DATA INTELLIGENCE APPLICATIONS PRICING & ADVERTISING

DATA INTELLIGENCE APPLICATIONS

Andrea Bionda
Department of Computer Science and Engineering
Politecnico di Milano
andrea.bionda@mail.polimi.it

Damiano Derin
Department of Computer Science and Engineering
Politecnico di Milano
damiano.derin@mail.polimi.it

Andrea Diecidue
Department of Computer Science and Engineering
Politecnico di Milano
andrea.diecidue@mail.polimi.it

Antonio Urbano
Department of Computer Science and Engineering
Politecnico di Milano
antonio.urbano@mail.polimi.it

Enrico Voltan
Department of Computer Science and Engineering
Politecnico di Milano
enrico.voltan@mail.polimi.it

October 20, 2020

ABSTRACT

The goal of the project is to model a scenario in which a seller exploits advertising tools in order to attract more and more users to its website, thus increasing the number of possible buyers.

The seller has to learn simultaneously the conversion rate and the number of users the advertising tools can attract.

In this report, we walk through the description of the specific scenario we have studied and the definition of the algorithm design choices we adopted in order to reach our goal. Then the achieved experimental results will be presented via some useful plots and a final conclusion summarizing the whole work.

Contents

Chapter 1

Introduction: Scenario Setup

We have studied a possible real world scenario in which a seller wants to increase the number of possible buyers by exploiting advertising tools.

The product we have considered is a particular pair of shoes which has a production cost (without loss of generality we can assume that the production cost is null) and a sell price.

The analysed campaign consists of three sub-campaigns, each with a different ad to advertise the product and each targeting a different class of users. Each class is defined by the values of pre-defined features.

The feature space we have considered is characterized by two binary features:

- *Age<30* or *Age>30*
- *Profession* that can be either *student* or *worker*

So according to the values of the above described features, we can distinguish among the following classes of users:

- *Elagant*: a worker with age>30
- *Casual*: a student with age<30
- *Sport*: a worker with age<30

Chapter 2

Advertising: Clicks Maximization

2.1 Algorithm Design Choices and Results

This section deals with the description of the algorithm design choices we adopted and it also contains the most relevant reference to the code. For the description we will adopt the following structure: for each sub-section we will describe the actual scenario we are interested in by defining its goals and then we focus on how we have solved the problem and the results we have obtained.

Chapter 3

Advertising: Handling Abrupt Phases

Design a sliding-window combinatorial bandit algorithm for the case, instead, in which there are the three phases aforementioned. Plot the cumulative regret and compare it with the cumulative regret that a non-sliding-window algorithm would obtain.

Chapter 4

Pricing: Learning the Best Price

In this chapter we are going to focus our attention on the pricing and on the optimal price, that we can propose to our customers, to maximize the total revenue.

In common scenarios, in which there is a seller that has to sell products, the retail price is fixed without exploiting information from the buyers.
Sometimes, in physical markets, the customers have the possibility to negotiate and a skillful seller can maximize his revenue keeping the price as high as possible. This is a common practice, but experience is needed, to understand the limit of the buyers, and moreover it requires time, loosing the possibility to serve other clients.
In e-commerce scenarios, we can do better exploiting information from the users, from the big data and from the software automation. In particular, given the assignments 8, we are going to design an algorithm being able to learn the optimal price for our product, to maximize the total revenue.

In the following steps, we are going to:

- define the demand curves of the users;
- describe the environment setup;
- explain the algorithm that has been chosen; and
- comment the results obtained in the end of the campaign.

4.1 Demand Curves

In the economic literature is defined the concept of **demand curve**. It is a function that maps the price to a certain probability. In other words, given a price, the function returns the probability that the product will be sold. Therefore, if we plot the demand curve, we'll obtain on the x-axis the admissible prices and on the y-axis the probability (between 0 and 1).

In our simulated scenario, we defined three classes of users with the corresponding demand curves. The shapes of the curves are motivated by the intrinsic behaviors of the classes:

- *Elegant*: they are rich, thus they will buy the product with high probability whatever is the proposed price;
- *Casual*: they are students, thus they are able to buy the product only if the price is relatively low; and
- *Sports*: they are workers, but without a particular interest on the shoes, therefore the price is relatively irrelevant and the probability is low.

