POLITECNICO
MILANO 1863

Data Intelligence Application

Online Advertising

Bonfanti Luca, Crippa Dominic, Krasniqi Filip, Pina Alessandro

December 16th, 2019

Abstract

Online advertising is a problem that has become very popular since the digitalization of our society.

In a typical online advertising setting we have 3 different **roles**:

- the *user/customer*, the people that are targeted by the ads;
- the *publisher*, whoever owns the platform on which the ads are published;
- the *advertiser*, whoever wants to publish the ads.

There are many different formats for an advertising campaign, but we focus our attention on **search** and **social advertising**.

For both of these formats we choose as payment schema the *cost per click schema, i.e.* the advertiser pays the publisher only when the targeted user clicks on the ad displayed.

The **goal** of this report is to show an **algorithm** that maximizes the reward provided from clicks as a function of the budget in a simulated scenario containing different sub-campaigns, and an **algorithm** that selects how to group the users assuming that they are characterized by different features that imply different reward curves.

Environment

Description of the product (question 1)

One of the most selling garments in fashion brands are **shoes**. Many e-commerce websites, like Zalando, have indeed moved their core business to shoes selling.
For this reason, we decided to simulate an advertising campaign focused on shoes.

For the sake of simplicity, we consider just a small subset of all the possible types of shoes available in the market: *sneakers, football shoes* and *shoes for kids.*

Users: definition, disaggregate curve, aggregate curve (question 2)

We identified three **classes of users**, briefly explained below.
Each of them is characterized by the following set of **features**: *age, gender, interests, location.*

Class	Location	Age	Gender	Interests
User 1 U_1	**Big Cities**	**15-25**	**Male and Female**	**Streetwear**
User 2 U_2	**Irrelevant**	**10-35**	**Male**	**Sports**
User 3 U_3	**Irrelevant**	**20-40**	**Female**	**They're mothers, they've child**

Channel, ad		Targeting			Economics
Channel	**Ad**	**Place**	**Time**	**User's info**	**Bid**
Google	Sneakers	Milan	Morning	Cookies	€ 5
Google	Kids shoes	Italy	Afternoon	Cookies	€ 1
Facebook	Football shoes	Italy	Afternoon	Interest	€ 3
Facebook	Sneakers	Barzanò	Afternoon	Interest	€ 1
Google	Football shoes	Lazio	All day	Cookies	€ 2

In the table above we divide the advertising campaign into **five sub-campaigns**.
These sub-campaigns differ depending on the *channel of distribution*, *ad type*, *ad location*, *time to display the ad* and the *source of users information*.
In addition, for each sub-campaign, we defined the value of a single bid for the associated ad.

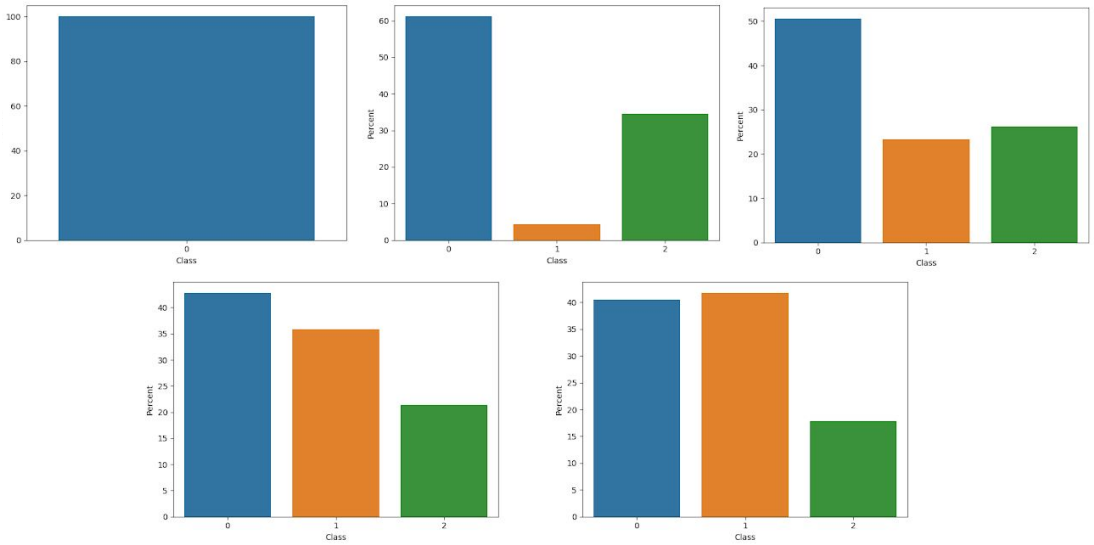
To characterize the curves, we show in the following table the parameters that explain how the curves of the different sub-campaigns are formalized.

