WonderMarket
Section A – Internal Report

Kenton Lam

MATH3202 Assignment 3
Due 27/05/2019 1:00 pm

Abstract

Our long-time client WonderMarket has approached regarding their foray into the refridgerator space. Competition in the area is fierce and they need us to optimise their fridge logistics. Given information about their costs and requirements, we created and optimmised a dynamic programming model using Python. This report describes the model and its solution.

Model Definition

WonderMarket wishes to start selling fridges. For the time being, they are focusing on 3 bespoke fridge options—named "Alaska", "Elsa" and "Lumi".

Data

The following data has been provided to us by WonderMarket.

F	the set of fridge types.
Profit_f	profit made by selling one of fridge f.
$\text{Expected}_{f,n}$	expected number of fridge f sold if n are displayed (comm 1 only). $(0 \leq n \leq 4)$.
$\text{DemandProbs}_{f,n}$	probability that n units of fridge f will be sold (comm 2). $(1 \leq n \leq 6)$.
StoreCost	cost of storing one fridge for one week (comm 2).
FridgesPerTruck	maximum fridges transported by one truck (comm 3).
TruckCost	cost of one truck (comm 3).
MaxTrucks	maximum number of trucks per week (comm 3).
MaxStore	maximum number of each fridge type handled per week (comm 3).

Communication 1

WonderMarket wishes to display fridges. They can display up to 8 fridges and the amount of each fridge bought depends on how many fridges of that type are displayed.

Here, the stages are the fridge type and the state is the number of fridges remaining. The action describes the number of the current fridge to display. We write

$$f = \text{current fridge}$$
$$r = \text{remaining fridges}$$
$$A = \{0, \ldots, 4\} = \text{all possible number of fridges display}$$

Mathematically, we can write the value function as

$$V_f(r) = \max_{a \in A \mid a \leq r} \left\{ \text{Profit}_f \times \text{Expected}_{f,a} + V_{f+1}(r-a) \right\}$$

with the base cases of

$$V_3(r) = 0, \qquad V_f(0) = 0 \qquad \forall r, f.$$

To optimise for WonderMarket, we computed $V_0(8)$ which will iterate through the stage of each fridge (0 to 2) and with 8 available fridges. This resulted in a maximum profit of \$1863.20 which is achieved by displaying 2 Alaska fridges, 3 Elsa fridges and 3 Lumi fridges.

Note that we assumed at most 4 of a fridge type would be displayed. This is reasonable as any fridges more than 4 do not increase expected sales. Therefore, it will always be better to display some fridges of another type than more than 4 fridges of one type.

Communication 2

In this communication, the focus moved from direct-to-customer sales to delivery. WonderMarket is running a 4 week trial of their refridgerator branch and needs to order fridges to ship to their customers. There are costs and constraints associated with storing fridges during this process.

In addition, they cannot know the exact demand ahead of time so require us to optimise for maximum expected profit.

We can consider each fridge independently of the others, so we write a value function for one specific fridge then combine them later. For each fridge, the stage is number of weeks elapsed, state is number of fridges stored at the start of that week and actions represent the number of fridges bought. We also have a set of possible demands.

We write

$$\begin{aligned} t &= \text{weeks elapsed} \\ s &= \text{fridges of type } f \text{ stored at start of week} \\ A &= \{0, \ldots, 6\} = \text{all possible number of fridges to order} \\ D &= \{1, \ldots, 6\} = \text{all possible demands} \end{aligned}$$

The value function for a particular fridge can be written as

$$\begin{aligned} V_{f,t}(s) = \max_{a \in A} \Big\{ &- \text{StoreCost} \times (s+a) \\ &+ \sum_{d \in D} \left(\text{DemandProbs}_{f,d} \times [\text{Profit}_f \times \min(d, s+a) + V_{f,t+1}(s+a-\min(n, s+a))] \right) \Big\} \end{aligned}$$

with base case $V_{f,4}(s) = 0$. This function was memoized using Python's lru_cache decorator. We then consider all fridges using

$$V_t(a, e, l) = V_{0,t}(a) + V_{1,t}(e) + V_{2,t}(l).$$

Computing $V_0(0, 0, 0)$ returns \$5282.92 as maximum expected profit which is obtained by ordering 4, 5, 5 of Alaksa, Elsa and Lumi fridges respectively in the first week. Because this is a stochastic dynamic programming problem, the complete list of optimal actions cannot be known in advance. See appendix for a Python script to interactively explore the solution.

Note that the summation computes an expected value over all demands. $\min(n, s+a)$ ensures that the number of fridges sold is limited by the smaller of n or $s+a$, the demand and number of fridges available respectively.

This cannot be directly compared to communication 1 because comm 1 was one week only and a non-stochastic DP problem.

We assumed at most 6 of a fridge type can be ordered per week. This is reasonable because at most, 6 fridges are sold per week. If 7 fridges are ordered, one would always be left over to next week. However, it is cheaper to only order 6 then buy one next week if required. Also, because WonderMarket is only just beginning their fridge trial, we assume they have no fridge stock stored.

Communication 3

In addition to the requirements of communication 2, WonderMarket needs to consider the logistics of transporting fridges to their warehouse. At most, 7 fridges (possibly of mixed type) fit on a truck and at most 2 trucks can be ordered per week. Each truck costs a flat fee of \$150. In addition, at most 8 fridges of each type can be in the warehouse at a time. This results in a maximum of 14 fridges ordered per week.

Again, the stage is number of weeks elapsed. Our state will now be a vector of fridges of each type currently stored. Similarly, the action is number of each fridge bought in the current week and the set of demands now considers all 3 fridge types.

