POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Solving the Cold-Start Problem in Digital Advertising

Tesi di Laurea Magistrale in
Computer Engineering - Ingegneria Informatica

Author: **Simone Orlando**

Student ID: 10530758
Advisor: Prof. Nicola Gatti
Co-advisors: Luca Alessandrelli,
Francesco Bacchiocchi,
Francesco Emanuele Stradi
Academic Year: 2021-22

A mia Madre, alla quale devo tutto.

Abstract

The online advertising market is experiencing the most flourishing period ever observed so far. The profit recorded in 2021 was 189 billion dollars, corresponding to an increase of 35.4% over the previous year. In order to win the best advertising slots, made available by search engines and social networks, advertisers must compete in an auction. Often, choosing the optimal bid is not trivial; consequently, more and more advertising agencies resort to the help of automated tools for managing the advertising campaigns of their customers. However, such systems suffer from a problem known as Cold-Start that occurs when they have to operate in situations where a minimum amount of historical data is available. The aim of the work is to develop a methodology that can mitigate this problem when an advertiser decides to launch a new advertising campaign for which no observations have yet been recorded. In particular, the proposed heuristic uses the information collected from the other sub-campaigns related to the advertiser to extract a characteristic average behaviour to be used as a starting point for the new sub-campaigns. Finally, the approach developed has been validated on real data provided by the advertising agency 'AdsHotel', highlighting a statistically significant improvement in performance compared to the scenario in which no Cold-Start management technique is used.

Keywords: Online Advertising, Cold-Start, Multi-Armed Bandit

Sommario

Il mercato pubblicitario online sta vivendo il periodo più florido mai registrato. Il profitto ottenuto nel 2021 è stato pari a 189 miliardi di dollari, corrispondente ad un aumento del 35.4% rispetto all'anno precedente. Per potersi aggiudicare i migliori spazi pubblicitari messi a disposizione da motori di ricerca e social network, gli inserzionisti devono concorrere in un' asta, per la quale, è necessario scegliere la puntata ottimale; di conseguenza, sempre più agenzie pubblicitarie ricorrono all'ausilio di strumenti di apprendimento automatizzato per la gestione delle campagne pubblicitarie dei propri clienti. Tali sistemi soffrono, però, di un problema noto come Cold-Start: quest'ultimo si verifica quando si ha a disposizione un quantitativo minimo di dati storici, e quindi, nelle fasi inziali dell'apprendimento, si tende ad avere una performance non soddisfacente. L'obiettivo di questo lavoro è quello di sviluppare una metodologia in grado di mitigare tale problematica, qualora un inserzionista decida di avviare una nuova campagna pubblicitaria per la quale nessuna osservazione è stata registrata. In particolare, l'euristica proposta sfrutta informazioni raccolte da altre sotto-campagne pubblicitarie, estraendo da esse un comportamento medio caratteristico da impiegare per le nuove sotto-campagne.
Infine, l'approccio sviluppato è stato validato su dati reali forniti dall'agenzia pubblicitaria 'AdsHotel', evidenziando un miglioramento statisticamente rilevante della performance rispetto allo scenario in cui non viene adoperata nessuna tecnica di gestione del Cold-Start.

Parole chiave: Pubblicità Online, Cold-Start, Multi-Armed Bandit

Contents

1 | Introduction

The use of the Internet has grown tremendously over the last two decades, and online advertising has become the most effective way to sponsor a product or an event. Internet advertising revenue in 2021 was 189 billion USD, a 35.4% increase over the previous year, and consequentially, the largest growth since 2006 [12]. The Internet channels' ability to reach a broad audience fast and efficiently while also targeting specific kinds of users with personalized sponsored announcements contributes to the success of this advertising approach. The key actors in this industry are the advertisers, who have a product or an event to market, the media agencies, whose job is to handle advertisement campaigns for the advertisers, and the Web publishers, whose responsibility is to supply slots in Web-pages dedicated for advertising purposes. Ad spots are typically provided by publishers through auctions, in which each advertiser must indicate the bid, i.e., how much he is willing to pay every time a consumer clicks on his ads, and the budget, i.e., the maximum amount of money he is ready to spend every day for his advertising campaign. The ability to target adverts to consumers very precisely, thanks to a massive amount of data on user activity available to advertisement platforms, is critical to the success of Internet advertising. However, such a large volume of data makes the challenge of identifying the optimal targeting unaffordable for humans, and, as a result, the use of automated approaches has become essential.

1.1. Motivation

Online advertisers face a critical challenge called the Cold-Start Problem. De facto, identifying the appropriate bid and budget combinations to utilize during the publisher-induced auction to clinch the best slots is one of the most essential aspects of the process of advertising a product or service online. In order to address this optimization problem, media agencies frequently employ automated tools based on machine learning techniques, which require a vast quantity of historical data to function properly.
The Cold-Start problem comes when these tools are tasked with optimizing fresh advertising campaigns launched by advertisers for whom no data or a minimum amount are

available. In the early days of a campaign, this frequently results in erroneous suggestions from automated systems until a significant number of observations are collected. As a result, the advertiser may suffer considerable economic losses, which may lead to a loss of confidence in the media agency and in the employment of new automated techniques.

1.2. Original Contribution

The contribution of this work consists of the development of a heuristic capable of mitigating the Cold-Start Problem for the advertiser in the internet advertising scenario. In particular, the approach we propose exploits the available information from the advertiser's open sub-campaigns, which are part of the same advertising campaign, to extrapolate an average behaviour to be used as a starting point for the new sub-campaigns that he could decide to open in future. More specifically, our goal was to extrapolate a way to produce an estimate of the click function, which is critical for the resolution of the advertiser optimization problem, that is more informative than the simple zero-centred prior function utilized in modern automated systems.
Using data provided by the media agency AdsHotel, we were able to test the effectiveness of the technique we proposed by simulating a real-world online scenario.

1.3. Document Outline

The document is structured as follows. Chapter 2 introduces the fundamental knowledge required to comprehend what follows, such as the Multi-Armed Bandit problem structure, the use of Gaussian Processes for regression tasks, and the dynamics that characterize the Online Advertising scenario from both the advertiser and the publisher's perspectives. Chapter 3 presents a literature review of the related works that have focused on the development of problem-solving techniques for the advertiser optimization problem. Chapter 4 gives a broad introduction to the Cold-Start problem for automated recommendation systems and then delves into how it manifests itself in the context of internet advertising. Chapter 5 discusses in detail the heuristic we designed, with a special emphasis on how it could be implemented with Gaussian Processes. Chapter 6 reports the results of the experiments we conducted to validate our technique, first on a synthetic environment and subsequently on real data provided by AdsHotel. Finally, Chapter 7 draws conclusions on this work and opens new directions for further projects.

2 | Preliminaries

This chapter presents the basic theoretical concepts required for the reader to understand the work presented in this document. Specifically, Section 2.1 illustrates the Multi-Armed Bandit (MAB) problem and the main algorithms used to solve it. Section 2.2 explores Gaussian Processes (GP) and their use in Combinatorial Bandits to set a correlation between arms' expected rewards. Finally, Section 2.3 provides an overview of the Online Advertising scenario and its internal dynamics both from the point of view of the publisher and the advertiser.

2.1. Multi-Armed Bandit

Multi-Armed Bandits also referred to as the K-armed bandit problem, consider a scenario in which an agent faces K independent actions or bandit's arms. In each time step, the agent must select one of these arms based on its current information receiving in return a reward sampled from an unknown distribution.
Furthermore, being MAB an online decision-making problem, it is mandatory to address the exploration-exploitation dilemma. Every time the agent needs to make a decision he can choose between two possible behavioural strategies:

- Exploration: gather more information choosing less explored actions
- Exploitation: select the action we consider to be the best one so far

Choosing the right balance between these two strategies is fundamental to achieve suitable performance.

2.1.1. Problem Formalization

A Multi-Armed Bandit problem can be seen as a tuple $(\mathcal{A}, \mathcal{R})$ where

- $\mathcal{A}$ is a set of $\mathcal{K}$ possible arms
- $\mathcal{R}$ is a set of real distributions, each associated with the rewards delivered by one of the arms

At every time t the agent selects a single arm $a_{i,t}$, then the environment generates a reward $r_{i,t}$ drawn from the distribution $\mathcal{R}(a_{i,t})$. Finally, the agent updates his information on pulled arms and the received rewards.
The final objective of the agent is to maximize the cumulative reward $\mathtt{v}$ over a given time horizon T:

$$\mathtt{v} = \sum_{i=1}^{T} r_{i,t}$$

If we call a^* the optimal arm and

$$R^* = R(a^*) = \max_{a_i \in \mathcal{A}} \mathbb{E}\left[\mathcal{R}(a_i)\right]$$

the expected reward associated with the optimal arm, we can reformulate the objective function in terms of *regret*, i.e. the average loss we incur at time t by playing the action $a_{i,t}$ instead of the optimal one a^*:

$$\mathbb{E}\left[\mathcal{R}(a^*) - \mathcal{R}(a_{i,t})\right] = R^* - R(a_{i,t})$$

Our goal is now to minimize the expected cumulative regret L_T suffered over a finite time horizon T:

$$L_T = TR^* - \mathbb{E}\left[\sum_{t=1}^{T} R(a_{i,t})\right]$$

Note that the maximization of the cumulative reward is equivalent to the minimization of the cumulative regret.
It is possible to express the lower bound of the regret [8] as

$$\lim_{T\to\infty} L_T \geq \log(T) \sum_{a_i | \Delta_i > 0} \frac{\Delta_i}{KL\left(R(a_i), R(a^*)\right)} \tag{2.1}$$

where $KL\left(R(a_i), R(a^*)\right)$ is the Kullback-Leibler divergence between the two reward distributions $R(a_i)$ and $R(a^*)$, and Δ_i is the average difference in reward between a generic arm a_i and the optimal one a^*.

$$\Delta_i := R^* - R(a_i)$$

This result states that regret grows at least logarithmically concerning the time horizon T, or more formally, $L_T = \Omega(log\ T)$. Thus, an algorithm is said to solve the MAB problem if it can match this lower bound, i.e. $L_T = O(log\ T)$.
In the next Subsections, we present two solving algorithms which try to approach this lower bound. In particular, in Subsection 2.1.2 we explore the Upper Confidence Bound 1 (UCB1) algorithm [1], and in Subsection 2.1.3 we go into the Thompson Sampling (TS)

algorithm [14].

2.1.2. UCB 1

The UCB1 algorithm, whose pseudo-code is provided in Algorithm 2.1, works by computing the upper confidence bounds for every arm a_i and selecting, at every time step t, the arm with the highest bound.

Algorithm 2.1 UCB1

Input: the number of arms K
Initialization:
$Q(a_i) \leftarrow 0$ for all actions a_i
$N(a_i) \leftarrow 0$ for all actions a_i
$t \leftarrow K + 1$
Play each of the K arms once, and update $Q(a_i)$ accordingly
while true do
    Play the arm a_i that maximizes (where t is the current time step):

$$Q(a_i) + \sqrt{\frac{2\ log\ t}{N(a_i)}}$$

    Receive a reward R and update $Q(a_i)$
    $N(a_i) \leftarrow N(a_i) + 1$
    $t \leftarrow t + 1$
end while

The upper confidence bound used by the algorithm

$$Q(a_i) + \sqrt{\frac{2\ log\ t}{N(a_i)}}$$

derives from Hoffding's inequality [5].
The first term of the bound, $Q(a_i)$, represents the mean reward obtained by playing the arm a_i so far and it is the pure exploitation part of the formula. Instead, the second term, $\sqrt{\frac{2\ log\ t}{N(a_i)}}$, is inversely proportional to $N(a_i)$, the number of times we played the arm a_i so far, and considers the exploration part of the algorithm. In fact, if two arms have a more or less similar average reward, but the first arm has been chosen more times than the second one, then the second term of the formula, being inversely proportional to $N(a_i)$, will propel us to the second arm.
We can estimate the performance of UCB1 by extrapolating an upper bound for the total

regret we get if we apply it to a stochastic MAB [1]:

$$L_T \leq 8\ log(T) \sum_{i|\Delta_i>0} \frac{1}{\Delta_i} + \left(1 + \frac{\pi^2}{3}\right) \sum_{i|\Delta_i>0} \Delta_i$$

This is a pretty good result because we obtain a bound of the same asymptotic order, $log(T)$, of Equation 2.1.

2.1.3. Thompson Sampling

The Thompson Sampling algorithm is structured as a classical Bayesian methodology for online learning. For each arm a_i we have a prior $f_i \sim Beta(\alpha, \beta)$ as a starting point. At each round t we sample from each one of the prior distributions a reward r_i and we play the arm $\hat{a}_i$ with the highest sampled value. Then, we update the prior incorporating the reward gathered by choosing arm $\hat{a}_i$.
The update is based on the following rules:

- In case of success: $f_i(t+1) = Beta(\alpha_t + 1, \beta_t)$
- In case of failure: $f_i(t+1) = Beta(\alpha_t, \beta_t + 1)$

We report the TS pseudo-code in Algorithm 2.2.

Algorithm 2.2 Thompson Sampling

Input: the number of arms K
Initialization:
Set $S(a_i) = 1$, $F(a_i) = 1$ for all arms a_i
$t \leftarrow 1$
while true do
    For each arm, sample $B_i(t) \sim Beta(S(a_i), F(a_i))$
    Play the arm a_i that maximizes $B_i(t)$
    Receive a reward $R \in [0, 1]$
    With probability R, $S(a_i) = S(a_i) + 1$ else $F(a_i) = F(a_i) + 1$
    $t \leftarrow t + 1$
end while

As for UCB1 also for TS, we can evaluate its performance by identifying its upper bound on the regret. [6]

$$L_T \leq O\left(\sum_{i|\Delta_i>0} \frac{\Delta_i}{KL(R(a_i), R(a^*))} (log(T) + log(log(T)))\right)$$

Note that the identified bound matches the asymptotic rate lower bound for the cumula-

tive regret given in Equation 2.1.

