SCUOLA DI DOTTORATO
UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA

Department of
Informatics, Systems and Communication

Ph.D. in Computer Science
XXXVI Cycle

Thompson sampling for Performance Marketing and delayed conversions

Surname: GIGLI Name: MARCO

Registration number: 869401

Tutor: Prof. Fabio Sartori

Supervisor: Prof. Fabio Stella

Coordinator: Prof. Leonardo Mariani

ACADEMIC YEAR 2022/2023

Contents

1 Introduction

Digital advertising expense in the US reached 189 billions USD in 2021, showing a staggering 35% year-over-year growth [1]. This was the highest level of growth seen since 2006 [2], and was to be partially imputed to COVID-19 restrictions and the consequent reliance on digital media: a deceleration in advertising revenues was thus to be expected. Moreover, the macroeconomic climate (high inflation rates, raising interest rates and economic uncertainty throughout 2022) impacted marketing budgets among others. Nevertheless, in 2022, far from decreasing, digital advertising expense reached a two-digit growth (10.8%) with respect to 2021, totalling 210 billions USD in the US [2]. These numbers show that digital marketing is an ever so important expense item for brands. Moreover, the current shift in focus from growth to profitability due to raising costs and interest rates means it is vital to efficiently and effectively manage digital marketing budget.

Digital marketing owes its appeal to several factors. First and foremost, given the current prevalence of online search and social media, digital marketing increases brand visibility. Moreover, when compared to traditional advertising channels like TV and billboards, it often offers better cost-efficiency, making it more accessible to smaller businesses. Most importantly for the scope of the present work, it offers measurable results: digital marketing provides extensive data and analytics tools that allow companies to track the performance of marketing campaigns in real-time. In particular, it is most often *performance marketing*: as the name suggests, the advertiser is charged on the basis of measurable *metrics*, such as the *number of views*, *clicks*, *customer contacts*, or *sales*.

The challenge of optimising digital marketing campaigns has thus attracted the interest of the Machine Learning community (see for instance [3–5]). It is possible to reliably measure the impact of decisions, hence closing the data loop between the *action* a learning agent performs (e.g., targeting a certain segment of the audience, or choosing a certain creative content for adverts) and the *reward*, i.e.,

an increment in the metrics mentioned above. Moreover, tabular data consisting of many dimensions are appealing for learning algorithms, as opposed to human intuition. Finally, and crucially for the present work, the optimisation usually takes place while the marketing campaigns are running. In other words, the advertiser is asked to solve a *sequential decision making problem* by facing the *exploration-exploitation dilemma*: the algorithm in charge of optimising marketing expense (the *agent*) must continuously decide between dedicating resources to gathering more data on how different choices perform (exploration) and sticking to the choice that has proven the best so far (exploitation). Collecting more data allows making sharper decisions, but it comes at a cost.

This sequential decision making problem can be fruitfully formalised in the framework of *stochastic Multi-armed Bandits* (MABs) [6–8]. The stochastic MAB problem can be seen as a special case of Reinforcement Learning, in which the actions of the agent do not modify the environment in which it acts [9, 10]. Put differently, the simplifying assumption that the actions that are available to the agent, and the consequent rewards, are not affected by past decisions, has proven very effective both in general [11] and in digital marketing in particular [12].

However, during the course of my PhD I identified several aspects that limit the application of MABs to the optimisation of performance marketing campaigns. This dissertation is devoted to three aspects in particular:

1. The extension of previous results to a widespread format for digital marketing campaigns; this format endows marketing campaigns with a hierarchical structure, not covered by state-of-the-art methods.

2. The switch from a general purpose regression method to a newly developed model, crafted specifically for digital marketing campaigns; this new model makes learning from acquired data significantly more efficient with respect to the state of the art.

3. The study of the negative impact of delayed feedback (an intrinsic facet of digital marketing) on the learning process and the development of a new technique to counter this effect; this technique achieves significantly better performances with respect to the state of the art in extensive simulations.

The next section presents an outline of these results and of how they address needs previously unfulfilled by the specialised literature. Contributions 1 and 2, being closely connected, are presented together in subsection 1.1.1, while subsection 1.1.2 is dedicated to contribution 3. In section 1.2, we will see an outline of the structure of this dissertation.

1.1 Contributions

1.1.1 Hierarchical campaigns and data efficiency

Major digital advertising platforms share the same basic strategy for choosing which adverts get shown to internet users: every time a user is eligible for seeing

an ad, compatible advertisers compete in an automated auction. For every ad, the advertiser is thus called to choose wisely a *bid* for the auction and a maximum *daily budget* (i.e., the maximum total expense one plans to sustain for that ad in a given day).

Search Engine Marketing (SEM) keywords and ads that share a common theme are usually gathered by practitioners into *ad groups* [13], which are in turn grouped in *campaigns*, that can be used to specify which demographic group to target. The daily budget is a parameter of the whole campaign, i.e., during the course of the day it gets eroded as the users interact with its constituting ad groups. We assume a total daily budget is imposed externally as a constraint for a group of several campaigns. The bid is instead set at the ad group level. The task is then deciding how to split the total budget across campaigns, and which bid to choose for each ad group.

We will see in chapter 3 that it is quite natural to see the bids and the budgets as the possible *actions* available to a learning agent in a Multi-armed Bandit formulation. Indeed, a recent series of papers (see [12] and references therein) has formulated the daily choice of bids and budgets of SEM campaigns as a Multi-armed Bandit problem. In particular, we will see in section 2.2 and in greater detail in chapter 3 that, to effectively face the sequential decision making problem, the agent builds a *regression model* of how the expected reward functionally depends on the combination of all bids and budgets.

These state-of-the-art results, however, suffer several limitations that hinder practical application. First, they concentrate on campaigns without substructure, while, as seen above, campaigns are routinely divided by practitioners in ad groups: the target audience of the ads and the budget are set at campaign level, while the ad content, the keywords and consequently the bids are set at ad group level. Moreover, Gaussian Processes are employed in [12] to build the regression model that maps bids and budgets into expected reward:

- They are very expressive, but for this reason they are also data intensive.
- They need approximations or proxies to accommodate censored data (which occur naturally when the daily budget limits the number of clicks in a day).
- They are non-monotonic, hence they require global optimisation methods when selecting the best bid/budget combination.

Given the above, under this setting my first contribution was to extend the framework to the multi-ad group domain [14]. This is non-trivial, because budgets and bids are set at different levels (respectively campaigns and ad groups): this creates an effective interaction among ad groups, since raising the bid for one erodes the click count for the others belonging to the same campaign. Moreover, all clicks are not created equal: clicks pertaining to different ad groups typically carry different monetary value to the advertiser. The solution I developed solves the problem switching from modelling the mapping of a single bid to the number of clicks, to modelling a function that maps a vector of bids (one element for every ad group) to the total *value* of a whole campaign (which takes into account

the expected monetary value of clicks coming from different ad groups). A corresponding *ansatz* to simplify the regression problem had to be generalised too.

The second aspect I tackled is data efficiency. In lieu of Gaussian Processes, I devised a parametric Bayesian regression model, informed by domain knowledge. Few interpretable parameters reduce the variance, and thus the need for data, allowing for faster convergence. Moreover, the model naturally accounts for censoring. Since the dependence of the number clicks on the bid is modelled as monotonically increasing, this model allows for local optimisation, which is both more reliable and faster with respect to global optimisation. While the bid choice can be carried out with off-the-shelf methods, I developed a local constrained optimisation algorithm for the budget splitting.

In order to test the model, I designed and developed a simulation environment, using what is disclosed about Google automated auctions. Simulations agree with experience, and also with the parametric model. The results show that the proposed approach brings a clear improvement in revenue, especially for short time frames: this means that the parametric model converges much more quickly than the Gaussian Process one.

Real-world data have also been used to validate above results [15]. In particular, I used the Criteo Attribution Modeling for Bidding Dataset [16]. The dataset contains more than 16 million user interactions, but these have a very low *conversion rate* (less than 3‰), i.e., they very rarely lead to further signs of user engagement besides the click: advertisers are usually interested in maximising these long-term metrics, such as add-to-cart or purchase events. This low rate and the fact that the experiment only lasts 30 days (as opposed to 100 days as in previous simulations) make data efficiency even more important.

In general, to evaluate the performance of competing bandit policies on logged data one would need a full counterfactual analysis, and this dataset is no exception, as it contains only winning auctions, i.e. those for which the competing bids were low enough: in other words, it contains *truncated data*, and the bid distributions are biased. In line with relevant literature, I opted for *noise injection* to tackle the problem of selection bias.

Real data granted me the possibility to compare my parametric regression model and the theoretical curve: they adhere very closely, while Gaussian Processes struggle to converge. This reflects directly in the performance of the optimisation algorithm. I conducted a host of experiments varying one external parameter at a time (i.e. total budget, number of campaigns and number of ad groups per campaign): in every setting, on average the Gaussian Process method totals less reward with respect to the method herein proposed, and the average relative difference can reach in some settings a staggering 40%, especially when an optimisation algorithm is needed the most (i.e., with tight budget or many ad groups).

In the same setting, with an eye to applications and in view of further contributions, Bootstrapped Thompson Sampling [17], an easy to implement approximated Bayesian inference method, has been adapted to the contextual bandit object of study and tested.

1.1.2 Delayed feedback

A defining aspect of Search Engine Marketing is that the reward is often delayed for many days, in particular when optimising for contacts or sales (hereafter generically called *conversions*): prospect customers need time to evaluate the purchase. Both [12] and the contributions introduced in the previous subsection, however, are placed in the idealised setting of no delay.

To overcome this issue, the first challenge has been finding the appropriate way to model the problem of delayed reward in digital marketing. To this end, I performed an extensive review of the literature about bandits with delayed feedback. I focused in particular on the manifold of settings in which bandits with delayed feedback have been applied, and the consequent modelling choices. Differences regard for instance the probability distribution of delays (i.e., whether it is sub-Gaussian or heavy-tailed, whether it depends on the chosen action or not, and so on), whether the agent has access to intermediate signals that can act as a proxy for the reward, or whether the feedback is anonymous in some way.

This has permitted me to identify the simplest non-trivial setting which is relevant for Search Engine Marketing, and to pinpoint [18] as the state-of-the-art approach in this setting. I thus decided to study this setting for two groups of reasons, one which is general and one which is specific to the bid/budget selection problem described above.

General In this setting, the state-of-the-art approach [18] presents room for improvement:

- It ignores how delays are distributed: when available, data on the entity of delays can be valuable in modelling the environment.
- It employs a specific heuristic (called LinUCB, see section 2.2) to face the exploration-exploitation dilemma. Another heuristic, Thompson sampling, has proven more robust to delays and to model misspecification in a variety of other settings.

Specific Perfecting the approach to this simplified setting can pave the way to more complex scenarios:

- In this setting, avoiding some of the intricacies of the bid/budget selection problem (such as non-linearity, the hierarchical structure, and resource constraints), simulations are much more transparent, and can shed light on the specific effect of delays on the total reward.
- Having a clear state-of-the-art competitor, it is possible to test a new technique with the goal of generalising it to more complex settings.
- The LinUCB algorithm mentioned above requires calculations that are specific to the linear case (i.e., the case in which the function that maps actions to expected rewards is linear). Ideally, one wants a technique which can be extended to more complex settings with minor effort.

For these reasons, I have worked on extending the aforementioned Thompson sampling technique to linear bandits for delayed conversions. This setting, however, presents several challenges:

Complex reward structure In usual MAB models, the agent observes both positive and negative rewards. In the present setting, on the other hand, the agent cannot distinguish the case in which a prospect customer decides not to proceed with a purchase from the case in which the customer will proceed, but the sale is yet to happen. In other words, the reward is *partially observable*.

Long-tailed delay distributions Depending on the business area, the distribution of delays can present heavy tails. Hence, parametric distribution families could perform poorly, hindering straightforward application of Bayesian methods such as Thompson sampling.

An approximate version of Thompson sampling has been recently applied to partially observable delayed rewards in the case of finite and static sets of actions (i.e., not the linear case) [19]. The approach there proposed, however, presents some shortcomings, besides the need to be generalised to linear bandits:

- While the model is derived for a general distribution of delays, it is then specialised to exponential distributions. As specified above, the delay distribution can be heavy-tailed, hence the need for a non-parametric estimator.
- As the name suggests, when adopting Thompson sampling, the agent uses *samples* from a distribution to guide its next move. This distribution captures its uncertainty about the expected outcome of its possible actions. In [19] only the uncertainty over the link between action and reward is taken into account, while the uncertainty on the model of delays is ignored; this could hamper exploration.

Besides the aforementioned review of the literature concerning bandits with delayed feedback, my contributions in this setting are hence the following:

- I extended the model to include a standard Maximum Likelihood non-parametric estimator for delays, namely the Kaplan-Meier estimator. Since it is widespread, it has the advantage of being implemented in many Machine Learning libraries that deal with time data, such as [20].
- To take into account both the uncertainty on rewards and on delays, I applied the model to Bootstrapped Thompson Sampling. This has the advantage of requiring minimal assumptions and being way faster than standard approaches to Bayesian inference, such as Markov chain Monte Carlo.
- I applied the resulting model to linear bandits.
- I tested the proposed approach against the state of the art on a host of delay distributions. The proposed approach performed significantly better in the great majority of cases, behaving comparably to the state of the art in the remaining ones.

1.2 Thesis outline

This dissertation is organised as follows. In chapter 2, a refresher on stochastic Multi-armed Bandits is given. With some motivating examples, the exploration-exploitation dilemma is presented, and the general notion of *regret* as a measure of performance is introduced. In section 2.1, bandits with a finite number of possible actions are defined and lead to the introduction of some heuristics that recur in all the present work, namely Upper Confidence Bound (as representing *optimism in the face of uncertainty*) and Thompson sampling. In section 2.2, the important setting in which actions are described by features, which can help learning, is introduced. In section 2.3, it is shown how representing actions as vectors of features actually permits treating the case of an infinite number of possible actions.

Chapter 3 is devoted, after an introduction to the bid/budget selection problem in online marketing campaigns (section 3.1), to presenting the contributions outlined in subsection 1.1.1 above, namely:

- The extension of a state-of-the-art bandit to admit online marketing campaigns with hierarchical structure (section 3.2).
- A new parametric model for the dependency of number of clicks and cost on the bid, which eases the task of learning from past data and permits extrapolation on unseen data (3.3).
- How this parametric model can be used in a bandit setting, also in the context of an approximate, lightweight technique which had to be generalised to this setting (subsections 3.3.2 and 3.3.3).
- The details of the purpose-built simulation environment to test the proposed methods, and simulation results (section 3.4).
- The results of tests on real data, including an analysis of the impact of environment parameters on performance (section 3.5).

Chapter 4 is devoted to analysing the impact of delayed feedback on performance. Section 4.1 contains the aforementioned review of the bandit literature on delayed rewards. Section 4.2 restricts the scope to the problem at hand, i.e., that of delayed conversions, and positions the present contributions within the related literature, highlighting differences and similarities. In section 4.3 the supervised learning model for partially observable, delayed rewards is derived in detail. In section 4.4 the contributions outlined in subsection 1.1.2 are analysed, namely: the use of a non-parametric estimator, how the model can be used within approximated Thompson sampling and in particular in the context of linear bandits. Moreover, the positive results of tests on a manifold of delay distributions are given and commented upon.

Finally, chapter 5 is devoted to drawing the conclusions on the present work. In section 5.1, the contributions are summarised, emphasising the novelty and increased performance with respect to the state of the art. Section 5.2 is devoted

to discussing the limiting aspects of the proposed approaches, and suggesting some ways to circumvent some of them. Section 5.3 presents the promising future avenues of research that emerged from the present work.

2

Bandits and the exploration-exploitation problem

In this chapter, I am going to present the basic ideas behind Multi-armed Bandits (MABs), the workhorse that powers the rest of the work. Bandit algorithms constitute a highly effective framework for decision-making in the presence of uncertainty over extended time periods. Let us start from a stylised version of the problem we are going to tackle in chapter 3 and highlight the characteristics that make bandits the natural choice.

As we will see, digital advertisers must select, daily, a host of parameters for their marketing campaigns. Since the possibility to appear on a web page is regulated by automated auctions, among other parameters the advertiser must choose a *bid*. If the bid is too low, the advertisement will not appear on the page, and the opportunity to gain a click is lost. On the other hand, if the bid is too high, the user may click on the advertisement, but the cost paid by the advertiser will be high as well. In this scenario, we can frame the advertiser's goal in this way: maximising the total number of clicks (or contacts, or sales) given a certain budget.

Two traits of this challenge need to be stressed:

- When choosing a bid, the advertiser observes the *reward* (i.e., if the user clicks) only for that bid: they don't know what *would have happened* had they chosen another bid.

- Since every user and every auction is different, for a given bid the reward is afflicted by noise: selecting the same bid twice does not necessarily lead to the same reward.

We are now ready to define Multi-armed Bandits. In doing so, we will mainly follow the treatment of [8] and occasionally of [10] and [11]. We will privilege the

concepts we will need in this work, consciously trying to convey general concepts and intuition, rather than focusing on detailed mathematical proofs.

2.1 Multi-armed Bandits

In the most basic form of the MAB problem, an algorithm (also called the *agent*) plays a game over T rounds (T is also called the *horizon*): in each round $t = 1, \dots, T$, it must choose among K possible actions, for some finite K. Depending on its choice, it collects a reward: such reward is drawn from a probability distribution which depends only on the chosen arm, and in particular does not depend on time. Problem is, the agent does not know the K distributions corresponding to the possible actions. Moreover, since the agent receives only the reward corresponding to the played action, it receives no information on the distributions corresponding to unplayed actions. The goal of this game is maximising the total reward over the T rounds.

The term *Multi-armed Bandit* refers to a (rather old) colloquial name for slot machines, *one-armed bandits*, because of the lever used to operate them (and the fact that they typically steal money from gamblers). The name came into being to describe a decision-making problem where a gambler is faced with a row of slot machines with different payout probabilities. Due to this historical origin, the K actions are also called *arms*: in the following, the two terms are used interchangeably.

Given the definition above, we readily see how the bid selection problem maps into a bandit problem:

- If the possible bids are discrete and finite in number, they represent the set of allowed actions.
- The reward is 1 if the advertisement is clicked and 0 otherwise.

Note that our simplified auction example already points to possible generalisations of the bandit definition given above:

- The bid is more naturally modelled as a continuous quantity. Even if there is a smallest scale (e.g., one Euro cent), the environment will respond to consecutive bids in a similar way; this similarity is lost if every bid is represented as an isolated action.
- Some other information, besides the reward corresponding to the chosen bid, is revealed: if there is a click, the advertiser knows that, for that particular user, any higher bid would have resulted in a click (and vice versa for the no click situation and lower bids). In other words, the algorithm observes reward for some other arms, but not for all.
- A cost is associated with every action, namely, the amount paid to the advertising platform if the user clicks. The goal is maximising the total reward *subject to a given budget*.

We will touch upon these generalisations in subsequent sections and chapters. Nonetheless, we will see that the concepts here introduced set the foundations for all subsequent developments. Among these concepts, we see here the need of balancing *exploration* and *exploitation*. The agent needs to explore, both trying out unplayed arms and replaying past arms to fight the effect of noise. On the other hand, there is a tradeoff with leveraging the information acquired, to make best-guess decisions on which one is the optimal arm.

In what follows, we will focus on so-called *stochastic bandits*, characterised by independent and identically distributed (IID) rewards: in each round, the reward is drawn independently from the distribution corresponding to the played arm. At the opposite end of the spectrum lie *adversarial bandits*: in this setting, it is assumed that rewards are chosen by an adversary right before the agent makes its move, with the goal of *minimising* the agent's cumulative reward. Since advertising competitors are usually many and not coordinated, and since we can assume that the user deciding whether to click is not colluded with competitors, IID rewards seem the most natural way to frame the problem.

It must also be remarked that, in the basic stochastic bandit setting, arm distributions do not have an explicit dependence on time. On the other hand, we can expect many natural settings, and in particular the online advertising scenario, to show some degree of time dependence. We will touch upon time dependence in section 2.2 and chapter 3.

For concreteness, let us focus in this section on the most basic form of bandit: *Bernoulli bandits*. They owe their name to the fact that the reward distribution for each arm is a Bernoulli distribution, i.e., the reward can be either 1 or 0 (click or no click) and each arm a is characterised by its expected value $\mu(a)$. Different problem instances are then characterised by different expected value vectors $\mu(1), \dots, \mu(K)$.

