POLITECNICO
MILANO 1863

Data Intelligence Application Project

ACADEMIC YEAR: 2018/2019

Part II: Advertising

Davide Rutigliano - 903616

Contents

1 Introduction

In this project we focus on the study of pay-per-click advertising, in which an advertiser pays only once a user has clicked an ad. An advertising campaign is characterized by a its sub-campaigns each with a (potentially) different pair ad/targeting and by a cumulative daily budget (also called spending plan).

In this type of advertising campaign, the advertiser takes part into an auction specifying a bid and a value for the daily budget for each sub-campaign. The goal of advertisers is to select these variables to maximize the expected revenue they get from the advertising campaign.

According to the literature, common methods used to face this problem are the second price auction (GSP) and the Vickrey-Clarke auction (VCG). The proposed method uses combinatorial bandit algorithms techniques to face the problem in an online fashion.

We 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 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 model definition is:

- N number of sub campaigns composing the advertising campaign C
- $C = \{C_1, ..., C_N\}$ advertising campaign
- T time horizon (expressed in days)
- $t \in T$ time (expressed in days)
- X_j bids

- Y_j budget
- B spending plan where $B_{j,t}$ is the pair $\left[\underline{y}_{j,t}, \bar{y}_{j,t}\right]$ that are respectively the minimum and maximum cumulative daily budget for campaign $j \in N$ for the day $t \in T$
- v_j value per click for sub-campaign $j \in N$
- $n_j(x_j, y_j)$ number of clicks for sub-campaign $j \in N$ given values of bid $x_j \in X_j$ and budget $y_j \in Y_j$

In our specific case we run the experiment over a time horizon of $T = 100$ days, with $N = 5$ sub-campaigns and linearly spaced values of the budget $y_j \in [0, 99]$ and $|Y_j| = 10, \forall j \in N$. Furthermore, we do not take into account of the bids X_j because we assume that bidding is automatically performed by the advertising platform (i.e. we assume the bid is constant for each day and for each sub-campaign: $x_{j,t} = \bar{x}, \bar{x} \in X_j, \forall j \in N, \forall t \in T$); we also assume the values per click $v_j = \bar{v} = 1.0$, to be constant for each sub-campaign $j \in N$. In addition, we set $\underline{y}_{j,t} = 0$ and $\bar{y}_{j,t} = 99$.

Below in the document budget/click curves will be shown.

3 Algorithms

3.1 Multiple Choice Knapsack (Optimization Problem)

Notation

- $C_j : X_j \times Y_j$ the advertising campaign $j \in N, \forall j \in N$, where $C_j(x_j, y_j) = v_j n_j(x_j, y_j)$
- M_j : max number of clicks for each budget $y_j \in Y_j$ for each sub-campaign $j \in N \cup \varnothing$, where the first row of the matrix ($j = 0$) correspond to the initialization of the algorithm with all values set to 0.

Mathematical Formulation

- $\max_{x_{j,t}, y_{j,t}} \sum_{j \in N} v_j n_j(x_{j,t}, y_{j,t})$
  s.t.
- $\sum_{j \in N} y_{j,t} \leq \bar{y}_t, \forall t \in T$
- $\underline{x}_{j,t} \leq x_{j,t} \leq \bar{x}_{j,t}, \forall j \in N, \forall t \in T$
- $\underline{y}_{j,y} \leq y_{j,t} \leq \bar{y}_{j,t}, \forall j \in N, \forall t \in T$

Algorithm The algorithm takes in input all the advertising campaigns C_j the sets of bids and budget respectively X_j and Y_j, $\forall j \in N$ and a spending plan B.

1. For each sub-campaign $j \in N$, for each budget $y_j \in Y_j$ find the bid $x_j \in X_j$ such that:

   $z_j(y) = \max_{x_j \in X_j} C_j(x_j, y_j)$

2. Use the maximum bid found to calculate:
$$w_j(y) = \begin{cases} v_j n_j(z_j(y), y), & \underline{y}_{j,t} \leq y \leq \bar{y}_{j,t} \\ 0, & \underline{y}_{j,y} \geq y \vee y \geq \bar{y}_{j,t} \end{cases}$$
3. Compute the value of M_j for every budget $y \in Y$:

   $M(j, y) = \max_{y' \in Y, y' \leq y} \{M(j-1, y') + w_j(y - y')\}$
4. At the end of the procedure, the optimal solution of the MCK problem can be found in the cell corresponding to:

   $y^* = \max_{y \in Y} M(N, y)$

   To find the optimal allocation (assignment of the budget), it is sufficient to also store the partial assignments of the budget (i.e. $y - y'$) corresponding to the optimal value.