2.1.4. Combinatorial Bandits

The novelty of Combinatorial Bandits (CB) problems is the introduction of constraints on playable arms. In classical bandit problems, we can choose only one arm, while in CB we can pull a subset of arms under customary combinatorial constraints.
Formally the elements of a CB problem are:

- Super-arm: a collection of arms
- Feasible Super-arm: a Super-arm satisfying the combinatorial constraints
- Super-arm reward: the sum of the reward of the arms that constitute it
- Goal: maximize the cumulative expected reward

One of the most common combinatorial constraints is the Knapsack Constraint in which every arm has a known cost and a Super-arm can not have a cost larger than a given budget.
We can simply extend the Thompson Sampling approach, as reported in Algorithm 2.3, to take into account the combinatorial constraints.

Algorithm 2.3 Combinatorial Thompson Sampling

Input: the number of arms K
Initialization:
Set $S(a_i) = 1$, $F(a_i) = 1$ for all arms a_i
$t \leftarrow 1$
while true do
  For each arm, sample $B_i(t) \sim Beta(S(a_i), F(a_i))$
  Play the Super-arm $\hat{a}_i$ that maximizes $\sum_{a_i \in \hat{a}_i} B_i(t)$, $\hat{a}_i \in S$
  Receive a reward $R \in [0, 1]$
  With probability R, $S(a_i) = S(a_i) + 1$ else $F(a_i) = F(a_i) + 1$ for $a_i \in \hat{a}_i$
  $t \leftarrow t + 1$
end while

The regret of the Combinatorial Thompson Sampling

$$L_T = O\left(|\mathcal{A}|\frac{\log(T)}{\Delta_{min}}\right)$$

is essentially the same as the basic version of the TS, i.e. logarithmic dependence on time T and linear dependence on the number of arms $\mathcal{A}$.

2.2. Gaussian Processes

The Gaussian Processes model is a probabilistic supervised machine learning framework that has been widely used for regression and classification tasks. A Gaussian Processes Regression (GPR) model can make predictions incorporating prior knowledge and provide uncertainty measures over predictions [15].

2.2.1. Multivariate Normal Distribution

A random variable X is normally distributed, with mean μ and variance σ^2, if its probability density function (PDF) can be expressed as

$$p_X(x) = \frac{1}{\sqrt{2\pi}\sigma} \, exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

This behaviour of the random variable X is represented by the notation $X \sim \mathcal{N}(\mu, \sigma^2)$. We report the plot of the PDF of a univariate Gaussian distribution in Fig. 2.1.

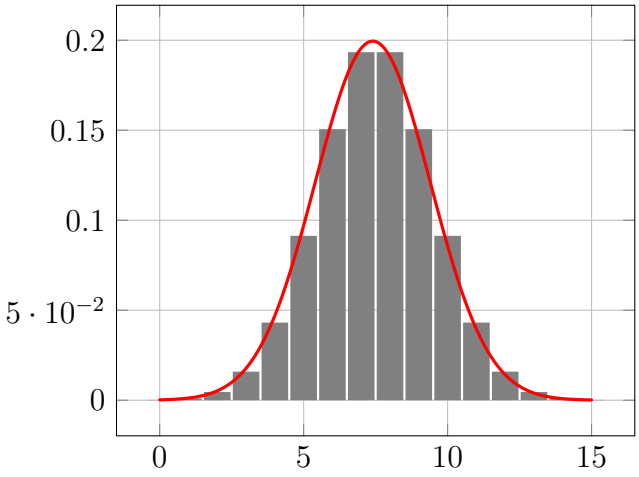
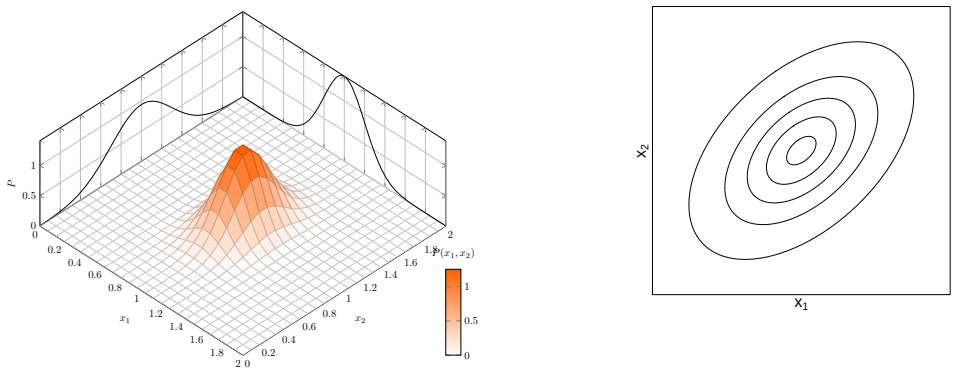


Figure 2.1: PDF of a random variable with a mean of 7.4 and a variance of 0.25.

In machine learning, more than one feature variable is commonly used to describe the samples inside a dataset and moreover, these variables are often correlated to each other. In order to model this scenario as one Gaussian model, we need to use a multivariate Gaussian (MVN) distribution.
The PDF of an MVN is defined as

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \, exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$$

where D is the number of features variables, $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{x}] \in \mathbb{R}^D$ is the mean vector and $\boldsymbol{\Sigma} = cov[\boldsymbol{x}]$ is the $D \times D$ covariance matrix. The $\boldsymbol{\Sigma}$ is a symmetric matrix that stores the pairwise covariance of all jointly modelled random variables with $\boldsymbol{\Sigma}_{ij} = cov(x_i, x_j)$ as its (i, j) element.
For visualization purposes, we focus on bivariate Gaussian (BVN) distributions in which we consider only two correlated feature variables (x_1, x_2). A BVN distribution can be visualized, as shown in Fig. 2.2a, as a three-dimensional (3-d) bell curve with heights representing the probability density.



(a) 3-d bell curve (b) 2-d ellipse contours

Figure 2.2: The PDF of a BVN visualization: (a) a 3-d bell curve with height represents the probability density, (b) ellipse contour projections showing the co-relationship between x_1 and x_2 points.

The ellipsoid shape of the 3-d bell level surfaces, shown in Fig. 2.2b, indicates the intensity of the correlation between the two variables x_1 and x_2. They show how sample points are arranged in the space, i.e. equally distributed within the ellipse, with decreasing probability from the central to the outermost ellipses.
Therefore, we can express the BVN in a compact way as

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \sigma_{11} & \sigma_{11} \\ \sigma_{21} & \sigma_{22} \end{bmatrix} \right) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

For regression problems, it is often more important conditional probability rather than joint probability. We can easily notice that if we fix a value for x_1 (or x_2) we still get for $P(x_2|x_1)$ (or $P(x_1|x_2)$) a Gaussian distribution, as shown in Fig. 2.3.

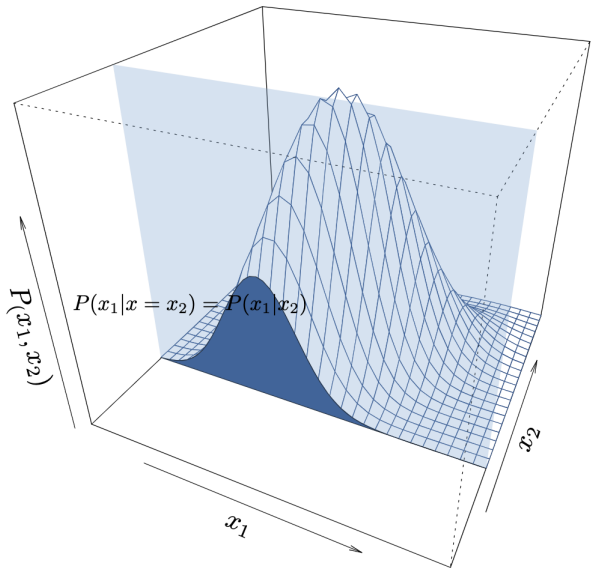


Figure 2.3: The conditional probability distribution $P(x_1|x_2)$ by cutting a slice on the PDF 3-d bell curve of a BVN.

2.2.2. Kernels

A kernel k, also called covariance function, is a positive-definite function that allows us to introduce our prior knowledge in the model:

$$Cov\left[f(x), f(x')\right] = k(x, x')$$

Informally, we can say that kernels specify the similarity between two values of a function, usually a mapping into a higher-dimensional features space, evaluated on each object. The most widely used kernel in Gaussian Processes is the radial basis function (RBF) kernel, also known as the squared exponential kernel, which is defined as

$$k(x, x') = exp\left(-\frac{(x - x')^2}{2}\right)$$

This is because it can be integrated against a wide range of functions due to its universal property.
If we consider two bivariate random variables X_1 and X_2 and we sample from X_1 a certain value, we can not sample from X_2 any random value with the same probability, because the two variables are correlated and which value pairs are more likely to sample is heavily influenced by the assigned kernel.

2.2.3. Gaussian Processes Regression

Gaussian processes address the regression task by defining a multivariate gaussian distribution over possible functions $\boldsymbol{f}$ that fit a set of points.
We initialize the model with the zero mean GP prior on the function variables:

$$p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}|0, \boldsymbol{K}) \tag{2.2}$$

where

$$\boldsymbol{K} = \begin{bmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & \dots & k(x_N, x_N) \end{bmatrix}$$

Our assumption is that the observed data are generated with Gaussian white noise around the underlying function f:

$$\boldsymbol{y} = \boldsymbol{f} + \mathcal{N}(0, \sigma^2)$$

Under this assumption and that noise is independent for each data point we can state that

$$p(\boldsymbol{y}|\boldsymbol{f}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma^2 \boldsymbol{I}_N)$$

Now we can compute the marginal distribution as

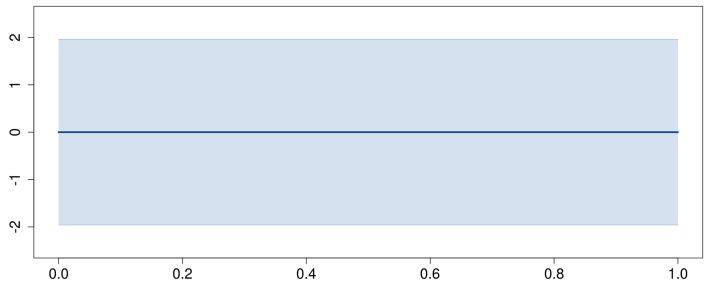
$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{f}) p(\boldsymbol{f}) d\boldsymbol{f} = \mathcal{N}(\boldsymbol{y}|0, \boldsymbol{C})$$

where N is the number of data point and $\boldsymbol{C} = \boldsymbol{K} + \sigma^2 \boldsymbol{I}_N$.
In order to make a prediction y_{N+1} for a new data point x_{N+1} we evaluate the predictive distribution

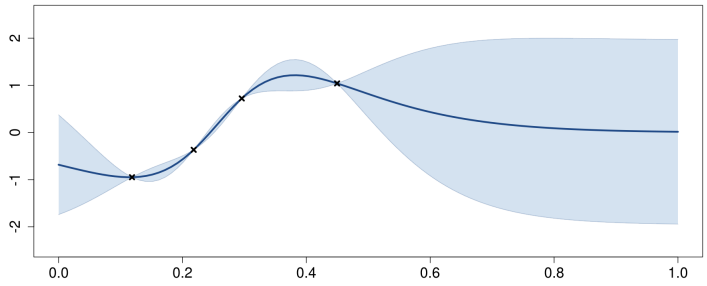
$$p(y_{N+1}|\boldsymbol{y}^{(N)}, x_1, ..., x_{N+1}) = \mathcal{N}(y_{N+1}|k^T \boldsymbol{C}^{-1} \boldsymbol{y}^{(N)}, c - k^T \boldsymbol{C}^{-1} k)$$

where $k = [k(x_1, x_{N+1}) \dots k(x_N, x_{N+1})]^T$ and $c = k(x_{N+1}, x_{N+1})$.
The mean value $k^T \boldsymbol{C}^{-1} \boldsymbol{y}^{(N)}$ of the prediction gives us our best estimate for y_{N+1}, and it is also known as the matrix of regression coefficients. The variance $c - k^T \boldsymbol{C}^{-1} k$ is, instead, an indication of the uncertainty of our estimation.
We can state from these results that the mean prediction is a linear combination of the observations $\boldsymbol{y}^{(N)}$, while the variance does not depend on the observed outputs $\boldsymbol{y}^{(N)}$, but only on the inputs $\boldsymbol{x}^{(N+1)}$.
In Fig. 2.4a we report how the prior distribution, defined in Equation 2.2, looks like, and

in Fig. 2.4b how its mean and variance are updated after adding four sample points to the training set.



(a) GP prior



(b) GP posterior

Figure 2.4: Process of inference in GPR a) Initial prior distribution over functions b) Posterior distribution over functions after observing four data points.

2.2.4. GP-Bandits

In a basic bandit problem, the reward of every arm is assumed to be independent of the reward of other arms. In GP-Bandit, instead, arms are correlated, and, moreover, the reward of an arm provides information on the reward of the arms close to it.
Gaussian processes can be used to express the correlation between arms due to their ability to represent a multivariate distribution over the possible reward functions.
In order to solve the GP-Bandit problem we can appeal to the natural evolution of UCB in the contextual bandit scenario. The GP-UCB [13] pseudocode is reported in Algorithm 2.4.