Let us denote the best arm as a^* and the corresponding best mean reward as μ^*. An oracle agent, which has access to the full vector of means, would always play a^* up to the horizon T: it can then expect a total reward of $\mu^* T$. On the other hand, a real agent that has to infer the arm distributions shall play a sequence a_t of arms for every round $t = 1, \dots, T$, totalling an expected reward $\sum_1^T \mu(a_t)$. Due to this dependence on the expected reward vector $\mu(1), \dots, \mu(K)$, some problem instances allow greater rewards than others. To have a metric of performance that is somewhat instance-independent, it is customary to define the *realised regret* (or *pseudo-regret*):

$$R(T) = \mu^* T - \sum_{t=1}^{T} \mu(a_t). \tag{2.1}$$

Since the reward for each turn is a random realisation coming from the corresponding distribution, an agent that adapts its strategy to observed rewards will result in a random sequence of played arms; moreover, we will see in section 2.1.3 that some bandit algorithms do not act deterministically given a sequence of realised rewards. For these reasons, one is usually interested in taking the expected value $\mathbb{E}[R(T)]$ of (2.1), called *regret*. However, the realised regret is oftentimes simply called regret, if the meaning is clear from the context.

We also need to remark that calculating the regret requires knowledge about the best arm, and this is possible only with simulated data, where we have control over the data generating process. In chapter 3, in order to compare two different algorithms without knowing the best action, we will calculate the regret suffered by one algorithm *for not behaving like the other*. Moreover, since even the regret can depend on the size of rewards, we will calculate the *relative* regret.

There is a clear overlap between MABs and supervised learning, in that the algorithm strives to learn which arms are best, and an intuitive way to do so is estimating the reward distribution of the arms, refining the estimates with the passing of rounds; this overlap will become even more apparent in sections 2.2 and 2.3, and this idea will be at the core of my contributions in chapters 3 and 4. Two aspects, however, make the bandit problem stand out. One is the already mentioned exploration-exploitation dilemma, which stems from the fact that the agent itself guides data collection, differently from the standard supervised learning problem, where the dataset is a given. The other differentiating aspect is the notion of regret itself, which endows the problem with a clear notion of performance with respect to standard supervised learning accuracy metrics. In supervised learning there are many possible way to measure model quality and identify the "best" possible model, and these metrics are often somehow detached from the final application of the model. On the other hand, in MABs the agent is focused only on totaling the highest cumulative reward at the horizon T, even if this comes at the expense of some notion of predictive accuracy.

After having defined precisely MABs and regret, we are in the position of going through the most important strategies to tackle the bandit problem. As it is for the problem itself, we will see that the strategies defined in this basic setting carry over to more general environments.

2.1.1 Epsilon-greedy strategy

The simplest heuristic we will define is the so-called epsilon-greedy algorithm. In this strategy, the balance between exploration and exploitation is explicitly encoded in a time dependent probability ε_t: for every turn $t = 1, \dots T$, the agent either explores with probability ε_t choosing an arm uniformly at random, or exploits (hence being *greedy*) with probability $(1 - \varepsilon_t)$, choosing the arm that has yielded the highest empirical average up to round t.

While the probability ε_t could in principle be (and sometimes is) kept constant, intuitively we want this probability to decrease over time, as the agent collects more data and becomes more and more certain about its estimates. Besides time, ε_t can depend also on the parameters of the problem that are known to the agent: the number of arms K and the horizon T. For the Bernoulli bandit one can prove that, if $\varepsilon_t = t^{-1/3} (K \log t)^{1/3}$, an upper bound on regret is minimised, and this bound is

$$\mathbb{E}[R(t)] \leq t^{2/3} O\left((K \log t)^{1/3}\right).$$

2.1.2 Optimism in the face of uncertainty: the UCB algorithm

In the previous section on the epsilon-greedy algorithm, the time-dependence of ε_t is known to the agent from the start, before having observed any reward. Intuitively, however, some problem instances will be "easier" than others, namely those where the difference between the mean of the best arm and the mean of suboptimal arms is great. In those cases, the agent will reach high *confidence* about which arm is best faster than in instancese where arms are very similar among each other. Moreover, in those easier instances "wasting" rounds exploring is also more costly in terms of regret, since choosing suboptimal arms the agent incurs in greater losses. On the other hand, in harder instances, where arm distributions are more similar among each other, it is less wasteful to play a suboptimal arm. For these reasons *adaptive* strategies, which evolve based on observed rewards, usually reach better performances.

Let us then focus on measuring confidence, which was mentioned above without being properly defined. Informally, we can define the *confidence radius* as a measure of how far the empirical average of an arm's rewards could be from the true expectation, with high probability. More formally, given an expected value μ, the empirical average $\hat{\mu}$ and a radius r, Hoeffding's inequality holds:

$$P\left(|\hat{\mu} - \mu| \geq r\right) \leq 2e^{-2nr^2},$$

where n is the number of samples used to calculate $\hat{\mu}$. As the radius r or the number of observations n increase, this bound decreases fast.

In particular, if we make the bound constant by defining $c = e^{2nr^2}$, it is easy to see that the inequality becomes:

$$P\left(|\hat{\mu} - \mu| \geq \sqrt{\frac{\log c}{2n}}\right) \leq \frac{2}{c}. \tag{2.2}$$

Keeping the bound constant and increasing the number of observations, the radius reduces.

With this notion, we are ready to introduce the Upper Confidence Bound (UCB) algorithm: for each round t, the agent acts *optimistically*, assuming that the true expectation of each arm is located at the upper border of the corresponding confidence interval, based on the observations collected up to t. In other words, in each round t, the agent chooses $\operatorname{argmax}_a \mathrm{UCB}_t(a)$, where $\mathrm{UCB}_t(a) = \hat{\mu}_t(a) + r_t(a)$.

Note that Hoeffding's inequality as it is written in (2.2) gives no prescription on how to choose the confidence radius $r_t(a)$: it just regulates the relationship between a given radius and deviation probabilities. As it was for ε_t in the previous section, also in this case we want to choose carefully the dependence of the confidence radius $r_t(a)$ on K, T, t and the rewards gathered up to t, in order to minimise regret. Before doing that, it is fruitful to gain an intuition of why the UCB algorithm makes sense.

The agent can choose an arm a at round t for one of two reasons (or both): because the empirical average $\hat{\mu}_t(a)$ is large, or because the confidence radius $r_t(a)$

is wide. If $\hat{\mu}_t(a)$ is large, it means the agent mostly got positive rewards playing this arm in past rounds, and we can expect it to behave similarly in future rounds. From inequality (2.2), if c does not depend on n or t (or, more generally, if this dependence is $o(n)$), the radius shrinks with n. Thus a radius $r_t(a)$ is wide if the arm a has not been explored extensively. Hence, the two terms in $UCB_t(a)$ transparently represent exploitation and exploration respectively.

Given that the radius in (2.2) shrinks with n, for each problem realisation, there is a round $\tilde{t}$ beyond which the algorithm always chooses the most promising arm, i.e., the one with highest empirical average. If we are interested in the asymptotic performance for large T, we want to be reasonably sure that, when $\tilde{t}$ comes, the best arm has been correctly identified as the most promising. In particular, we want the probability of locking in on the wrong arm to be small enough that the average regret coming from this case is negligible for high T. This suggests the following form for the inequality:

$$P\left(|\hat{\mu} - \mu| \geq \sqrt{\alpha \frac{\log T}{n}}\right) \leq \frac{2}{T^{2\alpha}}. \tag{2.3}$$

The constant α must then be carefully chosen to minimise regret.

We are now ready to state without proving the following: UCB algorithm with $\alpha = 2$, i.e., with confidence radius

$$r_t(a) = \sqrt{\frac{2 \log T}{n_t(a)}}$$

satisfies the regret bound

$$\mathbb{E}\left[R(T)\right] \leq O\left(\sqrt{KT \log T}\right).$$

2.1.3 Thompson sampling

We will now introduce the heuristic that is used in the rest of this work owing to its generality, conceptual simplicity and good practical performance: Thompson sampling. While rooted in Bayesian probability, as we will see it proves effective also in the frequentist setting we are considering.

The agent is endowed with a *prior distribution* $\mathbb{P}$ over problem instances $\mathcal{I}$. As the agent plays arms, a *history* of action-reward pairs is collected. Round by round, as this new information streams in, the agent updates its posterior distribution, narrowing the volume in the space of possible problem instances where that specific instance could live. The missing ingredient is how to turn this posterior belief in the action for the next round. This strategy can be characterised via two equivalent formulations

1. For each round t and arm a, the algorithm computes the posterior probability that a is the best arm, and samples arms according to these probabilities.

2. For each round t the algorithm samples an instance $\mathcal{I}$ from its posterior, and chooses the arm that would be best in that instance.

For concreteness, we will now specialise this algorithm on the Bernoulli bandit we have considered so far. In this setting, it is most often assumed that the priors are independent, i.e., according to the prior $\mathbb{P}$ the mean values $\mu(a)$ for each arm a are independent variables. It can be proven that, with this assumption, each arm can be updated separately: in other words, the posterior factorises across arms.

Thompson sampling in the second characterisation for the Bernoulli bandit can thus be formulated as follows. For each round, the agent samples mean rewards $\mu_s(a)$ for each arm a from the respective posterior distributions; it then chooses the best arm $\tilde{a}$ according to these sample means, $\tilde{a} = \operatorname{argmax}_a \mu_s(a)$. The posterior corresponding to $\tilde{a}$ gets then updated according to the received reward.

A very common prior, in the Bernoulli case, is the uniform prior over the interval $[0, 1]$. With this prior, the posterior at round t is a Beta distribution $\text{Beta}(\alpha_t, \beta_t)$: if arm a has been played n times and the total reward gathered from a is m, then $\alpha_t(a) = 1 + m$ and $\beta_t(a) = 1 + n - m$. More generally, since the Beta distribution and the Bernoulli distribution form a conjugate pair, one can assume a prior $\text{Beta}(\alpha_0, \beta_0)$ and the posterior at round t is just $\text{Beta}(\alpha_0 + m, \beta_0 + m - n)$. This is consistent with the uniform prior, as $\text{Beta}(1, 1)$ is the uniform distribution.

Note that, in the stochastic bandit setting we are considering, the problem instance $\mathcal{I}$ is considered fixed, and not governed by a prior distribution $\mathbb{P}$. In this sense, the prior can be seen just as a parameter of the model. Of course, the more the fixed instance $\mathcal{I}$ is "typical" with respect to the prior, the faster it is for the agent's posterior to concentrate around $\mathcal{I}$. We can now state the bound for the expected regret for Thompson Sampling in the frequentist setting:

$$\mathbb{E}\left[R(T)\right] \leq O\left(\sqrt{KT \log T} + K\right).$$

2.2 Contextual bandits

After having introduced the most basic MAB problem in section 2.1, we can now cover a very important extension, *contextual bandits*.

We will start from the following motivating example [21]. The learning agent represents a news website, which has the goal of optimising the experience for each visiting user. For simplicity, let us assume that the agent must choose only which headline to show at the top of the page. Of course, different users will have differing interests: rather than discovering the most interesting headline for its user base as a whole, the action must be *personalised*. To this end, the website observes several features regarding the user, such as location, demographics, device and so on: this information is called the *context*. As a proxy for user satisfaction, the end goal of the procedure is maximising the number of clicks on the displayed headline.

Not only users, but also the news pool from which to choose is changing constantly: in other words, the action set evolves with time. Rather than cold starting our algorithm every day with a new set of actions, it is thus convenient to represent news too via some features, and include these features in the context.

In order to make sense of this problem, we will have to place some regularity assumptions on the dependence of click probability on features. This is sensible: for instance, we can expect users with similar demographics and interests to react similarly to the same news story.

We can now distill the defining characteristics of the example above and define contextual bandits in general. As in the previous section, the algorithm must choose among K actions in T rounds. However, for each round t, the agent observes a context $x_t(a) \in \mathbb{R}^d$ for every arm $a = 1, \dots, K$, prior to choosing the action. After an arm is chosen, the agent collects a reward r_t. Rewards are drawn from a distribution which depends on the context $x_t(a)$. In particular, we will denote the expected reward given a context as $\mu\left(x_t(a)\right)$. Note that the contexts may change every round, but the functional dependence of rewards on context is the same for all rounds: this dependence is what the agent is interested in learning.

The most basic and common class of contextual bandits, and the one which we will cover in this section and in chapter 4, is that of *linear contextual bandits*. As the name suggests, in this setting there is a linear link between contexts and expected rewards: $\mu\left(x_t(a)\right) = \vartheta \cdot x_t(a)$, for some fixed vector ϑ, which is unknown to the algorithm.

To see how the example above can fit in this model, let us assume that incoming users are represented by feature vectors u and headline a is represented by vector $y(a)$ (dropping dependence on t for simplicity). If we also assume the expected reward to be linear both in u and $y(a)$, we can write: $\mu(u, y(a)) = \sum_{ij} u_i \Theta_{ij} y_j(a)$ for some matrix Θ. Let us then define an index $I = (i, j)$, the vector $x(a)$ with elements $x_I(a) = (u_i, y_j(a))$ and the vector ϑ with entries $\vartheta_I = \Theta_{ij}$. Our relation then becomes, with a slight abuse of notation, $\mu(x(a)) = \sum_I \vartheta_I x_I(a) = \vartheta \cdot x(a)$. In other words, if the environment at t (the user) and the arms (the headlines) are represented separately by feature vectors, the context for the linear contextual bandit is the outer product of these vectors.

In the definition above, we can see precisely in which sense contextual bandits generalise vanilla MABs. If we suppose that the set of contexts is kept constant, and corresponds to the canonical basis vectors $(e_i, 1, i = 1, \dots, d)$ of $\mathbb{R}^d$, we simply recover the $d-$armed bandit: every arm has its corresponding expected reward, ϑ_a. This simple remark will come handy when generalising the UCB algorithm to linear contextual bandits.

We can now define the regret for linear contextual bandits in a similar fashion as the one for vanilla MABs: if the agent were to know ϑ, the obvious optimal strategy would be picking, for each round t, the action $a_t^* = \underset{a}{\operatorname{argmax}}\, \vartheta \cdot x_t(a)$, accumulating reward $x_t^* = \vartheta \cdot a_t^*$ in expectation. The regret in this setting is then defined as:

$$R(T) = \sum_{t=1}^{T} \vartheta \cdot x_t^* - \sum_{t=1}^{T} \vartheta \cdot x_t(a_t)$$

We will not show how the ideas behind the algorithms designed for vanilla MABs can be fruitfully generalised to this linear setting.

2.2.1 LinUCB

The first algorithm we cover is Linear Upper Confidence Bound (LinUCB): it shows that the principles introduced in section 2.1, far from being limited to that settings, are actually quite general. Here we refer in particular to *optimism in the face of uncertainty*, i.e., acting *as if* unknown quantities are as favourable to the agent as they can plausibly be.

For linear contextual bandits, instead of constructing confidence bounds on the mean rewards of the single arms, we need to do something similar for the theta vector: we must devise a *confidence region* $\mathcal{C}_t$ such that, with high probability, $\vartheta \in \mathcal{C}_t$. For each incoming context $x_t(a)$, the agent then selects the ϑ in $\mathcal{C}_t$ such that its expected reward is maximal. Finally, it plays the most promising arm.

First, we note that the confidence region for the expected rewards of each arm in vanilla MABs is centered around the empirical average. To guide us in the linear setting, we recall that the empirical average is the estimator (without covariates) of a quantity that minimises the mean square error:

$$\hat{\mu}_t = \frac{1}{t}\sum_{s=1}^{t} r_s = \operatorname*{argmin}_r \sum_{s=1}^{t} (r - r_s)^2 ,$$

where r_s is the s-th reward. For the center of the confidence region, we do something similar, adopting the regularised least-squares estimator:

$$\hat{\vartheta}_t = \operatorname*{argmin}_{\vartheta \in \mathbb{R}^d} \left(\sum_{s=1}^{t} (r_s - \vartheta \cdot x_s)^2 + \lambda \|\vartheta\|_2^2, \right)$$

where x_s is the context corresponding to the s-th played arm.

At odds with the empirical average, here we *regularise* the estimation for two reasons. One is technical: choosing $\lambda > 0$ ensures that the loss function has a unique minimum even when the arm vectors do not span $\mathbb{R}^d$. The other is practical: we want to reduce noise. Since we are not placing bounds on the size of the context d, having a big number of features could mean that the above estimator would be overfitting (i.e., would be bad at generalising on unseen samples) for a great number of rounds. In essence, we are dealing some bias in exchange for a reduction in variance of our estimator. We will see, when extending Thompson sampling to the linear setting, that the regulariser λ has a transparent meaning in terms of a Bayesian prior.

It is a basic fact of supervised learning that the solution to the regularised least-squares problem is:

$$\hat{\vartheta}_t = V_t^{-1} \sum_{s=1}^{t} r_s x_s, \tag{2.4}$$

where

$$V_t = \lambda \mathbb{1} + \sum_{s=1}^{t} x_s x_s^\top . \tag{2.5}$$

Having selected the center of the confidence region $\mathcal{C}_t$, the next, harder, question is: how do we define "plausible" values of ϑ? In other words, how *far* from the

real ϑ could $\hat{\vartheta}_t$ be with high probability? Since we noted that d-armed bandits can be seen as special linear contextual bandits in $\mathbb{R}^d$, we resort to the confidence region (2.3). Rearranging terms, the confidence region of each arm is the set of values of ϑ_i such that $n_i|\hat{\vartheta}_i - \vartheta_i|^2 \leq \gamma_T$, where $\gamma_T = \alpha \log T$. If we consider all d conditions at once, they imply:

$$\sum_{i=1}^{d} n_i|\hat{\vartheta}_i - \vartheta_i|^2 \leq d\,\gamma_T. \tag{2.6}$$

Here we are just saying that $\hat{\vartheta}_i$ needs to be closer to ϑ_i for arms where n_i is large, i.e. for which the agent gathered more observations. In the case we are considering, it is easy to see that V_t is diagonal and (if $\lambda = 0$) has diagonal entries equal to n_i. This means we can rephrase condition (2.6) in a base independent way writing

$$\left(\vartheta - \hat{\vartheta}_t\right)^\top V_t \left(\vartheta - \hat{\vartheta}_t\right) \leq d\,\gamma_T,$$

or, more concisely, $\|\vartheta - \hat{\vartheta}_t\|^2_{V_t} \leq d\,\gamma_T$.

We then define the confidence region

$$\mathcal{C}_t = \left\{\vartheta \in \mathbb{R}^d \text{ s.t. } \|\vartheta - \hat{\vartheta}_t\|^2_{V_t} \leq \beta_T\right\} \tag{2.7}$$

For a general, linear contextual bandit, this confidence region is an ellipsoid, whose principal axes are the eigenvectors of V_t, and the lengths of these axes are the inverse of the eigenvalues. In (2.7), the bound is left more general than $d\,\gamma_T$: as it is for vanilla MABs, the bound must be carefully chosen to minimise regret. As it turns out, however, our intuitive analysis is correct for the leading term in T:

$$\beta_T \sim 2\log T \quad \text{for } T \to +\infty.$$

2.2.2 LinTS

We introduce here LinTS, the generalisation of Thompson sampling to linear contextual bandits. In its second formulation at page 14, Thompson sampling is already apt to be applied to linear contextual bandits. Since problem instances are characterised by the vector ϑ, the agent will be endowed with a prior belief over ϑ. As it pulls arms and collects data, its posterior gets updated. For every round, a problem instance is drawn from this posterior, and the best arm is selected accordingly.

We thus need to place a prior on ϑ. As it was for Bernoulli bandits, also in this case we can use a conjugate prior (assuming Gaussian noise). We assume that the reward distribution given context $x(a)$ for arm a is

$$r \sim \mathcal{N}\left(x(a)\cdot\vartheta, \sigma^2\right)$$

for some variance σ^2. We can then assume a multivariate Gaussian prior over ϑ:

$$\vartheta \sim \mathcal{N}(0, \Sigma), \qquad \Sigma = \sigma_0^2\,\mathbb{1}.$$

It is then well known that the posterior is (dropping arm reference for simplicity):

$$\mathbb{P}(\vartheta|x_1, r_1, \ldots, x_t, r_t) = \mathcal{N}\left(V_t^{-1}\sum_{s=1}^{t} r_s x_s,\, \sigma^2 V_t^{-1}\right).$$

This expression illuminates the formulas (2.4) and (2.5) of the LinUCB algorithm: $\hat{\vartheta}_t$ can be seen as the maximum a posteriori (MAP) of an agent with a Gaussian prior for a problem with Gaussian rewards; the parameter λ in (2.5) is the ratio $\frac{\sigma^2}{\sigma_0^2}$. Moreover, given the expression for the covariance of the posterior, the confidence region given above is just the ellipsoid in which the posterior probability density exceeds a certain threshold.

2.3 Static contextual bandits

As we have seen in section 2.2, contextual bandits have a close connection with supervised learning, as contexts are encoded in features and the relationship between these features and rewards is learnt from data, on the basis of some regularity assumption. Once the agent is confident enough about the model, it can reliably pick the next arm, even when the number K of arms is huge: this choice may be computationally challenging, but assuming the agent is able to efficiently optimise the reward function over the space of features the problem is essentially solved.

