Data Intelligence Applications

Pricing and Matching project – A.Y. 2020/2021

Letizia Brambilla, Francesca Pietrobon, Francesco Emanuele Stradi, Diego Savoia

Scenario

We considered as first item an Apple Watch, and as complementary second item a personalized wristband for the watch. We set up a Google Form in order to retrieve realistic data about the behaviour of potential customers, proposing a price of €300 for the first item, and a price of €50 for the second item; we considered as average daily customers the number of submissions and, given the answers, we computed the conversion rates. We opted for the following customer classes subdivision:

- Class 1: females up to 35 years old.
- Class 2: males up to 35 years old.
- Class 3: females older than 35 years old.
- Class 4: males older than 35 years.

As concerns the promos related to the second item:

- P0: no discount.
- P1: 10% discount.
- P2: 20% discount.
- P3: 50% discount.

Promos are received by all the customers that purchase the first item; trivially, giving promo P0 means that that specific customer will not receive a discount.

Note that, in the project steps, we considered vectors of candidate prices (and margins) for both the items, and related vectors of conversion rates, starting from the form data. Therefore, the optimal prices found in the project steps will not be necessarily the ones reported above.

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

This step is about solving the following pricing and matching problem in an offline manner.

Inputs:

- Vector of prices of the first item = [150, 200, 300, 400, 450]
- Vector of margins of the first item = [50, 75, 100, 125, 150]
- Vector of prices of the second item = [40, 50, 60]
- Vector of margins of the second item = [15, 20, 25]
- Conversion rates of the first item given the customer class.
- Conversion rates of the second item given the customer class and the received promo.
- Average number of daily customers for each class.
- Number of promos available for each type of promo, computed as a fraction of the buyers of the first item.

Outputs:

- Optimal price of the first item.
- Optimal price of the second item.
- Optimal matching, represented as the number of each type of promo given to each class of customers.
- Related total revenue.

Algorithm:

```
FOR EACH price_item1:
        FOR EACH price_item2:
                revenue_item2, matching = LINEAR_PROGRAM(margin_item2[price_item2],
                discounts, conversion_rates_item2[price_item2], daily_promos, customers *
                conversion_rate_item1[price_item1])
                total_revenue = revenue_item2 + revenue_item1
RETURN max(total_revenue), (price_item1, price_item2, matching) related to max(total_revenue)
```

LINEAR_PROGRAM (AMPL style since in Python it is less readable):

```
set I;                          # Promo codes (P0, P1, P2, P3)
set J;                          # Customer classes (C1, C2, C3, C4)

param r{I, J};                  # Conversion rates
param margin_item2;             # Margin of item 2
param P0;                       # Discount P0
param P1;                       # Discount P1
param P2;                       # Discount P2
param P3;                       # Discount P3
param numPromos{I};             # Number of promo codes available
param numCustomers{J};          # Number of daily customers

var n{I, J} >= 0;

maximize objective: (sum{j in J} (n["P0", j] * r["P0", j] * (1-P0)) +
                     sum{j in J} (n["P1", j] * r["P1", j] * (1-P1)) +
                     sum{j in J} (n["P2", j] * r["P2", j] * (1-P2)) +
                     sum{j in J} (n["P3", j] * r["P3", j] * (1-P3))) * margin_item2;

subject to numberOfPromos{i in I}: sum{j in J} n[i, j] = numPromos[i];

subject to numberOfCustomers{j in J}: sum{i in I} n[i, j] <= numCustomers[j];
```

Note that, even if from a conceptual perspective it should be an Integer Linear Program, we decided to opt for the standard Linear Program since the resulting matching matrix will be used as a weight matrix in the online steps of the project. Anyway, results are casted to int to be more readable.

Results:

```
Experiment 1: fractions [0.7  0.2  0.07 0.03]
Revenue: 47553.89487156687
Optimal price item 1: 400
Optimal price item 2: 60
Optimal assignment of promos to customer classes:
     Class1  Class2  Class3  Class4
P0     150       6      95       0
P1       0      58       0      12
P2       0       0       0      24
P3       9       0       0       0
```

```
Experiment 2: fractions [0.6  0.25 0.1  0.05]
Revenue: 48452.26627766729
Optimal price item 1: 400
Optimal price item 2: 60
Optimal assignment of promos to customer classes:
     Class1  Class2  Class3  Class4
P0     163       5      52       0
P1       0      69       0      22
P2       0       0      36       0
P3       0       0       0      17
```

Step 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.