2.3. Online Advertising

Due to Internet's popularity and its spread in everyday usage, Online advertising (OA) is now a primary solution for companies wishing to sponsor a good or service. Web

Algorithm 2.4 GP-UCB

Input: Input space D; GP Prior $\mu_0 = 0$, σ_0, k
for $t = 1, 2, ...$ **do**
  Choose $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x} \in D} \mu_{t-1}(\boldsymbol{x}) + \sqrt{\beta_t}\sigma_{t-1}(\boldsymbol{x})$
  Sample $y_t = f(\boldsymbol{x}_t) + \epsilon_t$
  Perform Bayesian update to obtain μ_t and σ_t
end for

technologies also make it possible for advertisers to produce ads that are specifically targeted at certain user demographics and to more effectively control expenses.
The main roles involved in the OA scenario are:

- Advertisers: those that are interested in advertising their goods or services online. They frequently create sub-campaigns to target various niche demographics and have well-defined objectives like boosting the quantity of goods or services sold and maximizing the income relative to costs
- Customers: ordinary people who run into advertisements while surfing the Internet and consequently can become potential buyers
- Publishers: act as middlemen between marketers and consumers, displaying their adverts. They hold auctions where advertisers compete for the slots with the best visibility in order to achieve their goal of maximizing earnings
- Media Agencies: companies that, as experts, are in charge of managing the tasks of the advertisers during the publisher auctions

Advertisers can choose between different online advertising formats among which are:

- Search: when conducting a search on a search engine, customers are displayed advertising. To improve the likelihood of a conversion, the displayed web page combines organic content with tailored advertising depending on the user's search query's keywords
- Social: advertising that is displayed to the consumer on a social network's website through ad-hoc posts, photographs, or videos based on the user's personal information and activity
- Display: adverts that are displayed to customers as banners, photos, and videos on common web pages with the goal of raising awareness of the goods or services they are advertising

The different formats differ for the customers' information they exploit and for the layer

of the AIDA model [4] they target.

2.3.1. Compensation Methods

The performance metrics and payment methods the publisher uses to determine how much the advertiser should pay him are frequently influenced by the format that is chosen.
The main performance indices used in the Online Advertising field are:

1. Impressions: indicates how many times an advertisement has been viewed by a customer
2. Clicks: indicates how many times a customer has clicked on an advertisement to learn more about the product or service on the advertiser's website
3. Leads: indicates the number of potential sales (i.e., registration form filled by the user)
4. Conversions: indicates the number of actual sales of the product or service obtained thanks to that specific advertisement

Based on the reported performance indices we can identify the three most used payment schemes:

- Pay-per-impression: the advertiser pays every time the advertisement receives an impression. It is the most used method for social and display advertising
- Pay-per-click: the advertiser pays every time the customer clicks on the advertisement. This is a very common method for social and search advertising
- Pay-per-conversion: the advertiser pays every time a customer actually buys the product or service offered by advertising. This scheme is very suitable for search advertising

In the rest of this document, we assume that the used format is search advertising and the payment scheme is pay-per-click.

2.3.2. Publisher's Optimization Problem

The publisher's objective is to maximize its revenue by selecting the most effective allocation of advertising among the available slots.
We can formally say that the objective function to be optimized is:

$$\sum_{a} \Lambda_{S(a)} q_a v_a$$

where $\Lambda_{S(a)}$ is the probability that the customer observes the slot in which the advertisement a is allocated (i.e., slot prominence), q_a is the probability that the customer clicks on the advertisement a given that he has observed it (i.e., ad quality) and v_a is the value-per-click assign by the advertiser to the advertisement a (i.e., ad value).
The ratio behind this formula is that customers follow a cascade model, which means they first observe slot number one, then with a given probability move to slot number two, and so on.
The probability to observe slot S_{i+1}, given that slot S_i is observed is given by:

$$\frac{\Lambda_{S_{i+1}}}{\Lambda_{S_i}}$$

where $i \in \{1, \ldots, k\}$ and k is the number of available slots.
The solution to the defined optimization problem:

$$\arg\max_{a} \sum_{a} \Lambda_{S(a)} q_a v_a \tag{2.3}$$

is straightforward. We first sort the advertisements in decreasing order based on the value of $q_a v_a$, and then we assign them to the slots according to that order.
The computational complexity of this solution is linear with respect to the number of advertisements n and logarithmic with respect to the number of slots k, i.e., $O(n\ logk)$.
It is important to note that the publisher frequently does not have the precise values of the parameters defining the optimization problem and he must estimate them. In particular, Δ_s and q_a are estimated using data from all allocated advertisements, whereas v_a is the advertiser's confidential information sent to the publisher in the form of a bid.

2.3.3. Advertiser's Optimization Problem

The advertiser's goal is to obtain the most relevant slots in order to increase his chances of selling his products or services while staying within certain financial constraints.
To begin, the advertiser must define an advertising campaign and all of its sub-campaigns. Each sub-campaign is characterized by:

1. the intended audience, i.e., the geographic area, age, interests, and so on of the customers
2. the bid, which is the maximum amount of money that the advertiser is willing to pay for each impression, click, or conversion based on the selected payment scheme
3. the daily budget, which is the amount of money that the advertiser is willing to

spend on a single sub-campaign

Frequently, the campaign specifies also a total budget, which is simply the sum of the daily budgets of the sub-campaigns that comprise it.
We can formalize the advertiser's optimization problem as an extended version of the Knapsack Problem as follows [10]:

$$\max_{x_{j,t},y_{j,t}} \sum_{j=1}^{N} v_j n_j(x_{j,t}, y_{j,t}) \tag{2.4a}$$

$$\text{s.t} \sum_{j=1}^{N} y_{j,t} \leq \overline{y_t} \tag{2.4b}$$

$$\underline{x}_{j,t} \leq x_{j,t} \leq \overline{x}_{j,t} \quad \forall j \tag{2.4c}$$

$$\underline{y}_{j,t} \leq y_{j,t} \leq \overline{y}_{j,t} \quad \forall j \tag{2.4d}$$

where:

- j denotes the index of a sub-campaign
- N is the total number of sub-campaigns in the campaign
- $x_{j,t}$ is the bid of sub-campaign j at time t
- $y_{j,t}$ is the daily budget of sub-campaign j at time t
- v_j is the estimated value per click for sub-campaign j
- n_j is a function returning the expected number of clicks for sub-campaign j given a chosen bid and a daily budget
- $\overline{y}_t$ is the daily budget for the entire campaign at time t
- $\underline{x}_{j,t}$ and $\overline{x}_{j,t}$ are the minimum value and the maximum value that the bid of sub-campaign j can assume at time t
- $\underline{y}_{j,t}$ and $\overline{y}_{j,t}$ are the minimum value and the maximum value that the daily budget of sub-campaign j can assume at time t

The objective function reported in Equation 2.4a represents the sum of each sub-campaign's expected revenue. The constraint in Equation 2.4b is a budget constraint, requiring that the sum of sub-campaign daily budgets not exceed the total campaign daily budget. Finally, constraints in Equation 2.4c and Equation 2.4d simply force the bid and daily

budget of sub-campaign j to fall within the specified value range at time t.
Obviously, the advertiser does not know the click function a priori, and he must estimate it by collecting observations by applying a bid and a daily budget and observing the effective number of clicks returned by the environment.

2.4. Hotel Advertising's Setting

The end consumer is inextricably linked to the modern hotel industry's core business concept and philosophy. As a result, marketing plays a key role in the hotel's development plans for its own market [2].
The hotel industry inherits everything we've already discussed about the general setting of Online Advertising, but it also has some unique characteristics. Advertisers are specifically performance advertisers, which means they are primarily concerned with short-term yields in order to keep their rooms fully booked.

2.4.1. Publishers

Google Hotel Ads and Tripadvisor are the two most well-known publishers in the Hotel Online Advertising sector.
In the case of Google Hotel Ads, the advertisement is displayed to the user when he searches for a hotel similar to the one advertised on Google Search or Google Maps.
A campaign for this publisher is defined by the following fields:

- The start and the end date
- The total budget
- The payment scheme
- The countries that the campaign must target
- The user devices that the campaign must target (i.e. desktop, mobile, and tablet)

The following payment methods are available to the advertiser:

- Manual cost-per-click (CPC): each time a consumer clicks on the advertisement, the advertiser receives a set amount of money
- Percentage CPC: each time a customer clicks on the advertisement, the advertiser pays a portion of the hotel's room booking price
- Optimized CPC: the bid is automatically changed to optimize conversions

- Pay-per-conversion: each time a consumer makes a reservation, the advertiser pays a fixed proportion of the hotel's room booking price
- Pay-per-stay: after a consumer has stayed in the hotel the advertiser pays a specified proportion of the booking price of the hotel's room

Google Hotel Ads allows you to fine-tune your advertising campaign even further by specifying the three additional criteria listed below:

– Length of Stay (los): the number of days the client wants to hire the accommodation

– Check-In Day: the day the consumer checks into the hotel

– Booking Window Days: the number of days between the day of booking and the day of check-in

The tuple *(hotel, user-country, device, los, check-in day, booking window days)* identifies Google sub-campaigns, and the advertiser can set a separate bid value for each combination of these features.
When a user searches on Tripadvisor's website for a certain destination or the name of a hotel, advertising is displayed to him. Campaigns are established in the same way as Google Hotel Ads, except the advertiser cannot specify some of the booking parameters in this case.
Therefore, a Tripadvisor sub-campaign is identified by the tuple *(hotel, user-country, device)*, and a different bid can be set for each combination of those parameters.

2.4.2. Advertising Campaigns Definition

In this subsection, we present the concept of sub-campaign that we use in our setting due to the fact that we use a slightly different structure from those presented in Subsection 2.4.1.
In our scenario, a campaign is identified by the tuple *(hotel, metasearch)*, where the metasearch represents the advertising channel (i.e., Google Hotel Ads or Tripadvisor).
We do not take into account more complex factors in the definition of sub-campaigns to simplify the treatment of the problem, such as Length of Stay, Check-In Day, and Booking Window Days, so that the sub-campaign is defined by the tuple *(hotel, metasearch, user-country, device)*.
In Figure 2.5, we display a tree depiction of the campaign structure. Each campaign is represented in this tree by a sub-tree anchored at the metasearch layer. Furthermore, each sub-root-to-leaf tree's path indicates a sub-campaign. Thus, in the case depicted, the hotel has two campaigns, one for Google Hotel Ads and one for Tripadvisor, each with

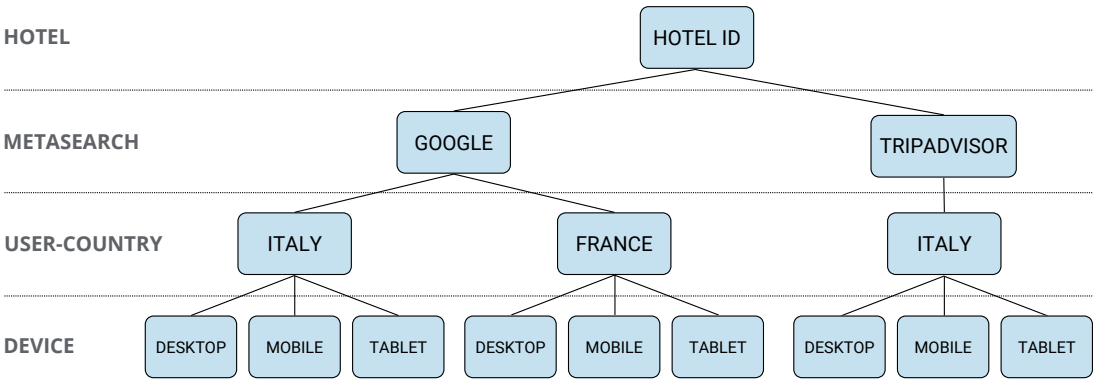


Figure 2.5: Tree representation of the campaign's structure for a specific hotel.

six sub-campaigns.

3 | Related Works

In this chapter, we present a literature review on the bid optimization problem that the advertiser must deal with. In Section 3.1 we focus on the work done by Castiglioni et al. [3], in which a safe approach to the problem is presented, with Return-on-Investment (ROI) Constraints and Budget Constraints subject to uncertainty. In Section 3.2, instead, we introduce the work done by Nuara et al. [11], where are presented techniques, characterized by theoretical guarantees, for the automation of the bid/daily budget optimization of pay-per-click advertising campaigns over multiple channels.

3.1. Safe Online Bid Optimization

The work's key contribution is the introduction of the safety concept for algorithms that choose the bid allocation each day. In particular, the offer should meet, with a high probability, some daily ROI and budget constraints imposed by company business units. Because the constraints' parameters are not known a priori, they are prone to uncertainty. As a result, the goal is to maximize the revenue by satisfying the uncertain constraints. This particular condition is called safety.

3.1.1. ROI and Budget Constraints

Advertisers' goal in online marketing is usually a trade-off between maximizing sales and achieving high profitability. Because the parameter values are uncertain and must be estimated during the sequential arrival of data, this trade-off can be naturally modelled as a combinatorial optimization problem subject to ROI and budget constraints that must be solved online.
Given an advertising campaign $\mathcal{C} = \{C_1, \dots, C_N\}$ with $N \in \mathbb{N}$, and a finite time-horizon $T \in \mathbb{N}$, we can formalize the scenario as follows:

$$\max_{x_{j,t} \in X_j} \sum_{j=1}^{N} v_j n_j(x_{j,t}) \tag{3.1a}$$

$$\text{s.t} \quad \frac{\sum_{j=1}^{N} v_j n_j(x_{j,t})}{\sum_{j=1}^{N} c_j(x_{j,t})} \geq \lambda^* \tag{3.1b}$$

$$\sum_{j=1}^{N} c_j(x_{j,t}) \leq y_t \tag{3.1c}$$

where $n_j(x_{j,t})$ and $c_j(x_{j,t})$ are the expected number of clicks and the expected cost given the bid $x_{j,t}$ for sub-campaign C_j, respectively, and v_j is the value per click for sub-campaign C_j. Constraint 3.1b is the ROI constraint, forcing the revenue to be at least λ^* times the incurred costs, while Constraint 3.1c is the budget constraint ensuring the daily expense to be under a predefined overall budget y_t.
We concentrate on the typical case in which $n_j(\cdot)$ and $c_j(\cdot)$ are unknown functions whose values must be estimated online. This problem can be naturally modelled as a multi-armed bandit, with the available arms being the various bid $x_{j,t} \in X_j$ values that satisfy the optimization problem's combinatorial constraints. A super-arm is an arm profile that specifies a single bid for each sub-campaign. A learning policy $\mathfrak{U}$ for such a problem is an algorithm that returns a set of bids $\{\hat{x}_{j,t}\}_{j=1}^{N}$ for each round t. Because policy $\mathfrak{U}$ can only use estimates of the unknown number-of-click and cost functions created during the learning process, the solutions returned may be suboptimal and/or violate Constraints (3.1b) and (3.1c) when compared to the true values.

