Data Intelligence Applications
Pricing and Matching

Ivan Cavadini (941927),
Simone Marforio (944320),
Nicolò Molinari (942404)

2020/2021

Contents

Introduction

0.1 Scenario

Consider the scenario in which a shop has a number of promo codes to incentivize the customers that buy an item to buy a different item. The customers can belong to different classes and the promo codes can provide different discounts.

0.2 Environment

Imagine two items (referred to as first and second items; for each item we have an infinite number of units) and four customers' classes. The daily number of customers of each class is described by a potentially different (truncated) Gaussian probability distribution. Each class is also associated with a potentially different conversion rate returning the probability that the user will buy the first item at a given price.

Once a buyer has bought the item, she/he can decide to buy the second item that can be or not promoted. There are four different promos P0, P1, P2, P3, each corresponding to a different level of discount. P0 corresponds to no discount. Given the total number of customers, the business unit of the shop decides the number of promos as a fraction of the total number of the daily customers and is fixed (use two different settings in your experiments that you are free to choose). Each customers' class is also associated with a potentially different conversion rate returning the probability that the user will buy the second item at a given price after she/he has bought the first. The promos will affect the conversion rate as they actually reduce the price. Every price available is associated with a margin obtained by the sale that is known beforehand. This holds both for the first and the second item. The conversion rates will change during time according to some phases due to, e.g., seasonality.

0.3 Steps

1. Provide a mathematical formulation of the problem in the case in which the daily optimization is performed using the average number of customers per class. Provide an algorithm to find the optimal solution in the offline case in which all the parameters are known. Then, during the day when customers arrive, the shop uses a randomized approach to assure that a fraction of the customers of a given class gets a specified promo according to the optimal solution. For instance, at the optimal solution, a specific fraction of the customers

2

of the first class gets P0, another fraction P1, and so on. These fractions will be used as probabilities during the day.

2. Consider the online learning version of the above optimization problem, identify the random variables, and choose a model for them when each round corresponds to a single day. Consider a time horizon of one year.

3. Consider the case in which the assignment of promos is fixed and the price of the second item is fixed and the goal is to learn the optimal price of the first item. Assume that the number of users per class is known as well as the conversion rate associated with the second item. Also assume that the prices are the same for all: the classes (assume the same in the following) and that the conversion rates do not change unless specified differently below. Adopt both an upper-confidence bound approach and a Thompson-sampling approach and compare their performance.

4. Do the same as Step 3 when instead the conversion rate associated with the second item is not known. Also assume that the number of customers per class is not known.

5. Consider the case in which prices are fixed, but the assignment of promos to users need to be optimized by using an assignment algorithm. All the parameters need to be learnt.

6. Consider the general case in which the shop needs to optimize the prices and the assignment of promos to the customers in the case all the parameters need to be learnt.

7. Do the same as Step 6 when the conversion rates are not stationary. Adopt a sliding-window approach.

8. Do the same as Step 6 when the conversion rates are not stationary. Adopt a change-detection test approach.

0.4   Context Modeling

We have considered a bike shop that sells bikes as first item and Garmin as second item. The optimization problem is based on a time horizon of one year, taking into account three seasons that change the conversion rates of the two items. We split the customers into four categories:

1. Sport addicted: who loves and practices cycling

2. Gifter: who wants to give away the two items

3. Worried: who pays a lot of attention to the price of the items

4. Amateur: who sometimes practices cycling

Formal Model

0.5 Assumption

This is the mathematical formulation for the pure pricing problem of maximization of the total reward. We consider the production costs of both the item equals to zero.

