Approximate Dynamic Programming in the context of e-mobility

Freie wissenschaftliche Arbeit
zur Erlangung des akademischen Grades
"Master of Science"
Studiengang: Finanz- und Informationsmanagement

an der
Wirtschaftswissenschaftlichen Fakultät
der Universität Augsburg

– Lehrstuhl für Analytics & Optimization –

Eingereicht bei:	Prof. Dr. Robert Klein
Betreuer:	tbd
Vorgelegt von:	Stefan Glogger
Adresse:	tbd
Matrikel-Nr.:	tbd
E-Mail:	stefan.glogger@student.uni-augsburg.de
Datum:	June 18, 2019

Contents

Chapter 1

Main chapter

1.1 Ideas

1. capacity consumption $a \in \mathbb{N}_0$ statt $a \in \{0, 1\}$

1.2 The problem

Here, we want to lay out the classical revenue management problem an the "brute force" approach to solve.

Consider a firm that produces products $j = 1, \dots, n$ with revenues $\boldsymbol{r} = (r_1, \dots, r_n)^T$. Resources $h = 1, \dots, m$ are used for production. In order to produce one unit of product j, resources $\boldsymbol{a}_j = (a_{1j}, \dots, a_{mj})^T$ are necessary, with $a_{hj} = 1$ if resource h is needed for production of product j and $a_{hj} = 0$ otherwise. Initially, the capacity is described by $\boldsymbol{c}^0 = (c_1^0, \dots, c_m^0)^T$.

The booking horizon is modelled by sufficiently small time periods $t = 0, \dots, T-1$, such that in each time period at most one customer arrives. This customer also purchases at most one product. If product j is purchased at time t, the capacity reduces to $\boldsymbol{c}^{t+1} = \boldsymbol{c}^t - \boldsymbol{a}_j$. Time moves forward, such that the last selling might occur at time $T-1$.

The firm wants to increase the value of the products sold and has flexibility in the sets offered. Thus, the decision variables at each time point t are given by $\boldsymbol{x}^t = (x_1^t, \dots, x_n^t)^T$ with $x_j^t = 1$ if product j is offered at time t and $x_j^t = 0$ otherwise.

One popular method of describing the probabilities of purchases is to have each customer belonging to one customer segment $l = 1, \dots L$, each of which following a multinomial logit model (MNL). A customer of segment l arrives with probability λ_i. His preference weights are given by $\boldsymbol{u}_l = (u_{l1}, \dots, u_{ln})^T$ and no purchase preference of u_{l0}. Note: $u_{lj} > 0$ if consumer of segment l might purchase product j (the higher, the more interested) and $u_{lj} = 0$ if customer is not interested in product. The probability of purchasing product j when set $\boldsymbol{x}$ is offered is given by $p_{lj}(\boldsymbol{x}) = \frac{u_{lj}x_j}{u_{l0}+\sum_{p\in[n]} u_{lp}x_p}$ and the no-purchase probability is given by $p_{l0}(\boldsymbol{x}) = 1 - \sum_{p\in[n]} p_{lp}$. Together with the uncertainty of which customer segment arrives (if any), we end up at a purchase probability for product j given $\boldsymbol{x}$ of $p_J(\boldsymbol{x}) = \sum_{l\in[L]} p_{lj}(\boldsymbol{x})$.

1.2.1 Examples

Single-leg flight example

For reasons of comparability, we use the same example as in **?**. An airlines offers four products with revenues $\boldsymbol{r} = (1000, 800, 600, 400)^T$ over $T = 400$ periods. Only one customer segment exists with arrival probability of $\lambda = 0.5$ and preference weights $\boldsymbol{u} = (0.4, 0.8, 1.2, 1.6)^T$.

Different network loads can be analyzed by varying initial capacity $c^0 \in \{40, 60, \ldots, 120\}$ and varying no-purchase preference weights $u_0 \in \{1, 2, 3\}$.

1.3 Implementation

Here, we present the results of the exact calculation for the single leg flight example.
Storage folder: `"C:/Users/Stefan/LRZ Sync+Share/Masterarbeit-Klein/Code/Results/singleLegFl`
Log:

```
Time (starting): 2019-06-11 09:17:42.448064
Example :  singleLegFlight
use_var_capacities :  True


Total time needed:
 151.31108283996582 seconds =
 2.521851380666097 minutes
```

Results:

	capacity	no-purchase preference	DP-value	DP-optimal offer set
0	40	1	35952	(1, 1, 0, 0)
1	60	1	49977.1	(1, 1, 0, 0)
2	80	1	59969.8	(1, 1, 1, 0)
3	100	1	65956.4	(1, 1, 1, 0)
4	120	1	68217.2	(1, 1, 1, 1)
5	40	2	35952	(1, 1, 0, 0)
6	60	2	49977.1	(1, 1, 0, 0)
7	80	2	59969.8	(1, 1, 1, 0)
8	100	2	65956.4	(1, 1, 1, 0)
9	120	2	68217.2	(1, 1, 1, 1)
10	40	3	35952	(1, 1, 0, 0)
11	60	3	49977.1	(1, 1, 0, 0)
12	80	3	59969.8	(1, 1, 1, 0)
13	100	3	65956.4	(1, 1, 1, 0)
14	120	3	68217.2	(1, 1, 1, 1)

1.4 Approximate Dynamic Programming

1.4.1 Approximate Policy Iteration

Line 16 umschreiben. Weil fuer den update werden alle Parameter auf einmal geupdatet und nicht fuer jeden Zeitschritt separat.