3.1.2. Gaussian Combinatorial multi-armed Bandit

In this section, we describe Castiglioni et al. [3] proposed algorithm, Gaussian Combinatorial Multi-armed Bandit (GCB), whose pseudocode is given in Algorithm 3.1, for solving the problem in Equations (3.1a-3.1c) online.
This solution is composed of three parts: Gaussian Processes (GPs) to model the unknown parameters, an estimation subroutine to generate estimates of the parameters from the GPs, and an optimization subroutine, as described in Algorithm 3.2, to solve the optimization problem once the estimates are known.

The noisy realization of the actual number of clicks $\tilde{n}_{j,h}(\tilde{x}_{j,h})$ collected from each sub-campaign C_j for each previous round $h \in \{1, \ldots, t-1\}$ is used by GPs to generate estimates for the expected value $\hat{n}_{j,t1}(x)$ and the standard deviation of the number of clicks $\hat{\sigma}^n_{j,t-1}(x)$ for each bid $x \in X_j$. Similarly, using the noisy realizations of the actual cost $\tilde{c}_{j,h}(\tilde{x}_{j,h})$, with $h \in \{1, \ldots, t-1\}$, GPs generate estimates for the expected value $\tilde{c}_{j,t1}(x)$ and the cost standard deviation $\tilde{\sigma}^C_{j,t1}(x)$ for each bid $x \in X_j$.
The estimation subroutine returns the vector $\boldsymbol{\mu}$, which is made up of the GP estimates. The vector $\boldsymbol{\mu}$ is then passed as input to the optimization subroutine, which solves the

Algorithm 3.1 GCB

Input: sets of bid values $X_1, \ldots, X_N$, ROI threshold λ, daily budget β
Initialize the GPs for the number of clicks and costs
for $t \in \{1, \ldots, T\}$ **do**
  for $j \in \{1, \ldots, N\}$ **do**
    for $x \in X_j$ **do**
      estimate $\hat{n}_{j,t-1}(x)$ and $\hat{\sigma}^n_{j,t-1}(x)$ using the GP on the number of clicks
      estimate $\hat{c}_{j,t-1}(x)$ and $\hat{\sigma}^c_{j,t-1}(x)$ using the GP on the costs
    end for
  end for
  Compute μ using the GPs estimates
  Call the optimization subroutine $Opt(\mu, \lambda)$ to get a solution $\{\hat{x}_{j,t}\}_{j=1}^N$
  Set the prescribed allocation during round t
  Get revenue $\sum_{j=1}^N v_j \tilde{n}_j(\hat{x}_{j,t})$
  Update the GPs using the new information $\tilde{n}_j(\hat{x}_{j,t})$ and $\tilde{c}_j(\hat{x}_{j,t})$
end for

problem posed by Equations (3.1a - 3.1c) and returns the bid strategy for the next round t. Finally, after implementing the strategy, the revenue $\sum_{j=1}^N v_j \tilde{n}_j(\hat{x}_{j,t})$ is obtained, and the stochastic realization of the number of clicks $\tilde{n}_{j,t}(\hat{x}_{j,t})$ and costs $\tilde{c}_{j,t}(\hat{x}_{j,t})$ is observed and provided to the GPs in order to update the models that will be used at round $t+1$.
The GCB algorithm is based on the idea of building the $\boldsymbol{\mu}$ vector so that the parameters corresponding to the reward are statistical upper bounds to the expected values of the random variables, and those corresponding to the costs are statistical lower bounds. The reasoning is that this option fulfils the optimism vs. uncertainty principle.
Formally, we have:

$$\overline{w}_j(x) = \underline{w}_j(x) := v_j \left[\hat{n}_{j,t-1}(x) + \sqrt{b_{t-1}}\hat{\sigma}^n_{j,t-1}(x)\right]$$

$$\overline{c}_j(x) := \hat{c}_{j,t-1}(x) - \sqrt{b_{t-1}\hat{\sigma}^c_{j,t-1}(x)}$$

where $b_t := 2\log\left(\frac{\pi^2 NQTt^2}{3\delta}\right)$ is an uncertainty term used to guarantee the confidence level required by GCB.
Thus, Castiglioni et al. [3] formally demonstrate that the GCB algorithm suffers from a sub-linear pseudo-regret, and it may violate the constraints a linear number of times.

3.1.3. GCB-Safe

To ensure safety, a variant of GCB, called GCB$_{safe}$, is proposed, which relies on different values of $\boldsymbol{\mu}$. Specifically, optimistic estimates for the objective function's parameters and

Algorithm 3.2 Optimization subroutine

Input: sets of bid values $X_1, \dots, X_N$, set of cumulative cost values Y, set of revenue values R, vector $\boldsymbol{\mu}$, ROI threshold λ
Initialize M empty matrix with dimension $|Y| \times |R|$
Initialize $\boldsymbol{x}^{y,r} = \boldsymbol{x}^{y,r}_{next} = [\,], \forall y \in Y, r \in R$
S(y,r) $= \cup\{x \in X_1 | \overline{c}_1(x) \leq y \ \wedge \ \underline{w}_1(x) \geq r\} \ \forall y \in Y, r \in R$
$\boldsymbol{x}^{y,r} = \arg\max_{x \in S} \overline{w}_1(x) \ \forall y \in Y, r \in R$
$M(y,r) = \max_{x \in S} \overline{w}_1(x) \ \forall y \in Y, r \in R$
for $j \in \{2, \dots, N\}$ **do**
  for $y \in Y$ **do**
    for $r \in R$ **do**
      Update $S(y,r)$
      $\boldsymbol{x}^{y,r}_{next} = \arg\max_{s \in S(y,r)} \sum_{i=1}^{j} \overline{w}_i(s_i)$
      $M(y,r) = \max_{s \in S(y,r)} \sum_{i=1}^{j} \overline{w}_i(s_i)$
    end for
  end for
  $\boldsymbol{x}^{y,r} = \boldsymbol{x}^{y,r}_{next}$
end for
Choose (y^*, r^*)
Output: $\boldsymbol{x}^{y^*,r^*}$

pessimistic estimates for the constraints' parameters are used.
In formal terms, we have:

$$\overline{w}_j(x) := v_j \left[\hat{n}_{j,t-1}(x) + \sqrt{b_{t-1}}\hat{\sigma}^n_{j,t-1}(x)\right]$$

$$\underline{w}_j(x) := v_j \left[\hat{n}_{j,t-1}(x) - \sqrt{b_{t-1}}\hat{\sigma}^n_{j,t-1}(x)\right]$$

$$\overline{c}_j(x) := \hat{c}_{j,t-1}(x) - \sqrt{b_{t-1}\hat{\sigma}^c_{j,t-1}(x)}$$

Furthermore, GCB$_{safe}$ requires a default set of bids, that is known to be feasible with the actual values of the parameters. The pseudocode of GCB$_{safe}$ is the same as Algorithm 3.1 with the parameters vector $\boldsymbol{\mu}$ defined above.
At the cost of a linear pseudo-regret, this new version of the algorithm guarantees a constant upper bound on the number of constraint violations.

3.2. Online Joint Bid/Daily Budget Optimization

In their work, Nuara et al. investigate the automation of the joint bid/daily budget optimization of advertising campaigns in online fashion. They propose AdComB, an algorithm that uses Gaussian Processes to estimate the campaigns' performance, and combinatorial

bandit techniques to address the exploration/exploitation dilemma in bid/daily budget selection. Furthermore, it uses dynamic programming techniques to solve the allocation optimization problem. Ultimately, they give theoretical guarantees for the algorithm by providing high probability bounds on the regret of $O(\sqrt{T})$, where T is the learning process's time horizon.

3.2.1. Learning Problem Formulation

In real-world scenarios, the functions n_j in the optimization problem described in Equations (2.4a-2.4d) are not known a priori and must be estimated online. Thus, an algorithm must gather as much information about these functions as possible during the system's operating life without losing too much revenue in investigating inefficient bid/daily budget allocations. As a result, Nuara et Al. formulate the learning problem in a sequential decision fashion as a Combinatorial Semi Bandit problem (CSB).
In the stated problem, each arm corresponds to a pair of bid/daily budget, each superarm corresponds to a pair for each campaign, and the constraints consist of the maximum value for the daily budget and the feasible range of values for the bid and the daily budget. The revenue generated by setting a bid/daily budget pair is the payout for each arm. Everyday t, an advertiser chooses a feasible superarm and obtains the revenue expressed in terms of clicks and value per click.

3.2.2. AdComB Algorithm

The proposed algorithm, named Advertising Combinatorial Bandit (AdComB), takes as input the discrete set of bid values X, the discrete set of daily budget values Y, the models $\mathcal{M}_j^{(0)}$ that, for each campaign C_j, capture the prior knowledge of the learner about the function $n_j(\cdot,\cdot)$, a cumulative daily budget $\overline{y}$, and a time horizon T.
The pseudocode of AdComB is reported in Algorithm 3.3.
In the initial phase of the algorithm, the $\mathcal{M}_j$ models are updated according to the observations of the previous days $\{1,\dots,T\}$, with the exception of the first day where the models are set to the prior models $\mathcal{M}_j^0$ provided as input (Lines 4-8). In the second phase, the algorithm uses the updated models $\mathcal{M}_j$ to infer an estimate of the functions $n_j(\cdot,\cdot)$, for every value of the bid and the daily budget in X and Y, and of the parameters v_j. These estimates are denoted, respectively, with $\hat{n}_j(\cdot,\cdot)$ and $\hat{v}_j$ (Line 9). As is well known in the bandit literature, the naive choice of using the expected value obtained from $\mathcal{M}_j$ may not provide any guarantee of minimizing the regret. In order to ensure that the algorithm minimizes the cumulative expected regret, Nuara et Al. compute an estimation based on the uncertainty information provided by model $\mathcal{M}_j$. In particular, the click functions

Algorithm 3.3 AdComB

1: **Input:** set X of bid values, set Y of daily budget values, prior models $\mathcal{M}_j^{(0)}$, cumulative daily budget $\overline{y}$, time horizon T
2: **for** $t \in \{1, \dots, T\}$ **do**
3: **for** $j \in \{1, \dots, N\}$ **do**
4: **if** $t = 1$ **then**
5: $\mathcal{M}_j = \mathcal{M}_j^{(0)}$
6: **else**
7: $Update(\mathcal{M}_j)$
8: **end if**
9: $(\hat{n}_j(\cdot,\cdot), \hat{v}_j) = Exploration(\mathcal{M}_j, X, Y)$
10: **end for**
11: $\{(\hat{x}_{j,t}, \hat{y}_{j,t})\}_{j=1}^N = Optimize(\{\hat{n}_j(\cdot,\cdot), \hat{v}_j, X, Y\}_{j=1}^N, \overline{y})$
12: $Pull(\hat{x}_{1,t}, \hat{y}_{1,t}, \dots, \hat{x}_{N,t}, \hat{y}_{N,t})$
13: **end for**

$n_j(\cdot)$ are replaced in the optimization problem, stated in Equations (2.4a-2.4d), by:

$$u_{j,t} := \hat{\mu}_{j,t}(x) + \sqrt{b_{j,t}}\hat{\sigma}_{j,t}(x)$$

where $\hat{\mu}_{j,t}(x)$ and $\hat{\sigma}_{j,t}(x)$ are the mean and the variance provided by the GP modelling $n_j(\cdot)$ and $b_{j,t}$ is a non-negative sequence of values necessary for the convergence of the algorithm to the optimal solution.
In the third phase, the algorithm solves the optimization problem, by exploiting the estimates $\hat{n}_j(\cdot,\cdot)$ and $\hat{v}_j$, through a modified version of the algorithm by Kellerer et al.[7] for the knapsack problem, and returns the bid/daily budget allocation for the next day t (Lines 11-12).

3.2.3. Regret Analysis

Nuara et Al. provide, furthermore, a theoretical finite-time analysis of the regret of the proposed algorithm.
For the AdComB algorithm with $b_t = 2\log\left(\frac{\pi^2 N|X|t^2}{2\delta}\right)$ and $b'_t = 2\log\left(\frac{\pi^2 N t^2}{2\delta}\right)$, for every $\delta \in (0,1)$, applied to an advertisement bid/daily budget allocation problem over T rounds, holds that:

$$L_T \leq \left\{ TN \left[\overline{c}_1 b_T \sum_{j=1}^{N} \gamma_T(n_j) + \overline{c}_2 b_T \sum_{j=1}^{N} \gamma_T(e_j) + \overline{c}_3 b'_T (2sy_{max}\sqrt{b_T} + 2\sigma\sqrt{b_T} + n_{max}^{sat})^2 \sum_{j=1}^{N} \log\left(\frac{\xi}{\Psi_j^2} + T\right) \right] \right\}^{\frac{1}{2}}$$

where the functions $n_j^{sat}(x)$ and $e_j^{sat}(x)$ model, respectively, the maximum number of clicks and the number of clicks per unit of daily budget.
The coefficients $\overline{c}_1$, $\overline{c}_2$ and $\overline{c}_3$ are defined as:

$$\overline{c}_1 := \frac{12v_{max}^2}{\log\left(1+\frac{1}{\lambda}\right)} \quad , \quad \overline{c}_2 := \frac{12v_{max}^2 y_{max}^2}{\log\left(1+\frac{1}{\lambda'}\right)} \quad , \quad \overline{c}_3 := 12\xi$$

Moreover, ξ, λ and λ' are, respectively, the variance of the value per click, of the measurement noise on the maximum number of clicks, and of the number of clicks per unit of daily budget.
Thanks to the reported result, it can be stated that AdComB suffers from a sub-linear pseudo-regret since the terms $\gamma_T(n_j)$ and $\gamma_T(e_j)$ are bounded by $O((\log T)^2)$ and, consequently, the bound for L_T is $O(N(\log T)^2\sqrt{T})$.

