POLITECNICO
MILANO 1863

Data Intelligence Application Project

Academic Year: 2018/2019

Part I: Pricing

Davide Rutigliano - 903616

Contents

1 Introduction

This project is about the study of A/B testing and Multi-Armed Bandit (MAB) approaches for pricing applications, where a seller needs to identify the best price for a particular kind of item that maximizes the profit without knowing the buyer demand. This is an interesting problem from the seller point of view, since the profit depends by the pricing policy and the best pricing policy is determined by the buyers' preferences, which are usually unknown a priori. In this work we make the assumption that the conversion probability , that is the probability that a buyer purchases the good, is decreasing as the revenue margin is increasing.

We decided to study the case of a laptop sold on the internet.

This type of good is regularly sold all over the year and there are some specific periods (from now on "seasons" or "phases") that are worth to analyse in terms of demands in order to increase the profit in each phase.

We identified three season denoted with *before christmas*, *christmas* and *after christmas*, respectively: the period before Christmas (for instance from mid November to start of December), the Christmas time (for example from the first days of December up to the 25th) and the period after Christmas (from the 26th of December on).

We also differentiate customers basing on their class identified by the features.

Features are summarized in the following table:

Feature	Value
Age	Young, Old
Nation	Italy, USA

Each class can occur with a specific probability, in this study we consider:

- 30% of users are from Italy, 70% are from USA;
- 60% customers are young, the remaining 40% are old.

Thus, we have (for every season) four different probability distribution and dis-aggregate demand curves, one for each combination of age and gender:

- Italy - Young (18%)
- Italy - Old (12%)
- USA - Young (42%)
- USA - Old (28%)

The probability with which each class can occur is reported in brackets.

2 Formal Model

The formal model definition is:

- $p \in \mathbb{R}^+$ price

- $q(p)$ aggregate demand for price p
- $c(q(p))$ cost (for seller) of q(p) items, in this experiment we consider $c(q(p)) = 0, \forall p$
- $pq(p)$ revenue at price p
- $pq(p) - c(q(p))$ profit at price p

The objective of the seller is the maximization of the profit.

Given the prices (or "candidates"), the probability with which each feature can occur and the conversion rate of each feature for each season, we can calculate the probability distribution for each class of customers for each season.

Furthermore, we calculate the revenue without taking into account of the cost of the item sold (for the seller) , that is in our case the revenue is the same as the profit. The problem can be trivially extended to the general case where there's a cost for the seller for the item sold.

3 Algorithms

3.1 A/B Testing

We explain here all the steps of A/B testing. In this work we propose Sequential A/B testing approach in which all candidates are tested in pairs.

Generate data We randomly generate data for the experiment by using a Bernoulli random variables (of parameter the conversion rate associated to the candidate price) to simulate that an user buy or not the product at a given price with a given probability.

Note: if $X \sim \mathcal{B}(p)$, then $\mathbb{E}[X] = p$ and $\text{VAR}[X] = p(1-p)$

Formulate hypothesis We are considering two candidates at time in order to have a test that states if one candidate is better than the other or not.

In order to take into account of both conversion rates and prices and therefore select the candidate that gives to the seller the highest revenue, we weighted the conversion rate (the probability of buying or not buying) for the price.

We first define the means of (the Bernoulli random variable associated to) candidate C_1 and C_2 weighting the conversion rates for the prices:
$\mu_{C_2} = \frac{cr_2 \times p_2}{p_1 + p_2}$ and $\mu_{C_1} = \frac{cr_1 \times p_1}{p_1 + p_2}$;

where p_1 and p_2 are the prices of respectively variant 1 and variant 2.