Inputs:

- Time horizon = 365 days.
- Round = 1 day.
- Vector of prices of the first item = [150, 200, 300, 400, 450]
- Vector of margins of the first item = [50, 75, 100, 125, 150]
- Vector of prices of the second item = [40, 50, 60]
- Vector of margins of the second item = [15, 20, 25]

Outputs:

- Optimal price of the first item.
- Optimal price of the second item.
- Vector of total revenues for each round of the time horizon.

Variables to learn:

- Conversion rates of the first item.
- Conversion rates of the second item.
- Number of daily customers.
- Optimal price of the first item.
- Optimal price of the second item.
- Daily matching.

General idea (that will be deepened in the next steps using Bandits):

For each round:

1. Update the estimation of the conversion rates, optimal prices, and daily customers.
2. Solve the linear program.
3. Normalize the resulting matrix of the matching.
4. Assign promos to the customers of the next day using the previously computed normalized matrix.

TODO: maybe explain how the environment works.

Step 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.

Inputs:

- Optimal price (margin) of the second item.
- Average daily number of customers for each class.
- Conversion rates of the second item.
- Assignment of promos (weight matrix).

Outputs:

- Revenue related to the learning of the optimal price of the first item.
- (Estimate of the conversion rates of the first item).

Models:

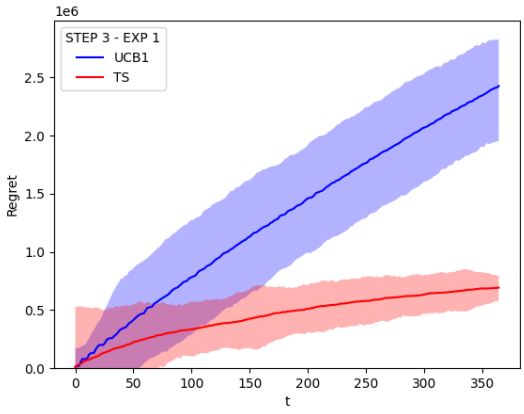
- UCB1:
    - Arms: margins of the first item.
    - Empirical mean: estimation of the conversion rates of the first item.
    - Confidence: standard confidence of the UCB1 Bandit.
    - Upper bound: revenue given the estimation of the conversion rates of the first item (empirical mean plus confidence) and the known optimal parameters received as inputs.

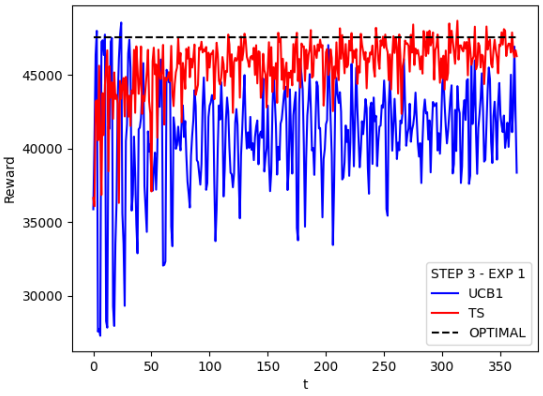
- Thompson Sampling:
    - Arms: margins of the first item.
    - Beta distribution: estimation of the conversion rates of the first item.
    - Pulled arm: based on the revenue given the estimation of the conversion rates of the first item (extraction from Beta distribution) and the known optimal parameters received as inputs.

Results:

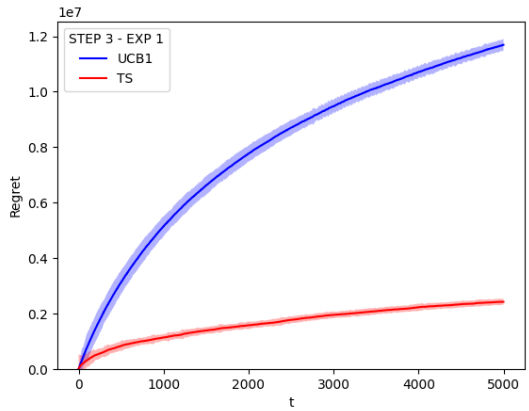
Note that the variance of the regret is multiplied for a large constant in order to make it more visible.

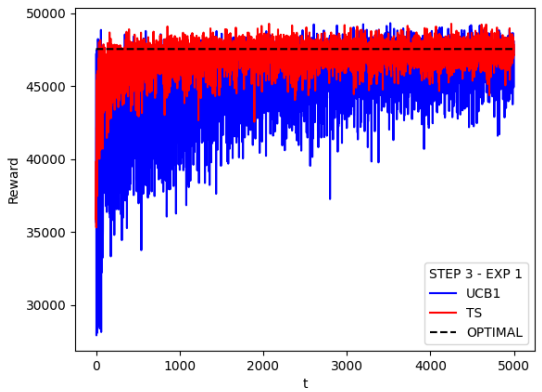
First experiment





Time horizon = 365





Time horizon = 5000

Second experiment

ADD IMAGES

Step 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.