0.6 Variables definition

- $i=$ user class

- $j =$ promo code -> $P0 = 0\%$, $P1 = 10\%$, $P2 = 20\%$, $P3 = 30\%$

- $p1 =$ full price item 1

- $p2 =$ full price item 2

- $p2_j =$ price of item 2 when applied the promo j

- $c1 =$ production cost item $1 = 0$

- $c2 =$ production cost item $2 = 0$

- $q1_i(p1) =$ conversion rate for user class i, for product 1, based on price p1

- $q2_i(p2) =$ conversion rate for user class i, for product 2, based on price p2

- $s_{ji}(p2) =$ discounted price of item 2, for user class i, based on promo j

- $d_{ij} =$ amount of promo j distributed to user class i

- $maxd =$ max number of promos to distribute -> #P1+#P2+#P3

- $avgCustomer_i=$ avg number of customers for class i

0.7 Formulation of elaborated variables

- $p1 * q1_i(p1) * avgCustomer_i =$ revenue at price p for product 1

- $s_{ji}(p2) * q2_i(s_{ji}(p2)) * d_{ij} * avgCustomer_i =$ revenue for the couple promo-class for item2

- $(p1 * q1_i(p1) - c1 * q1_i(p1)) * avgCustomer_i =$ reward item 1

- $(q2_i(p2) * (s_{ji}(p2) * q2_i(s_{ji}(p2)) * d_{ij} - q2_i(s_{ji}(p2))) * c2) * avgCustomer_i =$ reward item 2

0.8 Objective Function

$$\max\left(\sum_{i=0,j=0}^{i=4,j=4}[(p1 * q1_i(p1) - c1 * q1_i(p1) + q2_i(p2)(s_{ji}(p2) * q2_i(s_{ji}(p2)) * d_{ij} - q2_i(s_{ji}(p2)) * c2)) * avgCustomer_i]\right)$$

$$\textbf{s.t: } \forall j > 0 : [\sum_{i=0}^{i=4} d_{ij}] = maxd$$

We have fixed the full prices of the two items: $p1$, $p2$. We retrieve the discounted prices of $p2$, applying the promos j.
 We know: the average number of customers per class i $avgCustomer_i$, the conversion rate for both products ($q1_i(p1)$, $q2_i(p2)$) and the maximum number of promos to distribute ($maxd$). We are assuming that the production costs of the two items are zero ($c1$, $c2 = 0$). It is possible to retrieve the total revenue for product 1 as the product between the price of the item 1, the conversion rate for the considered class and the average number of customers for that class: ($p1 * q1_i(p1) * avgCustomer_i$). For the second item the calculation of the reward is the same except for the fact that the product is buyed only if also the first one is purchased (so we multiply also the conversion rate of the first item) and the price is discounted by the selected promo.

The solution of our optimization problem consists in the distribution of the fraction of promo codes among the user classes.

0.9 Promo Assignment Assumption and Implementation

Now, we consider the constraints that the shop uses a randomized approach to assure that a fraction of the customers of a given class gets a specified promo according to the optimal solution. We achieve the solution of the problem using a matching approach. To reach the optimal solution we have used an iterative approach: we build a matrix class promotion containing the mean expected rewards for every couple, calculated

as the conversion rate of the second item multiplied with its discounted price. We exploit this matrix to perform the matching between the customer classes and the promo types, in order to maximize the total reward. We select the best reward for every class, for 4 times, retrieving, at every iteration, the 4 combination promo-class and assigning an infinite weight to the obtained sub-optimal matching. Every matching is represented by a reward configuration that maximize the total reward. Every iteration is weighted and represent a different goodnesses of the solution, the first is the best, the last is the worst. Through the sub-optimal matchings, we have retrieved the fractions of different promos to assign to every customer class, based on the proportional weight of the previous sub-optimal matching. The proportions retrieved, are normalized class per class.