Overview of parameters:

1. θ_t optimization parameter (offset)
2. π_t optimization parameter (bid price for each resource)

Algorithm 1 Approximate policy iteration {alg-API}

1: Set $\theta_t = 0$ and $\pi_t = \mathbf{0}$ $\forall t = 1, \ldots, T$
2: **for** `k = 1 to K` **do**
3: Set $\hat{V}_t = 0$ and $\hat{\mathbf{C}}_t = 0$ $\forall t = 1, \ldots, T$
4: **for** `i = 1 to I` **do**
5: Set $\hat{r}_t = 0$ and $\hat{\mathbf{c}}_t = 0$ $\forall t = 1, \ldots, T$
6: Initialize $\mathbf{c} = \mathbf{c}^0$
7: **for** `t = 1 to T` **do**
8: $\hat{\mathbf{c}}_t \coloneqq \mathbf{c}$
9: Compute $\pi(t, \mathbf{c})$ {alg-API-cal
10: Compute $\mathbf{x} = \text{determineOfferset}(\pi(t, \mathbf{c}), \epsilon_t)$
11: Simulate a sales event $j' \in \{0, 1, \ldots, n\}$
12: **if** $j' \in \{1, \ldots, n\}$ **then**
13: $\hat{r}_t = r_{j'}$ and $\mathbf{c} = \mathbf{c} - \mathbf{a}_{j'}$
14: Compute $\hat{V}_t^i = \sum_{\tau=t}^{T} \hat{r}_t \quad \forall t = 1, \ldots, T$
15: Assign $\hat{\mathbf{C}}_t^i = \hat{\mathbf{c}}_t \quad \forall t = 1, \ldots, T$
16: $(\theta_t, \pi_t) = \text{updateParameters}\left(\hat{V}_t, \hat{\mathbf{C}}_t, \theta_t, \pi_t, k\right) \quad \forall t = 1, \ldots, T$ {alg-API-upd
return $(\theta_t, \pi_t) \quad \forall t = 1, \ldots, T$

3. $\hat{V}_t = 0$ all sample revenues to go for each sample for each time
4. $\hat{\mathbf{C}}_t = 0$ all sample available capacities for each sample for each resource for each time
5. r_t sample revenue generated at time t
6. $\mathbf{c}$ available capacities for each resource at current time
7. $\mathbf{c}^0$ starting capacities
8. $\mathbf{x}$ offerset at current time
9. ϵ_t epsilon used at current time

Calculations:

$\pi(t, \mathbf{c}) = \pi_h(t, c_h)$ for $h \in [m]$

$$\pi_h(t, c_h) = \begin{cases} \infty & \text{if } c_h = 0 \quad (1.1) \\ \sum_{s=1}^{S_h} \pi_{ths} \mathbb{1}_{\left(b_h^{s-1}, b_h^s\right]}(c_h) & \text{otherwise.} \quad (1.2) \end{cases}$$

Fuer Line 9 verwende Zeit **t** statt **t+1**. Grund: Kenne Informationen zur Zukunft nicht.

The function determineOfferset(π, ϵ) calculates the set to offer depending on the current bid prices π via the greedy algorithm layed out in . To account for the exploration vs exploitation dilemma, an epsilon-greedy strategy is used. With a probability of $\epsilon/2$ either no product is

offered at all or all products with positive contribution $r_j - \sum_{h\in[m]} a_{hj} \cdot \pi_h$ are offered. With a probability of $1-\epsilon$, the proper calculated set is offered.

A sales event is simulated by first having one or zero customer arrive at random. In case a customer arrives, its preference function given the offer set determines the probability according to which one product is sold ($j' \in \{1,\dots,n\}$) or no product is sold ($j'=0$).