On the other hand, in the standard MAB problem, for large number of arms K, achieving low regret is a hopeless endeavour: while similar contexts (and, by extension, similar arms) in a contextual bandit problem share information, in a standard MAB each arm is isolated from the others.

For these reasons, the contextual bandit problem is useful even when contexts do not change from one round to the next, if one is able to associate a fixed feature vector to each arm, and posit that the reward function is regular in some sense. This setting is called for obvious reasons *static contextual bandit*.

When dealing with a static contextual bandit, the agent reduces the problem of learning K independent distributions to learning a function belonging to some family of "well behaved" functions. Hence, we can also consider the case in which K approaches infinity, or even the case of continuously many arms: *continuum-armed bandits*. As we have seen commenting the motivating example in section 2.1, continuum-armed bandits are quite appropriate for the auction problem of online marketing: in that case, arms correspond to bids, and similar bids are expected to yield similar rewards.

One typical assumption for reward functions is linearity, also in the case of static, continuum-armed bandits: the learning problem (and thus either the confidence region of LinUCB or the posterior of LinTS) is exactly the same as in the finite-armed case. What changes is just the set of actions, which can be for instance a closed subset of $\mathbb{R}^d$.

Besides linear functions, another important class of continuum-armed bandit problems is that of *Gaussian process bandits*: given some kernel, it is assumed that

the reward function is sampled from a Gaussian Process (GP). Much of the interest for GP bandits stems from the problem of optimising an unknown, noisy function that is expensive to evaluate: given a maximum number of evaluations T, the task is approximating the true maximum as close as possible.

There are several ways to evaluate an optimisation algorithm designed for such problem; the most natural one is perhaps the *minimum simple regret*, i.e. calculating how far the value of the best action considered so far is from the true maximum: $\min_{t=1,\dots,T} [f(x^*) - f(x_t)]$, where f is the function to be optimised. However, the cumulative regret is often used: by minimising the cumulative regret instead of the minimum simple regret, we punish exploration in regions that are unlikely to contain the optimum, thus ensuring progress towards optimality uniformly over t. Moreover, the *average* cumulative regret (obtained simply dividing the cumulative regret by T) is an upper bound on the minimum simple regret. Finally, focusing on the cumulative regret, the problem becomes a bandit one, and one can re-use results and techniques from the bandit literature.

Among these techniques, GP bandits, too, admit variants of the UCB and Thompson sampling strategies. In particular, since GPs are a Bayesian model, GP-Thompson sampling (GP-TS) is perhaps the most natural: the posterior over functions is updated as more data points are gathered, and then a function is sampled from this posterior; the next action corresponds to the maximum of the sampled function over the action space.

The GP-UCB decision rule [22] again follows the optimism in the face of uncertainty principle: the next point to query x_t in the domain D is given by

$$x_t = \operatorname*{argmax}_{x \in D} \left[\mu_{t-1}(x) + \alpha_t^{1/2} \sigma_{t-1}(x) \right] \tag{2.8}$$

where $\mu_{t-1}(x)$ and $\sigma_{t-1}(x)$ are the mean and standard deviation respectively of the posterior evaluated at x, given the data gathered up to round $t-1$. Here α_t is a time-varying parameter, that must be tuned (akin to what is done in UCB and LinUCB algorithms) to trade off exploitation and exploration and attain minimum regret.

We will see an application of GP-bandits to online advertising auctions in chapter 3, and how its state-of-the-art results can be surpassed by specifying a family of functions which is less rich than GPs, but models the problem well. Indeed, as noted in [23], to achieve low regret, we must choose a regression model which is both simple enough to be learned fast, and expressive enough to capture the behavior of the function mapping actions to rewards: for specific problems as the one tackled in chapter 3, it is possible to leverage domain knowledge to this end.

3

Multi-armed Bandits for Performance Marketing

This chapter is devoted to the contributions introduced in section 1.1.1 on the topic of optimising bids and budgets of a set of digital advertising campaigns. First, in section 3.1, we will cover the problem in detail and see how bandits have been successfully used to tackle it. We will also point to the limitations of state-of-the-art approaches. In section 3.2, the optimisation problem is formalised mathematically, and we will see how this formulation can be effectively generalised to encompass real world marketing campaigns: as mentioned in the Introduction and explained below, real world campaigns present a hierarchical structure, which must be properly taken into account to solve the optimisation problem. In section 3.3 we will go through my proposed parametric bandit approach for learning more efficiently from the data the agent accumulates during the course of the optimisation. This method is supported, as we will see in sections 3.4 and 3.5, by extensive numerical experiments, performed on synthetic and real world data: these experiments show that, on average, the proposed parametric bandit gains more conversions than the state-of-the-art approach. Moreover, gains in performance are particularly high when an optimisation algorithm is needed the most, i.e., with tight budget or many ad groups.

3.1 Bandits and the bid/budget selection problem

In 2022, only three digital advertising formats accounted for the overwhelming majority (93%) of the total digital advertising spend [2]:

Search ads (40.2%), i.e., the ads that appear in the search engine results page together with so-called organic results.

Display ads (30.3%), mostly made of *banner ads*, i.e., images prominently displayed on websites, that serve as clickable links to the advertiser's website.

Video ads (22.5%) in the format of short videos that are displayed on streaming platforms together with the actual content.

In all three cases, the most common payment model is *performance dependent*: this means that, in order for the advertiser to pay the advertising platform, the user that is exposed to the ad must have some measurable interaction with it. Such interaction is often *clicking* on it, but this notion can also encompass other format-specific forms: for instance, for video ads, a watching time that exceeds a given threshold could trigger the payment. In the following we will generally refer to clicks as the most common interaction mechanism, but any other form of immediate interaction is included in the treatment.

The other most common charging principle is, in a sense, performance related too. It is distinguished from performance marketing proper because it serves a different objective: rather than pushing users to interact with a website and hopefully buying a product, it aims at getting a message to a target audience, in so called *brand awareness* campaigns [24]. For this second objective, the most common charging principle is CPM (*cost-per-mille*, i.e., payments are calculated on the number of times an ad is seen, expressed in thousands), as opposed to CPC (cost-per-click) and CPA (cost-per-acquisition, i.e., payments are based on the number of contacts or sales).

A second characteristic trait of digital marketing is the way in which advertisers are chosen, among many others, to be shown on a web page. Typically, every time a user is eligible for being shown an ad, compatible advertisers take part in an automated *auction*. Compatibility is mainly decided through associating ads with *targets*, such as demographic groups or keywords. Besides choosing the target, the advertiser must thus decide on the *bid* for taking part in auctions and a maximum *daily budget* (i.e., the maximum total expense one wants to sustain for that ad in a day).

While a complete description of the different auction types is beyond the scope of this work, they mainly belong to three kinds: *Generalised Second Price*, *VCG* [25] and *First Price* [26]. For definiteness, in this work we focus on Generalised Second Price auctions with CPC payments, typical of Search Engine Marketing. In this setting, the bid represents the maximum cost the advertiser is willing to pay if the user clicks on the ad. However, as will become apparent, the methods can be easily adapted to other kinds of auction and charging principles. In fact, we will see that the auctions are treated as a black-box mechanism, which maps bids into clicks and costs in a probabilistic way.

Search engine marketing keywords and the accompanying ads that share the same theme are usually gathered together by practitioners into so called *ad groups*. These are in turn further grouped, to form *campaigns*. Campaigns are characterised by their user targeting (based on location, age, and so on). The daily budget is set as a parameter on whole campaigns. The bid is, instead, a parameter of ad groups: it is expected that the competition will be higher on some sets of keywords and

lower on others, so the bids must be adjusted accordingly. What follows is set under the assumption that a *total daily budget* is imposed as an external constraint: this total budget must be split among several campaigns. We will call *portfolio* the collection of campaigns over which this total budget is set.[1]

Two recent works [12, 27] have cast the daily bid/budget optimisation as a Multi-armed Bandit problem. In particular, the goal is to maximise the long term revenue, choosing daily the combination of bids and budgets for the whole portfolio. In this formulation, the whole combination of bids and budgets is the *arm* that the learning agent plays, in the terminology introduced in chapter 2.

The authors of [27] note that, when the agents plays one such combination, it does not observe just the total number of clicks and conversions gained by the portfolio over the day: it observes also the single numbers of clicks and conversions totalled by the each ad group. In other words, even if the space of possible actions is combinatorial in nature, a richer feedback with respect to the typical bandit feedback we have seen in chapter 2 makes the problem manageable, as the agent can learn how to associate the bids and budget of a single campaign to the expected number of clicks. Since the feedback is richer than in the pure bandit setting, it is called *semi-bandit feedback*, and the corresponding bandit problem is called *combinatorial bandit* [8, 28, 29]. As the whole combination of bids a budgets can be seen as a combination of arms of single, simpler bandits (the campaigns and the ad groups), it is sometimes called a *super arm*.

Campaigns are in turn seen as *static contextual bandits* in the language introduced in section 2.3: similar bid/budget combinations for a campaign share information. The fixed context (feature) vector is indeed given by the bids/budget pair.

The application of these results in practical settings is, however, limited by the following shortcomings:

1. The literature concentrates on campaigns without substructure (single ad group campaigns). This means that, besides the budget, only one bid must be chosen per campaign. However, the typical campaign is divided in ad groups, thus requiring generalisation. Given the regression model introduced in [27] this generalisation is non-trivial.

2. Agnostic nature of Gaussian Processes (GPs) with respect to the functional form which links bid to observed clicks. While this regression method is very flexible, it also reflects in the need for more data to converge to a sensible posterior, if compared to a more informed model (in particular, a parametric model).

3. The need to extrapolate, from the observed clicks, the so-called *saturation clicks*, i.e. the number of clicks the agent would have observed if there were

[1] We specify that the term "portfolio" is less established in the industry with respect to terms such as "ad group" and "campaign": we use it here just to refer to the collection of campaigns involved in the optimisation. If these make up the totality of an advertiser's campaigns, one could use the term *account* interchangeably.

no budget constraints. The authors of [27] suggest using the time of the day at which the daily budget run out and an estimate of how searches are distributed during the day to perform the extrapolation (see also [12] where the example of a constant distribution is given). However, this information on time is not available, and ad showing frequency is explicitly reduced throughout the day to make sure the budget lasts until the end of the day [13]. Other metrics could be used to this end, but they are inherently noisy. Perhaps more importantly, this extrapolation method needs a certain share of the total budget to be *always* reserved for exploration (akin to the ε-greedy strategy we saw in section 2.1). The need for this missing data imputation stems from the use of vanilla GPs (i.e. with a Gaussian likelihood [30]). On the other hand, *censored regression* is a principled way to avoid the need for missing data imputation. While GPs can accommodate non-Gaussian likelihoods, this requires giving up exact update formulas, and switching to approximate methods [31].

Reasons 2 and 3 point to parametric regression, exploiting a functional form suggested by domain knowledge, as a way to use data more efficiently and without resorting to proxies. In this spirit, I have overcome the aforementioned shortcomings making the following contributions:

- To address point 1 above, I developed a multi-ad group generalisation of the relation between bid/budget and clicks (section 3.2), suitable regardless of the regression model one employs.
- To tackle points 2 and 3 above, I devised an informed alternative to GP regression, namely a parametric regression model, which accounts for censoring in a principled way and with interpretable parameters (section 3.3.1).
- I explored the use of such a model in the context of bandits (and specifically Thompson sampling) (section 3.3). In particular, bid and budget selection are recast as local constrained optimisation problems, as opposed to global optimisation required by GPs: this brings advantages both in terms of resource requirements and accuracy of the found optimum.
- To test and compare performances of the proposed approach, I developed a simulation environment, built on what is known about the inner workings of the auctions (section 3.4.1). Numerical results reported in section 3.4.2 confirm the advantages of the proposed method.
- The performance of the proposed approach has also been extensively tested on real world data by exploiting the Criteo Attribution Modeling for Bidding Dataset [16] (described in section 3.5.1).
- With an eye to applications, Bootstrapped Thompson Sampling [17, 29], an easy to implement approximated Bayesian inference method, has been adapted to our contextual bandit and tested (see section 3.3.3).

- The impact of various parameters on model's performance has been studied on the Criteo dataset. Furthermore, an ablation experiment, to study the effect of the model alone on single-ad group campaigns, has been performed (section 3.5.2).

The rest of the chapter is organised as follows. In section 3.2 we will set the notation and define the multi-ad group optimisation problem. In section 3.3 we will see how the optimisation can be carried out using a parametric Bayesian regression model. Sections 3.4 and 3.5 are devoted to testing the proposed technique in an artificial simulated environment and on real-world data respectively.

3.2 Optimisation problem for multi-ad group campaigns

To generalise the single-ad group model to multiple ad groups per campaign. let us first establish the notation in the single-ad group regime, and proceed with the generalisation below. Following [27] assume for now that we have a portfolio of N campaigns, each with just one ad group. Let us call $n_j(b_j, B_j)$ the average number of clicks obtained by the j-th campaign with budget B_j and bid b_j. Let v_j be the average value of one click from the j-th campaign. The task of maximising the revenue can then be formulated as the following constrained optimisation problem:

$$\max \sum_{j=1}^{N} v_j n_j(b_j, B_j) \qquad \text{s.t.} \sum_{j=1}^{N} B_j \leq B, \quad b_j \in [\underline{b}_j, \overline{b}_j] \, \forall j \tag{3.1}$$

where the maximum is taken over the budgets $B_1, \ldots, B_N$ and the bids $b_1, \ldots, b_N$, B is the total budget and $\underline{b}_j$ and $\overline{b}_j$ are the (possibly campaign-dependent) minimum and maximum allowed bids.

Both the functions n_j and the values v_j are unknown, and must be estimated from the collected data, hence the need to balance exploration and exploitation. Early estimates could be inaccurate and lead to sub-optimal decisions, but one does not want to spend too many resources on data gathering either, since this comes at the expense of exploiting acquired knowledge.

Note that, in accordance with [27], we are here assuming *stationarity*, i.e. that the probability distributions of click value and number of clicks given a bid and budget don't change with time. This is a realistic approximation only on short time spans: as detailed in section 3.4.2, this motivates the need for fast-learning models, as the one presented in section 3.3.1.

In order to reduce the burden of exploration, in [27] an ansatz for the form of n_j is proposed, reducing the complexity of a two-variable regression to two one-variable functions:

$$n_j(b_j, B_j) \approx n_j^{\text{sat}}(b_j) \, \min\left(1, \frac{B_j}{c_j^{\text{sat}}(b_j)}\right). \tag{3.2}$$

Here the function n_j^{sat} denotes the *saturation clicks*, i.e. the number of clicks a campaign would obtain if there were no budget limits. Likewise, c_j^{sat} denotes

the *saturation cost*, i.e. the cost faced in the same situation. Since the right hand side depends non-linearly on n^{sat} and c^{sat}, and given that we are speaking about averages, the equality in 3.2 strictly holds only in the deterministic case.

The issue in generalising the problem (3.1) to the multi-ad group setting is that the budget is shared by all the ad groups of the same campaign, as stated in section 3.1. While we can let an index k run over ad groups, and define v_{jk} as the value of a click from the ad group k of campaign j and do similarly for the bid b_{jk}, we cannot define a corresponding click function $n_{jk}(b_{jk}, B_j)$: the number of clicks gathered by an ad group depends also on the bids of all the other ad groups belonging to the same campaign. Intuitively, raising the bid b_{jk} will bring more clicks for the corresponding ad group, but it will also erode the budget B_j more quickly, thus lowering the clicks received by the other ad groups. This difficulty can be circumvented introducing the *total value* function $V_j(\boldsymbol{b}_j, B_j)$ of a campaign, which depends on the whole vector of bids $\boldsymbol{b}_j$. Therefore, the optimisation problem (3.1) generalises to

$$\max \sum_{j=1}^{N} V_j(\boldsymbol{b}_j, B_j) \quad \text{s.t.} \sum_{j=1}^{N} B_j \leq B, \quad b_{jk} \in [\underline{b}_{jk}, \bar{b}_{jk}] \, \forall j, k. \tag{3.3}$$

In order to preserve data efficiency, the ansatz (3.2) must be generalised too, linking the total value function to the corresponding saturation quantities. Note that the aforementioned interdependence among different ad groups is a consequence of a limited budget, while the dependence of saturation quantities n_{jk}^{sat} and c_{jk}^{sat} on the single bid b_{jk} is well defined. If the j-th campaign contains m_j ad groups, and we let

$$\begin{aligned} V_j^{\text{sat}}(\boldsymbol{b}_j) &= \sum_{k=1}^{m_j} v_{jk} n_{jk}^{\text{sat}}(b_{jk}), \\ c_j^{\text{sat}}(\boldsymbol{b}_j) &= \sum_{k=1}^{m_j} c_{jk}^{\text{sat}}(b_{jk}), \end{aligned} \tag{3.4}$$

then ansatz (3.2) generalises to

$$V_j(\boldsymbol{b}_j, B_j) \approx V_j^{\text{sat}}(\boldsymbol{b}_j) \, \min\left(1, \frac{B_j}{c_j^{\text{sat}}(\boldsymbol{b}_j)}\right). \tag{3.5}$$

Since the right hand side is not a sum over single-ad group contributions, this formula captures the interaction among different ad groups.

As a way to intuitively justify (3.5) for fixed bids and budget, we can think of the single n_{jk}^{sat} as the sizes of "reservoirs", one for each ad group, from which clicks are randomly drawn, up until the moment when the total cost paid matches the assigned budget. If the clicks pertaining to different ad groups are well mixed, each will bring approximately the same fraction of its *saturation value* $V_{jk}^{\text{sat}}(b_{jk}) = v_{jk} n_{jk}^{\text{sat}}(b_{jk})$ and saturation cost. The value of this fraction is found equating the total cost paid and the assigned budget B_j, hence (3.5).

3.3 Optimisation strategy with parametric bandits

In the previous section, we established how the optimisation problem is formulated, generalising it to the multi-ad group domain, which contains single-ad group as a special case. In this section, we explore an efficient way to perform the optimisation itself.

As seen in the Introduction and in section 3.2, we face here the exploration-exploitation dilemma, since the function we want to optimise must be learnt from the data. Among the strategies to solve this dilemma in the context of Multi-armed bandits, Thompson sampling [7] is of particular interest to practitioners, due both to its performance [32], generality and conceptual simplicity [29].

In general, as seen in section 2.1.3, Thompson sampling involves two steps:

1. Making optimal use of the data gathered thus far with Bayesian inference, establishing a posterior distribution on the space of parameters (section 3.3.1);

2. Sampling from the posterior distribution, and selecting the best arm acting *as if* the sample represented the reality (section 3.3.2).

We will examine these steps separately in the upcoming subsections, while in section 3.3.3 we will see how Bayesian inference can be simplified with an approximated technique.

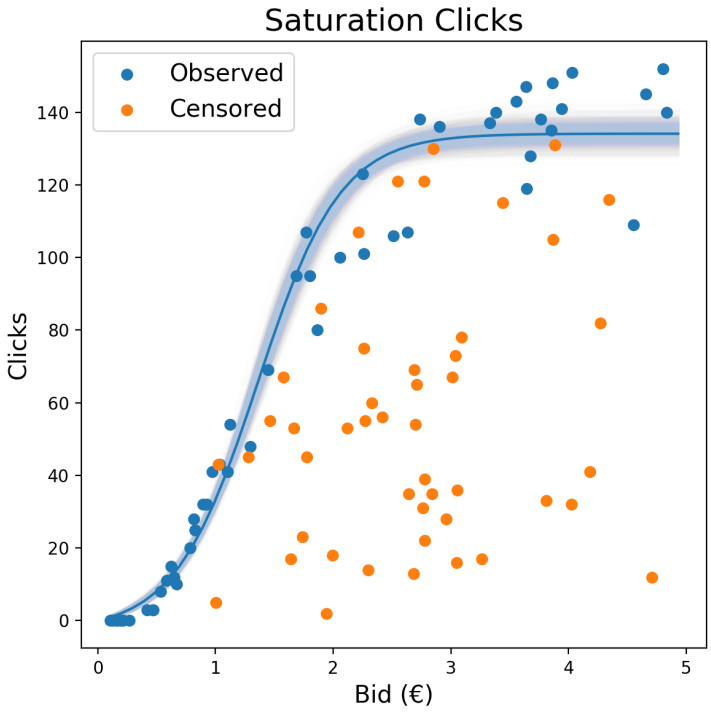
3.3.1 Parametric regression model

If we see the j-th campaign as a static contextual bandit (as defined in section 2.3), Thompson sampling requires performing Bayesian regression on the correspondence between $(\boldsymbol{b}_j, B_j)$ and the reward r_j. We can restrict the search space by placing few, sensible hypotheses on the shape of the functions n^{sat} and c^{sat}, introduced in equations (3.5) and (3.4) (we are here dropping indices for simplicity). These hypotheses will lead us naturally to a parametric model: Bayesian regression can then be conducted with Markov chain Monte Carlo (MCMC) [33].