4 | Problem Formulation

In this chapter, we outline the Cold-Start problem that we encountered and attempted to mitigate in our work. In particular, in Section 4.1 we describe the problem in the context of a generic recommendation system, whereas in Section 4.2 we delve into the specific case of the online advertising setting on which our experiments have been focused.

4.1. Cold-Start Recommendation

Recommender systems are Machine Learning applications that aim to recommend items/actions to users based on the information available. The Cold-Start problem occurs when the system is unable to establish any relationship between users and items/actions due to a lack of data [16].
There are two types of Cold-Start problems:

- User Cold-Start problem: when almost no information about the user is available
- Item Cold-Start problem: when almost no information about the item/action is available

There are three main reasons why the system fails to provide recommendations, as shown in Figure 4.1:

1. Systematic Bootstrapping: refers to the start of the system when the recommender has almost no information on which to rely. Because all of the items and users are new, this case is a hybrid of the Low Interaction and New Users cases
2. Low Interaction: when new catalogue items are added, they have no or very few interactions. This could be a problem because the algorithm recommends items based on their interactions. It can recommend even if only a few interactions are available, but the quality of those recommendations will be poor
3. New Users Entry: when a new user joins the system and the recommender is required to offer recommendations without relying on the user's previous interactions because none have yet been recorded. This is a critical issue because a user who receives

poor-quality recommendations may opt to the left of the system before allowing the recommender to understand his or her interests

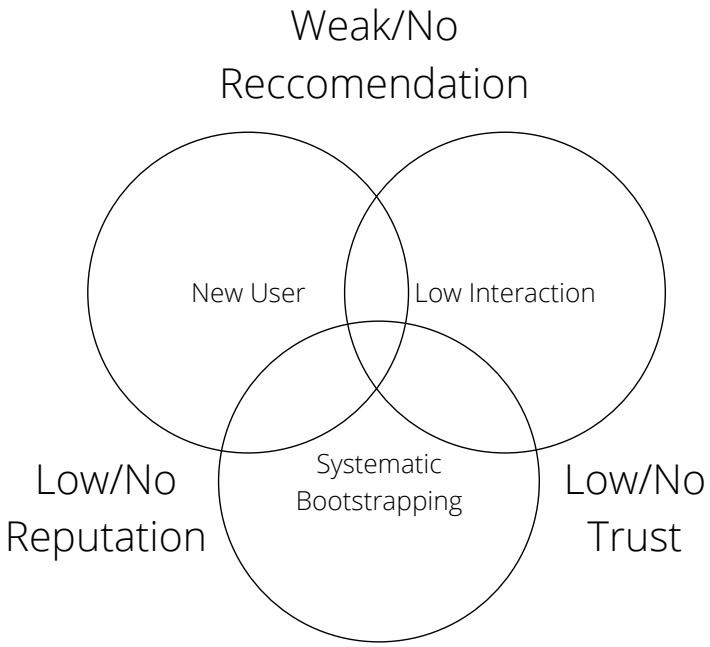


Figure 4.1: Main causes of Cold-Start in modern Recommender Systems.

4.2. Cold-Start in Online Advertising

The Cold-Start problem can arise for both the publisher and the advertiser in the context of online advertising. We present the problem from the perspective of the publisher in Subsection 4.2.1, and from the perspective of the advertiser in Subsection 4.2.2, which is the setting on which we focused in our work.

4.2.1. Publisher's Scenario

According to Subsection 2.3.2 the publisher must estimate both the probability that a slot can be observed $\Lambda_{S(a)}$ and the probability that a customer clicks on a specific advertisement q_a in order to solve its optimization problem, as stated in Equation 2.3. These two quantities can be combined into one called Click Through Rate (CTR), defined as:

$$CTR = \Lambda_{S(a)} q_a$$

As a result, instead of estimating the two individual quantities, the estimation problem can be reduced to CTR estimation.

To accomplish this, cutting-edge prediction algorithms rely heavily on historical data, and as a result, they perform poorly on an increasing number of new ads with no historical data [9]. This is an excellent example of the Item Cold-Start problem, in which the item for which we have no previous observations is the new ads, and thus we cannot make a precise estimate of its CTR.
Because of the importance of new ads, this creates a huge problem for the online advertising industry. Users quickly tire of old ads in fast-changing markets, so sellers must frequently update them. As a result, most advertisements have a limited lifespan. Furthermore, an increasing number of new sellers are hoping that their ads will be shown through ad networks. In this case, new ads account for a sizable portion of all ads. If the click prediction system does not pay enough attention to new ads, it will not be able to collect new user feedback on them, and the system may eventually enter a self-destructive cycle.

4.2.2. Advertiser's Scenario

The Cold-Start problem may occur at various levels in the advertiser's configuration. The advertiser, like the publisher, must estimate some parameters to solve the optimization problem in Equations (2.4a - 2.4d). The clicks function, which returns the expected number of clicks for a sub-campaign given a bid and a daily budget, necessitates a large amount of data for its estimation.
This means that when the advertiser decides to open new sub-campaigns to target a new audience, as shown in Figure 4.3, it must contend with the fact that the new sub-campaign lacks initial data. As a result, he finds himself in the Item Cold-Start problem, where the item about which he knows nothing due to low interaction is the new sub-campaign itself.

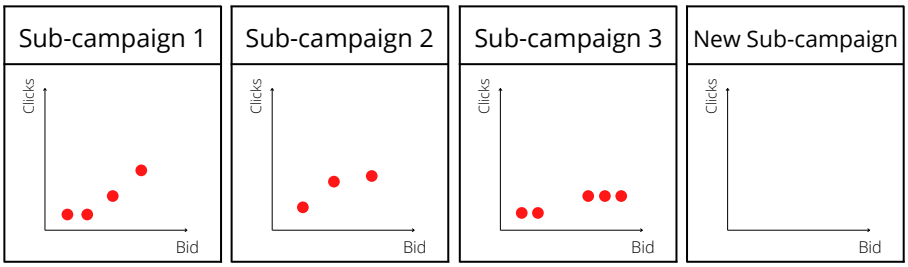


Figure 4.2: Cold-Start problem due to the opening of a new sub-campaign.

Solving the optimization problem with an inaccurate and identical clicks function for all

new sub-campaigns for which we have not yet collected enough data online may result in proposing bids that are far from optimal, resulting in huge economic losses for the advertiser and a loss of trust in the automation tools used.
User Cold-Start may occur in media agencies that manage multiple advertiser campaigns. In fact, these agencies frequently receive requests from new customers who decide to entrust them with the management of their online advertising campaigns. As a result, they are forced to solve the optimization problem for all new customers' sub-campaigns without any historical data on any of them, as shown in Figure 4.3.

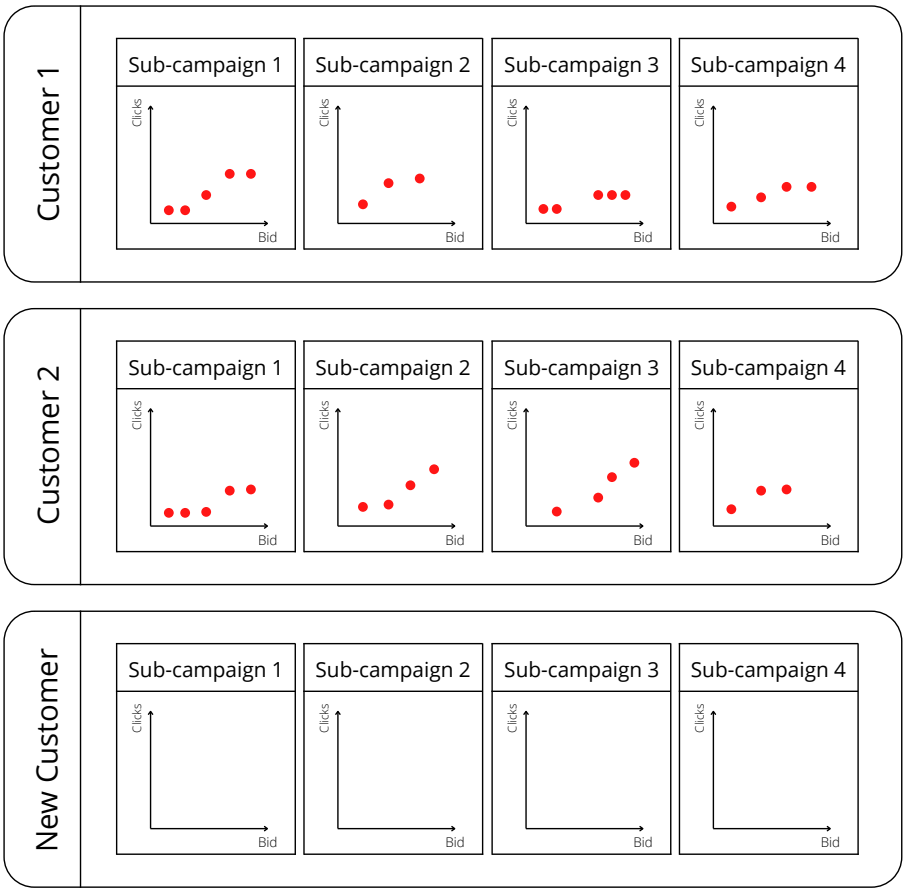


Figure 4.3: Media agencies' Cold-Start Problem due to the acquisition of a new customer.

Due to a lack of initial observations for each sub-campaign, completely different click functions may be used to solve optimization problems. As a result, the first solutions used by media agencies for new customers may be extremely far from the optimal real ones, risking losing the new customer who is scared of the initial economic loss.

5 | Proposed Solution

In this chapter, we present the methodology developed to address the Cold-Start Problem in the context of Online Advertising. Particularly, Section 6.1 focuses on the offline phase of the proposed approach, in which we build the starting GP to be used in Cold-Start situations. Section 5.2 concentrates on the online phase, and specifically on how we solve the optimization problem for the advertiser. Finally, Section 5.3 shows the overall structure of our algorithm.

5.1. Combining Gaussian Processes

Our strategy for solving the Cold-Start Problem involves combining the GPs of selected sub-campaigns to produce a starting GP that is more insightful than the conventional zero-centred prior. In-depth, when a hotel wants to initiate a new sub-campaign in a new country on a specific device, we combine the GPs of the N sub-campaigns opened in other countries on the same device to determine the starting GP of the new sub-campaign.
In particular, fixed the bid's vector $\boldsymbol{x} = (x_1 \dots x_n)$, we have that the number of clicks of the i-th sub-campaign is distributed according to a multivariate normal distribution:

$$n_i(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$

The convex combination $n_{\boldsymbol{\alpha}}$ of the n_i is still a multi-variate normal distribution of the type:

$$n_{\boldsymbol{\alpha}}(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i n_i(\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\alpha}}, \boldsymbol{\Sigma}_{\boldsymbol{\alpha}})$$

where

$$\boldsymbol{\mu}_{\boldsymbol{\alpha}} = \sum_{i=1}^{N} \alpha_i \boldsymbol{\mu}_i \tag{5.1a}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\alpha}} = \sum_{i=1}^{N} \alpha_i^2 \boldsymbol{\Sigma}_i \tag{5.1b}$$

The $\alpha_i \in [0, 1]$ coefficient is the weight of the convex combination associated with the i-th GP for which applies that:

$$\sum_{i=1}^{N} \alpha_i = 1 \quad , \quad \alpha_i \geq 0 \qquad \forall i$$

The fundamental purpose of combining GPs is to identify the typical behaviour that a specific hotel can exhibit on a given user device. In other words, it is legitimate to suppose that each hotel has its own distinctive trend that expresses itself and assumes different subtleties in each country. The idea is that starting with a GP that includes the hotel's intrinsic behaviour instead of the zero-centred prior may help the optimization routine in the exploration phase leading to more appropriate bid suggestions even from the early stages of the optimization.
It is important to underline that each engineering implementation works in a discrete space, while the mean function and the kernel of the Gaussian Process are operators defined in a continuous one. Accordingly, the mean vector $\boldsymbol{\mu_\alpha}$ and the covariance matrix $\boldsymbol{\Sigma_\alpha}$ returned by Equation 5.1a and Equation 5.1b have to be exploited to estimate the GP's characteristic parameters.
The mean function can simply be obtained by interpolation of the points contained in the vector $\boldsymbol{\mu_\alpha}$. Instead, to estimate the length scale parameter l defining the kernel function we observe that:

$$k(x_i, x_j) = \exp\left(-\frac{d(x_i, x_j)^2}{2l^2}\right) = \boldsymbol{\Sigma}^{\boldsymbol{\alpha}}_{i,j} \quad \forall x_i, x_j$$

taking the log on both sides of the equation we obtain:

$$-\frac{1}{2l^2} d(x_i, x_j)^2 = \log\left(\boldsymbol{\Sigma}^{\boldsymbol{\alpha}}_{i,j}\right) \qquad \forall x_i, x_j$$

then, due to the fact that the two matrices are symmetric and positive-definite, we can rearrange the elements situated in the main diagonal and above it in two vectors, respectively $\boldsymbol{D}$ and $\boldsymbol{\Sigma^\alpha}$:

$$-\frac{1}{2l^2}\boldsymbol{D} = \log\left(\boldsymbol{\Sigma^\alpha}\right)$$

solving the resulting system with the Least Squares method and making explicit $\hat{l}$ from the equation, we finally get:

$$\hat{l} = \sqrt{-\frac{1}{2(\boldsymbol{D}^T\boldsymbol{D})^{-1}\boldsymbol{D^T}\log\left(\boldsymbol{\Sigma^\alpha}\right)}} \tag{5.2}$$

In Algorithm 5.1, we present the pseudocode of the described heuristic, which builds the resulting gaussian process $n_{\boldsymbol{\alpha}}$ using the GPs fitted on the data gathered up to time t_0 for the N sub-campaigns.