Hypothesis	Symbol	Value
Null	$\mathcal{H}0$	$\mu_{C_2} - \mu_{C_1} = 0$
Alternative	$\mathcal{H}1$	$\mu_{C_2} - \mu_{C_1} \geq \delta$

Test statistic The idea behind this test statistic is this: we have two normal distributions the null hypothesis and the alternative hypothesis. We consider the value z that is the realization of the random variable Z defined below: it represent the empirical difference between the two candidates divided by the pooled standard deviation. When this value is smaller than the 95% of the null hypothesis we accept $\mathcal{H}0$, otherwise we decide for $\mathcal{H}1$.

$$Z = \frac{\bar{X_2} - \bar{X_1}}{\sqrt{\bar{Y} \times (1 - \bar{Y}) \times \left(\frac{1}{n_1} + \frac{1}{n_2}\right)}}$$

where the pooled mean:

$$\bar{Y} = \frac{n_1 \bar{X_1} + n_2 \bar{X_2}}{n_1 + n_2}$$

Choose accuracy of test statistics We decided to use an accuracy that is typical for A/B testing, that is 95% , or equivalently a I-type error probability of 5%. The I-type error probability is often denoted as α (or significance level) and it represent the probability of rejecting the null hypothesis when it's true (false positive rate).

The other parameter we set before the experiment is the power of the test $1 - \beta = 80\%$ and the subsequent type-II error or false negative rate, given by $\beta = 20\%$.

Select random samples We first calculate the minimum number of samples needed for the test, then we set the number of samples of our experiment to be at least the minimum sample size.

Given α and β defined above, the standard deviation of the process σ and the minimum variation δ, we have:

$$n = \frac{(z(1-\alpha) + z(\beta))^2 \sigma^2}{\delta^2}$$

where $z(q)$ is the value of the Gaussian distribution for quantile q.

Note: in order to calculate the standard deviation and the minimum difference between candidates we also used the conversion rates weighted for the selling prices.

Calculate test statistics on given samples

3.2 Upper Confidence Bound One

Notation

- t time
- A set of arms
- a arm
- a_t arm played at time t
- a^* optimal arm

- X_a random variable (Bernoulli) associated to arm a
- μ_a expected value of random variable X_a
- $x_{a,t}$ realization of rv X_a at time t
- x_a realizations of X_a
- $\bar{x}_a$ empirical mean of x_a
- $n_a(t)$ number of samples of arm a at time t

Pseudocode

1. Play once each arm $a \in A$
2. At every time t play arm a such that:

$$a_t \leftarrow \arg\max_a \left\{ \left[\bar{x}_a + \sqrt{\frac{2log(t)}{n_a(t-1)}} \right] \times a \right\}$$

3.3 Sliding Window Upper Confidence Bound One

Notation

- t time
- τ sliding window
- $\{b_1, ..., b_m\}$ breakpoints
- A set of arms
- a arm
- a_t arm played at time t
- a_t^* optimal arm at time t
- $X_{a,t}$ random variable (Bernoulli) associated to arm a
- $\mu_{a,t}$ expected value of random variable X_a
- $x_{a,t}$ realization of rv X_a at time t
- $\bar{x}_{a,t,\tau}$ empirical mean of x_a computed with samples in $\{t - \tau, ..., t\}$
- $n_a(t, \tau)$ number of samples of arm a at time t in $\{t - \tau, ..., t\}$

Pseudocode

1. Play once each arm $a \in A$
2. At every time t play arm a with $n_a(t) = 0$ if any, otherwise play arm a such that:

$$a_t \leftarrow \arg\max_a \left\{ \left[\bar{x}_{a,t,\tau} + \sqrt{\frac{2log(t)}{n_a(t-1,\tau)}} \right] \times a \right\}$$

The idea behind these algorithms is that at each round the arm (price) with the highest upper confidence bound value (multiplied by the corresponding price) is chosen in order to maximize the total revenue among all the possible arms.

