $j = 1, \ldots, 6$.

Solve using backward induction.

Source: Tijms, Henk (2012), "Stochastic games and dynamic programming," *Asia Pacific Mathematics Newsletter*, 2 (3), 6–10.

Stage i corresponds to toss i $(i = 1, \ldots, 5)$

State j corresponds to the number j rolled on a toss $(j = 1, \ldots, 6)$

$V(i, j) =$ payoff on toss i if you roll j

$$V(i, j) = \max\{j, E[V(i+1)]\}$$

$E[V(i)] =$ expected value of payoff for stage i

$$E[V(i)] = P(j > E[V(i+1)]) \cdot j + P(j \leq E[V(i+1)]) \cdot E[V(i+1)]$$

Stopping rule: stop at toss i if it is the first time $V(i, j) = j$.

Solution using backward induction

$i = 5$ (last toss). Take what you roll.

$$V(5, j) = j$$

$$E[V(5)] = \sum_{j=1}^{6} \frac{1}{6} \cdot j = 3.5$$

$i = 4$ (fourth toss). Stop only if $j > 3.5$ (that is, if $j = 4, 5$ or 6).

$$V(4, j) = \max\{j, E[V(5)]\}$$

$$E[V(4)] = \frac{3}{6} \cdot 3.5 + \sum_{j=4}^{6} \frac{1}{6} \cdot j = 4.25$$

$i = 3$ (third toss). Stop only if $j > 4.25$ (that is, if $j = 5$ or $j = 6$).

$$V(3, j) = \max\{j, E[V(4)]\}.$$

$$E[V(3)] = \frac{1}{6} \cdot 5 + \frac{1}{6} \cdot 6 + \frac{4}{6} \cdot 4.25 = 4.67$$

$i = 2$ (second toss). Stop only if $j > 4.67$ (that is, if $j = 5$ or $j = 6$)

$$V(2, j) = \max\{j, E[V(3, j)]\}$$

$$E[V(2, j)] = \frac{1}{6} \cdot 5 + \frac{1}{6} \cdot 6 + \frac{4}{6} \cdot 4.67 = 4.944$$

$i = 1$ (first toss). Stop only if $j > 4.944$ (that is, if $j = 5$ or $j = 6$.

Maximal expected payoff is

$$E[V(1, j)] = \frac{1}{6} \cdot 5 + \frac{1}{6} \cdot 6 + \frac{4}{6} \cdot 4.944 = 5.129.$$

Two-class model for revenue management

Capacity C, two product classes, no cancellation or overbooking. Class i has price p_i and demand D_i, where the probability $P(D_i \leq x) = F_i(x)$ is an increasing function of x. A single request is received from class 2 when the remaining capacity is x. Revenue from selling $= p_2$. Should you sell?

Newsboy problem: Sell only if revenue from selling is no less than the expected revenue from not selling: $p_2 \geq p_1(1 - F(x))$

$$\Rightarrow F(x) \geq 1 - \frac{p_2}{p_1} \Rightarrow x \geq F^{-1}\left(1 - \frac{p_2}{p_1}\right)$$

Littlewood's rule: the smallest value $x = y_1^*$ that satisfies this condition is the optimal protection level y_1^* for class 1.

Example: If $F_1(x) \sim N(\mu, \sigma)$, then $y_1^* = \mu + \sigma z$, where $z = \Phi^{-1}(1 - (p_1/p_2))$ and Φ is the standard Normal cdf.

n-class model:. Class j has price p_j, where $p_1 > \cdots > p_n$.
Periods $t = 1, \ldots, T$. At most one customer arrives in each period.
$R(t) = p_j$ if demand for class j arrives in period t with probability $\lambda_j(t)$, where $\sum_{j=1}^{n} \lambda_j(t) \leq 1$ (no demand may arrive in period t).