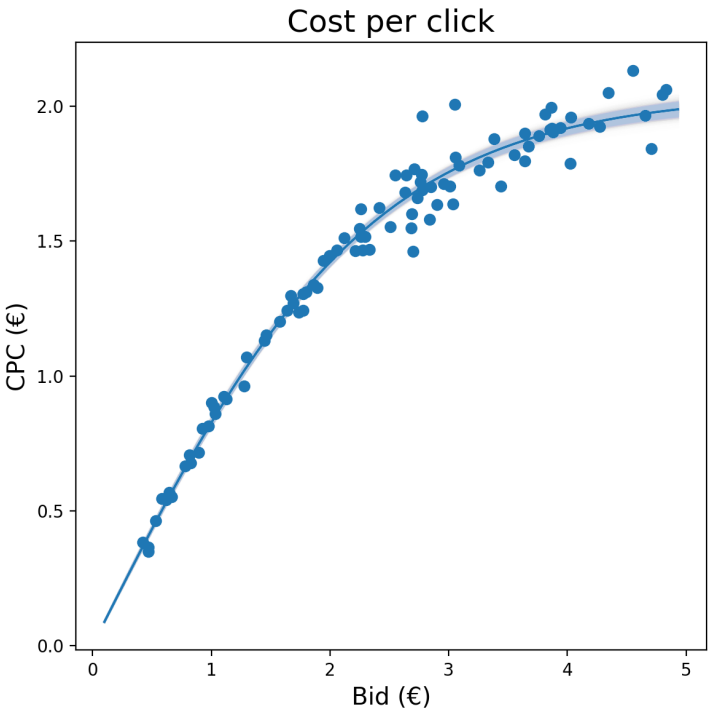
Clicks and cost paid are of course highly correlated, so to be able to perform separately the two regressions it is convenient to introduce the cost-per-click (CPC) function $\varphi(b) = \frac{c^{\text{sat}}(b)}{n^{\text{sat}}(b)}$. Both functions n^{sat} and φ must be positive, be monotonic increasing with the bid, saturate for high enough bid and vanish for vanishing bid. Moreover, φ was empirically found to be linear for small bids (in accordance with the law of diminishing returns), and must be strictly smaller than the identity (because of the meaning of bid as maximum CPC). These considerations suggest to use a properly shifted and scaled logistic function. Starting from the saturation clicks,

$$n^{\text{sat}}(b) = \underbrace{k(1+e^{-ac})}_{\text{scale factor}} \left(\underbrace{\frac{1}{1+e^{-a(b-c)}}}_{\text{logistic function}} - \underbrace{\frac{1}{1+e^{ac}}}_{\text{vert. shift}} \right). \tag{3.6}$$

An example of this function is shown in figure 3.1a.



(a) Saturation clicks regression



(b) CPC regression

Figure 3.1: Bayesian regression for models (3.6) and (3.7). Data have been simulated as in section 3.4.1. Orange dots denote *censored* quantities: the number of saturation clicks is greater or equal than the observed number of clicks (see section 3.3.1).

The term in parentheses in the scale factor has the goal of providing a meaningful k, which is the *saturation value*, i.e., the maximum number of clicks one can expect when setting a very high bid. The coefficients a and c have the meaning of an inverted length scale and of a horizontal shift. In order to give them a more intuitive meaning (since it is required to place priors on them), we can link them to the elbows of the curve, which can be identified as the maximum and minimum of the second derivative of the function. The left elbow can be interpreted as the threshold below which the bid yields a negligible number of clicks, while above the right elbow the function effectively saturates. For a standard logistic function (with $a = c = 1$) such elbow points are: $x_\pm = \log(2 \pm \sqrt{3})$. For a general logistic function, the elbows b_- and b_+ are linked to the parameters a and c via: $a = \frac{x_+ - x_-}{b_+ - b_-}$, $c = b_+ - \frac{x_+}{a}$. Switching to the CPC function φ (figure 3.1b), the additional hypothesis of being linear near the origin suggests the same functional form as (3.6), with $c = 0$:

$$\varphi(b) = 2\kappa \left(\frac{1}{1 + e^{-\alpha b}} - \frac{1}{2} \right). \tag{3.7}$$

Here κ is the maximum CPC which can be paid, and the same considerations connecting α with elbows apply.

In order to perform regression, one needs to model the likelihood of the data given the parameters. Since n^{sat} is a *count* of clicks at saturation, a natural choice is the Poisson distribution, centered around the mean given by (3.6). We note however that this count is often *censored*, i.e. only partially known: this happens when the assigned budget is less than the saturation cost. In these cases, all we know is that saturation clicks are greater or equal than observed clicks, and one needs a principled way to take these data into account, without introducing systematic bias. In the bandit model one can assume *non-informative censoring* [34]. Thus, just a simple change in the likelihood is needed: censored data enter the likelihood via the so-called *survival function*, i.e. the complementary of the Poisson CDF. Finally, a natural model for the CPC is offered by the lognormal distribution.

Summing up, this model has the following advantages: *i*) lower variance (with a small bias increase), *ii*) closed form functions, *iii*) it forces monotonicity, which helps optimisation to choose the next action, *iv*) transparent hyperparameters make it easy to elicit priors, and finally *v*) parametric Bayesian regression easily accommodates *censoring*.

Contextual bandits have been mostly studied in the linear [35, 36], generalised linear [37] and kernelised domain [22, 38]. More recently, deep neural networks have been explored for the regression step [39]; their expressive power is, however, balanced by the large need of data. To the best of my knowledge, this is the first instance of contextual bandits with full-fledged Bayesian regression on a parametric function which is not (generalised) linear.

3.3.2 Next super-arm selection

We now turn to the problem of sampling from the posterior distribution, and selecting the best arm accordingly: this means drawing a particular instance of the

functions introduced in (3.4) and (3.5) and solving the optimisation problem (3.3) for those instances. As noted in [27], since the constraint acts only on the budgets, the optimisation problem (3.3) can be decoupled as follows:

$$\max \sum_{j=1}^{N} V_j(\boldsymbol{b}_j, B_j) = \max_{B_1,\dots,B_N} \left(\sum_{j=1}^{N} \max_{\boldsymbol{b}_j} V_j(\boldsymbol{b}_j, B_j) \right).$$

In other words, if we are able to find the bid vector $\boldsymbol{b}_j = \boldsymbol{b}_j(B_j)$ which maximises the value $V_j(\boldsymbol{b}_j, B_j)$ for a fixed budget B_j, we are then left only with the constrained optimisation on the budget splitting $B_1, \dots, B_N$.

While the grid search approach suggested in [27] works well in one dimension (i.e. for single-ad group campaigns), it scales badly with increasing dimensionality. If a GP regression model is employed, owing to the non-monotonic nature of extracted samples, one must recur to *global* methods, as opposed to local ones. On the other hand, employing the monotonic functions (3.6) and (3.7), the function $V_j(\boldsymbol{b}_j, B_j)$ with fixed budget was empirically found to have only one local maximum, which is also global (see figure 3.2a). Therefore, optimisation is amenable to local methods: when applicable, these are both faster and more reliable. Such methods can be applied in practice as follows. Starting from (3.5), $V_j(\boldsymbol{b}_j, B_j)$ can be rewritten as a piecewise function:

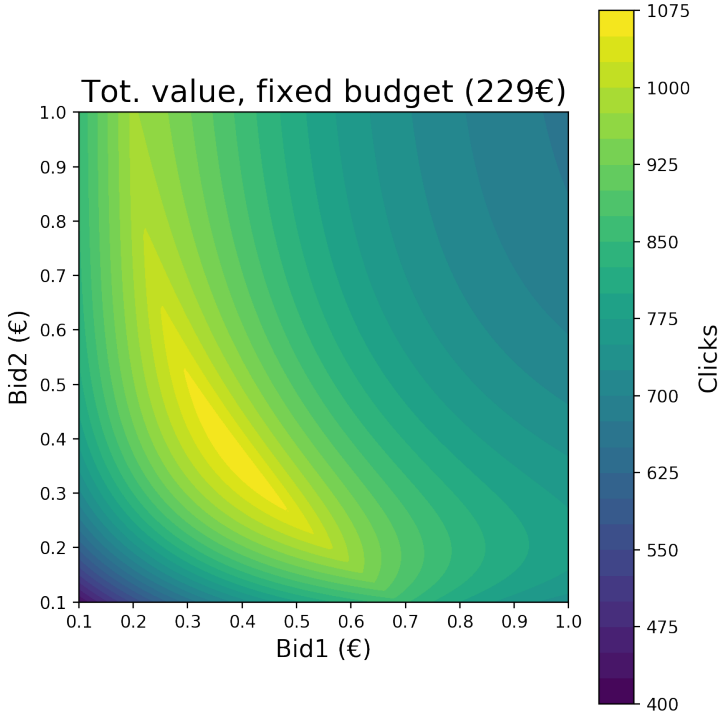
$$V_j(\boldsymbol{b}_j, B_j) = \begin{cases} V_j^{\text{sat}}(\boldsymbol{b}_j), & c_j^{\text{sat}}(\boldsymbol{b}_j) \leq B_j \\ B_j \frac{V_j^{\text{sat}}(\boldsymbol{b}_j)}{c_j^{\text{sat}}(\boldsymbol{b}_j)}, & c_j^{\text{sat}}(\boldsymbol{b}_j) \geq B_j \end{cases}. \tag{3.8}$$

Note that, on the boundary $\{c^{\text{sat}}(\boldsymbol{b}_j) = B_j\}$ between the two regions, the functions coincide. On the other hand, traversing the boundary the gradient changes abruptly, thus hindering the direct application of gradient-based optimisation methods. We will see, however, that a constrained optimisation on just one region is sufficient. First we note that, if both regions are non-empty, the global maximum of $V_j(\boldsymbol{b}_j, B_j)$ is given by the maximum between the two maxima of the function on the two regions. Moreover, every directional derivative of $V_j^{\text{sat}}(\boldsymbol{b}_j)$ (sum of monotonic single-variable functions) is strictly positive. If the boundary $\{c^{\text{sat}}(\boldsymbol{b}_j) = B_j\}$ is not empty, the maximum of $V_j^{\text{sat}}(\boldsymbol{b}_j)$ then lies on said boundary. This, in turn, means that the maximum over the region $\{c^{\text{sat}}(\boldsymbol{b}_j) \leq B_j\}$ is less than or equal to the maximum over the region $\{c^{\text{sat}}(\boldsymbol{b}_j) \geq B_j\}$, i.e. that, if the latter region is not empty, it suffices to search the maximum there.

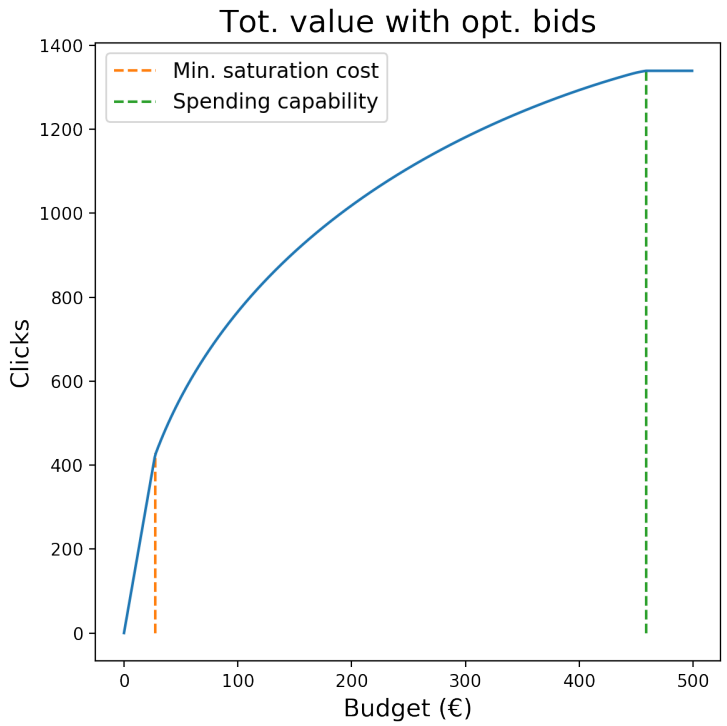
Up to now we dealt with finding the optimal bids for a campaign given the budget, thus finding a function $\boldsymbol{b}_j = \boldsymbol{b}_j(B_j)$. We must now solve the following optimisation problem:

$$\max_{B_1,\dots,B_N} \sum_{j=1}^{N} V_j(\boldsymbol{b}_j(B_j), B_j) \quad \text{s.t.} \sum_j B_j \leq B. \tag{3.9}$$

The terms $V_j(\boldsymbol{b}_j(B_j), B_j)$ in the sum are single-argument functions that depend only on the budget of the campaign.



(a) Total value function for fixed budget



(b) Total value function for varying budget, selecting optimal bids for each budget

Figure 3.2: Thompson sample of the total value function $V_j(\boldsymbol{b}_j, B_j)$ of a campaign with two ad groups (click values v_{jk} are set to 1 for simplicity).

For budgets B_j greater than the *spending capability* $c_j^{\text{sat}}(\bar{b}_{j1}, \dots \bar{b}_{jm_j})$ of the campaign, such functions become constant, as can be seen by (3.8). For budgets below the spending capabilities, the functions have been empirically found to be downwards concave (see figure 3.2b), in agreement with the law of diminishing returns.

Interestingly, this property, which in the present work is *emergent* from the characteristic of the click and cost-per-click functions, is instead *assumed* in [40]: that work, however, deals with the optimisation of budgets only, implicitly assuming that bid choice is either unnecessary or taken care of by some dedicated (and undescribed) mechanism. [2]

Convexity also means that this optimisation step is amenable to local gradient methods too. If, however, the optimisation over bids is performed with numerical methods, extra care must be taken in choosing the step size over budgets: small errors in the first step translate to a small noise in the function $V_j(\boldsymbol{b}_j(B_j), B_j)$. To control this issue, I developed an intuitive optimisation procedure which generalises the budget splitting strategy presented in [41] to the case of non-constant return on investment (i.e. non-linear functions): this procedure is outlined in Algorithm 1.

Algorithm 1 Local budget splitting optimisation

Input: Spending capabilities of every campaign

1: **for** every campaign j **do**
2: Assign j-th spending capability to j as initial budget
3: **end for**
4: **while** total assigned budget is greater than B **do**
5: **for** every campaign j **do**
6: Calculate discrete derivative of $V_j(\boldsymbol{b}_j(B_j), B_j)$
7: **end for**
8: Find campaign with smallest discrete derivative
9: Subtract one unit from its assigned budget (e.g. one euro)
10: **end while**

The procedure keeps subtracting budget from a campaign until the discrete derivative matches or becomes smaller than the discrete derivative of another. This means that the procedure effectively searches for the point in the budget splitting simplex where the derivatives are approximately equal: this is the solution of the Lagrange problem corresponding to (3.9).

[2] Disregarding the more limited scope with respect to optimisation, [40] is interesting for a manifold of reasons. First, the authors associate time-dependent contexts to campaigns, to tackle the possibility of portfolios that evolve in time and the consequent *cold-start problem* for campaigns with limited or no historical performance data. Moreover, they consider more general constraints beside fixing the total budget. Further, the dataset augmentation they employ is somewhat reminiscent of the Bootstrapped Thompson Sampling technique we will see in subsection 3.3.3, though the goal (transfer learning) and implementation details are different. Finally, they consider the problems of non-stationarity and delayed rewards, as we will see in section 4.1.

3.3.3 Bootstrapped Thompson Sampling

While Markov chain Monte Carlo [33] has become the *de facto* standard for Bayesian inference when no closed formula for the posterior exists, it presents challenges that could hinder applications. First, some tuning is often required on the models to ensure that the Markov process converges, and that the sampled data approximate the posterior well. Second, it is resource-intensive, and requires much time to converge (if compared to maximum likelihood models).

To bridge this gap, I adapted an approximated, lighter alternative, i.e. Bootstrapped Thompson Sampling [17, 29]. As we will see in section 3.3.3 below, this setting provided also the testbed for comparing Bootstrapped Thompson sampling with exact Thompson sampling, before using the former in a setting where Markov chain Monte Carlo cannot be used (since a proper Bayesian formulation is lacking completely).

Being crucial for the contributions of chapter 4, we will introduce Bootstrapped Thomspon Sampling in more detail there: we introduce here a high level view of the technique. The main idea behind this method is to approximate sampling from the posterior with the Statistical Bootstrap. In the case of our parametric model, this would mean sampling with replacement from the history of data gathered so far, and finding the maximum likelihood estimate of the parameters for said sampled history. This procedure, however, would present two issues [29]: the prior over parameters is ignored and, more importantly, the uncertainty over parameters is underestimated in the initial rounds of the optimisation process, when data points are scarce. Since Thompson sampling leverages this uncertainty to produce next action decisions, this naive application of the bootstrap is known to lead to linear regret [17]. In particular, in the first days of optimisation the agent could conclude that an ad group is underperforming with respect to the others just by chance, and stop allocating budget to that ad group altogether, never allowing it to recover.

To overcome this issue, I adapted Algorithm 3 of [17] with Bayesian bootstrapping to the present setting. The adapted algorithm is reported in Algorithm 2.

3.4 Numerical simulations

In order to test the model and compare it with the state of the art, I designed and developed an environment which tries to capture what is disclosed about the ad placing auctions [13]: this environment is described in section 3.4.1. In section 3.4.2 we will analyse the results of the simulations.

3.4.1 Simulation environment

To build the simulation environment herein described, I had to make some simplifying assumptions. Nevertheless, the click and CPC dependence on bids agrees with experience on actual auctions. Moreover, the goal was proving the ability of the optimiser to adapt to an environment which is similar enough to reality. In

Algorithm 2 Bootstrapped Thompson Sampling

Input: Prior P over parameters of models (3.6) and (3.7); ad groups list $(a_1, a_2, \dots)$.

Data $D_0 = ()$
for $t = 0, 1, \dots$ **do**
  for $a = a_1, a_2, \dots$ **do**
    Uniformly sample artificial bid $\tilde{b}$ over allowed range
    Sample model parameters $(k, a, c, \kappa, \alpha) \sim P$
    Sample artificial saturation clicks $\tilde{n} \sim \text{Pois}\left(n^{\text{sat}}(\tilde{b})\right)$
    Sample artificial CPC $\tilde{\varphi} \sim \text{Lognormal}\left(\varphi(\tilde{b})\right)$
    Sample $t+1$ weights $w_i \sim \text{Gamma}(1, 1), \quad i = 0, \dots, t$
    Weighted Maximum Likelihood regression over data $D_t \cup (\tilde{b}, \tilde{n}, \tilde{\varphi})$
  end for
  Use ML parameters for choosing next bids and budgets as in section 3.3.2
  Update data D_{t+1} with observed clicks and CPCs
end for

particular, note that clicks and CPC are not simulated according to the model that is being tested (i.e., the one proposed in section 3.3.1) to avoid biasing the results.

For every day, the number of searches compatible with an ad group is sampled from a Poisson distribution. Then, for each search an auction is simulated. The number of competing advertisers is again sampled from a Poisson distribution (with a different mean). The ads belonging to different advertisers are ranked according to the product of three quantities. The first is the bid: for competing advertisers, the bid is sampled from an exponential distribution. The second is a *static quality score*, which measures the intrinsic quality of the ad: it is sampled from a triangular distribution. The third is an *instantaneous quality score*, which measures the affinity between the single search and the ad group. It is modeled as an angle between vectors, which is extracted from a rescaled beta distribution; then the quality score is calculated as the scalar product between said vectors.

After the ads have been ranked, the first ones appear on the search engine result page: whether they are clicked or not is determined by a Bernoulli distribution. Then, in keeping with the meaning of the bid as maximum CPC, the advertiser that has received a click pays the minimum amount necessary to appear in that position. The budget is then updated accordingly, until either available searches end or the budget is finished. A second Bernoulli distribution governs which clicks turn into contacts. What distinguishes various simulations are the parameters of the manifold of the probability distributions involved.

To run a comparison between the parametric regression model introduced in section 3.3.1 and the GP model introduced in [27], the latter needs some additional metric to extrapolate the saturation clicks of the day from the observed clicks, as stated in section 3.1. To this end, I have chosen *lost impression share*, an estimate of the fraction of times the ad was eligible for appearing in a search, but did not due to limited budget. To capture the fact that it is inherently a noisy quantity, a convex combination of the actual lost impression share with random fractions was

used, with varying coefficients.

The code of the simulation environment is available at https://github.com/MarcoGigli/sem-simulation.