	Users	P	Bid	Slope	Max clicks	Max budgets	V
Sub campaign 1	**User 1**	1	15 €	0.05	2	200	1
	User 2	0	10 €	0.03	1		
	User 3	0	20 €	0.01	1		
Sub campaign 2	**User 1**	0.2	20 €	0.03	1.5	200	1.3
	User 2	0.1	30 €	0.01	1		
	User 3	0.7	15 €	0.04	2.5		
Sub campaign 3	**User 1**	0.3	15 €	0.04	2	140	1.6
	User 2	0.6	25 €	0.04	2.5		
	User 3	0.1	30 €	0.01	1		
Sub campaign 4	**User 1**	0.2	27 €	0.05	2	80	1.9
	User 2	0.7	22 €	0.05	2.5		
	User 3	0.1	14 €	0.02	1		
Sub campaign 5	**User 1**	0.3	13 €	0.03	1.5	200	2.2
	User 2	0.65	29 €	0.03	2		
	User 3	0.05	33 €	0.01	1		

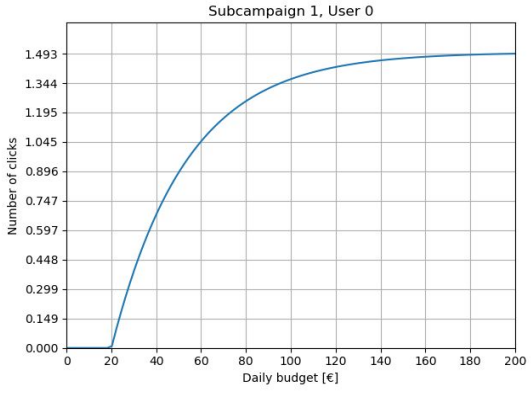
Fixed a sub-campaign and a class of user, we have four parameters describing the curves:

- **P:** it describes, given a sub-campaign, the probability distribution for each class of users. We compute it in the following way: we define a variable called *batch_size* representing the number of users drawn for each sub-campaign. In this way, we do not define a priori the users, but we draw them with the associated probabilities.

  The next graph displays how the three classes of users are drawn for each sub-campaign, according to the values of P (on the y-axis of the box plot you can see the percentage).



- **Bid:** It describes the minimum amount of money that the advertiser should bid to start getting clicks from a given class of user belonging to the corresponding sub-campaign.
- **Slope:** It describes how fast the clicks curve rises, assuming the curve to non-monotonically increase. The bigger the value, the faster the curve reaches its *max_clicks* value.
- **Max_clicks:** It represents the value that the curve can reach asymptotically.
- **Max_budgets:** It represents the maximum daily amount of budget we want to invest in a sub-campaign. This is useful if we don't want to invest a lot of money in a specific sub-campaign, but more specifically we added this parameter because it is present in the formalization of the optimization problem.
- **V:** Value per click associated to a sub-campaign.



Algorithms

Combinatorial GP-TS: definition, application on aggregate curve, regret (question 3)

The Thompson-Sampling algorithm belongs to the class of **MAB**s, i.e. it is an algorithm that estimates the values of a function in a discrete domain by using samples with the goal of minimizing the regret.

What differentiates one MAB algorithm from the others is the selection of the so-far optimum arm. TS does that by estimating a probability distribution for each arm, describing where the y-value lays. More samples for each arm equals a more accurate estimate of the probability distribution for each arm; every round, a sample is drawn from each distribution and the selected arm is the one providing the greater drawn value.

A MAB algorithm estimates each arm separately, if we are estimating a continuous curve we consider many points of this as separate arms. Therefore a MAB does not take into account the fact that we have a continuous curve, so these points will have to be connected to each other in some way. The best way to model this type of connection between the arms is done by adding a **Gaussian Process** to a MAB.

A **Gaussian Process** is a collection of random variables indexed by time or space, such that every finite collection of those random variables has a multivariate normal distribution, i.e. every finite linear combination of them is normally distributed. The distribution of a Gaussian process is the joint distribution of all those random variables, and as such, it is a distribution over functions with a continuous domain.