Note: In our case, where the bidding is automatically performed by the advertising platform we assume to be already given the number of clicks of the best bid $\bar{x}_j \in X_j$ for each budget $y_j \in Y_j$. Thus, the algorithm we implement differ from the one presented above because it skips step 1 of the algorithm as we already have the best bid for each budget.

3.2 Gaussian Process Thompson Sampling

Notation

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

Gaussian Process The parameters that define a Gaussian process are:

- $m : Y_j \rightarrow \mathbb{R}$ its mean
- $k : Y_j \times Y_j \rightarrow \mathbb{R}$ its co-variance matrix

If no prior information is available at the beginning we set $m(y) = 0$, otherwise we can model the mean of the Gaussian process in order to include the prior information about the model.

The co-variance function $k(y, y') = \mathbb{E}\left[(f(y) - m(y)) \cdot (f(y') - m(y'))\right]$,

We decided to use a squared exponential kernel, that is:

$$k(y, y') = \theta^2 e^{-\frac{(y-y')^2}{2l^2}}$$

where θ is the scale factor and l is the length-scale, using initial values for $\theta = 1$ and $l = 1$ (common choice when no prior knowledge on the process is available). Furthermore we add a noise to the process, setting initially its variance to $\sigma_n = 10$.

We use a Gaussian process defined as above in order to estimate the budget/click curve:

$n(y) = \mathcal{GP}(m(y), k(y, \cdot)), \forall y \in Y_j$

Pseudocode

```
1. For each day $t \in T$, do:
2. For each sub-campaign $j \in N$, For every arm $a$:
    • $\tilde{\theta_a} \leftarrow Sample(\mathbb{P}(\mu_a = \theta_a))$
    • $a_t \leftarrow \arg\max_a \left\{\tilde{\theta_a}\right\}$
    • $n_j(y) \leftarrow Sampling(\mathbf{a})$
3. $\{\hat{y}_{j,t}\}_{j \in N} \leftarrow Optimize(\{(v_j, n_j(y), Y_j\}_{j \in N}, \bar{y}_t)$
```

The algorithm works in the following way: the first step for each day for each arm we choose the best arm according to Thompson Sampling selection criterion, then use the arms pulled and rewards obtained so far to estimate (by meaning of Bandit algorithms, GP-TS in our case, for each sub-campaign) the budget/click curve $n_j(y)$. Then, for each day the optimal solution for each sub-campaign can be found by solving the Multiple-Choice Knapsack problem with the procedure described above.

3.3 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

Value of a context structure $v = \sum_{c \in \mathscr{P}} p_c J_c^*(y)$
Where:

$J_c^*(y) = \sum_{j=1}^{N} v_j^* n(y_j^*)$
(J_c is the objective function of the MCK problem for context c)

Split condition $\underline{p}_{c_1} \underline{J}_{c_1}^*(y) + \underline{p}_{c_2} \underline{J}_{c_2}^*(y) \geq \underline{J}_{c_0}^*(y)$
Where:

$\underline{p}_{c_i}$ is the lower bound of the probability of context c_i and,

$\underline{J}_c^*(y)$ is the lower bound of the (optimal) objective function of the MCK problem defined before in context c_i: $\underline{J}_c^*(y) = \sum_{j=1}^{N} \underline{v}_j^* \underline{n}(y_j^*)$

Lower bound The lower bound is calculated as the Hoeffding bound, that 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

We use a function in order to imagine the budget/click curve:

$n(y) = c_{max} \cdot e^{-\alpha(y-\beta)}$

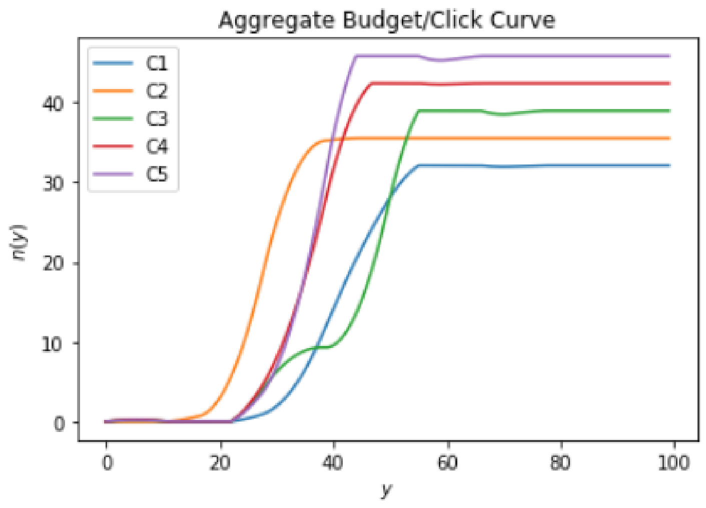
where c_{max} is the maximum number of clicks, alpha controls how rapidly the click reaches the maximum number of clicks (i.e. the point in which increasing the budget does not increase the number of clicks) and beta is a parameter that controls at which budget the clicks starts increasing (with the meaning that when the bid is too small the ad is not displayed in any auction). The idea is that the curve is zero when the bid is too small, then when starts increasing depends linearly from the budget and from a certain point (budget value) on the dependency starts to be constant.

In order to obtain different curves for different classes we combine different functions defined above (different values of alpha and beta and max no. of clicks).

To get the aggregate curve instead, we combine the base classes curves taking the weighted average on the classes:

$\bar{n}_j(y) = \frac{\sum_{k \in F} n_{k,j}(y) \cdot p_k}{\sum_{k \in F} p_k}, \forall y \in Y_j, \forall j \in N$

Where p_k is the probability and $n_{k,j}(y)$ is the budget/click curve of class $k \in F$.



5 Results

5.1 Gaussian Process Thompson Sampling

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.

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:

- $\mathbb{E}\left[\sum_{j=1}^{N} v_j n_j(\hat{y}_j)\right]$
- $R_t(\mathfrak{U}) = TG^* - \mathbb{E}\left[\sum_{j=1}^{N} v_j n_j(\hat{y}_j)\right]$

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

- $\mathbb{E}\left[\sum_{j=1}^{N} \sum_{t=1}^{T} v_j n_j(\hat{y}_j)\right]$
- $R_T(\mathfrak{U}) = TG^* - \mathbb{E}\left[\sum_{j=1}^{N} \sum_{t=1}^{T} v_j n_j(\hat{y}_j)\right]$

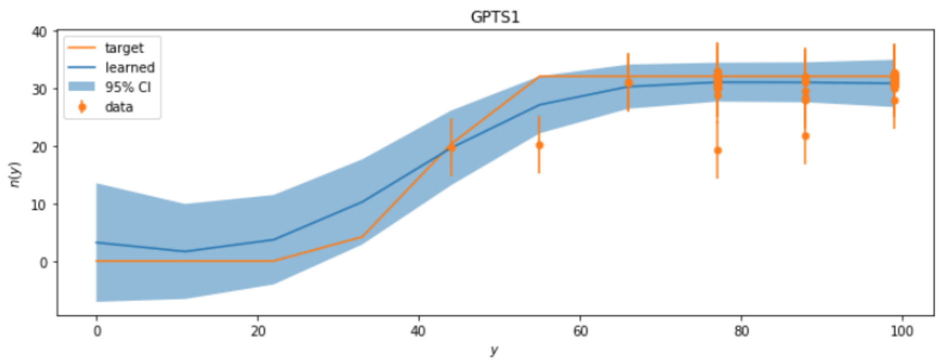
Where $G^* = \sum_{j=1}^{N} v_j^* n_j(y_j^*)$ is the (optimal) value provided by the clairvoyant algorithm.