Random Varaibles

Daily customers: Gaussian Distribution
Normalized gaussian parameters per class (normalizing factor:1000), average and variance:

Customer Category	Average	Variance
Category 1	0.15	0.03
Category 2	0.20	0.03
Category 3	0.40	0.05
Category 4	0.25	0.04
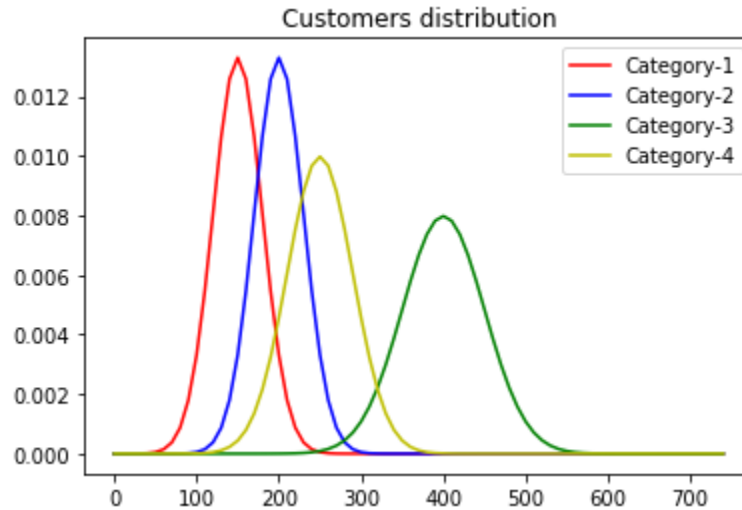
- Buy item1(price) : Bernoulli $\tilde{0}$,1

- Buy item2(price) : Bernoulli $\tilde{0}$,1

- Reward item1 = buy item 1 * price item 1

- Reward item2 = buy item 1 * buy item 2 * price item2

- Time horizon = 365 days

The following graphs are the demand curves of the two item: the first three graphs are the demand curves of the first item, associated to the three seasonality (spring-summer, autumn, winter); the last three graphs are the demand curves of the second item with its respective seasonality.