Bellman equation

$$V_t(x) = E\Big[\max\{R(t) + V_{t+1}(x-1), V_{t+1}(x)\Big]$$

Let $\Delta V_t(x) = V_t(x) - V_t(x-1)$ denote the marginal value of capacity at stage t. Then $V_{t+1}(x-1) = V_{t+1}(x) - \Delta V_{t+1}(x)$ and

$$V_t(x) = E\Big[\max\{R(t) + V_{t+1}(x) - \Delta V_{t+1}(x), V_{t+1}(x)\Big]$$

$$= V_{t+1}(x) + E\Big[\max\{R(t) - \Delta V_{t+1}(x), 0\}\Big]$$

Implies bid-control policy: accept request from class j in period t if $R(t) = p_j > \Delta V_{t+1}(x)$

Properties of $\Delta V_t(x) = V_t(x) - V_t(x-1)$

(i) an extra unit of capacity is more valuable when the available capacity is lower: $\Delta V_t(x+1) \leq \Delta V_t(x)$

(ii) an extra unit of capacity is more valuable when there are more periods remaining: $\Delta V_{t+1}(x) \leq \Delta V_t(x)$

Implies time-varying protection levels:

$$y_j^*(t) = \max\{x : p_{j+1} < \Delta V_{t+1}(x)\}, j = 1, \ldots, n-1$$

where $y_j^*(t)$ is the protected capacity for class j at time t, and

$$y_1^*(t) \leq \cdots \leq y_{n-1}^*(t).$$

Sequential buying

You want to buy a new car, add a room to your house, upgrade the kitchen, buy new furniture, vacation in Tanzania, ...

In what order should you make these (say) n purchases?

Buying i gives you reward (undiscounted utility) r_i.

If you buy i, you need to wait t_i days (a positive integer) before you have enough money to make the next purchase.

If you buy in the order $1, \ldots, n$, you get total discounted reward

$$r_1\beta^{t_1} + r_2\beta^{t_1+t_2} + \cdots + r_N\beta^{t_1+\cdots+t_n},$$

where $0 < \beta < 1$ is the discounting factor.

Job scheduling (from Richard Weber)

n jobs are to be processed successively on one machine. In what order should they be processed?

Job i has processing time t_i, a positive integer.

On completion of job i, reward r_i is obtained.

If processed in the order $1, \ldots, n$, total discounted reward is

$$r_1\beta^{t_1} + r_2\beta^{t_1+t_2} + \cdots + r_N\beta^{t_1+\cdots+t_n},$$

where $0 < \beta < 1$.

Source: Weber, Richard (2013), "Tutorial: Bandit Processes and Index Policies," Young European Queueing Theorist workshop on scheduling and priorities in queueing systems, Eindhoven, Nov. 4-5-6.

Dynamic programming solution

Let $S_k \subset \{1, \ldots, n\}$ be a subset of k uncompleted jobs.
The dynamic program is

$$V(S_k) = \max_{i \in S_k} \left[\beta^{t_i} r_i + \beta^{t_i} V(S_k - \{i\}) \right],$$

where $V(\emptyset) = 0$.

We can use an index policy to solve the problem. The index is based on an interchange argument.

Interchange argument

Consider processing jobs in the order

$$i_1, \ldots, i_k, i, j, i_{k+3}, \ldots, i_n.$$

Consider interchanging jobs i and j:

$$i_1, \ldots, i_k, j, i, i_{k+3}, \ldots, i_n.$$

Rewards under the two schedules are

$$R_1 + \beta^{T+t_i} r_i + \beta^{T+t_i+t_j} r_j + R_2$$

$$R_1 + \beta^{T+t_j} r_j + \beta^{T+t_j+t_i} r_i + R_2$$

where $T = t_1 + \cdots + t_{i_k}$ and R_1 and R_2 are the rewards accruing from jobs coming before and after i and j.

Index policy

Reward of first schedule is greater if

$$\frac{r_i \beta^{t_i}}{1-\beta^{t_i}} > \frac{r_j \beta^{t_j}}{1-\beta^{t_j}}$$

A schedule can be optimal only if jobs are taken in decreasing order of their indices $r_i \beta^{t_i}/(1-\beta^{t_i})$.

Index policy: total discounted reward is maximized by processing an uncompleted job of greatest index

$$G_i = \frac{r_i \beta^{t_i}(1-\beta)}{1-\beta^{t_i}}.$$

Since

$$\frac{1-\beta^{t_i}}{1-\beta} = 1 + \beta + \cdots + \beta^{t_i - 1}$$

$G_i \to r_i/t_i$ as $\beta \to 1$.

Multi-armed bandit problems

Embody conflict between immediate payoff and information.

Example

Website morphing (Hauser, Urban, Liberali and Braun 2009)

Automatically match the "look and feel' of a website, not just the content, to the cognitive styles of visitors. The morph (1) must be based on relatively few clicks, (2) learn which characteristics are best for which customers, (3) use prior information, and (4) be implemented in real time.