3.4 Thompson Sampling

Notation (in addition to the one of classical UCB1)

- $\mathbb{P}(\mu_a = \theta_a)$ prior of the expected value of X_a
- θ_a variable of $\mathbb{P}(\mu_a = \theta_a)$
- $(\alpha_{a_t}, \beta_{a_t})$ parameters of the beta distribution $P(\mu_a = \theta_a)$

Pseudocode

1. At every time t for every arm a:

$$\tilde{\theta_a} \leftarrow Sample(\mathbb{P}(\mu_a = \theta_a))$$

2. At every time t play arm a_t such that:

$$a_t \leftarrow \arg\max_a \left\{ \tilde{\theta_a} \times a \right\}$$

3. Update beta distribution of arm a_t
$(\alpha_{a_t}, \beta_{a_t}) \leftarrow (\alpha_{a_t}, \beta_{a_t}) + (x_{a_t,t}, 1 - x_{a_t,t})$

3.5 Sliding Window Thompson Sampling

Notation is the same as above (in addition to the one of sliding window UCB1)

Pseudocode

1. At every time t for every arm a:

$$\tilde{\theta_a} \leftarrow Sample(\mathbb{P}(\mu_a = \theta_a))$$

2. At every time t play arm a_t such that:

$$a_t \leftarrow \arg\max_a \left\{ \tilde{\theta_a} \times a \right\}$$

3. Update beta distribution of arm a_t:

$$\text{if } t < \tau : (\alpha_{a_t}, \beta_{a_t}) \leftarrow (\alpha_{a_t}, \beta_{a_t}) + (x_{a_t,t}, 1 - x_{a_t,t})$$
$$\text{if } t > \tau :$$
$$(\alpha_{a_t}, \beta_{a_t}) \leftarrow \arg\max \big\{(1,1), (\alpha_{a_t}, \beta_{a_t}) + (x_{a_t,t}, 1 - x_{a_t,t}) - (x_{a_{t-\tau},t-\tau}, 1 - x_{a_{t-\tau},t-\tau})\big\}$$

The idea behind these algorithms is that at each round the arm (price) with the highest value of the prior (multiplied by the corresponding price) is chosen in order to maximize the total revenue among all the possible arms.

3.6 Context Generation

Notation

- t time
- l features, we assume each feature can have a binary value (e.g. for feature young, we have that a customer is either young or not)
- $F \subseteq \{0,1\}^l$ space of attributes
- $c \subseteq F$ context
- $\mathscr{P} = \{c : c \subseteq F, c \cup \varnothing, c \cap F\}$ context structure
- a_c^* optimal arm for context c
- $X_{a,c}$ random variable (Bernoulli) associated to arm a in context c
- $\mu_{a,c}$ expected value of random variable $X_{a,c}$
- $x_{a,c,t}$ realizations of $X_{a,c}$ at round t
- $\bar{x}_{a,c}$ empirical mean of $x_{a,c,t}$

Value of a context structure $v = \sum_{c \in \mathscr{P}} p_c \mu_{a_c^*,c}$

Split condition $\underline{p}_{c_1} \underline{\mu}_{a^*_{c_1},c_1} + \underline{p}_{c_2} \underline{\mu}_{a^*_{c_2},c_2} \geq \mu_{a^*_{c_0},c_0}$

Where:
$\underline{p}_{c_i}$ is the lower bound of the probability of context c_i and,
$\underline{\mu}_{a^*_{c_i}}$ is the lower bound of the reward of the optimal arm a_c^* in context c_i.

Lower bound The lower bound is calculated as the Hoeffding bound, that for Bernoulli distribution is:

$$\bar{x} - \sqrt{-\frac{log(\delta)}{2|Z|}}$$

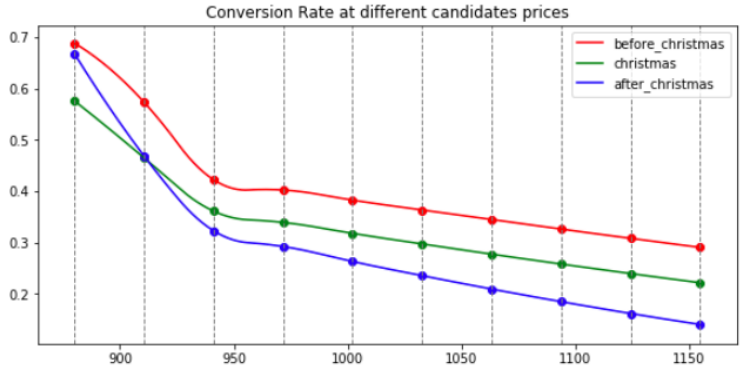
where δ is the confidence and Z is the dataset.