Conversion rate: second item
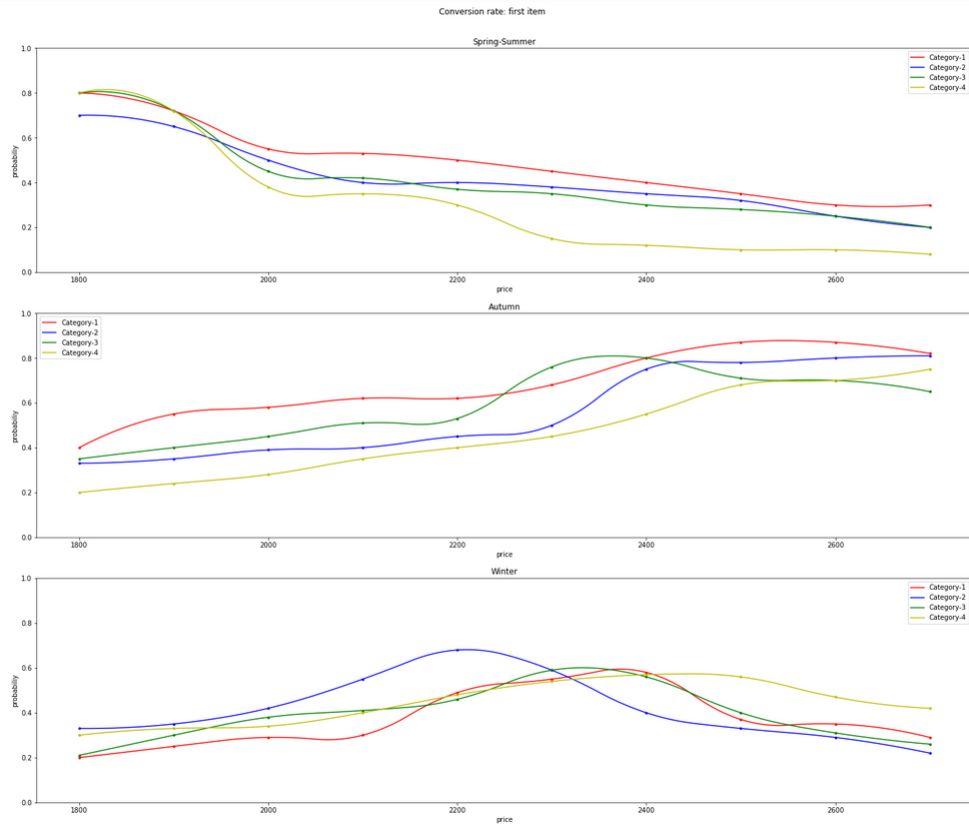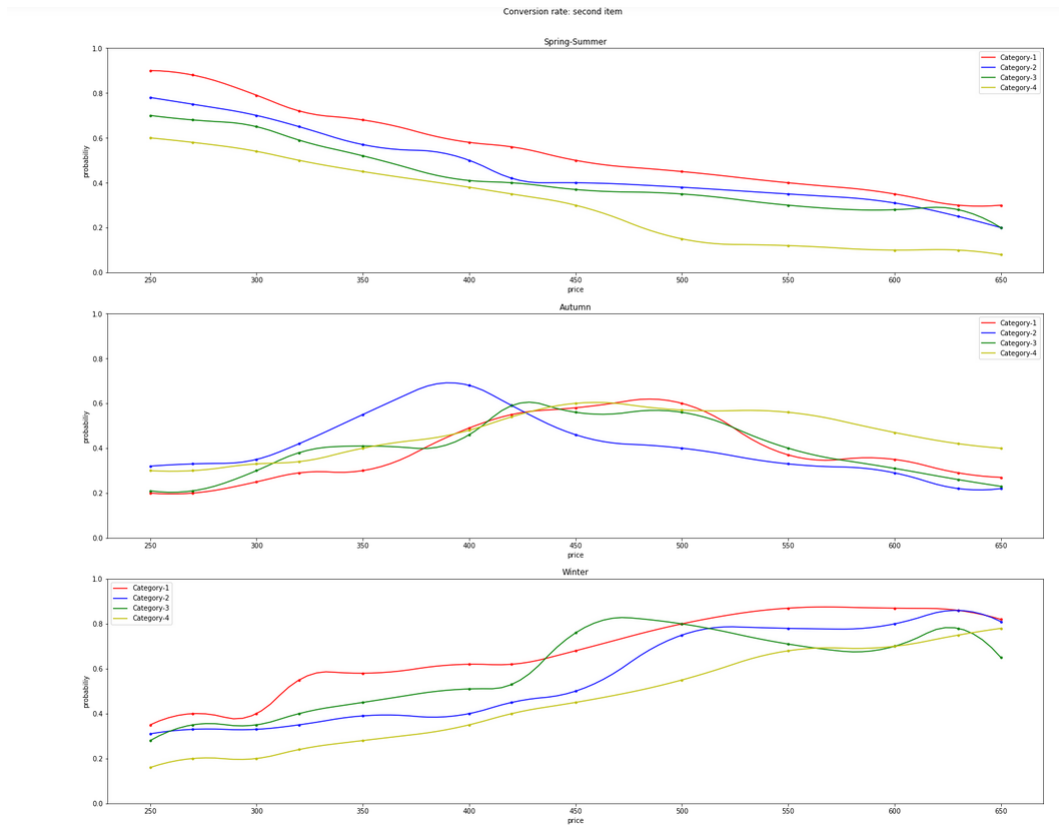
0.10   Online Approach

Daily we extract the items prices using an online learner algorithm (pricing or matching), then we simulate the current day estimating the number of customers per class through the gaussian distribution. The arrival of every customer is simulated with a random approach. The system proposes the first item to every customer, simulating the purchase of it exploiting the associated Bernoulli distribution probability. The second item is proposed to the customer with the same approach, only if the first item will be bought. It is possible to exploit the information obtained in the previous steps to calculate the rewards and use them to update the learner. This procedure is repeated for every daily customer, for a time horizon of 365 days and during this period the purpose is to find a solution (composed by the prices and the assignment of promotions) that maximise the total reward.