In figure 4.1 there are the curves designed following the above descriptions. [1]

In the same figure, the *aggregate* curve has been reported (colored in *orange*), it represents the mean between the other curves. (It will be useful in the next subsections.)
Moreover, points representing the theoretical optimum are reported. The area below them is maximum w.r.t. the curves.

[1]A tool as been implemented to draw the curves manually. It requires few points and it performs polynomial regression to augment the sampling definition. It can be found in the attached code.

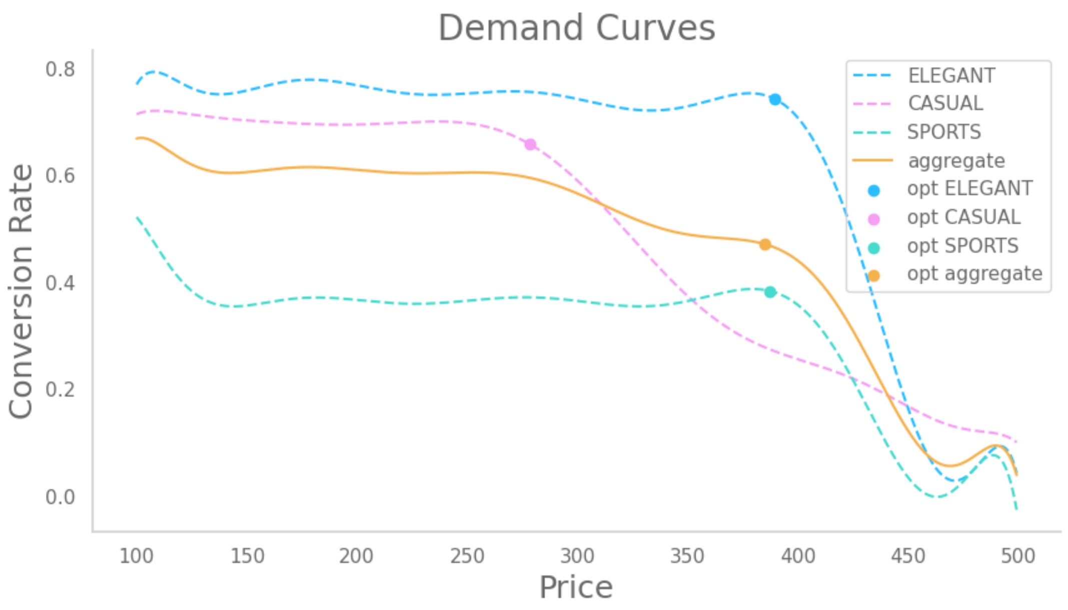


Figure 4.1: **Demand Curves**.
They are used to model the probability (*conversion rate*) that a certain class, of customers, will buy or not the product w.r.t. a proposed price.
The dashed lines represent the demand curves of the corresponding classes: *blue* for the *Elegant* class; *pink* for the *Casual* class; and *light green* for the *Sports* class.
The *orange* curve is the mean between the others and it is used to study the *aggregate* model.
The points are displayed to show the optimal price w.r.t. the conversion rate: the area below them is maximum.

4.2 Environment

In our implementation, the environment is a black box from the point of view of the learning algorithm. It is used to iterate over the days, for the entire duration of the campaign. It schedules the time and the number of users reaching our theoretic website.

Every user is randomly taken from the three classes and the class is not communicated to the learner. The environment, knowing the original class of the users, computes the optimal revenue and returns the effective reward obtained by the learner.

How the learner will handle these information will be explained in the next subsections.

4.3 Learner Algorithm Selection

This context of application can be solved in different ways, but the most common choice is to use a *multi-armed bandit (MAB)* algorithm.

We mainly tested two algorithms:

- **Thompson Sampling (TS)** algorithm; and
- **Thompson Sampling with Sliding Window (SWTS)** algorithm.

We tested both the algorithms during the implementation phase, but since the SWTS is more sensible to the window size, and finding a good one requires a lot of testing, we kept the TS as final choice.
This decision has been motivated by the fact that there aren't abrupt phases, thus there is no reason to "forgot the past" as done by SWTS.