A **Gaussian Multi Armed Bandit** is done at a high level as a MAB.
At each step I select the best arm according to the learner, but rather than updating only the selected arm, all the arms get updated through the GP.
The closer an arm will be to the chosen arm, the more information the GP will extract about it.

So a **GP** can be used in combination with Thompson Sampling to achieve two properties that suit our problem:

- Very mild assumptions over the function
- A probability distribution returned over the outcome

A GP-MAB is useful only to learn one curve.

In our problem we want to learn 5 different ads curves (1 for each sub-campaign) to be able to invest our money in these five sub-campaigns in the best possible way, i.e. we want to find the best combination of money given the estimation of rewards per budget returned by the GP-MAB.

For this reason, we need to add a combinatorial optimization step to the algorithm.
In our case, we can apply an extended version of the **Knapsack algorithm.**

We make two initial assumptions:

- The performance of every sub-campaign is independent of the performance of the other sub-campaigns. Note that this is not true in general.
- The values of the bid and daily budget are finite. This is also not true in general but it is reasonable in practice.

The extended Knapsack algorithm returns the best allocation of money divided into n sub-campaigns, after receiving in input the rewards in function of the budget for each sub-campaign.

This is the formal model of the optimization problem:

$$\max_{x_{j,t}, y_{j,t}} \sum_{j=1}^{N} v_j \; n_j(x_{j,t}, y_{j,t})$$

$$\text{s.t.} \sum_{j=1}^{N} y_{j,t} \leq \overline{y}_t$$

$$\underline{x}_{j,t} \leq x_{j,t} \leq \overline{x}_{j,t}$$

$$\underline{y}_{j,t} \leq y_{j,t} \leq \overline{y}_{j,t}$$

J: is a sub-campaign
N: Number of sub-campaign
$\mathbf{X}_{j,t}$: Bid of sub-campaign j
$\mathbf{Y}_{j,t}$: Budget of sub-campaign j
$\mathbf{V}_j$: Value per click of sub-campaign j
$\mathbf{n}_j$: Number of clicks of sub-campaign j given the value of x_j y_j
$\mathbf{y}_t$: Cumulative budget constraint
$\underline{\mathbf{x}}_{j,t}$, $x_{j,t}$: Box constraint for the bid of sub campaign j
$\underline{\mathbf{y}}_{j,t}$, $\mathbf{y}_{j,t}$: Box constraint for the budget of sub campaign j

High-level algorithm

1. Sampling ➔ Invest money in every sub-campaign and Update all the GPs
2. Combinatorial Optimization ➔ Solve the Knapsack problem given the GPs values. The results are the arms to pull the next cycle / day

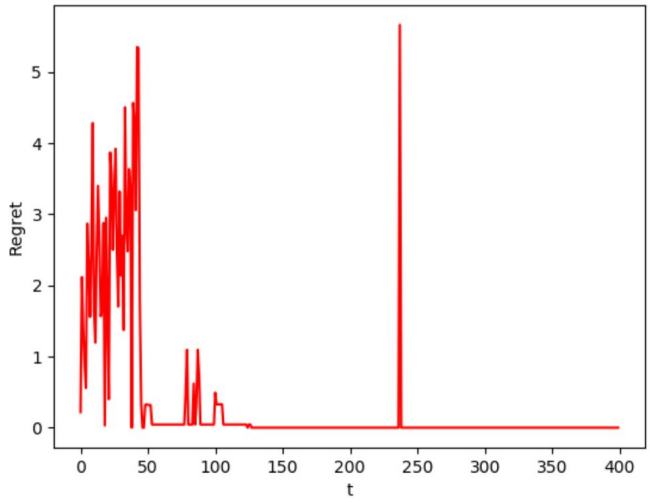
3. Repeat 1, 2 for T times

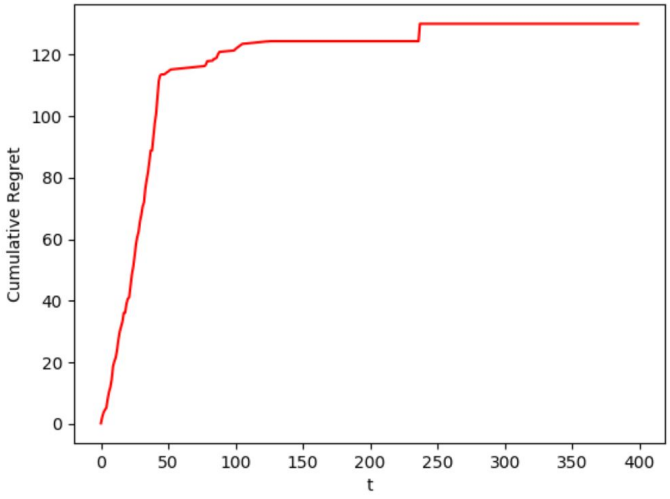
We provide in the following table the values of the constraints and of the discretization:

- **Total budget**: 200
- **Min. daily budget for a single sub-campaign**: 0
- **Max. daily budget for a single sub-campaign**: 200
- **N. arms**: 21
- **Arms values**: {0, 5, ..., 190, 195, 200}

The left image represents how the regret varies in time, while to the right we have the cumulative regret.

We can see that the algorithm converges to an optimal solution.



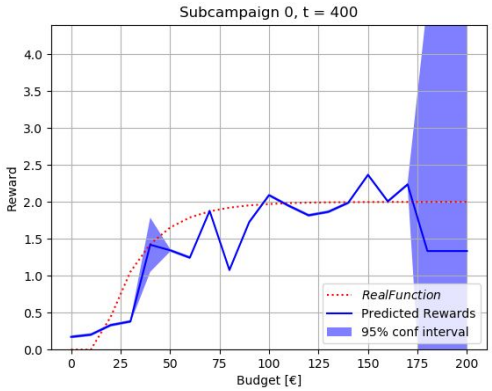


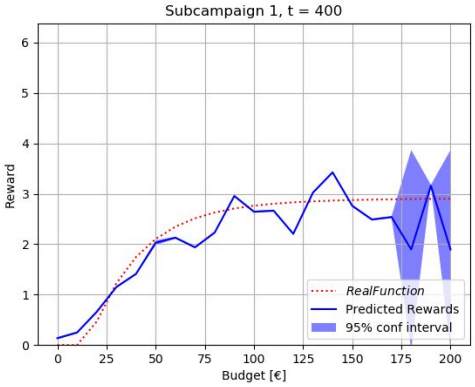
Single sub-campaign: Regression error of GP (question 4)

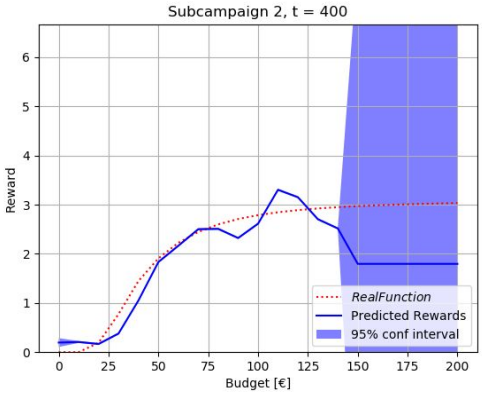
We provide 6 plots to show the quality of the regression of the GPs.

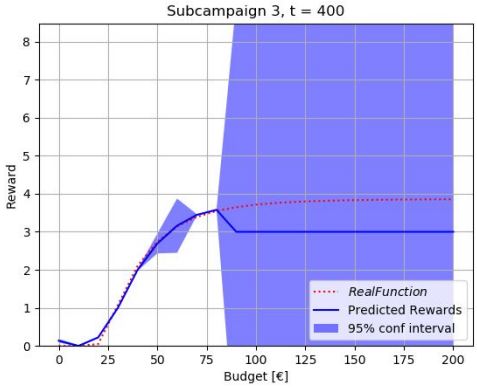
The first 5 plots show the resulting curve obtained by the GPs for each sub-campaign compared to the real curve. The confidence interval is also displayed.

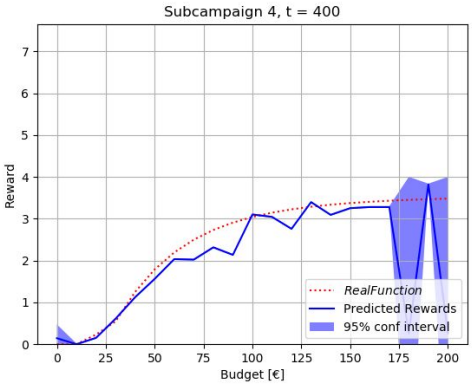
The fact that some values are drawn less frequently is due to the fact that the algorithm converges to other more promising solutions, or those values don't satisfy the constraints.