Algorithm 5.1 Combine GPs

1: **Input:** set of gaussian processes GPS, vector of bids $\boldsymbol{x}$, vector of weights $\boldsymbol{\alpha}$
2: **for** $i = 1, \dots, N$ **do**
3: $\quad \boldsymbol{\mu}, \boldsymbol{\Sigma}$ =GPS$[i].predict(\boldsymbol{x})$
4: $\quad \boldsymbol{\mu_\alpha}$ += $\boldsymbol{\alpha}[i] * \boldsymbol{\mu}$
5: $\quad \boldsymbol{\Sigma_\alpha}$ += $\boldsymbol{\alpha}[i]^2 * \boldsymbol{\Sigma}$
6: **end for**
7: **for** $i = 1, \dots, \boldsymbol{x}$ **do**
8: $\quad$ **for** $j = i + 1, \dots, \boldsymbol{x}$ **do**
9: $\quad\quad \boldsymbol{D}[i] = (\boldsymbol{x}[i] - \boldsymbol{x}[j])^2$
10: $\quad\quad \boldsymbol{\Sigma_\alpha}[i] = \boldsymbol{\Sigma_\alpha}[i, j]$
11: $\quad$ **end for**
12: **end for**
13: Compute $\hat{l}$ as

$$\hat{l} = \sqrt{-\frac{1}{2(\boldsymbol{D}^T\boldsymbol{D})^{-1}\boldsymbol{D}^T \log(\boldsymbol{\Sigma_\alpha})}}$$

14: $K = RBF(\hat{l})$
15: $f = interpolate(\boldsymbol{\mu_\alpha})$
16: **return** GP(f, K)

Given the set GPS of the gaussian processes of the N sub-campaigns, the vector $\boldsymbol{x}$ representing the discretized bid space and the vector $\boldsymbol{\alpha}$ containing in i-th position the weight associated with the i-th GP, the algorithm returns the gaussian process $n_{\boldsymbol{\alpha}}$ obtained as a convex combination of the N GPs in GPS.
In the beginning, we compute the mean vector and the covariance matrix of the resulting GP, according to Equations (5.1a - 5.1b), by extracting the mean vector and the covariance matrix for the i-th GP on the discretized bid space $\boldsymbol{x}$ (Lines 2-6). Then, we build the distances vector $\boldsymbol{D}$ and the covariance vector $\boldsymbol{\Sigma_\alpha}$ that allow us to move to the matrix representation of the problem (Lines 7-11). Subsequently, we compute the length scale parameter $\hat{l}$ according to Equation 5.2 (Line 13), we initialize the radial basis kernel with the computed length scale (Line 14) and we estimate the mean function f by interpolation of the points in $\boldsymbol{\mu_\alpha}$ (Line 15). Finally, we instantiate and return the resulting Gaussian Process according to the calculated parameters (f, K) (Line 16).

5.2. Optimization Problem Resolution

In this section, we describe the algorithm we employ in the online phase to solve the advertiser's optimization problem stated in Equations (2.4a - 2.4d).
It is necessary to make the following two assumptions in order to reduce the complexity of the problem and make it more tractable:

1. The performance of every sub-campaign is independent of other sub-campaigns performance
2. The values of the bid and daily budget are finite

These are not true in general, given that the sub-campaigns can target users at different levels of the AIDA model and advertisers could choose any bid and budget value in real space, but they are satisfactory approximations in most real-world situations.
Once we fix a discrete set of values $x_1, x_2, \dots, x_n$, and $y_1, y_2, \dots, y_m$ respectively for the bid and for the budget, then we can summarize the values for a sub-campaign in a table in which the cell in position (i, j) contains the product $v \cdot n(x_i, y_j)$, where v is the value per click of the considered sub-campaign and $n(x_i, y_j)$ is the value returned by its click function for the bid x_i and the budget y_j. Table 5.1 shows an example for a sub-campaign with $n = m = 8$.

$v \cdot n(x, y)$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
y_1	100	110	115	108	100	95	90	90
y_2	120	130	140	132	125	120	115	115
y_3	135	140	143	145	138	130	125	120
y_4	148	153	156	160	152	145	137	130
y_5	152	155	158	163	165	157	140	135
y_6	158	160	163	165	168	165	150	145
y_7	164	168	170	174	178	170	163	158
y_8	166	170	174	178	181	184	171	160

Table 5.1: Representation of the values assumed by $v \cdot n(x, y)$ for a given sub-campaign for all possible combinations of bid and budget in their discretized and finite spaces.

In order to solve the optimization problem, the first step of the algorithm is to remove the dependency on the bid. The easiest way to achieve this is to take, for each daily budget, the maximum among all the possible values of the bid. In other words, for each row of Table 5.1 we take the cell with the highest value, as reported in Table 5.2.

$v \cdot n(x,y)$	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	max
y_1	100	110	115	108	100	95	90	90	115
y_2	120	130	140	132	125	120	115	115	140
y_3	135	140	143	145	138	130	125	120	145
y_4	148	153	156	160	152	145	137	130	160
y_5	152	155	158	163	165	157	140	135	165
y_6	158	160	163	165	168	165	150	145	168
y_7	164	168	170	174	178	170	163	158	178
y_8	166	170	174	178	181	184	171	160	184

Table 5.2: Removal of the dependency on the bid value by the addition of the max column that takes into account the maximum value associated with each budget.

Once we remove the dependency on the values of the bid we can visualize the problem with a table, where in the rows we have all the sub-campaigns of the given campaign, in the columns we have the values of the daily budget and in the cell (i, j) the j-th element of the max column of the i-th sub-campaign. Moreover, in this new representation, we use minus infinity for the non-feasible cases, i.e. values of the daily budget which are smaller than its lower bound or larger than its upper bound. Table 5.3 shows an example for a Campaign C with five sub-campaigns $\mathrm{C}_1, \ldots, \mathrm{C}_5$ and a discretized budget space from 0 to 70 with a step size of 10.

	0	**10**	**20**	**30**	**40**	**50**	**60**	**70**
C_1	$-\infty$	90	100	105	110	$-\infty$	$-\infty$	$-\infty$
C_2	0	82	90	92	$-\infty$	$-\infty$	$-\infty$	$-\infty$
C_3	0	80	83	85	86	$-\infty$	$-\infty$	$-\infty$
C_4	$-\infty$	90	110	115	118	120	$-\infty$	$-\infty$
C_5	$-\infty$	111	130	138	142	148	155	$-\infty$

Table 5.3: Representation of the considered campaign's sub-campaigns deprived of the dependence on the bid values and with the exclusion of non-feasible cases.

The next step of the algorithm is basically an extension of the dynamic programming algorithm used to solve the Knapsack Problem. We build a new table where the first row r_0 is initialized with all zeros and every other row r_i corresponds to the adding of the i-th sub-campaign C_i to the problem. The value of the cell (i, j) is given by taking the maximum value between the sums of the values provided by the best solution of the problem solve in the previous row and the new considered sub-campaign C_i such that the daily budget sums to the j-th value of the discretized budget space. Table 5.4 reports the

	0	**10**	**20**	**30**	**40**	**50**	**60**	**70**
0	0	0	0	0	0	0	0	0
$+C_1$	$-\infty$	90	100	105	110	110	110	110
$+C_2$	$-\infty$	90	172	182	190	195	200	202
$+C_3$	$-\infty$	90	172	252	262	270	275	280
$+C_4$	$-\infty$	$-\infty$	180	262	342	362	372	380
$+C_5$	$-\infty$	$-\infty$	$-\infty$	291	373	453	473	492

Table 5.4: Algorithm's final table containing the optimal solution of the optimization problem as the maximum value of its last row.

results computed by the algorithm for the campaign presented in Table 5.3.

Algorithm 5.2 Optimize Campaign

Input: set of sub-campaigns $C_1, \ldots, C_k$
Initialization:
matrix M_i of type $n \times m$ for each sub-campaign C_i
matrix A of type $k \times m$
matrix U of type $(k+1) \times m$
for $i = 1, \ldots, k$ **do**
  for $j = 1, \ldots, m$ **do**
    for $l = 1, \ldots, n$ **do**
      $C_i[j, l] = v_i n_i(x_l, y_j)$
    end for
  end for
end for
for $i = 1, \ldots, k$ **do**
  for $j = 1, \ldots, m$ **do**
    if $\underline{y_j} \leq y_j \leq \overline{y_j}$ **then**
      $\overline{A}[i, j] = max(M_i[j, :])$
    else
      $A[i, j] = -\infty$
    end if
  end for
end for
for $i = 1, \ldots, k$ **do**
  for $j = 1, \ldots, m$ **do**
    $U[i, j]$ set to the max of $U[i-1, p] + A[i, q] \quad \forall\, p, q : y_p + y_q = y_j$
  end for
end for
return $max(U[k+1, :])$

The maximum value associated with the last row of this table, the one which takes into

account all the sub-campaigns, is the solution to the optimization problem of the advertiser for the campaign C.
The pseudo-code of the presented approach is reported in Algorithm 5.2.
First, we initialize three empty matrices used to store the results of each procedure step (Line 2). In particular, the matrix M_i is used to store for the i-th sub-campaign the structure presented in Table 5.1, the matrix A is used to store the campaign's new representation with sub-campaigns deprived of the dependence on the bid values, Table 5.3, and ultimately matrix U is the final table computed by the algorithm containing the optimal solution of the optimization problem, as shown in Table 5.4.
Then, for each of the k sub-campaigns, we fill the respective M_i matrix according to its value-per-click and click function (Lines 3-9). Hence, we supply the values for the A matrix by taking, for each sub-campaign C_i, the maximum of each row of the relative matrix M_i where the row is associated with a feasible budget and $-\infty$ for the row associated with a non-feasible budget (Lines 10-18).
Finally, we compute iteratively the row u_i of the U matrix associated with the dynamic program by taking the maximum of all the feasible cross-sums between the row u_{i-1} of the U matrix itself and the i-th row of the A matrix (Lines 19-23).
In the end, the algorithm returns the optimal solution by taking the maximum value contained in the last row of the U matrix (Line 24).

5.3. Proposed Algorithm

The algorithm we propose, whose pseudocode is reported in Algorithm 5.3, can be broken down into two main phases. The first phase consists of the initialization of the GPs associated with each sub-campaign of the campaign to be optimized.
In particular, the sub-campaigns for which observations are available are associated with the GPs fitted on that data, while the new sub-campaign is associated with a starting GP generated through the convex combination of the GPs of other correlated sub-campaigns, as explained in Algorithm 5.1 (Lines 2-8).
The second phase, instead, is the actual online part of the algorithm and consists of three steps that are repeated each day $t \in \{1, \dots, T\}$. The first step is to obtain the optimal bids $\tilde{x}_1, \dots, \tilde{x}_N$ for each sub-campaign, according to the algorithm presented in Algorithm 3.2, based on the GPs assigned in the previous phase.
Then, the second step is to use the optimal bids and collect the real number of clicks $o_1, \dots, o_N$ for each sub-campaign from the environment. Finally, the third step is the updating of the GPs according to the collected observations and their use as starting estimations for the following day (Lines 9-13).

Algorithm 5.3 Proposed Algorithm

Input: Campaign C, new sub-campaign index j
for $i = 1, 2, \ldots, N$ **do**
  if $i \neq j$ **then**
    $n_i = fit(C_i)$
  else
    $n_i = combine(C, j)$
  end if
end for
for $t = 1, 2, \ldots, T$ **do**
  $\tilde{x}_1, \ldots, \tilde{x}_N = optimize(n_1, \ldots, n_N)$
  $o_1, \ldots, o_N = play(\tilde{x}_1, \ldots, \tilde{x}_N)$
  $update(n_1, \ldots, n_N, o_1, \ldots, o_N)$
end for

In the optimization phase of the algorithm we use as click function $n_{i,t}(\cdot)$ for the i-th sub-campaign at time t:

$$n_{i,t}(x) = \mu_{i,t}(x) + \sqrt{b_{i,t}}\sigma_{i,t}(x)$$

where $\mu_{i,t}(x)$ and $\sigma_{i,t}(x)$ are the mean and the variance provided by the GP associated with the sub-campaign. The coefficient $b_{i,t} \in \mathbb{R}^+$ is a non-negative value required to set optimistic bounds necessary for the convergence of the algorithm to the optimal solution. We refer to the work by Nuara et al. [11], reported in Section 3.2, for its definition and properties.

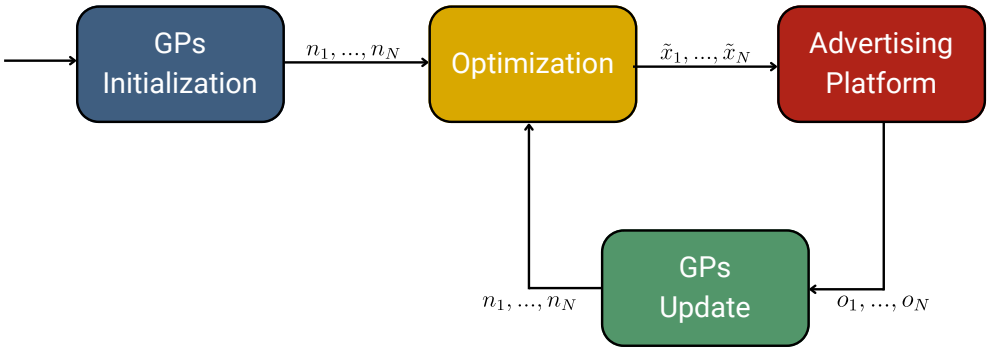


Figure 5.1: The information flow in the proposed algorithm along with its main phases.