3.4.2 Simulation results

I run 120 experiments randomly drawing the parameters introduced in section 3.4.1 The number of campaigns of the portfolio varied between 2 and 8 and, for each campaign, the number of ad groups varied between 1 and 4. For each parameter setting, both the parametric and the GP model optimised the total value for 100 virtual days.

In these numerical simulations the parametric model runs with full-fledged Bayesian inference, performed with MCMC: tests using the approximated Bootstrapped Thompson Sampling introduced in section 3.3.3 were deferred to the real-world experiments (see section 3.5.2).

As explained in section 2.1, the most commonly used figure of merit to evaluate the performance of bandit algorithms is *regret*. The latter, however, requires knowing which action on average leads to the best reward.

In these simulations, however, the mean reward is an emergent property that is not easily deduced by the parameters that characterise a problem instance: for this reason I evaluated the regret of using the GP model *instead of the parametric one*:

$$R_n = \sum_{t=1}^{n} \left(r_t^{\text{par}} - r_t^{\text{GP}} \right).$$

Here r_t^{par} and r_t^{GP} represent the rewards received at day t using the parametric and GP model respectively. In particular, it is given by the number of contacts.

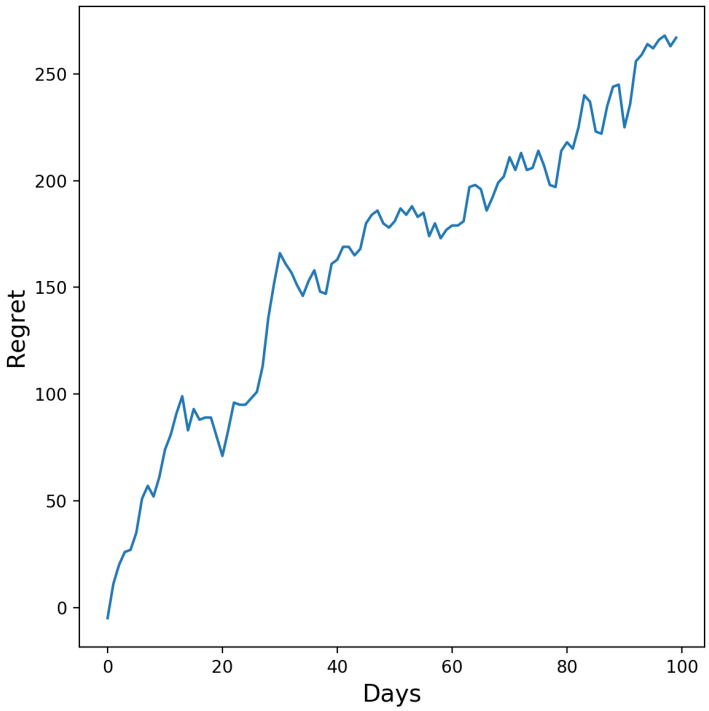
Moreover, to meaningfully compare performances of experiments with different parameters, I also calculated the *relative regret*: $\rho_n = \frac{\sum_{t=1}^{n} \left(r_t^{\text{par}} - r_t^{\text{GP}} \right)}{\sum_{t=1}^{n} r_t^{\text{par}}}$.

In figure 3.3, the behavior in time of regret and relative regret is shown for a particular set of parameters, i.e. one of the 120 experiments. After a short time in which, due to random fluctuations, the GP model gathers more contacts, the relative regret quickly raises to 40%. Then, as both models are given more data, the relative difference in performance gradually tapers off, and converges to approximately 10%.

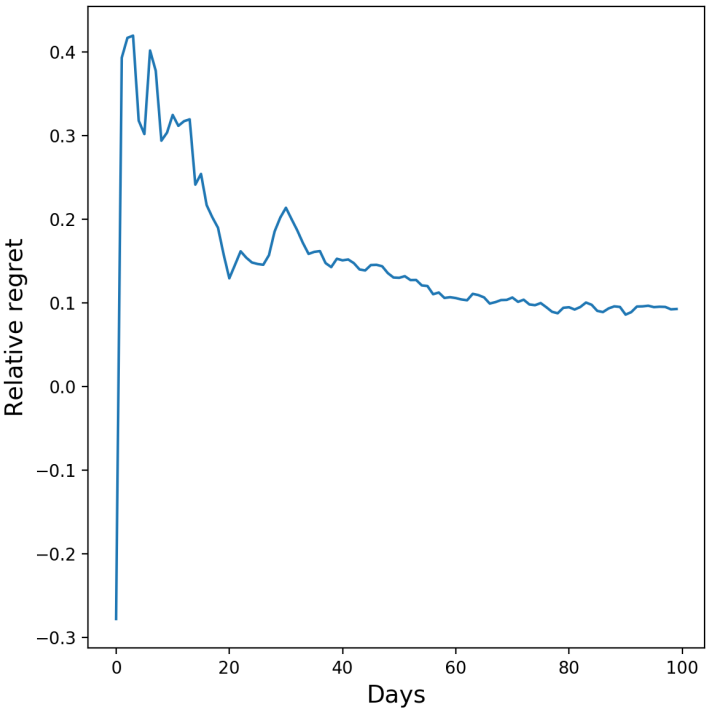
This example is typical, as can be seen in figure 3.4 and in table 3.1: at $n = 10$ days, only in 11 experiments the regret is negative, and in most cases the relative regret ranges from 8% to 55%.

Fast convergence is especially important if a sliding window strategy is employed to retroactively take time dependence into account, as in [12] and [42]. At $n = 100$ days, the relative regrets are much less spread out and lower on average, and again they are negative only in roughly one in ten cases (13 runs).

Upon closer inspection the most important feature in determining regret is the number of ad groups per campaign (see figure 3.5). This shows that the ability



(a) Regret



(b) Relative Regret

Figure 3.3: Time dependence of regret and relative regret suffered by the GP model when compared to the parametric model.

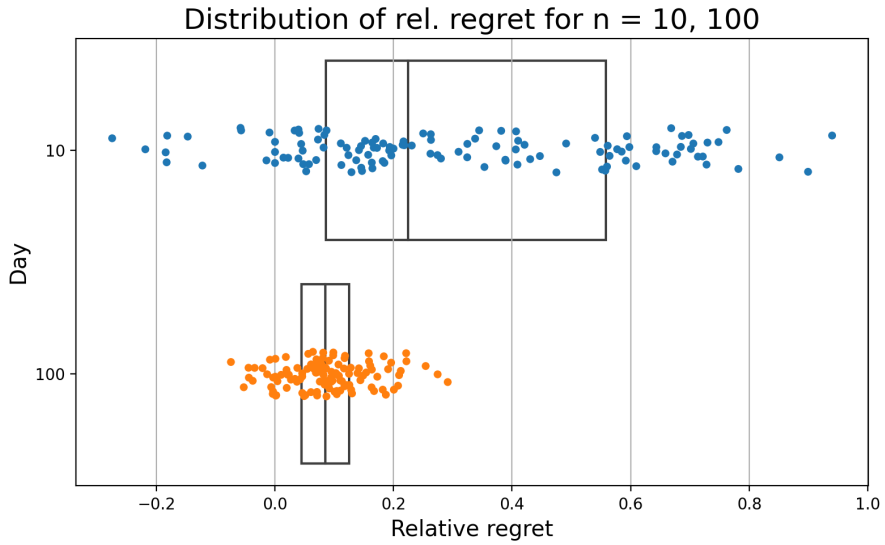


Figure 3.4: Distribution of the relative regret at $n = 10$ days and $n = 100$ days for all synthetic experiments. The boxes show the first quartile, median and third quartile.

Table 3.1: Descriptive statistics of the relative regret at $n = 10$ days and $n = 100$ days.

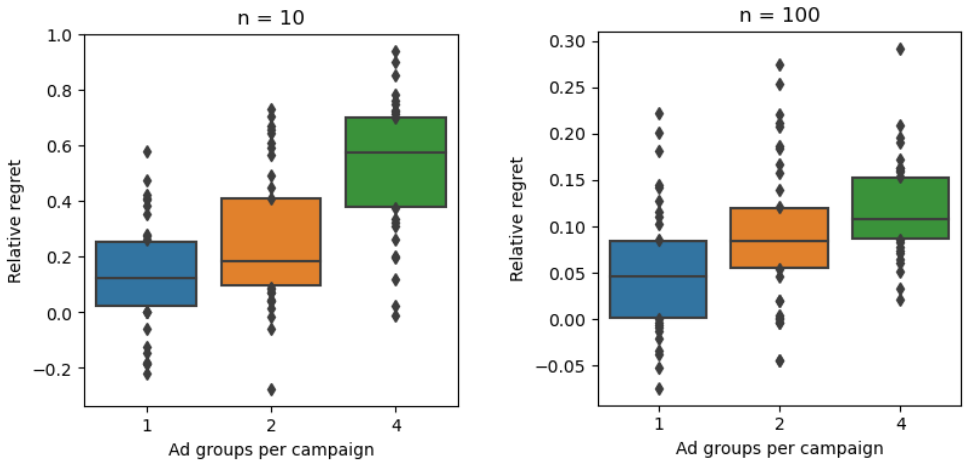
Days	Relative regret		
	First quartile	Median	Third quartile
10	8.6 %	22.4 %	55.8 %
100	4.5 %	8.5 %	12.5 %

to efficiently search for the optimal bid combination given a certain budget (as described in section 3.3.2) is crucial for the performance of the agent.

Moreover, higher percentages of noise in the extrapolation metric are associated with a higher relative regret (especially at $n = 100$ days), as is to be expected from the discussion of section 3.2. Other features do not show a clear link with relative regret.

3.5 Real-world data

While the simulation enviroment described in section 3.4.1 has the benefit of giving full control over the parameters of the simulation, it is at risk of being too idealised and relying too heavily on assumptions, particularly with regard to probability distributions and the presence of outliers. To counter this, I performed extensive experiments over the Criteo Attribution Modeling for Bidding Dataset [16], which contains one month worth of (subsampled) Criteo display advertising data. Dealing with real data, as opposed to synthetic, means the problem is not shielded from non-stationarity; moreover, the imposed 31-days time frame makes data efficiency



(a) After n=10 days

(b) After n=100 days

Figure 3.5: Distribution of the relative regret at $n = 10$ days and $n = 100$ days varying the number of ad groups per campaign. The boxes show the first quartile, median and third quartile.

all the more important.

The next subsection is devoted to describing the dataset and how it can be used to evaluate bandit performances (section 3.5.1). The results of the experiments are then analysed in section 3.5.2.

3.5.1 The Criteo dataset

Every line of the dataset contains contextual features regarding the user, the website and the ad. These (anonymised) categorical features are used to feed a supervised model that predicts the probability of conversion (see [16] for details). This predicted probability is the analogue of the product of static and instantaneous quality score described in section 3.4.1: it is multiplied by the bid to determine the *ad rank*, and thus the right to be shown on page. The dataset was thus split in two: one half was used to train the predictive model, while the other half was used to test the bandit algorithms. The training set was also used to learn the priors, both over the parameters of the proposed Bayesian model and over the hyperparameters of the GPs.

The data points correspond to auctions that were *won* by the production policy; in other words, the dataset is *truncated*, since lines corresponding to lost auctions are absent. This selection bias means that the logged distribution of minimum winning bids is not representative of the true one. More generally, to evaluate the performance of competing bandit policies on logged data one would need a full counterfactual analysis [43], which could be unfeasible if the production policy is deterministic. In this case, however, it has become standard practice to tackle the problem of selection bias by injecting noise into the distribution of competing bids [16, 44], and I adopted this simple approach.

It was thus possible to effectively replay the auctions. The bid chosen by the agent for a given day and ad group is compared, after multiplication by the predicted probability of conversion, with the best competing ad rank. If the auction is won and the ad in the dataset received a click, then the click count for that day is increased by one. The same holds for conversions.

As an ablation experiment, and to keep matters as close as possible to the original dataset, I fed the GP model with the actual lost impression share, without adding noise. This benefits the GP model, as one of the concerns raised in section 3.1 was here removed.

Nevertheless, we will see in section 3.5.2 that the parametric Bayesian model performs best on average in every examined setting.

Before delving into comparing the performances of the bandit algorithms on this dataset, we can study the two regression models taken in isolation. First note that the right hand side of formula (3.6) with $k = 1$ is the probability of winning a single auction, as predicted by the parametric model. The actual probability of winning is given here by the cumulative distribution of competing ad ranks (which is unknown to the agent). A comparison of the two curves shows the expressive power of the proposed model (see figure 3.6 for an example).

Switching to the number of clicks in a day, one can compare the proposed parametric model with a GP trained on the same data (see figure 3.7). Here we see that the fewer restrictions placed on the GP (as mentioned in section 3.1) mean much slower convergence to sensible shapes.

3.5.2 Results on the Criteo data

In order to study how the main parameters of the experiment affect the regrets, I varied them one at a time, keeping the others fixed: budget per ad group (figure 3.8a), number of campaigns (figure 3.8b), and number of ad groups per campaign (figure 3.8c).

For each parameter combination, I sampled 120 realisations from the Criteo dataset, i.e., randomly extracting portfolios with the chosen number of campaigns and ad groups, and replayed its auction as described in section 3.5.1. Since the campaigns of the Criteo dataset present no substructure, to test multi-ad group scenarios I treated them as ad groups, and randomly clustered them to form campaigns.

The experiments letting budget and number of campaigns vary were conducted with one ad group per campaign, to perform an ablation experiment and study the effect of the parametric model alone, independently of the multi-ad group generalisation introduced in section 3.2.

Overall, in every setting I studied, the average regret suffered by either the GP method or Bootstrapped TS at the end of the simulation is positive. As shown in figure 3.8, the average percentage of conversions lost due to not using the parametric Bayesian method can reach in some settings 40% (figure 3.8c).

As can be seen from figure 3.8a, keeping the number of campaigns and ad groups fixed and increasing daily budget, relative regret quite steadily decreases

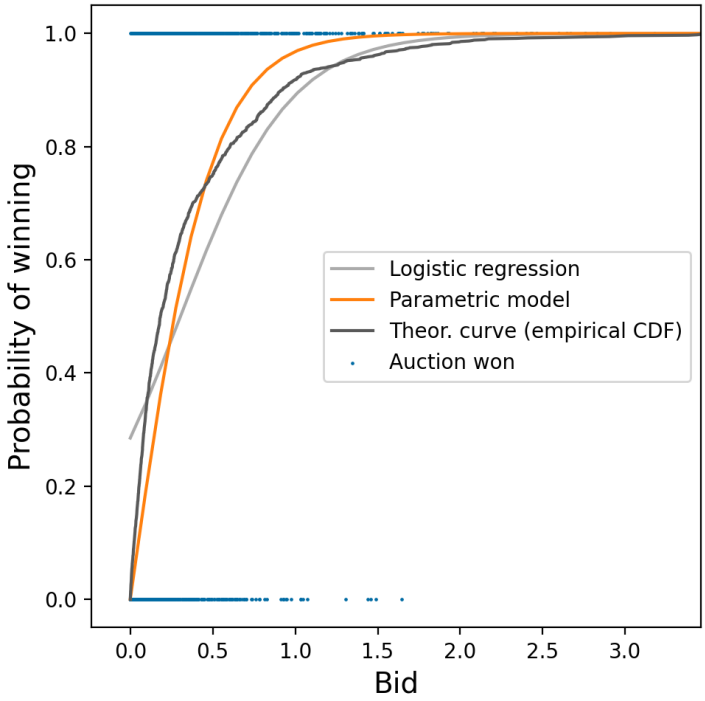


Figure 3.6: Predicted versus actual dependence of the probability of winning an auction on the bid, for a campaign of the Criteo dataset.

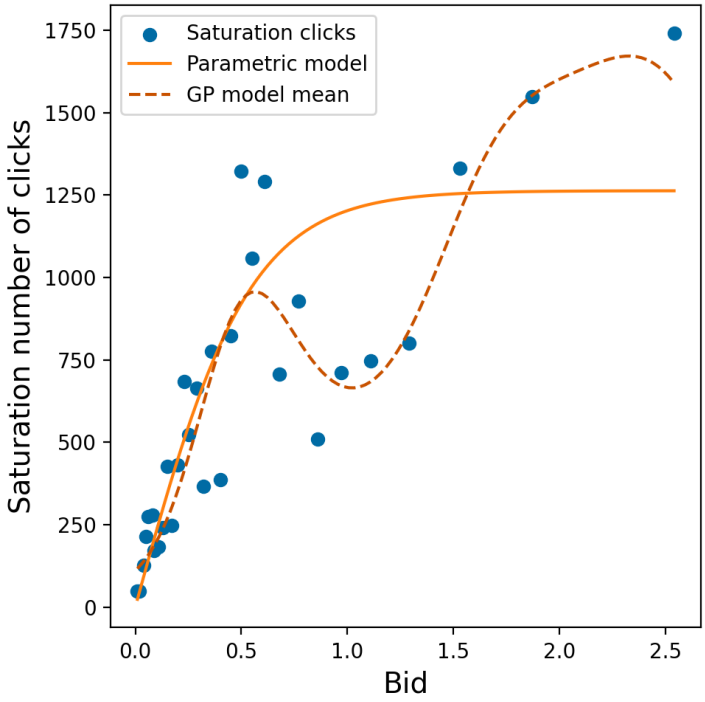
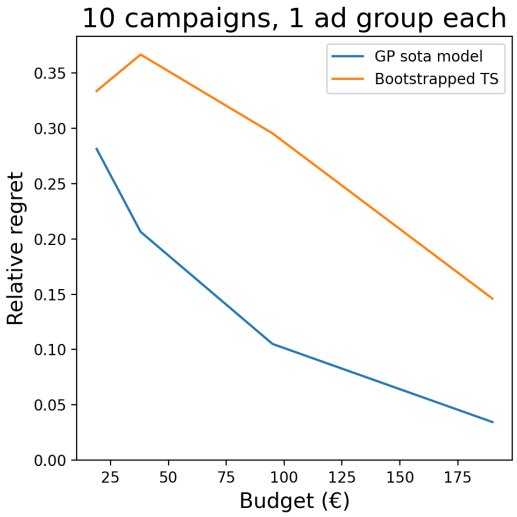
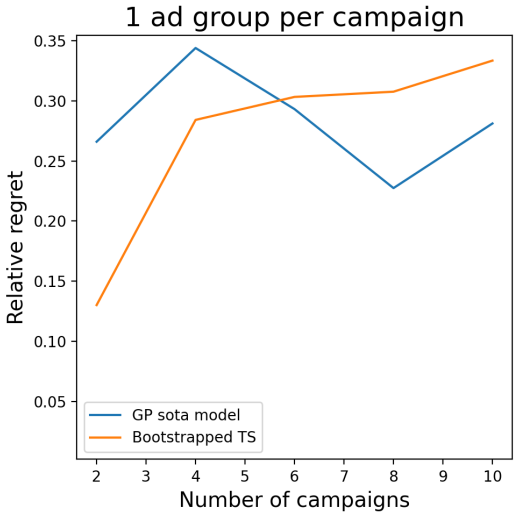


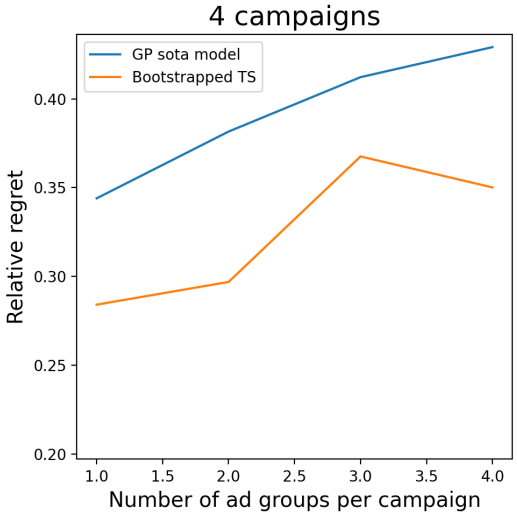
Figure 3.7: Daily saturation number of clicks for a given bid with the prediction of the parametric model and the GP, for a campaign of the Criteo dataset.



(a) Varying budget



(b) Varying campaign number



(c) Varying ad groups per campaign

Figure 3.8: Behavior of the average relative regret on the Criteo dataset varying one parameter at a time.

both for GPs and Bootstrapped TS. This is to be expected, since increasing daily budget means moving closer to saturation cost, which in turn means that

- errors in splitting budget have less impact,
- more budget is available for exploration, which means getting more accurate data.

Put differently, while the proposed method gathers more conversions on average for all the budgets we tested, its advantages are more clear cut when daily budget is tight.

The relative order between the two curves in figure 3.8a is in turn an effect of the relatively high number of campaigns involved, as can be seen in figure 3.8b: while the regret for GP is somewhat independent by the number of campaigns, Bootstrapped TS suffers from increasing this number, so that the two curves cross.

If we instead let the number of ad groups per campaign vary (figure 3.8c), while both methods show increasing average relative regret with increasing number of ad groups, Bootstrapped TS suffers less: while the Bayesian posterior is approximated, next action selection is the same as in the full-fledged parametric Bayesian algorithm, and can thus benefit from the same local optimisation method.