Algorithm

1. For every feature:
    - Split the context c (according to the split condition defined before) if it's worth doing it
    - Evaluate the value v of the context after the split
2. Select the feature with the highest value v if larger than the non split case

4 Experiment Setup

In this work we test our algorithms in the environment described in the introduction. The idea is that we have three different season (behaviour of the customer) and different features for each customer. All these attributes define different demand curves for each customer in each season. Aggregate conversion rates at the different candidate prices (dotted lines) are reported in the figure below (for dis-aggregate plots refer to github/colab).



Furthermore, we decided to run the experiment for a time period of 50 days, considering an average number of 24 customers per day. Thus, the horizon of our experiment (, that is the total number of users), $T \in \mathbb{N}$ is: $T = 50 \times 24 = 1200$ (customers).

Subsequently, we define the number of candidates prices according to the formula:

$K = \lfloor \sqrt[4]{T \cdot log(T)} \rfloor$

The sliding window should be proportional to the square root of the time horizon ($\tau \propto \sqrt{T}$) and cannot depends on unknown parameters.

In this work we decided to use $\tau = 15 \cdot \sqrt{T}$ that is slightly bigger than the phase length (about 40% of the time horizon).

5 Results

We report here (only synthetic) results of the execution of the algorithms for the experiment explained above. For complete results (i.e. instantaneous rewards/regrets and histograms) please refer to the project repository on Github or to the iPython notebook on Google Colab.

5.1 A/B Testing

For showing results we plot the distributions of the hypothesis and the confidence interval of $H0$ in order to graphically show the power of the test; furthermore we plot the confusion matrix for the I- and II- type error probabilities (in %) and calculate the p-value, defined as the area of the normal distribution associated to the null hypothesis given the value of the test statistic z.

As explained above, before running the tests we first set the I- and II- type error probabilities (5% and 20% in this case) and then calculate the standard deviation σ and the minimum difference δ between the probabilities of each pair of candidates, in order to calculate the minimum number of samples needed for each test.

We can observe that when the difference between the candidates tested (A/B) is not very high (as for example with candidates 4 vs. 5, 5 vs. 6 and so on) the number of samples required in order to reach the desired confidence/power is very high (for example variants D vs. C in phase 2 where the difference in revenue is 3% and the number of samples required is **60931**) with respect to tests comparing candidates with bigger differences in probabilities (for instance variants C vs. B in phase 2 where the difference in the total revenue is 25% and the number of samples is **991**).

This essentially points out the fact that A/B Testing requires a very large amount of samples when the difference in probabilities is not big enough and that, more in general, A/B testing is not suitable for pricing problems as it will require a very large amount of samples to converge to the optimum.

Indeed, when using $\frac{T}{2}$ as minimum number of samples in A/B testing (in order to have total number of samples for each test equal to 1200, that is the horizon for other experiments) we get very low precision and high p-values. We also notice that the algorithm does not reach the optimum at all.

We run the experiment for 100 independent runs then we average the results.

5.2 MAB Algorithms

We run for each algorithm 100 independent experiments and then average the results.

Given a policy $\mathfrak{U}$, we define the instantaneous reward and regret as the difference between the optimal candidate (clairvoyant solution) and the expected value of the reward given by the MAB:

- $\mathbb{E}\left[\mu_{a_t}\right]$
- $R_t\left(\mathfrak{U}\right) = T\mu_a^* - \mathbb{E}\left[\mu_{a_t}\right]$

And cumulative (or pseudo-) reward and regret, respectively:

- $\mathbb{E}\left[\sum_{t=1}^{T} \mu_{a_t}\right]$
- $R_T\left(\mathfrak{U}\right) = T\mu_a^* - \mathbb{E}\left[\sum_{t=1}^{T} \mu_{a_t}\right]$

Where μ_a^* is the (optimal) value provided by the clairvoyant algorithm.