Figure 5.1 reports the scheme of the presented algorithm, highlighting its main phases and the information flow among them. The optimization phase is fed by the starting

estimations $n_1, \ldots, n_N$ generated by the initialization phase. Then, the loop is entered and repeated T times. The advertising platform receives the optimal bids $\tilde{x}_1, \ldots, \tilde{x}_N$ computed by the optimization phase and produces the observations $o_1, \ldots, o_N$ on the real number of clicks collected. Finally, the last phase exploits the observations to update the estimations and provides them to the optimization phase of the next iteration.

6 | Experiments

In this chapter, we outline the experiments conducted to validate the proposed approach. In particular, in Section 6.1 we describe the methodology used to create the online environment simulation in which to evaluate the performance of our heuristics. Then, in Section 6.2 we focus on the preliminary experiments done in the synthetic environment to verify whether a more informed prior can positively influence the solution of the optimization problem. Finally, in Section 6.3 we move to the experiments performed in the real-world scenario through the data provided by the advertising company AdsHotel.

6.1. Online Environment Simulation

The impossibility to connect with a live online scenario is the first issue we ran into when we had to evaluate our approach. Indeed, media agencies are frequently hesitant to test new techniques on their customers for economic and reliability reasons.
In order to overcome this limitation we simulate the online environment with the data at our disposal. In detail, we pretend not to have the observations O_h of a given sub-campaign C_h, which we assume to be the one where the advertiser wants to make the new opening. Hence we exploit the gaussian process GP_h, trained on the observations O_h and estimating the click function $n_{h,t}$ of C_h, to simulate the values that the advertiser would get day by day.
More formally, at any time t, for the newly opened sub-campaign C_h, we state that, fixed the bids vector, the real number of clicks $n_{h,t}$ obtained by playing the bid $x_{i,t}$ is given by:

$$n_{h,t} \sim \mathcal{N}(\mu_{GP_h}(x_{i,t}), \sigma_{GP_h}(x_{i,t}))$$

where $\mu_{GP_h}(x_{i,t})$ is the value of GP_h's mean function for the bid $x_{i,t}$, and $\sigma_{GP_h}(x_{i,t})$ is the GP_h's estimated uncertainty for the number of clicks associated to the bid $x_{i,t}$.
Assuming GP_h is the process that generates real data for the sub-campaign C_h allows us to build a framework in which to evaluate our approach without interfacing directly with a real online environment and having to wait days to collect an adequate number of observations.

6.2. Synthetic Data Setting

In order to understand the influence that an informed prior can have on solving the optimization problem, we performed preliminary tests in a synthetic environment. In particular, we chose n arbitrary functions that we pretended to be the click functions associated with n sub-campaigns. Figure 6.1 shows an example with four sub-campaigns.

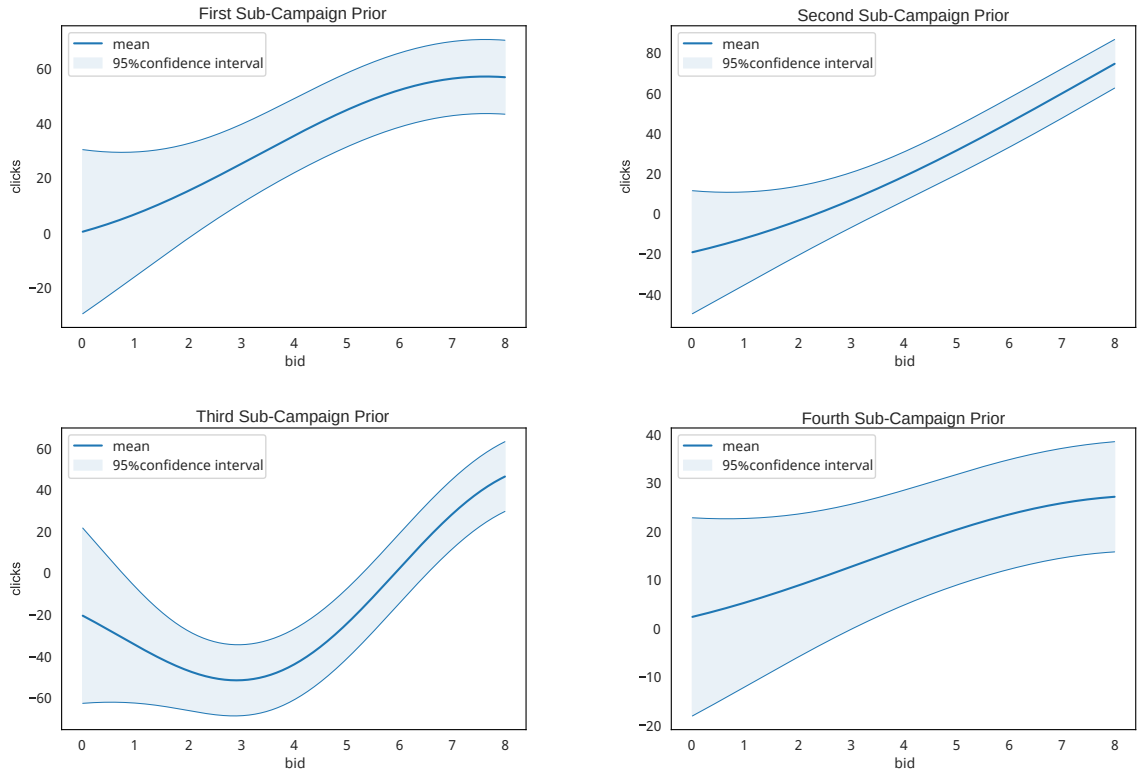


Figure 6.1: Synthetic clicks functions associated with the four sub-campaigns characterizing the fictitious advertising campaign to be optimized online.

The first step of our approach is to run the optimization routine, reported in Algorithm 5.2 on the n sub-campaigns each with its assigned real click function. In this way, we obtain the optimal solution to the problem identified by the optimal bid and the optimal budget for each of the campaign's sub-campaigns.
After that, we simulate not having the click function associated with the $(n-1)$-th sub-campaign and pretend that it corresponds to the new opening desired by the advertiser. Then, we put aside the GP associated with that click function and we use it to simulate the online scenario as explained in Section 6.1.

The next step is to run in parallel the campaign optimization for a time horizon T using as the starting GP of the sub-campaign to open in one case the zero-centred prior and in the other case, an informed prior characterized by a GP similar to the true click function of the sub-campaign.
For each day $t \in T$ we collect observations related to bids and budgets suggested by the optimization program and re-fit the GPs associated with each sub-campaign according to these newly collected data. The new GPs obtained in this way are then used to solve the optimization problem in the following day $t + 1$.
In Figure 6.2 we report the four starting GPs used for the campaign illustrated in Figure 6.1 in case we pretend that the advertiser has three open sub-campaigns, respectively those associated with the first three GPs, and wants to open a new sub-campaign, the fourth, for which the zero-centred prior is used as the initial GP.

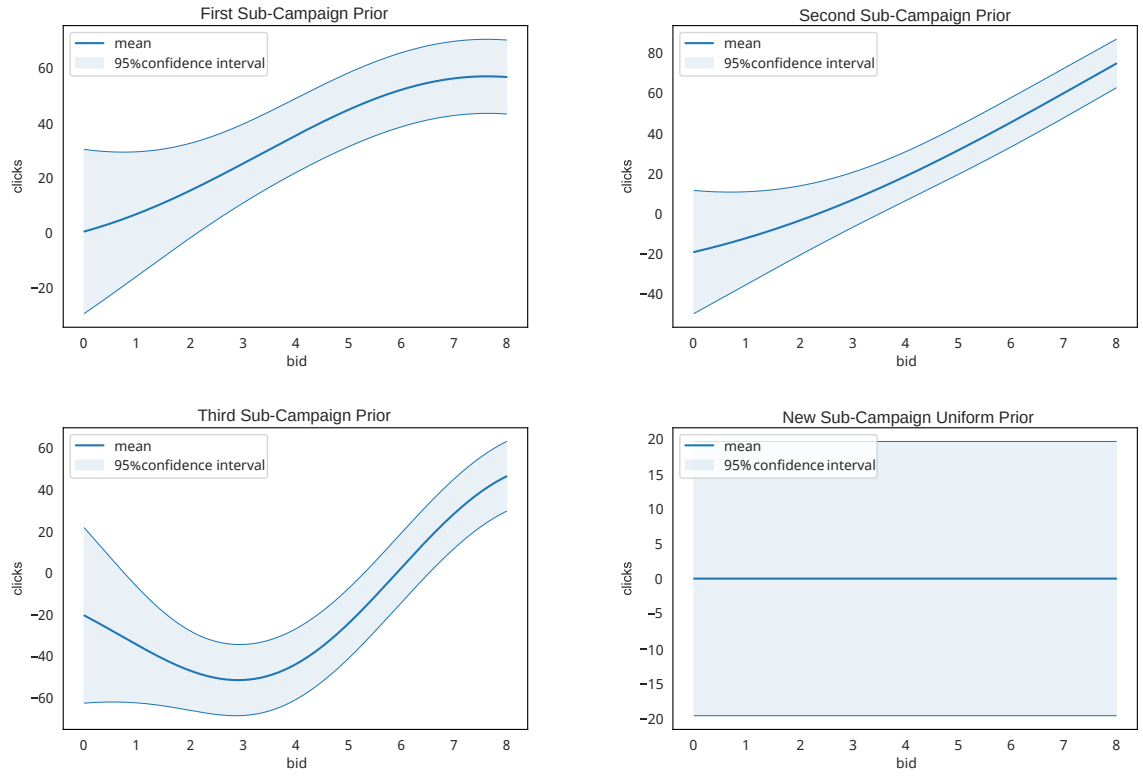


Figure 6.2: The four starting GPs used in the optimization process for the four sub-campaigns in case the zero-centred prior is used for the new sub-campaign to be opened.

In Figure 6.3, instead, we report the four starting GPs in the same scenario described above, but when a GP similar to the actual click function is used for the new sub-campaign to be opened.

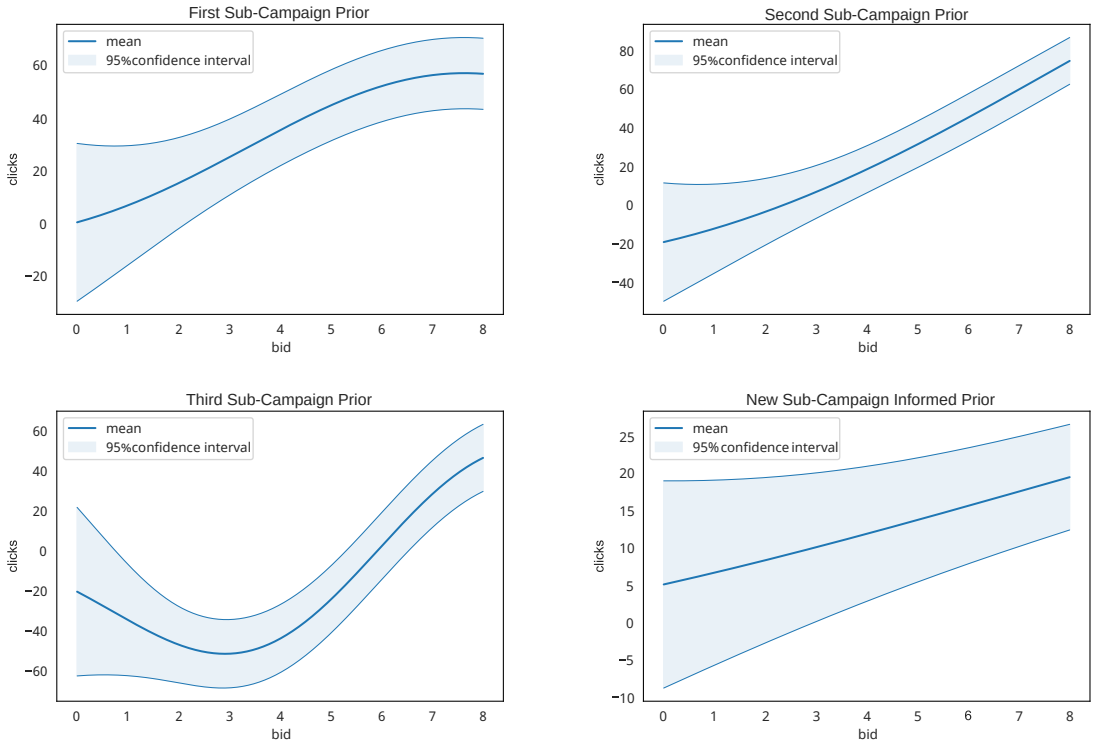


Figure 6.3: The four starting GPs used in the optimization process for the four sub-campaigns in case a GP similar to the real click function is used for the new sub-campaign to be opened.

At the end of the chosen time horizon T, we evaluate the performance of the two approaches by comparing the respective cumulative regret and cumulative rewards collected.

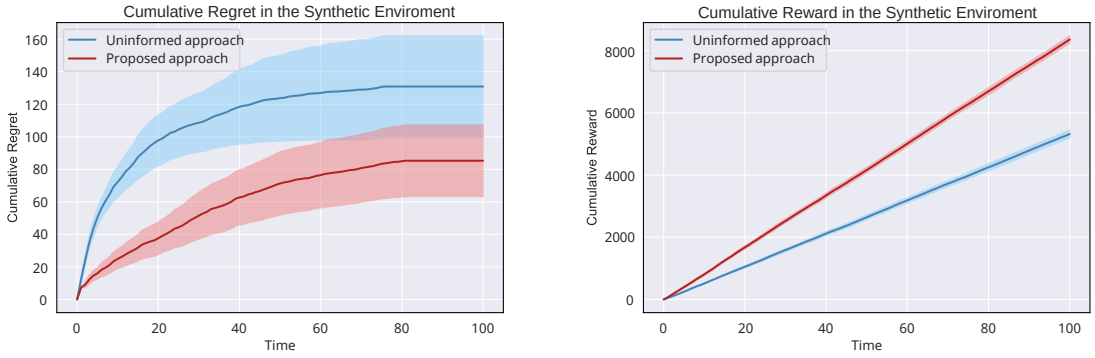


Figure 6.4: The average cumulative regret (left) and the average cumulative reward (right) for the synthetic campaign with a time horizon T=100 and a number of repetitions of the experiment equal to one hundred. The curves in blue correspond to the results for the uninformed approach, while those in red correspond to the informed approach.