Experiment Context

Online pricing for first item

Problem explaination   The problem is to learn the optimal price of the first item, especially comparing the adoption of a Thompson-sampling approach and an upper-confidence bound approach, in the following scenario:

- Assignment of promos is fixed

- Price of second item is fixed

- Number of users per class is known

- Conversion rate associated with second item is known

- Prices are the same for all the classe

- Conversion rates do not change

Strategy   The problem described is a combinatorial bandit problem,which is a decision making problem in which the decision maker selects one single arm in each round, and observes a realization of the corresponding unknown reward distribution. Each decision is based on past decisions and observed rewards. The objective is to maximize the expected cumulative reward over some time horizon by balancing exploitation and exploration. We can solve it, through an online algorithm, where only for the feasible solutions we will have precise estimations. We implement a script that works on a time horizon of ten days, repeating the experiment ten times. We simulate a random arrival of the customers, providing to them the prices given by the two learners (UCB and TS) for the first item and simulating the purchase of it. In case the customer buys the first item, we retrieve the reward of the second item as the product between the conversion rate of that customer and the respective discounted price We calculate the expected reward and we update the learners with the results.

Upper-Confidence Bound (UCB1)   Main idea:

- Every arm is associated with an upper confidence bound

- At every round, the arm with the highest upper confidence bound is chosen

- After having observed the realization of the reward of the arm, the upper confidence bound is updated

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

Thompson Sampling (TS)   Main idea:

- For every arm, we have a prior on its expected value

- In the case the arms' rewards are Bernoulli distribution, the priors are Beta distributions

- Notice that, with the opportune parameters, a Beta distribution is a uniform distribution

- For every arm, we draw a sample according to the corresponding Beta

- We choose the arm with the best sample

- We update the Beta distribution of the chosen arm according the observed realization

Notation (in addition to classical UCB):

- $\mathbb{P}(\mu_a = \theta_a)$ 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
$(\alpha_{a_t}, \beta_{a_t}) \leftarrow (\alpha_{a_t}, \beta_{a_t}) + (x_{a_t,t}, 1 - x_{a_t,t})$

Results

Considerations

Online pricing for first item with purchase simulation

Problem explaination   The goal is the same of the previous problem, but there are few changes in the scenario:

- The conversion rates associated to the second item are not known

- The number of customers per class is not known

Strategy   The script that we have implemented is similar to the previous one: it works on a time horizon of ten days, repeating the experiment ten times. We simulate a random arrival of the customers, providing to them the prices given by the two learners (UCB and TS) for the first item and simulating the purchase of it. In case the customer buys the first item, instead of retrieving the reward of the second item as the product between the conversion rate of that customer and the respective discounted price, we simulate the purchase of it. We calculate the expected reward and we update the learners with the results. The algorihm that we have used are the same of the previous one: UCB and TS.

Results

Considerations

Matching problem: promo assignment

Problem explaination   The problem requires to optimize the assignment of the problem using an assignment algorithm, in the following scenario:

- The prices are fixed

- All the parameter need to be learnt

Strategy   We simulate the randomly arrival of the customers and the purchase of the first item. In case of purchase of the first item, we retrieve the optimal matching for the user categories from the learner (UCB matching) and we propose the second item to him, at the discounted price based on the matching suggested by the learner. The learner provides us a set of arms, so we cannot update it with the reward given by the single customer. We introduce two matrixes of shape (|User category |, |Type of promo|), every cell represents the matching between a specific customer category and a specific promo.The first matrix contains the sum of the rewards for that matching, the second contains the number of occurences that the matching has been chosen. At the beginning of the simulation, we use the first 1000 samples (customers) to initialize the two matrix, forcing all the possible combinations of matching. In this way we can update the learner, customer by customer, using the average reward for every matching.

UCB Matching

Results

Considerations

Pricing and Matching problem

Problem explaination

Strategy

Results

Considerations

Seasonal Pricing and Matching problem: Sliding Window

Problem explaination

Strategy

Results

Considerations

Seasonal Pricing and Matching problem: Change Detection

Problem explaination

Strategy

Results

Considerations