4.4 Experiments

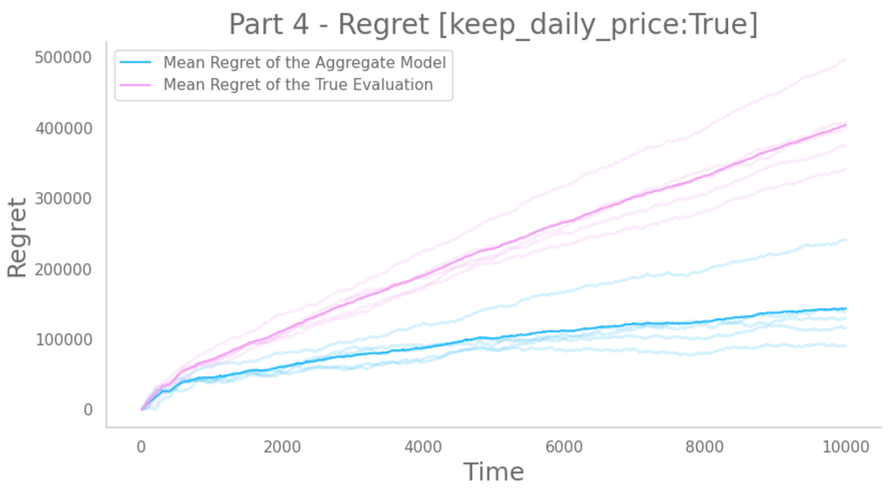
Summarizing, we have to learn the best price to propose, to the users, to maximize the total revenue.
We have discussed about the environment, that generates incoming users and allows us to iterate over the days of the campaign, and finally, we presented the learner adopted.
At this point we can explain how the tests have been initialized.

The experiments has been configured in two ways:

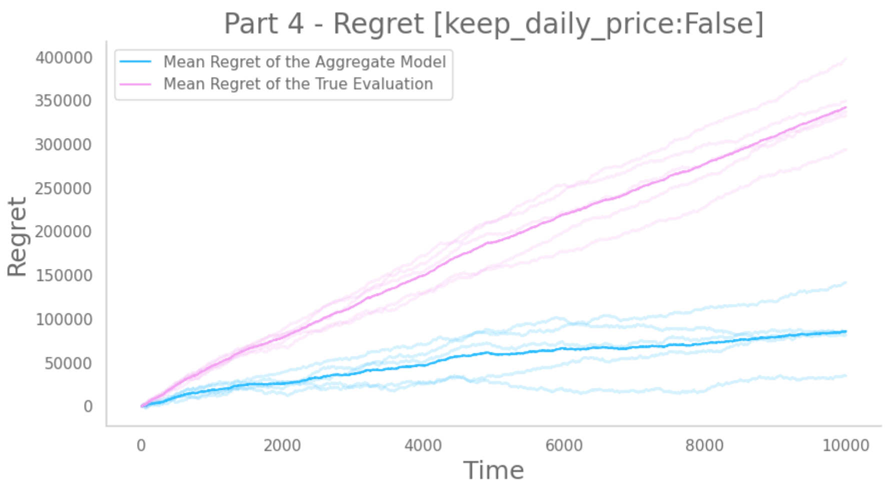
- keeping a fixed daily price: we pulled an arm only for the first user of the day; for the other users we proposed always the same price. This means that for the entire day we pull and update always the same arm of the TS;
- one price for each user: maybe, this is an unpractical option in a real world scenario, but it allows to obtain a total revenue outperforming the previous configuration.

In the end of the tests, the figure 4.2 reports the obtained regrets. The sub figure *(a)* for the first configuration and the sub figure *(b)* for the second.