We show here results, after the execution of each algorithm, in 4 different plots respectively:

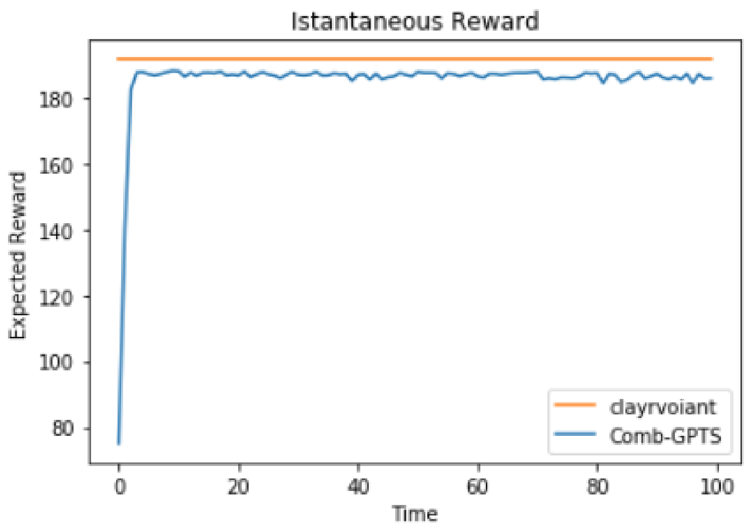
- Instantaneous Reward vs. Clayrvoiant
- Instantaneous Regret

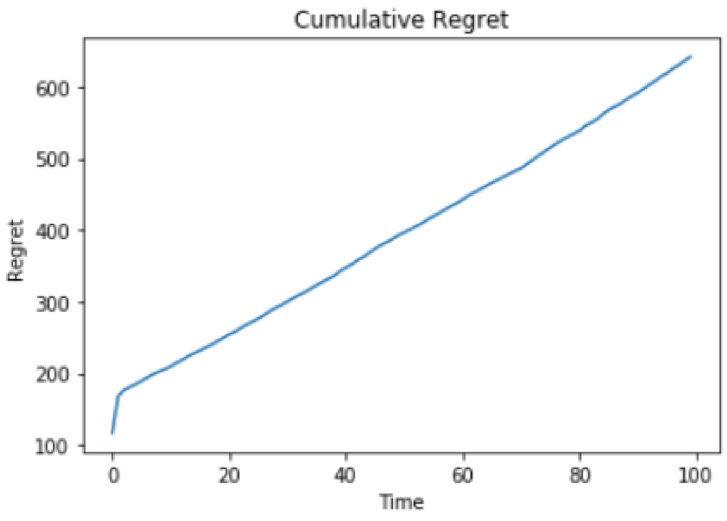
- Cumulative Reward vs. Clayrvoiant
- Cumulative Regret

We also plot data points, target and estimated functions for each sub-campaign. We only report results for one of the learner (or equivalently, sub-campaign), for complete results please refer to the repository on git.

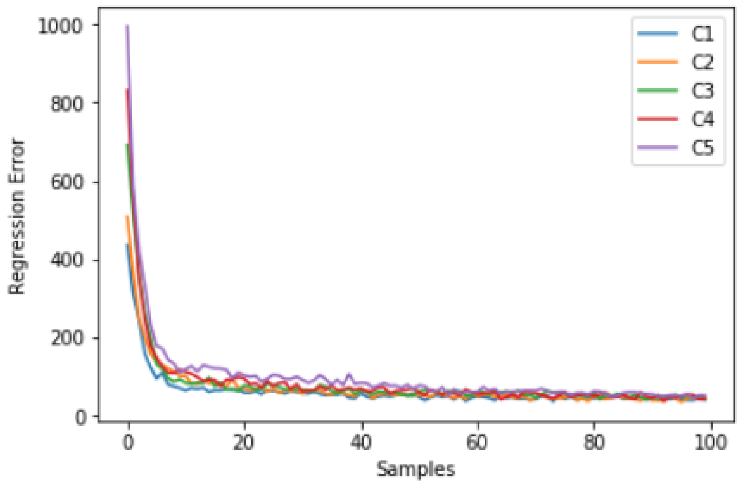


Then we show how reward and regret vary in time:





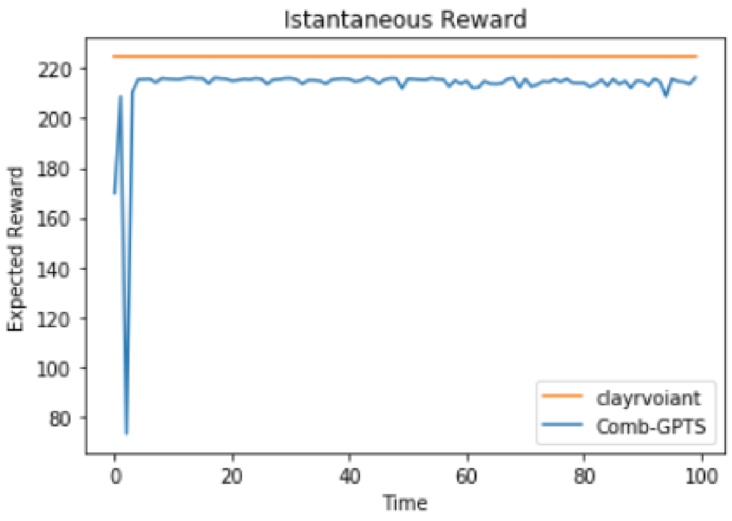
Furthermore, we plot the regression error for each sub-campaign as the number of samples:

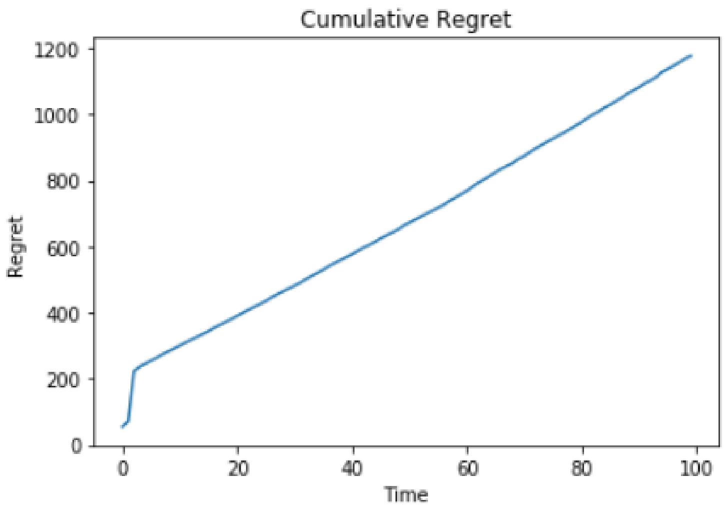


5.2 GP-TS with Context Generation

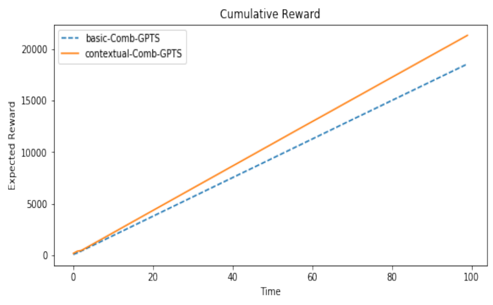
We decided to apply the context generation algorithm each week, supposing we have 100 customers per day.

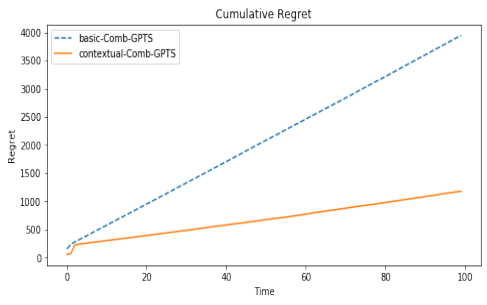
We first show how reward and regret vary in time:





Then we show how the contextual version benefits of the information about the context resulting in higher reward and lower regret:


Cumulative Reward
basic-Comb-GPTS
contextual-Comb-GPTS
Expected Reward
Time


Cumulative Regret
basic-Comb-GPTS
contextual-Comb-GPTS
Regret
Time