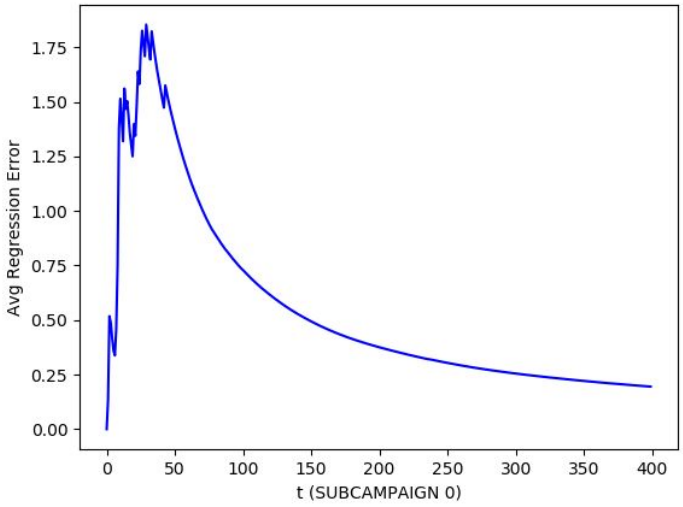








This plot shows the average regression error of the first sub-campaign.



Driving Exploration by Maximum Distribution in Gaussian Process Bandits (question 5)

This part was done by Dominic Crippa for his thesis work and replaces the actual part 5 of the assignment.

The algorithm proposed in this part is a variant of the DAGP-UCB algorithm (ideated by Alessandro Nuara and Francesco Trovò, implemented in Dominic Crippa master thesis).

The **DAGP-UCB** algorithm decides the next arm to pull by taking into account the value of some weights w(x,x') that give more importance to the region of the domain D with the highest probability of being optimal.

The uncertainty term S(x,x') encourages the exploration of those arms that contribute most to the reduction of the standard deviation of the GP over each point in the input space D.

Without going into details, the weights are estimated using a Monte Carlo algorithm and they are updated at each step. The variant designed for the Ads environment uses a Monte Carlo algorithm too but with some modifications.

Instead of using a Monte Carlo algorithm to estimate the DAGP-UCB weights, this variant uses a MC algorithm to estimate the values to be passed to the Knapsack one.

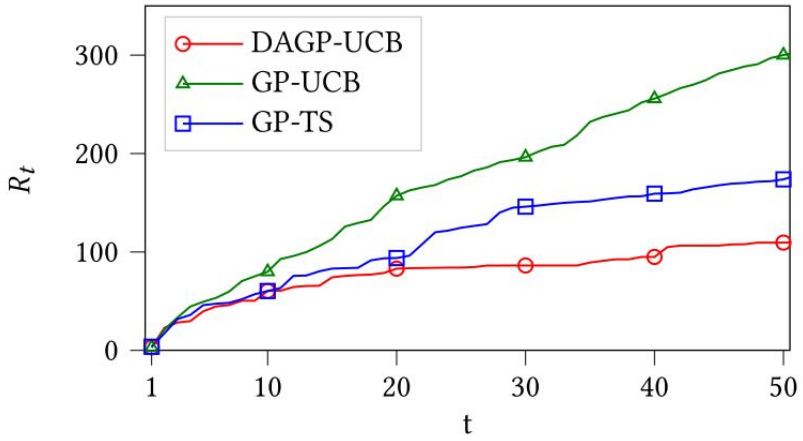
The algorithm simply samples k times each arm of each aggregate curve and runs the Knapsack algorithm, thus counting the number of times each arm is chosen as the best one by the Knapsack Algorithm.

By dividing the number of times that an arm has been chosen as optimal by the number of iterations of Monte Carlo k performed, we obtain the values to pass to the Knapsack.

High-level algorithm:

1. Sampling ➔ Invest money in every sub-campaign and Update all the GPs
2. Monte Carlo ➔ Compute arm values for the Knapsack
3. Combinatorial Optimization ➔ Compute the Knapsack problem given the Monte Carlo results. The result will be the arm to pull the next cycle/day
4. Repeat 1,2,3 for T times

This is a plot that shows the regret on the advertising application.



The performance of the algorithm is compared to others most commonly used nowadays.
In these specific experiments an advertiser has to maximize the number of clicks over a set of C = 3 sub-campaigns, each of which corresponds to a specific ad.