The GP method and Bootstrapped TS show comparable performances across the settings studied. One can interpret this as showing the separated effects of both a more efficient model (shared by Bootstrapped TS and our method) and full Bayesian inference (shared by GPs and our method): one needs both in order to increase performance in this environment.

4

Approximate Thompson Sampling for delayed feedback

As briefly touched upon in the Introduction, delayed feedback is an intrinsic property of digital advertising: the metrics to be maximised are most-often measured a non-negligible interval of time after the *impression* (i.e., after the instant when the user first sees the advertisement). Delays cannot be ignored in the sense that, during the aforementioned interval of time, the agent in charge of carrying out the optimisation has to take a number of other actions, without being able to benefit from the information about its past choices. In the language of Multi-armed Bandits (MABs) introduced in chapter 2, the learner faces several *rounds* before being able to update its strategy with the most recent feedback.

Delays are unavoidable both from the perspective of the advertiser and the *advertising platform* (i.e., the platform that gets paid by the advertiser to show advertisements to end-users). Starting from the latter perspective, assume an algorithm is in charge of deciding which ads to serve to the users of a website. This problem is very similar to the one of section 2.2. Even if the algorithm has the goal of maximising the number of *clicks*, which usually happen within minutes since impression, it needs to continue serving ads to other users in the meanwhile.

Going back to advertisers, the delay between impression and click is usually not problematic if, as in chapter 3, one round corresponds to a calendar day: the overwhelming majority of clicks happens within the same day as the impression, so that we can simply ignore the small fraction that does not. However, as hinted upon in chapter 3, clicks are a suboptimal proxy to optimise: advertisers are interested in increasing the number of customer contacts or, if at all possible, sales directly.

Depending on the business area, the delay between first click and contact or sale (two events that can be collectively called *conversions*) can reach several

weeks. In particular, we can expect this delay to be greater in high-ticket market segments [45]. Waiting for a time-cohort of customers to *close*, i.e., waiting for the feedback of all customers from a specific time period, means being unable to use fresh data and adapt to market shifts [46].

Finally, both sides (advertisers and advertising platforms) often do not re-train their algorithms *online*, i.e. at every user interaction. This is due both to performance reasons (a re-train can take longer than the typical time separation between consecutive rounds) and also safety reasons: the re-trained model often needs to be tested before deployment, to avoid user-facing issues [40]. For this reason, the learning algorithm usually receives data in *batches*.

Notwithstanding the above, in chapter 3 we have seen an idealised version of the bid/budget optimisation problem, in which *immediate* feedback was assumed, with the aim of tackling one facet of the problem at a time. Since MABs proved an effective tool in carrying out the optimisation, I scanned the literature about MABs with delayed rewards in search for indications on how to properly treat delayed conversions: the next section is devoted to such literature review.

A setting emerged which, while simpler than the full problem of optimising bids and budgets with delays, has several of its characteristic traits and showed potential for improvement with respect to the state of the art, while potentially paving the way for tackling the full bid/budget optimisation problem. Section 4.2 is devoted to introducing this setting. In the subsequent sections 4.3 and 4.4 we will go through the model behind my approach and the comparison of results with the state of the art.

4.1 Stochastic bandits with delayed feedback: a review

4.1.1 Early works

Insights about the problem of Multi-armed Bandits with delayed feedback go back at least to the influential paper [32]. The authors empirically proved the effectiveness of Thompson sampling (TS) in a manifold of bandit problems, thus contributing greatly to the re-discovery of the technique by the bandit community [29]. In particular, they compared TS to the UCB algorithm in the presence of fixed, deterministic delays of varying entity via simulations. TS proved robust to delays, while the performance of UCB matched that of TS only in the case of immediate feedback. The reason is that, being UCB a deterministic algorithm, the same action is played over and over while waiting for feedback, leading to redundant exploration. The randomised nature of TS, on the other hand, makes it explore much more effectively.

The first work that studied theoretically the effect of delays on bandits is [47]. In the contextual bandit setting, again with constant, deterministic delays, the authors developed a proof-of-concept policy that showed it is possible to obtain regret bounds that only increase *additively* with delay: this is desirable, as it means that the penalty becomes negligible as the horizon T grows. However, their policy

is computationally intractable, and requires exact knowledge of the probability distribution of contexts.

In the non-contextual, finite-armed setting, the authors of [48] developed a queue based wrapper algorithm which transforms, in a black-box fashion, immediate-feedback bandit algorithms into algorithms capable of handling the delayed case. In addition, they analyse the straightforward adaptation of the UCB algorithm that just uses the available feedback, ignoring actions that did not receive feedback yet (as was done empirically in [32]). For both proposals, assuming that the distribution of delays has bounded expectation, they show that the regret incurs an additive penalty. On the other hand, both proposed algorithms act deterministically while waiting for feedback: this is true even if one uses the first algorithm to wrap TS, which is random in nature. As noted above, this trait is sub-optimal.

Also the authors of [49] provide a queue based wrapping algorithm, but noting the harming nature of determinism in the presence of delays, they design a balancing mechanism to avoid long queues (i.e., to avoid repeating the same action for too long), and they also allow for the use of external heuristics. While regret bounds are the same as in [48], via simulations they show that the practical performance of their algorithms is significantly better. Moreover, in experiments they note that pure TS performs as good as their proposed algorithm.

4.1.2 Gaussian Process bandits

The authors of [23] study delays in the context of Gaussian Process (GP) bandits. They too note that UCB suffers from repeating the same action while waiting for feedback, and introduce a technique to correct for this in the context of bounded delays (which include batch re-training and deterministic delays as special cases). The key observation is that, in GPs, the posterior variance depends only on *where* observations are made, but not on *which values* of the function are observed: in other words, even while waiting for delayed feedback, the agent is able to predict by how much the uncertainty will reduce. A natural way to avoid redundant exploration in GP-UCB is then to keep the mean of the posterior unaffected by the unavailable observations, while updating the uncertainty estimate using all observations (including those that still have to receive feedback). This is equivalent to *hallucinating* the missing rewards, substituting them with the posterior mean: in this form, the technique could actually be applied to MABs in general. The authors of [23] note, however, that this brings to an *overconfident* posterior, which does not contain the true reward function with high probability. To correct for this and obtain low regret bounds, they inflate the multiplicative factor α_t of the uncertainty in the action choice rule (2.8). Interestingly, however, in experiments they do not inflate α_t, and indeed they introduce a small premultiplier (0.05 or 0.1), which further reduces exploration. This signals some detachment between rigorous bounds and practical purposes: while bounds are set on worst case scenarios, one is usually interested in average performance in some sense. We will see further signs of this when discussing [50] and in section 4.2.

From [23] a whole stream ignited, aimed at studying GP bandits with delays. The authors of [51] note that, in the case of batches of known size n, where the agent can plan the whole set of actions in advance, the hallucination rule introduced in [23] is *ad hoc*: a more principled strategy would be to collectively plan the n actions to be played strategically. However, due to the combinatorial nature of the problem, this planning task scales exponentially in n. If, for instance, the batches originate from the parallelisation of function evaluations to speed up the optimisation, the time saved could be offset by the time needed to calculate the next n points to query. They thus propose an approximate technique that achieves good performance, and the degree of approximation can be tuned with respect to the cost of the single function evaluation.

The authors of [52] observe that the need for variance reduction via hallucinating rewards in UCB stems from the necessity to explicitly enforce diversity of query points: ultimately, this is a consequence of the deterministic behavior of UCB. On the other hand, the inherent randomness of GP-TS suffices to avoid redundant function evaluations: they prove this for simple Bayesian regret. They also prove empirically that the variant of GP-TS that employs the hallucination diversity scheme performs either about the same as or slightly worse than pure GP-TS. After [32], [49] and [52], we will encounter again proofs of the good practical performance of TS in the presence of delays when commenting [50].

In the opposite direction, the authors of [53] modify the hallucinating algorithm of [23], reducing the width of the confidence interval, possibly building upon the observation that was made in [23], namely that in empirical tests they had to make the algorithm exploit more aggressively. They also provide an hallucinating version of GP-TS: besides feeding GP-TS with the expected reward while waiting for feedback, also the related uncertainty gets shrunk, as in their GP-UCB based algorithm. With these algorithms, they improve upon the bounds derived in [23] in the frequentist setting.

Continuing in the GP-bandit stream, the authors of [54] concentrate on feedback with *stochastic* delays, which contain batch updates of previous works as a special case. Building upon a strategy contained in [18] (which we will consider later on in the review), they modify the hallucination mechanism of [23] and [53]: unobserved feedback gets substituted with the *minimum* for the function (assuming it is known) rather than the posterior mean. Doing so, they foster exploration, even more so than hallucinating the mean. They consider both a UCB-like and a TS-like variant. In both cases, the posterior uncertainty is multiplied by a properly tuned factor. Restricting to a large class of GP kernels, they prove tighter regret bounds with respect to those in [53]. These bounds depend on quantiles of the distribution of delays: in doing so they avoid placing assumptions on the shape of the distribution. In experiments, however, they consider a Poisson distribution for delays, i.e., a light-tailed one. Moreover, interestingly, they don't explicitly compare UCB-like algorithms to TS-like ones, and don't compare their algorithms with the UCB-like algorithm of [53]. Moreover, GP-TS with delays as introduced in [52] seems to perform competitively to their TS-like algorithm, without the need for carefully tuning the time-dependent uncertainty pre-factor α_t.

4.1.3 Partially observable rewards

Going back to finite-armed bandits, the authors of [55] focus on bandits to optimise *conversions* in digital marketing, i.e., sales or customer contacts. They start from an remark contained in [24], namely that conversions are only *partially observable*: after a user has clicked on an advertisement, they may decide to buy the product after some time, thus providing positive feedback; if they decide *not to buy*, on the other hand, no explicit feedback is ever sent to the system. The author of [24] deals with the Maximum Likelihood Estimation (MLE) of the conversion probability in such a setting, assuming that the delay distribution is exponential, but without assuming that the average delay is known: the distribution of delays is modelled together with the conversion probability. The authors of [55], on the other hand, treat this partially observable feedback in the bandit realm. They propose a UCB-like algorithm, which however needs *full knowledge* of the delay distribution, in order to re-weight observed feedback.

This strong (and oftentimes unrealistic) assumption is significantly relaxed in [45]. The authors study again the finite-armed bandit problem with partially observable feedback, but only assume that the Cumulative Density Function (CDF) of the delay distribution is bounded from below by the CDF of a α-Pareto distribution, thus allowing for infinite expected delays if $\alpha < 1$. It is assumed that the agent knows α. Moreover, they admit the case of arm-dependent distributions (i.e., one value of α per arm). Besides the problem of *censoring* caused by partially observable delays (due to which the learner has to deal with an unknown number of missing rewards), they point out an *identifiability* issue: having two distributions at play (the distribution on delays and the Bernoulli distribution on rewards), the same observed data could be produced by pairs of distributions which differ in their parameters. We will deal with this identifiability problem in section 4.3.

The authors propose a UCB-like algorithm. The empirical average in this algorithm is biased, as all past actions are taken into account, even if some may be waiting for positive feedback: all missing rewards are treated as zeroes. To deal with this, they inflate the upper confidence bound with a term that handles the extra bias due to delays.

The problem of partially observable rewards in the linear contextual bandit setting has been tackled by the authors of [18] with a somewhat similar logic. They employ a LinUCB-like algorithm that estimates the linear coefficient vector ϑ we saw in section 2.2 using all played actions, and the confidence interval gets inflated by a quantity that reflects the bias. To calculate this quantity, the algorithm takes as input a *window parameter* m: if feedback has not been received within m rounds, the algorithm labels it as a non-conversion. Though in some situations this window parameter could be externally imposed to the algorithm due to memory constraints, its presence is somewhat unfortunate, as in general it should be carefully tuned depending on the delay distribution.

The authors do not place restrictions on the shape of the delay distribution, as the regret bound depends on a quantile of the distribution (as in [54] that we saw above, which was inspired by this work). On the other hand, the delay

distribution does not depend on the action (contrary to [45]). They also provide a LinTS-like heuristic, though theirs cannot be considered a pure TS approach, as it does not sample from a posterior distribution: they obtain it heuristically from the confidence region of their proposed LinUCB-like algorithm. Finally, they test their algorithms on geometrically distributed delays and on an empirical distribution of delays inferred from the real-world dataset of [16].

We will thoroughly analyse this linear contextual setting with partially observable rewards in sections 4.3 and 4.4. We will see empirical proof on a host of delay distributions that a TS approach, albeit approximated, yields at worst comparable and at best significantly better regret. The TS algorithm I propose shares several traits with [19], though the authors of the latter work analyse the finite-armed bandit setting, rather than the linear contextual one. In this practitioners' paper, the authors move their steps from the MLE model in [24]: they generalise the derivation to generic delay distributions, though they then apply the model assuming exponential delays. In this model, the distributions of delays and rewards decouple and are learned separately. They propose a TS-like approach, which considers only the uncertainty on the distribution of rewards: the delay model is assumed to be precisely known, even if it is learned with the passing of rounds together with the distribution of rewards. This strategy is practical because it permits using a Beta-Bernoulli conjugate on rewards corrected using the distribution of delays; on the other hand, ignoring uncertainty on the delay distribution could lead to insufficient exploration, especially in the case of heavy-tailed delay distributions.

Rather than providing rigorous regret bounds, being interested in average-case performance, they conduct a host of numerical experiments, comparing their solution to the straightforward extension of TS which ignores the partial observability problem and to the UCB-like algorithm of [55]. Since the latter model needs the delay distribution (assumed to be known) as input, they provide it with the *learned* distribution. In all reported experiments, their TS-like approach outperforms the others. It must be noted, though, that they focus on light-tailed distributions in experiments, and they don't compare their method to the one proposed in [45].

The last work we are going to cover in the partially observable setting is [56]. The authors study the problem of deciding how to allocate existing vaccines to target new variants of an infectious disease. Feedback is partially observable, because only negative rewards are observed (i.e., infections). Since the decision is among a finite set of vaccines, they study the finite-armed bandit case, without personalising vaccines (i.e., without contexts). This problem is interesting in that the rate of infection depends on the ebb and flow of the pandemic: it is a non-stationary problem. This model can also capture the evolution in time of the prevalence of a specific variant, if only infections by the given variant are treated as events. The authors model the problem as a proportional hazards model: there is a baseline risk for an unvaccinated person to get infected, and arms capture the protective effect of vaccines. Via a partial likelihood approach, they are able to cancel out the non-stationary baseline risk. By then applying TS, they are able to minimise infections over time, with better performances with

respect to randomised control trials: they prove this performing simulations on real COVID-19 data.

In their model, they assume that the vaccine efficacy does not wane with time since vaccination: if the baseline risk were constant, this would correspond to an exponential distribution of times between vaccination and infection. We will see in section 4.2 that this setting is, in a sense, opposite to ours: we will assume stationarity of the baseline risk, but general (in particular, not necessarily exponential) delay distribution, modelled with a Kaplan-Meier estimator.

4.1.4 Aggregated anonymous feedback

With respect to the partially observable setting, the authors of [57] consider an even more challenging setting, that of *aggregated anonymous feedback*. Namely, the learner observes the sum of the rewards that arrive at the same round, without being able to disentangle to which arms the single rewards correspond. The motivating example is again related to online advertising: in practice, it is not always possible to associate a conversion to the ad that produced it. To estimate the association between arms and expected rewards, the authors group rounds into long stretches: the agent consecutively plays the same action over these stretches. If the stretches are sufficiently long compared to the mean delay, the rewards received during one stretch will mostly come from the action played in that stretch. However, this algorithm heavily relies on the precise knowledge of the expected delay.

The authors of [58] further generalise the problem. In the setting they consider, the reward for pulling an arm can be spread over multiple time steps. This would be the most natural way to introduce delays in the model considered in chapter 3, if the advertiser is unable to identify incoming users and their corresponding conversions. The authors of [58] tackle the simpler, non-contextual setting. Moreover, as in [57], it is crucial for their algorithm that the expected delay is precisely known to the agent.

With this limitation in mind, the authors of [59] modify the algorithm proposed in [57] and [58]: the amount of rounds per stretch in which the same action is repeated over and over is not fixed, but adaptively increases with time.

4.1.5 Contextual bandits

Turning again to fully observable rewards, the first work after [47] that deals with (non-static) contextual bandits in the presence of delays is [60]. The authors consider the setting of generalised-linear contextual bandits (which include linear and logistic bandits as special cases). The authors significantly relax the assumptions in [47] regarding knowledge of the distribution of incoming contexts and constant delays: contexts are assumed sampled IID from a distribution which is unknown to the learner, and delays are stochastic. They propose a UCB-like algorithm, which depends only on those actions for which the feedback has already been observed. To obtain their regret bound, they design a delay-adaptive confidence parameter which depends on how many rewards are missing up to the current

time step. Moreover, for their analysis they do not assume a finite number of actions: their result can be extended to generalised-linear contextual bandits with infinitely many arms. They go on proposing an algorithm designed specifically for linear contextual bandits with delays, which attains a tighter regret bound, at the price of being usable only with a finite number of arms.

In the same setting, the authors of [61] both relax assumptions contained in [60] and obtain a tighter regret bound. Whereas in [60] the penalty due to delays grows with the horizon T, we have seen in other settings that we can hope for constant, additive penalties: the authors of [61] note, in this regard, that we can expect delays to become irrelevant once the learner has obtained a good enough estimate of the reward function. Indeed, they propose a UCB-like algorithm that attains a constant regret penalty that depends on the expected value of the delay distribution. Moreover, they are able to relax the assumptions on the tails of the delay distribution to cover so-called sub-exponential delays (which include the exponential distribution observed in [24]). The algorithm proposed in [60] both places stronger assumptions on these tails and needs some prior knowledge of the delay distribution, to calculate a number of rounds in which the agent explores randomly. A regularisation term akin to the one we saw in section 2.2 grants the authors of [61] the opportunity to avoid this costly exploration phase altogether. They back up their theoretical results with experiments on linear and logistic contextual bandits, with uniformly and exponentially distributed delays.

4.1.6 Intermediate observations

The authors of [42] consider the interesting case of non-stationary, delayed bandits with intermediate observations. Namely, as we have seen in the introduction to the present chapter, it is often the case that the agent has access to *immediate* proxies (like, for instance, clicks) besides the delayed feedback it aims to optimise. These intermediate signals are helpful if the environment is non-stationary, but we can assume that the long-term behavior of the system *given the intermediate signal* is stationary. This is the main assumption of [42]: the authors build upon a model introduced in [62], according to which the final feedback is independent of the agent's action given the intermediate feedback. Although the motivating example is again in the realm of digital advertising, this work does not consider partially observable feedback. This is not inconsistent with [55], since the conversion signal could be, for instance, usage of an advertised app after a given number of weeks (see also [63] for another practical example). Indeed, this way of measuring reward is similar to the approach used in [18] where, as we have seen above, the algorithm employs a fixed windows after which observations are discarded. Hence, they concentrate on fixed, deterministic delays. They deal with non-stationarity with a sliding-window approach. The disentangling between action and final reward given the intermediate signal is modelled via a Markov Decision Process (MDP), for which they build a UCB-like algorithm.

The presence of intermediate signals is assumed also in [40], whose authors tackle a somewhat simpler problem with respect to the one of chapter 3, namely,

that of optimising the budgets (but not the bids) of a portfolio of marketing campaigns (see subsection 3.3.2 for details). The problem of delayed conversions is dealt with substituting the missing rewards with *surrogate rewards*, which are produced as follows. A series of models are trained to predict the probability of conversion before fixed time-horizons (one per model). A (properly re-scaled) logistic regression is then trained on the predictions of the sub-models, to extrapolate to times other than the fixed time-horizons: the predictions of this meta-model are the surrogate rewards that are fed to the agent of this bandit problem. Although non-standard, this procedure effectively estimates the dependence on time of the CDF of rewards. As was the case for [19] and as we will see in [46], the Bayesian uncertainty on this estimated CDF is not taken into account when applying Thompson sampling downstream. Moreover, the use of logistic regression places a strong assumption on the distribution of delays (logistic distribution). Interestingly for practitioners, they comment on the need for *guardrails* in Thompson sampling, which restrict the exploration to the central quartiles of posterior distributions: while this could be detrimental purely in terms of regret, this procedure gives more stable next-action choices, which can be beneficial in a production environment and to foster trust from marketing and operations teams. Moreover, they deal with non-stationarity with an exponential penalty on old data, and they give estimates for the typical timescales involved.