The function $(\theta_t, \pi_t) = \text{updateParameters}\left(\hat{V}_t, \hat{\mathbf{C}}_t, \theta_t, \pi_t, k\right)$ really optimizes the following least squares optimization problem for all parameters ($t = 1,\dots,T$) at the same time.

$$V_t(\theta_t, \pi_t, \mathbf{c}_t) \coloneqq \theta_t + \sum_{h=1}^{m}\sum_{s=1}^{S_h} \pi_{ths} f_{hs}(c_h) \tag{1.3}$$

$$f_{hs}(c_h) \coloneqq \begin{cases} 0 & \text{if } c_h \le b_h^{s-1} \\ c_h - b_h^{s-1} & \text{if } b_h^{s-1} < c_h \le b_h^s \\ b_h^s - b_h^{s-1} & \text{if } b_h^s < c_h \end{cases} \tag{1.4}$$ {def-f}

Equation (1.4) describes the occupied amount of capacity of interval $\left(b_h^{s-1}, b_h^s\right]$.

The following optimization problem depends on the old parameters $\theta_t = \theta_t^k$ and $\pi_t = \pi_t^k$ to determine the optimal parameter θ_t^{update} and π_t^{update}.

$$\min \sum_{i=1}^{I}\sum_{t=1}^{T} \left(\hat{V}_t^i - V_t(\theta_t, \pi_t, \mathbf{c}_t^i)\right)^2 \tag{1.5}$$

$$s.t. \tag{1.6}$$

$$\theta_t \ge 0 \qquad \forall t \tag{1.7}$$

$$\max_{j=1,\dots,n} r_j \ge \pi_{ths} \ge 0 \qquad \forall t,h,s \tag{1.8}$$

$$\pi_{ths} \ge \pi_{th,s+1} \qquad \forall t,h,s = 1,\dots,S_h - 1 \tag{1.9}$$

$$\theta_t \ge \theta_{t+1} \qquad \forall t = 1,\dots,T-1 \tag{1.10}$$

$$\pi_{ths} \ge \pi_{t+1,hs} \qquad \forall t = 1,\dots,T-1 \tag{1.11}$$

The final parameters θ_t^{K+1} and π_t^{K+1} can obtained via two possible equally possible ways. One is the so called exponential smoothing, where in each iteration k the parameter for the next iteration $k+1$ is calculated via:

$$\theta_t^{k+1} = \left(1 - \frac{1}{k}\right)\theta_t^k + \frac{1}{k}\theta_t^{update} \tag{1.12}$$

$$\pi_t^{k+1} = \left(1 - \frac{1}{k}\right)\pi_t^k + \frac{1}{k}\pi_t^{update} \tag{1.13}$$

The other one uses $\theta_t^{k+1} = \theta_t^{update}$ and $\pi_t^{k+1} = \pi_t^{update}$ and averages at the very end.

$$\theta_t^{K+1} = \frac{1}{K} \sum_{k=1}^{K} \theta_t^k \tag{1.14}$$

$$\pi_t^{K+1} = \frac{1}{K} \sum_{k=1}^{K} \pi_t^k \tag{1.15}$$

Proof.

Den Beweis sauber ausfuehren. Hier sind die Inhalte. Die Aussage stimmt, falls die gefundenen optimalen Loesungen in jeder Iteration stets dieselben sind. Dies ist der Fall, wenn es stets nur ein Minimum gibt. Wir haben eine quadratische Zielfunktion, die sozusagen eine mehrdimensionale, nach oben geoeffnete Parabel zeigt, die genau ein globales Minimum besitzt. Weitere noetige Punkte: eindeutiges globales Minimum ueber positiv definite Hesse-Matrix. Optimum auch zulaessig (schwierig?)

■

1.4.2 Greedy Heuristic to determine offerset

The following algorithm is based on the ideas of the greedy heuristic for the column generation subproblem outlined in **?**.

Our goal is to determine a reasonable set of products to offer in a fast manner. Thus, we use a heuristic and cut down the amount of products to consider as fast as possible.

Value Funktion gebuendelt dargestellt und ueberall mit λ. Vergleiche zu Bront et al. 4.2.2, wo in 3. ohne λ und in 4.a mit λ.

Algorithm 2 Greedy Heuristic {alg-GreedyH

1: $\text{Value}(X) \coloneqq \sum_{l=1}^{L} \lambda_l \frac{\sum_{i \in X} (r_i - A_i^T \pi) u_{li}}{\sum_{i \in X} u_{li} + u_{l0}}$
2: $S \coloneqq \emptyset, \quad S' \coloneqq \left\{ j \in N : r_j - A_j^T \pi > 0 \right\}$ {alg-L1}
3: $j^* \coloneqq \arg\max_{j \in S'} \text{Value}(\{j\})$
4: **repeat**
5: $\quad S \coloneqq S \cup \{j^*\}, \quad S' \coloneqq S' \backslash \{j^*\}$
6: $\quad j^* \coloneqq \arg\max_{j \in S'} \text{Value}(S \cup \{j\})$
7: **until** $\text{Value}(S \cup \{j^*\}) \leq \text{Value}(S)$
8: **return** S

Let S' be the set of products with positive reduced costs, i. Line 2

Nomenclature

MNL multinomial logit model