In detail, given the stochasticity of the experiment, we repeat it several times for the same campaign and take into account as the final result the average values of regret and reward with their relative confidence interval. Figure 6.4 shows the results collected for the approaches defined in Figure 6.2 and Figure 6.3 respectively.
The results we have collected in the synthetic environment, presented in this section, indicate that the cumulative regret of the informed approach is always lower than that of the uninformed approach. Similarly, the cumulative reward of the informed approach is increasingly greater than the uninformed approach.
In light of this, we can conclude that using a starting GP that has additional information inside it results in better performance in the online optimization problem than using the same uninformed GP for all new sub-campaigns.

6.3. Real Data Setting

The next step is to test whether the heuristic proposed by us actually generates an initial GP that captures the average behaviour of the advertiser and that is similar to the true click function associated with that sub-campaign. In order to do this we exploited the data provided by AdsHotel.

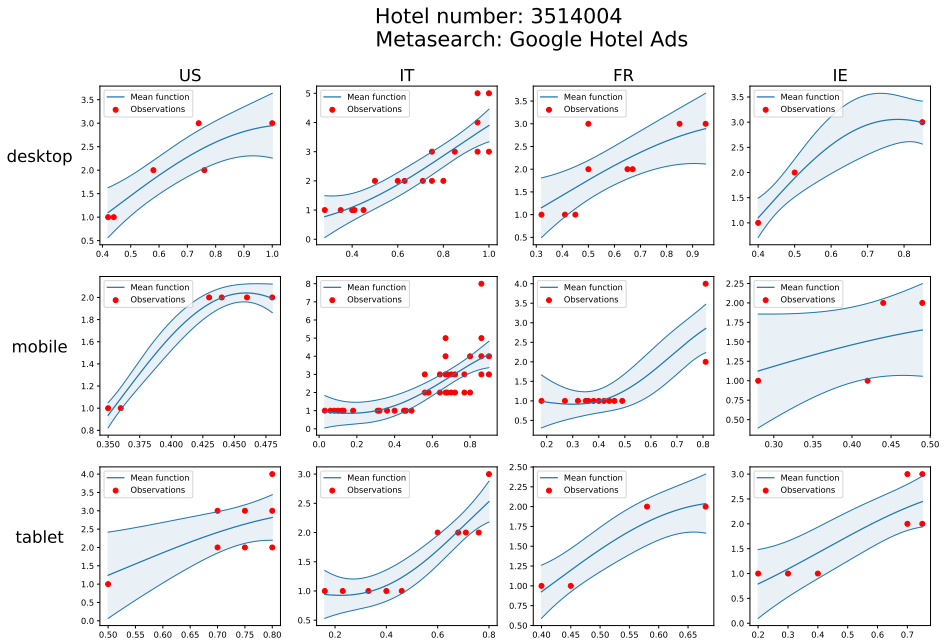


Figure 6.5: The estimated click functions for the sub-campaigns opened by hotel *3514004* for the Google Hotel Ads metasearch in the United States, Italy, France, and Ireland on desktop, mobile, and tablet devices.

In particular, for each hotel's campaign open on a given metasearch we consider the observations of its sub-campaigns and we fit a GP on them to estimate their respective click functions. Figure 6.5 shows the estimated GPs for the twelve sub-campaigns of hotel *3514004* on the metasearch Google Hotel Ads.
Then, we fix the user device and we take into account only the sub-campaign opened in different countries on that device. In other words, we focus on one row of the campaign's matrix representation reported in Figure 6.5.
We choose, among the selected sub-campaigns, one that we pretend to be the new one that the advertiser wants to open and for which we do not yet have any observation. Figure 6.6 shows the case in which the chosen device is desktop and the new sub-campaign to open is in Ireland (IE).

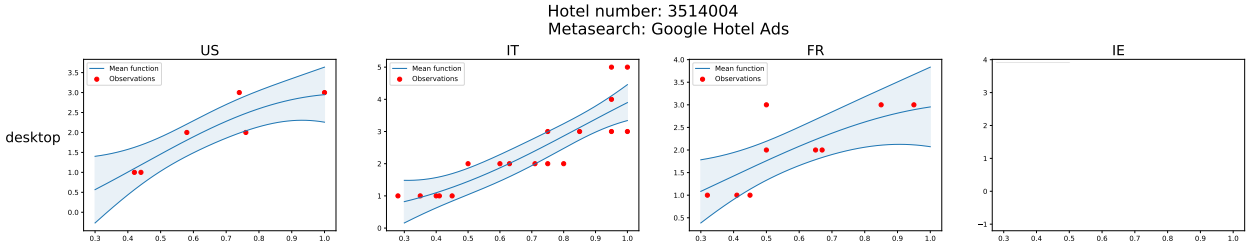


Figure 6.6: Advertising campaign characterized by three already opened sub-campaigns for which observations are available, respectively in the United States (US), Italy (IT) and France (FR), and a new sub-campaign, without observations, to open in Ireland (IE).

We associate as starting GP to this sub-campaign the one generated by our heuristics, reported in Algorithm 5.1, combining the GP of the other campaign's sub-campaigns on the selected device, as shown in Figure 6.7.

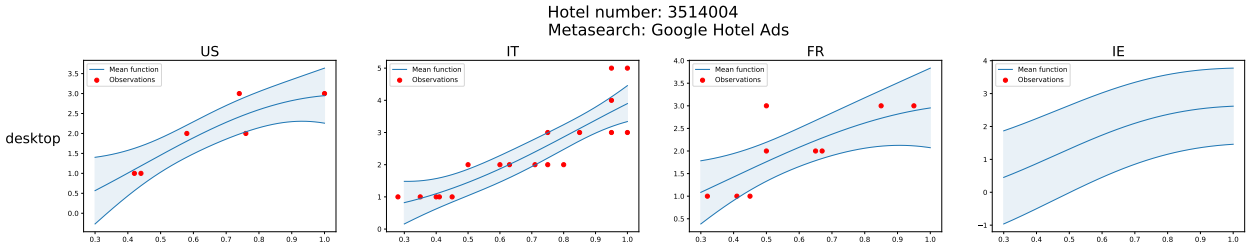


Figure 6.7: Initialization of the new sub-campaign in Ireland (IE) on desktop devices through a GP generated by the combination of GPs related to other sub-campaigns opened on the same device, respectively in the United States (US), Italy (IT) and France (FR).

Finally, we run the optimization of the campaign using as the initial GP for the new sub-campaign in one case the zero-centred prior and in another, the GP proposed by our heuristic.

As for the experiments in the synthetic scenario, also in this case we use the cumulative regret and the cumulative reward to evaluate and compare the performance of the two approaches. Figure 6.8 reports the cumulative reward and the cumulative regret for the campaign shown in Figure 6.7, in the not informed case (in blue) and in the informed case (in red).

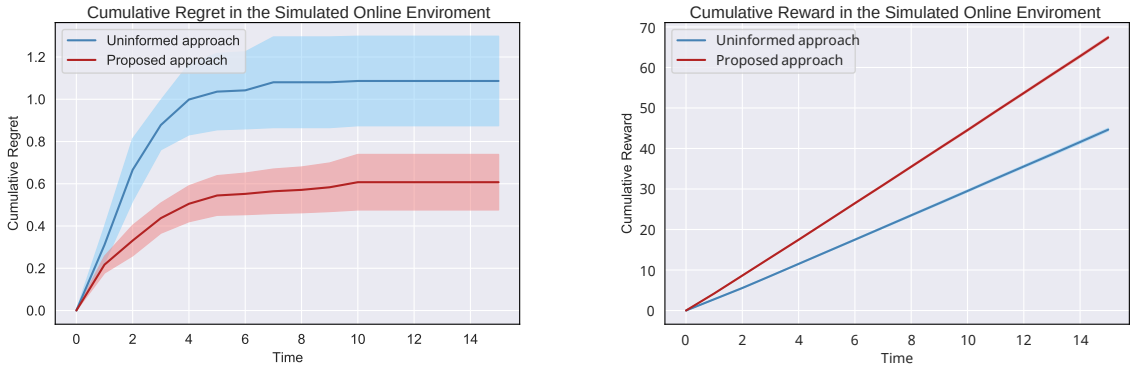


Figure 6.8: Comparison between the cumulative regrets (left) and the cumulative reward (right) when the zero-centred prior (in blue) and the GP generated by our heuristic (in red) are used as starting GPs for the new sub-campaigns.

The results collected on the advertising campaigns of the hotels contained in the AdsHotel database have shown a significant improvement in performance in case the GP, produced by our heuristic, is used as the starting GP for the new sub-campaigns. In particular, we can observe that the informed GP aids the optimization algorithm more during the exploration phase leading to a lower average cumulative regret and a higher average cumulative reward than the uninformed case.

7 | Conclusions and future developments

In this chapter, we draw some conclusions about our work, and we suggest some potential future projects that we believe would be a good starting point for improving the proposed approach.

7.1. Final Considerations

In this document, we presented and outlined one of the major issues affecting Machine Learning algorithms, known as the Cold-Start Problem. We specifically concentrated on the Online Advertising scenario from the advertiser's perspective with an emphasis on hotel advertising. In order to establish a specific initial informed GP to employ when a new sub-campaign is created, we, therefore, developed a methodology based on the combination of Gaussian Processes relevant to other related sub-campaigns. Finally, we validated our methodology using real data given by the marketing agency AdsHotel, which allowed us to simulate the online environment.

We compared the results by optimizing the same campaigns, first using a zero-centred prior GP for the new sub-campaigns for which we had no data, and then the GP generated by our heuristic. The experiments then demonstrated that using an informed GP with some hotel-specific behaviour as the initial GP for the new sub-campaigns is in all cases more effective and results in significantly lower regret than the uninformed case.

Our work has thus highlighted how the use of techniques capable of generating initial GP tailored for each hotel, as the heuristic we presented, can effectively lead media agencies, that use automated tools, to make fewer mistakes in the early stages of new sub-campaigns, limiting their customers' economic losses and increasing their future confidence.

7.2. Future Works

The first obvious extension of what has been presented is to test the proposed approach in a real-world online environment without the use of a simulator. This will allow more accurate data on the effectiveness of the developed methodology to be collected. However, it should be noted that finding an advertising company willing to test new techniques by making them available to its customers is not trivial, because potential economic losses experienced by the customers during the test could cause the latter to lose confidence in the use of automated tools as well as in the media agency itself.
Another extension of what was done, which could lead to further refinement of the initial informed GP, could be to estimate similarity metrics between different hotels using physical information such as geographical coordinates, the number of stars, and so on. When a hotel wants to open a new sub-campaign in a given country, we can exploit and combine the sub-campaigns' GPs in that country of the more similar hotels, weighed by the magnitude of the similarity. The premise behind this technique is that generally similar hotels will behave similarly in the same countries and this correlation will become even more intense as hotels have more analogous traits.

Bibliography

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[2] I. Batinić. The role and importance of internet marketing in modern hotel industry. *Journal of Process Management. New Technologies*, 3:34–38, 2015.

[3] M. Castiglioni, A. Nuara, G. Romano, G. Spadaro, F. Trovò, and N. Gatti. Safe online bid optimization with return-on-investment and budget constraints subject to uncertainty. *CoRR*, abs/2201.07139, 2022. URL `https://arxiv.org/abs/2201.07139`.

[4] J. E. K. Strong. *The Psychology of Selling and Advertising*. Number p. 349 and p. 9. MIT press Cambridge, MA, New York 1925.

[5] W. Hoeffding. Probability inequalities for sum of bounded random variables. 1961.

[6] E. Kaufmann, N. Korda, and R. Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In N. H. Bshouty, G. Stoltz, N. Vayatis, and T. Zeugmann, editors, *Algorithmic Learning Theory*, pages 199–213, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-34106-9.

[7] H. Kellerer, U. Pferschy, and D. Pisinger. *The Multiple-Choice Knapsack Problem*, pages 317–347. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-24777-7. doi: 10.1007/978-3-540-24777-7_11. URL `https://doi.org/10.1007/978-3-540-24777-7_11`.

[8] T. L. Lai, H. Robbins, et al. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[9] K. Mo, B. Liu, L. Xiao, Y. Li, and J. Jiang. Image feature learning for cold start problem in display advertising. IJCAI'15, page 3728–3734. AAAI Press, 2015. ISBN 9781577357384.

[10] A. Nuara, F. Trovo, N. Gatti, and M. Restelli. A combinatorial-bandit algorithm for

the online joint bid/budget optimization of pay-per-click advertising campaigns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[11] A. Nuara, F. Trovò, N. Gatti, and M. Restelli. Online joint bid/daily budget optimization of internet advertising campaigns. *Artificial Intelligence*, 305:103663, 2022. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2022.103663. URL https://www.sciencedirect.com/science/article/pii/S0004370222000030.

[12] P.Coopers. Internet advertising revenue report. 2022.

[13] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 1015–1022, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

[14] W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 12 1933. ISSN 0006-3444. doi: 10.1093/biomet/25.3-4.285. URL https://doi.org/10.1093/biomet/25.3-4.285.

[15] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[16] A. Ziesemer, L. Müller, and M. Silveira. Just rate it! gamification as part of recommendation. 06 2014. ISBN 978-3-319-07226-5. doi: 10.1007/978-3-319-07227-2_75.

List of Figures

List of Tables

List of Algorithms