The authors of [46] consider a related setting (though in the stationary regime) where the reward for the agent is a linear combination of intermediate, immediate signals and long-term feedback. They consider two sources of delay: one is natural (i.e., the time for the conversion to happen), and the other is due to batch training. As in [42], they only consider fully observable feedback. When natural delay places the feedback beyond the re-training time, the time for feedback is considered censored, and modelled with Survival Analysis methods (assuming an exponential distribution of delays). They then modify the UCB algorithm for the batch setting, and use the fitted distribution of delays to re-weight observed feedback. As in [19], they do not account for uncertainty on the distribution of delays. Note that, assuming daily re-training and weeks-long feedback time, this model cannot be applied to the setting we saw in chapter 3, as the overwhelming majority of data would be censored.

The reward model of [46] is taken to the extremes in [63]: the authors of this practitioners' paper consider a setting in which the reward is a linear combination of a number Δ of signals, that become progressively available during Δ rounds, where Δ is a fixed integer. As a motivating example, they consider maximising user engagement with a recommended podcast after 60 days, measured as the number of days in which users listen to said podcast. They model this *progressive feedback* as sampled from a multivariate Gaussian distribution, and corrupted by zero-mean Gaussian noise. After learning the hyperparameters for these Gaussian distributions from past-data, they can thus update the Bayesian posterior upon the long-term reward incorporating all information available to date. In this way, they are able to leverage TS in this recommendation problem.

4.1.7 Unrestricted delay distributions

With the exception of [45], [18] and [54], most of the works we have covered assume well-behaved delay distributions (bounded or sub-Gaussian), with at most sub-exponential tails [61] and well defined expected value; in some cases, delays are even deterministic. The authors of [64], on the contrary, place no restrictions on the shape of the distribution of delays: they do not assume it comes from any parametric family of distributions, they allow for infinite expectation and also infinite delays (i.e. the support of the distribution can be $\mathbb{N} \cup \{+\infty\}$). Moreover, as in [45], they allow the delay distribution to be arm-dependent. They place their work in the finite-armed, fully observable setting. Unlike most of the works above, they don't use a UCB-like algorithm: they use the straightforward extension of the Successive Elimination (SE) algorithm to the delayed case, which only considers observed feedback. SE, too, exploits the concept of confidence bounds we have seen in section 2.1. It maintains a set of active arms, where initially all arms are active. It pulls arms equally and, whenever there is a high confidence that an arm is sub-optimal, it eliminates it from the set of active arms. Unlike UCB, SE samples all active arms in a round-robin fashion, and not just the most promising arm: this resolves the redundant exploration problem which, as we saw above, UCB suffers in the presence of delays. Using SE, the authors obtain an additive penalty due to delays that depends on a quantile of the distribution of delays, which can be chosen at will (thus not assuming that the expectation value exists). Moreover, they do so without the need of knowing parameters of the delay distribution (as in [45]) or setting a window hyperparameter for discarding observations (as in [18]). In simulations with fixed delays, the authors compare SE with the UCB adaptation of [48]: as expected, SE is a clear winner. Interestingly, an approach to UCB with hallucinated rewards (as the one we saw in [23, 53, 54]) does not seem to exist in the literature for finite-armed bandits: we can only assume it would be a fairer competitor of SE. The authors also compare SE with the approach of [45] on α-Pareto distributions: they note that this competitor was designed for a more challenging setting (i.e., partially observable rewards), but it was the only algorithm with a regret bound on delay distributions with polynomial tail bounds and infinite expectation.

The authors also consider the more challenging case in which the delay distribution depends on the reward, even for fixed arm. No restriction is placed on the joint delay-reward distribution. In clinical trials this could happen if, given a treatment, some side effects take longer than others to surface. They first note that, if delays are *not* reward dependent, using only the observed rewards for inference yields an unbiased estimate, and this translates in the added regret due to the time to receive enough feedback. On the other hand, with reward-dependent delays, the observed empirical mean is a biased estimator of the expected reward; in the worst case, the direction of the bias can be opposite between different arms. To deal with the reward-dependent case, they widen the confidence bound of SE to account for this bias, and obtain an instance-dependent regret bound. As we will see in detail in section 4.2, also in the partially observable setting the distribution

of delays can be seen as reward-dependent; on the other hand, the authors of [64] place themselves in the fully observable regime, so this example does not apply to their work.

In view of the robust empirical performance of TS in the presence of delays [32, 52], the authors of [50] study its performance both theoretically and via simulation in the reward-independent setting of [64], namely: arm-dependent delay distributions with no restriction, for a finite-armed MAB. They derive for pure TS an instance-dependent regret bound that is similar to that of [64]. On the practical side, they test TS with a manifold of delay distributions.

- In the fixed delay setting, they compare it to the UCB adaptation of [48] and to the SE adaptation of [64]: TS significantly outperforms both (and the dominance of SE with respect to UCB asserted in [64] is confirmed).

- With α-Pareto distributed delays, the authors compare TS to SE, UCB and the algorithm of [45] (though the latter is designed for the partially observable setting). While also in this case TS outperforms all competitors, SE outperforms the algorithm of [45] only for small $\alpha = 0.2$ (the only setting that was tested in [64]). Moreover, interestingly UCB stably outperforms SE and the algorithm of [45] with this heavy-tailed distribution.

- Considering a packet-loss distribution (i.e. the feedback could be "lost" and have infinite delay with some probability), TS slightly outperforms UCB, and both outperform SE.

- Considering a geometric distribution, TS again outperforms all competitors. Interestingly, UCB has in this case the worst performance, also with respect to SE.

- A similar scenario happens for uniform delays.

All in all, the good performance of TS is confirmed in this variety of settings, while the relative performance of SE and UCB seems to defy intuition, and might warrant further in-depth analysis.

4.2 Goals and related work

As we have seen in the introduction to this chapter and in chapter 3, previous state-of-the-art approaches to bid/budget optimisation in performance marketing were grounded in the assumption of *immediate* rewards. While this assumption is reasonably verified in scenarios where the goal is to maximise clicks, owing to their occurrence within the same browsing session as the initial impression, it is likely violated when the goal is to maximise conversions.

Violations can manifest to varying extents, depending on the nature of the marketed product. For instance, experience suggests the majority of users deciding on the purchase of a low-cost consumer item tend to act on the same day as the

exposure to the advertisement and, in many cases, immediately thereafter. Conversely, this pattern does not hold true for higher-end goods such as automobiles or real estate properties: in these cases, the decision on whether to proceed can take several weeks.

As we have seen in section 4.1, there is a rich literature on delayed rewards in stochastic bandits. As highlighted in [18, 19, 45, 55] and detailed in section 4.1, the context of online marketing stands out, in that the feedback is only partially observable: while positive feedback, represented by purchase decisions, becomes accessible to the agent after a certain delay, negative feedback is never explicitly observed; in other words, the agent cannot distinguish between users who do not convert and those who have not converted yet, but will do so in the future.

The context of application of [40] is the closest to ours, since the authors deal with the optimisation of digital marketing campaigns in the presence of delayed rewards. However, some concerns limit its direct application. On the theoretical side, it is unclear what justifies substituting missing rewards with the predictions of a model (*surrogate rewards*), and what effect does this substitution have on the Thompson sampling procedure. In addition, training several models for fixed time-horizons, and then joining them with a scaled logistic regression seems *ad hoc*: the estimation of the probability of conversion over time is properly modeled as a Survival Analysis problem.

Crucially, as noted in section 4.1, this technique ignores the agent's uncertainty on the distribution of delays. If the experimenter has access to a rich dataset *before* the optimisation starts (as seems to be the case for [40]), then the distribution of delays can be assumed to be known precisely: this setting becomes similar to the one of [55], albeit in the contextual bandit case. In other words, rather than feeding surrogate rewards to the agent in place of missing ones, one can re-weight observations as in [55]. If, on the other hand, the distribution has to be estimated during the optimisation (as is the case for a pure bandit setting as the one considered here), ignoring this uncertainty could hinder exploration.

Additionally, the use of a scaled logistic function to extrapolate the estimated CDF of the distribution of rewards severely limits the shape this distribution can take: the logistic function is the CDF of the *logistic distribution*, which has exponential tails.

Finally, the technique proposed in [40] gets tested only on one setting (internal, undisclosed data of one company) and compared with a previous, undisclosed, algorithm in use in the same company. The figure of merit for comparison is non-standard, since the authors use the percentage improvement in CPA (as defined in section 3.1), rather than regret.

We will see in section 4.3.6, however, that surrogate rewards can be seen as a special case of the approach hereby proposed, when the distribution of delays is precisely known. Moreover, deriving this technique in a principled way shows how the delay distribution modelling problem can be tackled with the tools of Survival Analysis, without the need for *ad hoc* techniques or severe limitations on the shape of the distribution [65].

The setting of [18] is very close to ours: besides taking explicitly into account the

complex structure of partially observable rewards, the authors deal with contextual bandits (which, in turn, can host a continuous set of actions, as we have seen in section 2.3). However, some limitations hinder the direct application of this work to our setting:

Linear bandit Their approach requires carefully deriving Upper Confidence Bounds, and they do so in the linear case: it is nontrivial to extend their approach to the non-linear functions that link bid and budget to the number of clicks.

UCB Their approach involves a version of UCB to solve the exploration-exploitation dilemma, and it is nontrivial to integrate it with the Thompson sampling treatment presented in chapter 3. While they also present a sampling approach, it is an *ansatz* that uses a heuristic.

Discarded information Their approach ignores data about the magnitude of the delays. When available, this added information could increase performance.

Hyperparameter Their proposed method involves setting beforehand a number of rounds m beyond which the feedback is considered zero by default. While it is conceivable that such window parameter may be externally imposed for memory reasons, in all other cases it is not clear how it could be tuned, without previous knowledge of the distribution of delays.

Biased conversion rate Besides maximising the number of conversions, practitioners are often interested in the estimate of conversion rate itself (i.e., the probability of a conversion to happen). In [18], this estimate is biased by design.

Among other settings, the context of [64] presents some similarities with ours. The authors consider the case of *reward-dependent* delays, where realised delays may depend on the stochastic rewards. The partially observable rewards of online marketing can be in fact alternatively formulated in terms of a probability distribution $p(R, D)$ over rewards R and delays D. With a slight abuse of notation:

$$p(R, D) = p(D|R)p(R = 1) = \begin{cases} p(D|R = 1)p(R = 1), & \text{if } R = 1 \\ \delta(D - \infty)(1 - p(R = 1)), & \text{if } R = 0 \end{cases}$$

where $p(D|R = 1)$ is the distribution of delays for positive rewards, $\delta(D - \infty)$ is the Dirac delta concentrated at positive infinity, and $p(R = 1)$ is the probability of getting positive reward. However, as we have noted in section 4.1, they assume *full observability*, i.e., they assume that the agent receives both positive and negative feedback. If one tries and apply their proposed approach to this rather extreme case of delay-reward dependence, one obtains a degenerate model: the empirical mean of observed rewards is identically equal to one for all arms.

The authors of [64] and [50] stress the importance of being able to handle *unrestricted* delay distributions ($p(D|R = 1)$ in the above notation), meaning in particular long-tailed distributions. On the other hand, also [50] cannot be

readily applied to our setting, since the authors explicitly exclude reward-delay dependence, and pure TS was empirically proven to be suboptimal in the partially observable setting [19].

As we have seen in section 4.1, the authors of [56] investigate delayed rewards for bandits in the setting of vaccine trials: the structure of the feedback is similar to ours, since only negative feedback is observed (infections). However, they concentrate on being able to handle time-dependent risk: they do so at the expense of being able to model the dependence of risk on the time since exposure (to the vaccine in their case, to the advertisement in ours). In other words, they restrict the space of possible delay distributions. Moreover they too, as in [50, 64], work in the discrete case (multi-armed bandit with finite number of arms).

Considering all the above, I developed an approximate Thompson sampling approach to delayed partially observable rewards in contextual linear bandits. The reasons for this choice are the following:

- As commented in [50], oftentimes bandit approaches to delays require either modifying familiar MAB algorithms to account for delays or propose new delay-robust algorithms that are likely unfamiliar to practitioners; on the other hand, TS is popular among practitioners [19, 29, 40, 60, 63].
- As noted above, algorithms based on confidence bounds (either UCB or SE) require complex derivations of the optimal form of the confidence bound itself, while TS can work "out of the box".
- TS has shown good practical performance in a number of delayed feedback scenarios [32, 50, 52].
- TS is robust to non-standard information flows [50].
- As repeatedly noted in section 4.1, there is a sort of gap between theoretical regret bounds and empirical performance. While worst case rigorous bounds are useful, practitioners are often interested in average performance in "typical" scenarios [32, 60].

The closest approach to the one here proposed is treated in [19], whose authors propose a TS-like technique: starting from the MLE model of [24], they re-weight observations in view of the learned distribution of delays. After this is done, the problem is mapped to a standard Bernoulli bandit, and TS is carried out through a Beta-Bernoulli conjugate pair, as seen in section 2.1.3. These results cannot, however, be readily applied to the present setting:

- They consider finite-armed bandits with delays, while we aim to linear contextual bandits as a first proxy for the more complex contextual setting of chapter 3.
- While their model is derived for a general delay distribution, they then use an exponential distribution for experiments; we aim at handling a generic distribution, as in [50, 64], since in some practical cases the distribution of

delays is far from exponential [16, 18]. This can be done via a standard non-parametric estimator (the Kaplan-Meier estimator [34, 66]).

- As noted in section 4.1, they only consider (via a Beta posterior distribution) uncertainty on the distribution of rewards, while they ignore uncertainty on the learned distribution of delays. This could lead to insufficient exploration.

In the next section, we are going to cover the MLE model (first introduced in [24]) which powers my TS technique and the one in [19]. We will remain agnostic with respect to the space of actions, which means the model can be applied to standard multi-armed bandits with a finite set of actions as well as to linear and non-linear bandits.

4.3 Maximum Likelihood model for delayed conversions

The authors of [18] acknowledge that their proposed sampling approach does not strictly adhere to a conventional Thompson sampling methodology. Indeed, recall (section 2.2.2 and chapter 3) that TS involves sampling from a posterior distribution on problem instances. In their and our setting, the environment is described by two probability distributions (for delays and conversions respectively). Thus, the posterior would be a *distribution of distributions*. Even if the prior over delay distribution were independent from the prior over conversion distributions, independence would be lost in the posterior. For instance, the same data could be interpreted either as having already seen most conversions (belief peak on relatively small delays and low conversion probability) or as still missing most conversions (belief peak on high delays and conversion probability): the posterior would not factorise. We thus share their reservations regarding the possibility of efficiently computing such posterior distribution.

If we wanted to restrict ourselves to families of parametric distributions, we could in fact resort to Markov chain Monte Carlo (MCMC): all that is needed is a prior over parameters and a way of expressing the likelihood of data given the parameters. However, as hinted upon below, evaluating such likelihood is a nontrivial task for our problem, let alone feeding it to a MCMC algorithm. Moreover, as outlined in section 4.2, we need to be able to model general distributions, possibly with long tails and without assuming belonging to any parametric family of distributions. Furthermore, coupling such model with the MCMC model of [15] could prove nontrivial. Finally, with an eye to applications, it would be beneficial to leverage a more lightweight alternative to MCMC for carrying out posterior sampling [19, 40].

As proven by means of simulation in [15], Bootstrapped Thompson Sampling [17, 29] is a valid, albeit suboptimal, approximate alternative to exact posterior sampling. At its core, it involves calculating a Maximum Likelihood estimate of the parameters that describe the environment from randomly resampled data. In this section, we will hence describe in detail the technique introduced in [24] and [19] to derive a Maximum Likelihood Estimator for the coupled delay-reward

distribution. Maintaining the treatment general with respect to delay distributions and dependency on covariates, we will be able to encompass the results of both works, and then *extend* them to include the non-parametric Kaplan-Meier estimator.

In order to derive a Maximum Likelihood Estimator, we must first model how the dataset is generated. This is done in the next subsection.

4.3.1 Data generation model

Let us first describe the dataset that is fed to an agent that has to estimate the distribution of delays and rewards during the course of a marketing campaign. Each past interaction i with a user (i.e., each previous round) is associated to a context x_i that describes the action taken and, optionally, the user itself. Moreover, we assume the agent has access to the elapsed time δ_i since click until current round n. Finally, each past event is characterised by the following random quantities:

- A Boolean variable E_i, which indicates if a conversion was observed;
- A time T_i, which is the time between click and conversion if $E_i = 1$, and the time between click and current round n otherwise;
- A Boolean R_i, that tells us whether the user i does convert or not (irrespective of when);
- A time D_i, the delay between click and conversion (undefined if $R_i = 0$).

The variables R_i and D_i are *latent variables*: the agent observes only E_i and T_i. Only if $E_i = 1$ the agent is able to infer with certainty the value of R_i and D_i.

Given a context x_i, we assume that the conversion variable R_i is drawn from some distribution $p(R_i|x_i)$. If $R_i = 1$, then also the delay variable D_i is generated, according to $p(D_i|x_i, R_i = 1)$. Finally, the variables E_i and T_i are just deterministic functions of R_i, D_i and of the elapsed time δ_i:

$$E_i = \begin{cases} 0 & \text{if } R_i = 0 \\ \mathbb{1}(\delta_i \geq D_i) & \text{if } R_i = 1 \end{cases} \qquad T_i = \begin{cases} \delta_i & \text{if } R_i = 0 \\ \min(D_i, \delta_i) & \text{if } R_i = 1 \end{cases}$$

Given this data generation model, if the agent were to know the actual value of R_i for all $i = 1, \ldots, n$, the estimation problem for the two distributions of conversions and delays would decouple and be solvable with standard methods. Indeed, the agent could estimate $p(R_i|x_i)$ with the methods outlined in section 2.2, and it could estimate $p(D_i|x_i, R_i = 1)$ applying standard Survival Analysis techniques on those rows of the dataset such that $R_i = 1$.

Another point worth adding is that, even if the agent does not know the value of R_i, if the delay distribution $p(D_i|x_i, R_i = 1)$ is instead known it can be used to calculate an unbiased estimator for $p(R_i|x_i)$ (and this is precisely what the authors of [55] do in the finite-armed bandit setting). In turn, if the conversion probability $p(R_i|x_i)$ is known, it can be used to estimate the delay distribution.

These observations point to the Expectation Maximisation technique as a way to iteratively solve the estimation problem, refining the estimate of the two distributions at each iteration. This is what we will do in the next subsection.

4.3.2 Expectation-Maximisation technique

To fix the notation, let us collectively call ϑ the parameters that characterise the distribution of rewards and delays $p(R_i, D_i|x_i, \vartheta)$. Our goal is estimating ϑ given the available data.

Regarding the distribution of delays, we are not making here any assumption about the shape of ϑ: it could be anything from the rate parameter of an exponential distribution (as is done in [19, 24]) to the list of discrete hazards of a non parametric Kaplan-Meier estimator: we leave it unspecified, as this technique can be used "plug and play" with the estimator of choice, depending on the experimenter's belief (or lack thereof) on the shape of the delay distribution.

For brevity, given the n events up to the current round, we use the vector of conversions

$$\mathbf{R} = (R_1, \dots, R_n)$$

and similarly for the other variables. Moreover, we define the log-likelihood of the data (both observed and latent) given parameter ϑ:

$$l(\mathbf{E}, \mathbf{T}, \mathbf{R}|\mathbf{x}, \vartheta) = \log p(\mathbf{E}, \mathbf{T}, \mathbf{R}|\mathbf{x}, \vartheta).$$

Here we did not consider also the vector of delays $\mathbf{D}$ because, as remarked in the previous subsection, knowing $\mathbf{R}$ would be sufficient for our estimation problem. The following derivation would also work considering $\mathbf{D}$ among the latent variables, at the price of a more cumbersome notation.

The remark of the previous subsection can be thus formalised: if we had access also to latent variables, to estimate ϑ we could simply maximise the likelihood,

$$\hat{\vartheta} = \operatorname*{argmax}_{\vartheta} l(\mathbf{E}, \mathbf{T}, \mathbf{R}|\mathbf{x}, \vartheta).$$

Since we do *not* know $\mathbf{R}$, intuitively we can average the log-likelihood over its possible values $\mathbf{r}$, weighting each contribution by the posterior probability $p(\mathbf{R} = \mathbf{r}|\mathbf{x}, \vartheta, \mathbf{E}, \mathbf{T})$ given the observed data: in a sense, we would be considering "all possible worlds", and each would contribute to the log-likelihood proportionally to its posterior probability. However, calculating such posterior probabilities would in turn require already knowing ϑ. The way out of this circle is starting from a rough estimate ϑ_0 of ϑ, and iteratively refine this estimate:

$$\hat{\vartheta}_{k+1} = \operatorname*{argmax}_{\vartheta} \mathbb{E}_{\mathbf{R}\sim p(\cdot|\mathbf{x},\vartheta_k,\mathbf{E},\mathbf{T})}\, l(\mathbf{E}, \mathbf{T}, \mathbf{R}|\mathbf{x}, \vartheta). \tag{4.1}$$

The one described above is the Expectation-Maximisation (EM) technique [67], and one can indeed prove that, as k grows, it converges to a local maximum of the log-likelihood. In other words, if we are able to calculate the right hand side of (4.1)

$$Q_k(\vartheta) = \mathbb{E}_{\mathbf{R}\sim p(\cdot|\mathbf{x},\vartheta_k,\mathbf{E},\mathbf{T})}\, l(\mathbf{E}, \mathbf{T}, \mathbf{R}|\mathbf{x}, \vartheta) \tag{4.2}$$

we have gained a MLE, which we will later use in Bootstrapped Thompson sampling.