5.3 Comparison

We can observe that UCB-1 performs slightly better than TS, because UCB-1 uses the number of times an arm has been pulled during past explorations to estimate the upper bounds and for this reason the algorithms performs very well when the optimal arm does not change over the time horizon.

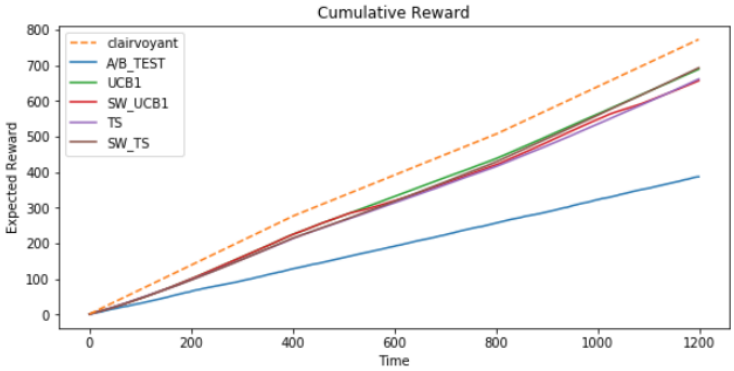
SW_UCB-1 instead, always have worse results compared to UCB-1 (and also among all the algorithms) because of the fact that the algorithm pays an exploration cost each time t reaches the sliding window size ($t \propto \tau$).

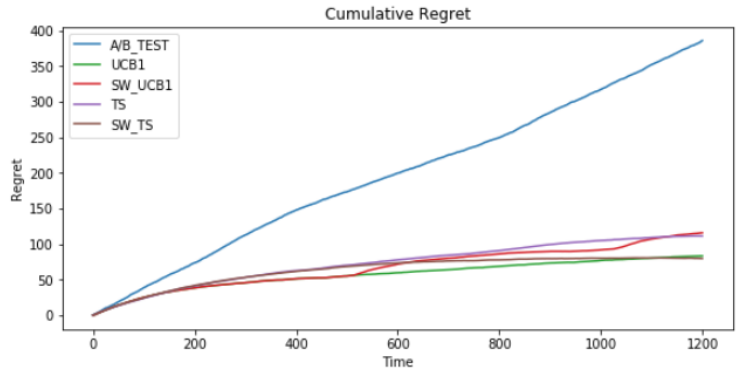
We also observe that SW_TS has slightly lower regret with respect to classical TS.

Moreover, even if at first sight UCB-1 seems to have smaller regret with respect to SW_TS but while the latter starts to have almost constant regret (around $t = 500$), the former's regret keeps increasing. Results could be more evident over a longer time horizon.

In general we can argue that Thompson Sampling gets more benefits with the sliding window, while SW_UCB-1 have worse performances with respect to UCB-1.

As pointed out before, all bandits performs better than A/B testing because the latter requires a lot of samples (, that is a very long time horizon) in order to reach the optimum. This also suggests A/B testing is not very suitable for this kind of problems.