We plotted two regrets computed with two different optimal revenues. The regret called **Mean Regret of the Aggregate Model** has been computed using, as optimal, the optimal point taken from the aggregate model. This is an optimistic regret because it doesn't take into consideration the information about the specific class of the users. In other words, the learner and the clairvoyant algorithm don't know the class of the customers. The learner is learning the aggregate model, thus this comparison seams fair, but
since the clairvoyant knows the original class of the users, we introduced the so called **Mean Regret of the True Evaluation**. Obviously, from the point of view of the learner, this is a pessimistic computation since it is unable to infer more information from the input data.
Anyway, these two ways, to compute the regrets, will be useful in the next chapter, since the learner will be able to infer more information from the data, creating contexts to exploit a finer profiling.
You will notice how the two regrets will be pushed down and the regret on the True Evaluation will be more similar to the regret of the Aggregate Model obtained here, proving that a finer profiling improves drastically the performances.



(a) Regret obtained proposing to the customers the same price for the entire day.



(b) Regret obtained proposing to the customers different prices for each visit. This method achieves better performances since the learner collects more data during the campaign.

Figure 4.2: Comparison between the regrets obtained during the campaign.
The **Mean Regret of the Aggregate Model** (colored in *blue*) is the regret computed keeping, as optimal, the area under the optimal point of the aggregate curve 4.1. In other words, both the learner and the clairvoyant don't know the class of the customers. This is an optimistic measure of the regret.
The **Mean Regret of the True Evaluation** (colored in *pink*) is the regret obtained computing the optimal value exploiting the original class of the customers.

Chapter 5

Pricing: Context Generation

In this part the assignment was to create a context generation algorithm for the pricing when each sub campaign had a fixed budget allocated to it. Contrary to the previous part, in which we had a single price for all three sub campaigns, here it is possible to have different prices for each context.

5.1 What is a Context?

The idea in this type of problems is that each class of users is identified by a single subset of features. In real problems we have a lot of features and so we can identify many classes of users. Context are useful because we can group these classes together based on the values of their features and we can use a price for each single context, instead of one for each single class, which might be unfeasible if the feature space is very large.

5.2 Our Approach

When we first approached the problem, we used an algorithm similar to that presented in class, with a tree structure. The idea is that we collect data for some time using the aggregate model of the previous part, then we use this data to make predictions. At the end of the week the algorithm chooses the most promising feature (the one with the highest expected reward) and splits accordingly, creating a new binary subtree, with one context for each leaf node of the tree. This process is repeated using the newly generated contexts instead of the ones used in the previous week, until it is not useful to split anymore (the expected reward is lower than without splitting or there are no more features).

However our problem has very little features (only 2) with very limited values (both are binary), so we chose a different approach. We always start with the aggregate model to collect data, but when the time to split arrives, we evaluate all possible partitions of our feature space, either obtaining again the aggregate model, splitting according to only one feature or splitting according to both features, thus producing one, two or three different contexts (it would be four but we only have three classes).

The graphs above measure the regret with respect to two different models: the Aggregate Model (we have only one price for each class) and the True Evalutation (where we use different prices for each class). As we can see, the algorithm performs very well with respect to the Aggregate Model, obtaining better results. On the other hand, the True Evaluation model performs better than the algorithm in the first weeks, but as time goes on the difference becomes smaller and smaller, as the graph grows more and more slowly.

Chapter 6

Adverting and Pricing: Combining the Algorithms

Design an optimization algorithm combining the allocation of budget and the pricing when the seller a priori knows that every subcampaign is associated with a different context and charges a different price for every context. Suggestion: the value per click to use in the knapsack-like problem depends on the pricing, that depends on the number of users of a specific class interested in buying the product. Notice that the two problems, namely, pricing and advertising, can be decomposed since each subcampaign targets a single class of users, thus allowing the computation of the value per click of a campaign only on the basis of the number of clicks generated by that subcampaign. Plot the cumulative regret when the algorithm learns both the conversion rate curves and the performance of the advertising subcampaigns.

Chapter 7

Advertising and Pricing: Fixed Prices

Do the same of Step 6 under the constraint that the seller charges a unique price to all the classes of users. Suggestion: for every possible price, fix this price and repeat the algorithm used in Step 6. Plot the cumulative regret when the algorithm learns both the conversion rate curves and the performance of the advertising subcampaigns.

Chapter 8

Assignments

Part 4: *Design a learning algorithm for pricing when the users that will buy the product are those that have clicked on the ads. Assume that the allocation of the budget over the three sub campaigns is fixed and there is only one phase (make this assumption also in the next steps). Plot the cumulative regret.*