Inputs:

- Optimal price (margin) of the second item.
- Assignment of promos (weight matrix).

Outputs:

- Revenue related to the learning of the optimal price of the first item.
- (Estimate of the conversion rates of the first item).
- (Estimate of the conversion rates of the second item).
- (Estimate of the number of daily customers).

Models:

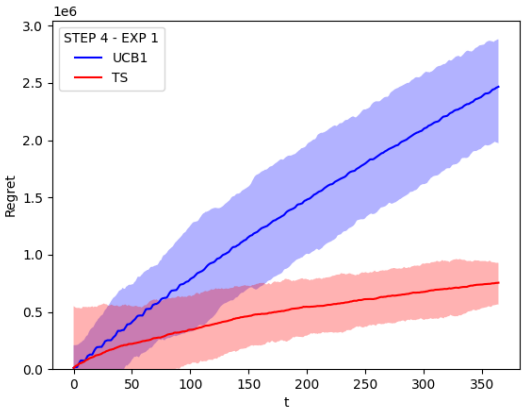
- UCB1:
    - Arms: margins of the first item.
    - Empirical mean: estimation of the conversion rates of the first item.
    - Confidence: standard confidence of the UCB1 Bandit.
    - Upper bound: revenue given the estimation of the conversion rates of the first item (empirical mean plus confidence), the estimation of the conversion rates of the second item (computing the mean of the daily conversion rates of the previous rounds) and number of customers (computing the mean of the daily customers of the previous rounds), and the known optimal parameters received as inputs.

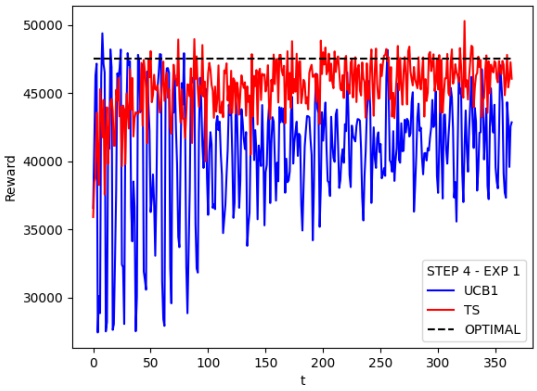
- Thompson Sampling:
    - Arms: margins of the first item.
    - Beta distribution: estimation of the conversion rates of the first item.
    - Pulled arm: based on the revenue given the estimation of the conversion rates of the first item (extraction from Beta distribution), the estimation of the conversion rates of the second item (computing the mean of the daily conversion rates of the previous rounds) and number of customers (computing the mean of the daily customers of the previous rounds), and the known optimal parameters received as inputs.

Results:

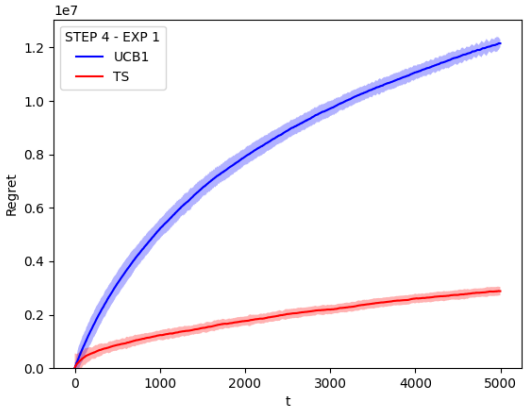
Note that the variance of the regret is multiplied for a large constant in order to make it more visible.

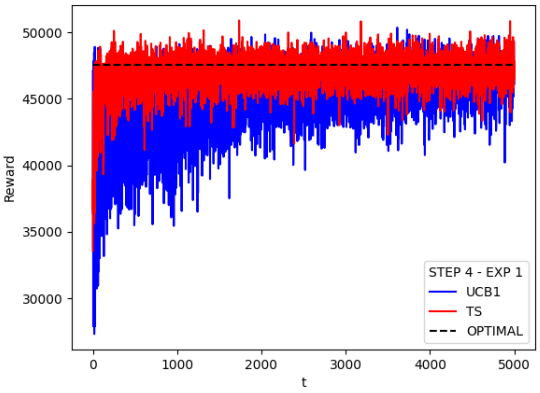
First experiment





Time horizon = 365





Time horizon = 5000

Second experiment

ADD IMAGES