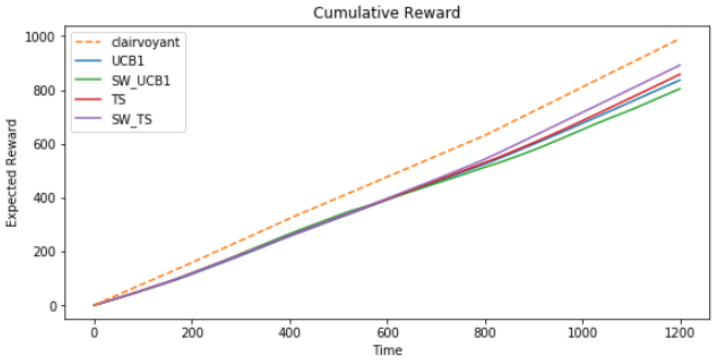


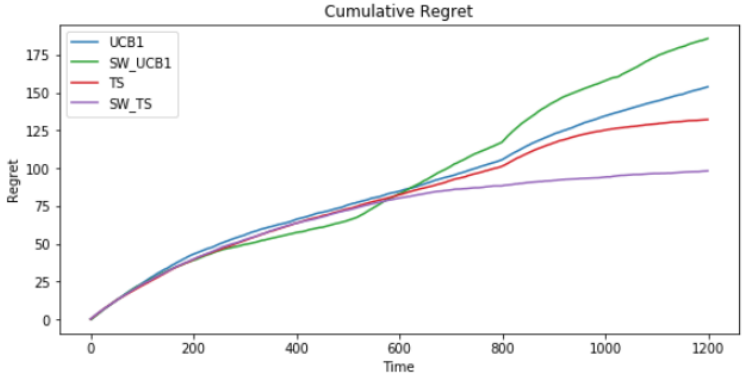


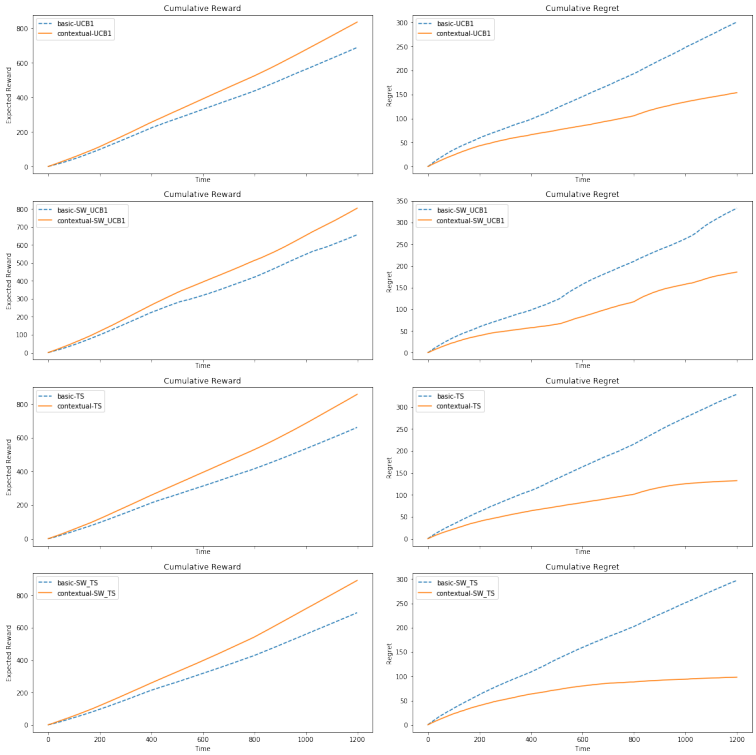
5.4 Algorithms with Context Generation

We decided to apply the context generation algorithm each week, so we have 24 customers per day and then we check if it's worth splitting the context or not each $24 \times 7 = 168$ customers.

In general, when compared with the non contextual version of the algorithms the gain in regret/reward is very significant as can be seen in plots below.






Cumulative Reward
basic-UCB1
contextual-UCB1
Expected Reward
Time
Cumulative Regret
basic-UCB1
contextual-UCB1
Regret
Time
Cumulative Reward
basic-SW_UCB1
contextual-SW_UCB1
Expected Reward
Time
Cumulative Regret
basic-SW_UCB1
contextual-SW_UCB1
Regret
Time
Cumulative Reward
basic-TS
contextual-TS
Expected Reward
Time
Cumulative Regret
basic-TS
contextual-TS
Regret
Time
Cumulative Reward
basic-SW_TS
contextual-SW_TS
Expected Reward
Time
Cumulative Regret
basic-SW_TS
contextual-SW_TS
Regret
Time