Let $\sum \mathbf{x}$ be the sum of elements of the vector $\mathbf{x}$. We write

$t =$ weeks elapsed

$\mathbf{s} =$ fridges stored at start of week
(given $\mathbf{s}, s_f =$ fridges of type f stored)

$A = \left\{\mathbf{a} \in \{0, \ldots, 14\}^3 \mid \sum \mathbf{a} \leq 14\right\} =$ all possible permutations of fridges to order
(given $\mathbf{a} \in A, a_f$ is number of fridge f ordered)

$D = \{1, \ldots, 6\}^3 =$ all possible demands
(given $\mathbf{d} \in D, d_f$ is demand of fridge f)

To keep the notation neat, we define the element-wise minimum of two n-dimensional vectors as

$$\operatorname{emin}(\mathbf{v}, \mathbf{u}) := \left(\min\{v_1, u_1\}, \min\{v_2, u_2\}, \ldots, \min\{v_n, u_n\}\right).$$

Also note that Profit can be treated as a 3-dimensional vector. Then, we write the value function as $V_4(\mathbf{s}) = 0$, and for $t < 4$,

$$\begin{aligned} V_t(\mathbf{s}) = \max_{\substack{\mathbf{a} \in A \\ \max(\mathbf{s}+\mathbf{a}) \leq \mathrm{MaxStore}}} \Bigg\{ & -\mathrm{StoreCost} \times \sum(\mathbf{s}+\mathbf{a}) \\ & - \mathrm{TruckCost} \times \left\lceil \frac{\sum \mathbf{a}}{\mathrm{FridgesPerTruck}} \right\rceil \\ & + \sum_{\mathbf{d} \in D} \Big[\Big(\prod_{f \in F} \mathrm{DemandProbs}_{f,d_f} \Big) \times \Big(\mathrm{Profit} \cdot \operatorname{emin}(\mathbf{d}, \mathbf{s}+\mathbf{a}) \\ & \qquad + V_{t+1}(\mathbf{s}+\mathbf{a} - \operatorname{emin}(\mathbf{d}, \mathbf{s}+\mathbf{a})) \Big) \Big] \Bigg\} \end{aligned}$$

Computing $V_0(\mathbf{0})$ results in an expected maximum profit of \$4229.47. WonderMarket should buy 4, 5 and 5 of Alaska, Elsa and Lumi respectively in the first week. Again, this

is a stochastic problem and further actions can only be stated with certainty once the outcome of the first week is known. An interactive explorer is included in the appendix.

It is expected that this is less then communication 2 as we are now considering truck costs on top of communication 2's costs. This reduces the expected profit by $1053.44.

The algorithm implemented differs slightly from the above formulation because it was further optimised to reduce runtime (see below).

Optimisations

Initially, the simple model ran in over 5 minutes. This was due to the large action space (680) and large stochastic event space (potential demand scenarios, approximately $6^3 = 216$). These compounded, resulting in 146880 iterations per call of V_t.

To shrink the stochastic event space, we defined the following memoized auxillary function.

$$\text{capped\_demand\_probs}(\mathbf{c}) := \{(\mathbf{d}, p(\mathbf{c},\mathbf{d})) \mid \mathbf{d} \in D,\ d_f \le c_f\ \forall f \in F,\ p(\mathbf{c},\mathbf{d}) \neq 0\}$$
$$\text{where} \qquad p(\mathbf{c},\mathbf{d}) := \sum_{\substack{\mathbf{e}\in D \\ \text{emin}(\mathbf{e},\mathbf{c})=\mathbf{d}}} \prod_{f\in F} \text{DemandProbs}_{f,e_f}$$

In words, for each tuple $(\mathbf{d}, p) \in \text{capped\_demand\_probs}(\mathbf{c})$, $\mathbf{d}$ is a tuple of demands which can be completely fulfilled given we currently have $\mathbf{c}$ fridges. Moreover, for each $\mathbf{e} \in D$ where $\mathbf{e}$ is not possible, we add the probability of $\mathbf{e}$ to the probability of $\text{emin}(\mathbf{e},\mathbf{c}) \in D$. Also, we remove demands with probability 0 from the set.

This has significant speed ups, especially for calls of $V_t(\mathbf{s})$ where the number of fridges held is small, this reduces the number of iterations required to compute the expected value. This reduced the runtime to under 10 seconds.

Additionally, we augmented the action set A with precomputed costs of storage and transport.

$$\text{ActionsWithCosts} := \left\{ \left(\mathbf{a}, -\text{StoreCost} \sum \mathbf{a} - \text{TruckCost} \left\lceil \frac{\sum \mathbf{a}}{\text{FridgesPerTruck}} \right\rceil \right) \ \middle|\ \mathbf{a} \in A \right\}$$

Then, the value function can be written as

$$V_t(\mathbf{s}) = \max_{\substack{(\mathbf{a},\text{action\_cost}) \in A \\ \max(\mathbf{s}+\mathbf{a}) \le \text{MaxStore}}} \left\{ -\text{StoreCost} \sum \mathbf{s} + \text{action\_cost} + \sum_{(\mathbf{d},p)\in \text{capped\_demand\_probs}(\mathbf{s}+\mathbf{a})} \left[p \times \left(\text{Profit} \cdot \mathbf{d} + V_{t+1}(\mathbf{s}+\mathbf{a}-\mathbf{d}) \right) \right] \right\}$$

In code, we manually expanded the vector operations.

Appendix

We have included an interactive Python script alongside the technical model so WonderMarket can explore the solution in the weeks to come. This is included in stochastic_explorer.py. This should be placed into the same folder as assignment_dp.py. Usage is described in the report to WonderMarket.