Derivation is traditionally divided in two parts: the *expectation step*, in which we calculate $p(\cdot|\mathbf{x}, \vartheta_k, \mathbf{E}, \mathbf{T})$ and the *maximisation step*, in which we calculate the quantity $Q_k(\vartheta)$ that gets maximised .

Before that, we show that we can work on single-round quantities separately, rather than on vectors, owing to the fact that the random variables at play are IID.

4.3.3 Round decoupling

We will now show that, as in standard log-likelihood maximisation, also in EM we can rewrite the quantity $Q_k(\vartheta)$ we want to maximise as a sum of terms, each of which refers to just one event.

Indeed, assuming IID observations, the full log-likelihood can be expanded as a sum,

$$l(\mathbf{E}, \mathbf{T}, \mathbf{R}|\mathbf{x}, \vartheta) = \sum_{i=1}^{n} l(E_i, T_i, R_i|x_i, \vartheta),$$

while the posterior probability of the latent $\mathbf{R}$ factorises as follows:

$$p(\mathbf{R}|\mathbf{x}, \vartheta_k, \mathbf{E}, \mathbf{T}) = \prod_{i=1}^{n} p(R_i|x_i, \vartheta_k, E_i, T_i).$$

Dropping the dependency on x_i, ϑ and ϑ_k for simplicity, (4.2) then becomes:

$$Q_k = \sum_{R_1 \in \{0,1\}} \cdots \sum_{R_n \in \{0,1\}} \sum_i l(E_i, T_i, R_i) \prod_j p(R_j|E_j, T_j).$$

Summing over the values of $R_1, \ldots, R_N$ means that, for every $i = 1, \ldots, n$, only the factor with $j = i$ survives, while all the others get summed to one:

$$Q_k = \sum_{i=1}^{n} \sum_{R_i \in \{0,1\}} l(E_i, T_i, R_i)\, p(R_i|E_i, T_i) = \sum_{i=1}^{n} Q_{ki} \tag{4.3}$$

In other words, Q_k can be expressed in terms of single-round quantities Q_{ki}: this is actually a general property of EM for IID variables.

In the next subsection we will calculate $p(R|x, \vartheta_k, E, T)$, while in 4.3.5 we will derive $l(E, T, R|x, \vartheta)$ and wrap up.

4.3.4 Expectation step

The problem of calculating the posterior probability $p(R|x, \vartheta_k, E, T)$ can be carried out mechanically, splitting it into cases. Since x and ϑ_k are fixed throughout the calculation, we will temporarily drop them for simplicity. We start from the case $R = 1, E = 1$: in this scenario, the user has not converted yet ($E = 0$), but they eventually will ($R = 1$). Using Bayes' theorem, we can write:

$$p(R = 1|E = 0, T) = \frac{p(E = 0, T|R = 1)p(R = 1)}{p(E = 0, T)}.$$

Not having observed yet the conversion of a user that will convert is equivalent to stating that the conversion event will happen in the future of T, i.e.

$$p(E = 0, T|R = 1) = S(T),$$

where S is the *survival function* of the distribution of delays (i.e., the complement to 1 of its CDF). For brevity we call p_1 the probability of generating reward,

$$p_1 = p(R = 1),$$

and we call $Z(T)$ the normalisation factor

$$Z(T) = p(E = 0, T).$$

We now switch to the case $R = 0$ and $E = 0$. Again resorting to Bayes' theorem,

$$p(R = 0|E = 0, T) = \frac{p(E = 0, T|R = 0)p(R = 0)}{p(E = 0, T)}.$$

Note that the probability of not having observed reward at time T for a user that does not yield reward at all is simply 1:

$$p(E = 0, T|R = 0) = 1.$$

Moreover, the probability of not generating reward is the complement of p_1:

$$p_0 = p(R = 0) = 1 - p_1.$$

Finally, the denominator is $Z(T)$ also in this case. We can then calculate $Z(T)$ requiring that probabilities sum to 1:

$$1 = p(R = 1|E = 0, T) + p(R = 0|E = 0, T) = \frac{S(T)p_1 + p_0}{Z(T)},$$

so that

$$Z(T) = S(T)p_1 + p_0.$$

As a last case, we now need to consider $E = 1$. Since having observed a reward naturally implies that the user yields one, we simply have:

$$p(R|E = 1, T) = \delta_{R,1} = R.$$

Summing up:

$$p(R|E, T) = \begin{cases} R & \text{if } E = 1 \\ \frac{p_1 S(T)}{p_0 + p_1 S(T)} & \text{if } E = 0, R = 1 \\ \frac{p_0}{p_0 + p_1 S(T)} & \text{if } E = 0, R = 0 \end{cases}$$

This expression can be further simplified, using the Boolean variables R and E as indicators:

$$p(R|E, T) = E\,R + (1 - E)R\frac{p_1 S(T)}{p_0 + p_1 S(T)} + (1 - E)(1 - R)\frac{p_0}{p_0 + p_1 S(T)}.$$

Note that in the right hand side of this expression, all probabilities depend on the context x and on ϑ_k:

$$p_1 = p_1(x, \vartheta_k), \quad S(T) = S(T|x, \vartheta_k).$$

We are here leaving open the possibility that also the distribution of delays is context dependent (as in [19, 45, 50, 64]).

In the next subsection, we turn to calculating the log-likelihood factor $l(E, T, R|x, \vartheta)$ of expansion (4.3).

4.3.5 Maximisation step

Again dropping x and ϑ for simplicity, to calculate $l(E, T, R)$ we start from the likelihood itself; proceeding along the same lines as above we find:

$$p(E, T, R) = p(E, T|R)p(R) = \begin{cases} p_0(1-E) & \text{if } R = 0, \forall T \\ S(T)p_1 & \text{if } R = 1, E = 0\,, \\ f(T)p_1 & \text{if } R = 1, E = 1 \end{cases}$$

where we denote by f the probability density function of delays, and S is again the survival function of delays. Switching to logarithms, we get:

$$l(E, T, R) = \begin{cases} \log p_0 + \log(1-E) & \text{if } R = 0, \forall T \\ \log p_1 + \log S(T) & \text{if } R = 1, E = 0 \\ \log p_1 + \log f(T) & \text{if } R = 1, E = 1. \end{cases}$$

Note that the term $\log(1-E)$ would be infinite for $E = 1$; however, since it appears only for $E = 0$, we will never observe such case in actual data: with a slight abuse of notation, we will drop this term from now on. In the same fashion as above, we can then express the log-likelihoods as

$$l(E, T, R) = R \log p_1 + (1-R) \log(1-p_1) + R(1-E) \log S(T) + RE \log f(T).$$

As customary in Survival Analysis, we switch to the *hazard rate* $\lambda(t) = \frac{f(t)}{S(t)}$. In this way, we can re-express the log-likelihood in a more familiar way:

$$l(E, T, R) = \underbrace{R \log p_1 + (1-R) \log(1-p_1)}_{\text{classifier log-likelihood}} + R \underbrace{\left(E \log \lambda(T) + \log S(T)\right)}_{\text{survival log-likelihood}}. \tag{4.4}$$

We have here highlighted a component which is simply a binary cross-entropy (i.e., the log-likelihood of a binary classifier), and another which is the log-likelihood of a Survival Analysis problem [34], in which the task is to estimate the distribution of delays and time data are subjected to censorship (for $E = 0$). This second component is multiplied by a factor R, that effectively selects only those data-points for which the notion of delay is meaningful (i.e., users that will convert).

Here, as in subsection 4.3.4, all probabilities on the right hand side of (4.4) depend on the context x. Differently from subsection 4.3.4, however, they depend also on ϑ, rather than the previous estimate ϑ_k.

Inserting our results in (4.3), we come to:

$$Q_{ki} = \sum_{R_i \in \{0,1\}} [R_i \log p_1 + (1 - R_i) \log(1 - p_1) + R_i(E_i \log \lambda(T_i) + \log S(T_i))] \times \left[E_i R_i + (1 - E_i) R_i \frac{p_1^k S^k(T_i)}{p_0^k + p_1^k S^k(T_i)} + (1 - E_i)(1 - R_i) \frac{p_0^k}{p_0^k + p_1^k S^k(T_i)} \right].$$

We have here denoted the quantities that depend on ϑ_k with a k superscript; the dependency on x_i is left implicit.

We are now able to carry out the summation over R_i: this is desirable, since we want our summands to depend only on observed quantities. The expression can be simplified isolating the probabilities that depend on the k-th estimate of ϑ, defining

$$q_1^k(T) = \frac{p_1^k S^k(T)}{p_0^k + p_1^k S^k(T)}, \qquad q_0^k(T) = 1 - q_1^k(T). \tag{4.5}$$

These are the probabilities (according to ϑ_k) that a user that has not yet converted will do so in the future and that they will never convert, respectively. We then come to:

$$Q_{ki} = \log(1 - p_1)(1 - E_i) q_0^k(T_i) + [\log p_1 + (E_i \log \lambda(T_i) + \log S(T_i))] \left[E_i + (1 - E_i) q_1^k(T_i) \right].$$

Finally, carrying out the product, we are able to re-express it as:

$$Q_{ki} = \underbrace{E_i \log f(T_i) + (1 - E_i) q_1^k(T_i) \log S(T_i)}_{\text{weighted survival log-likelihood}} + \underbrace{\log p_1 \left[E_i + (1 - E_i) q_1^k(T_i) \right] + \log(1 - p_1)(1 - E_i) q_0^k(T_i)}_{\text{weighted classifier log-likelihood}}. \tag{4.6}$$

We have thus proven that the quantity we want to maximise at step k is composed of two terms: one has the shape of a weighted classifier log-likelihood, while the other is the weighted log-likelihood of a Survival Analysis problem. Note that the two log-likelihoods are decoupled: we can maximise the former with respect to p_1 (our next estimate of the probability of conversion), and the latter with respect to $S(t)$ (our next estimate of the survival function of delays). In other words, any software that is able to handle weighted data either for binary classification or for Survival Analysis can be employed to solve the k-th step of the Expectation Maximisation algorithm.

We also note that the two weighted log-likelihoods have a transparent interpretation. In the survival one, users which still have to convert have a weight which expresses our current (at k-th step) belief that they will convert in the future; on the other hand, users that already converted enter fully in the estimation of the distribution of delays. In a similar fashion, in the binary cross-entropy, users that

have not converted contribute partially to the component with positive label, and partially to the component with zero label.

Finally, we again remark that we made no assumptions regarding the dependency of p_1 and $S(t)$ on covariates x. As such, depending on the problem at hand, this formulation can handle univariate models (i.e., no covariates are included), the division of the population into groups (with multiple univariate models, one for each group), and regression models, both linear and non-linear.

4.3.6 Fitting the model

In this subsection we are going to describe in detail how the two log-likelihoods on the right hand side of (4.6) can be maximised given a dataset, hence showing how the k-th step of EM can be carried out.

We start from the weighted classifier log-likelihood. First, we define the weight:

$$w_i = E_i + (1 - E_i) q_1^k(T_i). \tag{4.7}$$

Then, summing over the row index i, we get the familiar form

$$L_{\text{class}}(p_1) = \sum_{i=1}^{n} \{ w_i \log p_1 + (1 - w_i) \log(1 - p_1) \}. \tag{4.8}$$

If p_1 does *not* depend on a context x (e.g., because we are calculating the probability associated with a given arm in a vanilla MAB, as in [19]), this expression is simply maximised by the average of weights:

$$p_1 = \frac{1}{n} \sum_{i=1}^{n} w_i. \tag{4.9}$$

If, like in our case, p_1 is a function of the context, $p_1 = p_1(x)$, we need to solve a supervised learning problem. For this, however, we can rely on any supervised learning library that supports weighted samples, without the need of carrying out the log-likelihood maximisation explicitly: we just need to build the dataset appropriately. To see this, we rewrite the score $L_{\text{class}} = \sum_i l_i$:

$$l_i = \begin{cases} \log p_1(x_i) & \text{if } E_i = 1 \\ q_1^k(T_i) \log p_1(x_i) + q_0^k(T_i) \log(1 - p_1(x_i)) & \text{if } E_i = 0 \end{cases} \tag{4.10}$$

We then see that it is *as if* for every row such that $E_i = 1$ there is one row with label 1 and weight 1, and for every row such that $E_i = 0$ there are *two* rows: one with label 1 and weight $q_1^k(T_i)$, and the other with label 0 and weight $1 - q_1^k(T_i)$. We thus simply need to duplicate the unobserved rows and assign weights appropriately, and then we can exploit the supervised learning algorithm of our choice.

Formulas (4.9) and (4.7) point us to a way to recover the *surrogate rewards* technique of [40]. First note that, if the model for the distribution of delays is trained previously on a separate dataset and is sufficiently reliable, the Expectation Maximisation technique becomes trivial: assuming the distribution of delays is

known, one can use it in place of $q_1^{k=0}$, and only one iteration would be needed to estimate $p_1(x)$, maximising the classifier score L_{class} defined in 4.8. If, in place of the log-likelihood, one employs Brier score, which is another proper scoring rule, to estimate p_1, the quantity to be maximised becomes:

$$L_{\text{Brier}}(p_1) = -\sum_{i=1}^{n} (w_i - p_1(x_i))^2 .$$

This scoring rule avoids the need for duplicating some rows of the dataset, since the problem is transformed from training a classifier to training a regressor: Brier loss is nothing other than the well-known Mean Square Error loss. This means that labels do not need to be restricted to 0 or 1, but can be continuous. Given the definition (4.7), we see that w_i can be seen as a surrogate reward in this case: if the reward is known, the true value is used; if the reward is not known, the predicted probability of conversion is used instead. Note, however, that the authors of [19] assert that the approach of [55] is more stable than maximising the likelihood 4.8 in a similar setting.

We now turn to the weighted survival log-likelihood of (4.6). Here we do not need to transform the dataset in any way: once we choose a Survival Analysis model for $S(T)$ we just need to feed it with the dataset as-is, with weights w_i defined in (4.7):

$$L_{\text{surv}} = \sum_{i=1}^{n} w_i \big(E_i \log f(T_i) + (1 - E_i) \log S(T_i)\big). \tag{4.11}$$

In particular, in section 4.4 we will use the Kaplan-Meier estimator which, being nonparametric, is agnostic with respect to the shape of the delay distribution.

To wrap up, the whole Expectation Maximisation algorithm for delayed conversions is reported in Algorithm 3. The termination condition is not specified on purpose: it can be set as reaching a pre-determined number of iterations, or it can entail checking whether the change from ϑ_k to ϑ_{k+1} is below some threshold.

4.4 Proposed algorithm and results

4.4.1 Bootstrapped Thompson Sampling for delayed conversions

After having introduced, in section 4.3, the data generating process and an appropriate Maximum Likelihood method for handling partially observable delayed rewards, we will now cover the bandit setting. First, we will briefly describe how the MLE is applied to bandits, and then we will see the results of extensive simulations, that show how this technique improves over the state of the art.

As we have seen in chapter 3, Bootstrapped Thompson Sampling (BTS) [68] is an effective technique for approximating sampling from the posterior distribution, as required by TS. A number of approaches to BTS exist (see for instance [29] and the references therein). Again with an eye to practitioners, a version was chosen for the present work, which does not require further calculations besides the MLE,

Algorithm 3 Expectation Maximisation for delayed conversions

Input: Dataset of contexts x_i, times T_i and observation variables E_i as in subsection 4.3.1, for $i = 1, \dots, n$. First guess ϑ_0.

Iteration index $k = 0$
while True **do**
  for $i = 1, \dots, n$ **do**
    Calculate survival function estimate $S^k(T_i|x_i) = S(T_i|x_i, \vartheta_k)$
    Calculate conversion rate $p_1^k(x_i) = p_1(x_i, \vartheta_k)$
    Calculate probability $q_1^k(T|x_i)$ as in (4.5)
    Calculate weights w_i as in (4.7)
  end for
  Maximise weighted survival log-likelihood (4.11)
  Duplicate un-observed rows of dataset as described in subsection 4.3.6
  Maximise weighted classifier log-likelihood $L_{\text{class}} = \sum_i l_i$ of (4.10)
  $k \leftarrow k + 1$
  Concatenate estimated survival and classifier parameters into ϑ_k
  if Termination condition is reached **then**
    Break
  end if
end while

and is closest in spirit to the well known Statistical Bootstrap method [69, 70]: the approach we will follow is an extension to our setting of the one presented in [17] for the Bernoulli bandit.

Given a Maximum Likelihood Estimator (like the one we saw in section 4.3) and a dataset, the Statistical Bootstrap yields an estimate of the uncertainty on the estimated quantities (in our case, the probability function $p_1(x)$ and the survival function $S(t)$) by resampling the dataset with replacement many times, and applying the MLE to the resampled data. The main idea behind BTS is sampling from the outcoming distribution of Maximum Likelihood estimates, *as if* they represented the posterior distribution. Indeed, one can show [70] that the bootstrapped distribution approximates the posterior given a non-informative prior.

However, for this to work, the empirical distribution function of the observed data should approximate reasonably well the population distribution: this assumption breaks down when the size of the dataset is very small. The problem is that, in a bandit problem, in the first rounds this is precisely what happens. Moreover, as remarked in chapter 2, the learning agent is in charge of the exploration: underestimating the uncertainty in the first few rounds could mean exploring insufficiently, and exploiting suboptimal actions.

This was proven in [17] for the Bernoulli bandit: blindly applying the Statistical Bootstrap to approximate the posterior of TS leads to a regret which, on average, grows linearly with the horizon T. In essence, this means that the agents never gets to learn the reward function well enough that it can safely choose the next action.

Luckily, in that work there is also a simple heuristic solution: enrich the history of played arms and rewards with an *artificial history*, generated from a distribution which can be thought of as a "prior" of sorts. The way these artificial data points are generated is problem specific, so we will now go through the experimental setting.

Since, as seen in section 4.2, the only state-of-the-art competitors for the proposed algorithm are OTFLinUCB and OTFLinTS of [18], for a fair comparison I adopted the same data generation mechanism. The environment is described by a vector $\vartheta \in \mathbb{R}^d$, with $\|\vartheta\|_2 \leq 1$. At every round, the agent receives from the environment a set of K contexts $x_1, \dots, x_K$. For each context index $i = 1, \dots, K$, the scalar product with ϑ belongs to the unit interval: $x_i \cdot \vartheta \in [0, 1]$. Moreover, also the context norms are bounded: $\|x_i\|_2 \leq 1$. The reward is then sampled from a Bernoulli distribution with mean $x_i \cdot \vartheta$.

In their experiments, the authors of [18] choose $d = 5$ and $K = 10$. Moreover, the environment vector ϑ is fixed at $\vartheta = (1/\sqrt{d}, \dots, 1/\sqrt{d})$. Finally, the contexts are sampled independently at each round from $[0, 1]^d$ and then normalised.

As for the delays, in [18] two distributions were tested: a geometric distribution with varying mean, and an empirical distribution fitted with a Gaussian kernel on the dataset released in [16]. Regarding these real data, in the code accompanying [18] the authors concluded that the delay distribution does not significantly depend on the context, and we will make the same assumption here too. I extended this set of distributions to include the IID distributions considered in [50] to test pure TS in the presence of (fully observable) delays, namely: constant, deterministic delays, uniformly distributed delays over some interval, α-Pareto distribution and packet-loss distribution.

In this setting, the following mechanism for generating the artificial history was used across all experiments. For every round, n_{prior} data points were generated. Since these prior-like points have the only goal of making the learner aware that the observed data may not represent the whole population, n_{prior} was kept way smaller than the horizon T, which is greater than 1000 rounds in all experiments: I chose $n_{\text{prior}} = 10$ across all experiments.

Given the assumptions on the environment ϑ and the contexts x_i above, the following artificial history generation process was deemed natural: for each round, and for each $j = 1, \dots, n_{\text{prior}}$, both a ϑ_j and a x_j were generated uniformly over $[0, 1]^d$ and then normalised. The reward was then drawn from a Bernoulli distribution with mean $\vartheta_j \cdot x_j$. Finally, delays were sampled uniformly over $[0, D_{\max}]$, for a $D_{\max}$ one order of magnitude smaller than the horizon. Since in all experiments the BTS algorithm was compared with OTFLinUCB and OTFLinTS, both of which require a time parameter m, it seemed natural to fix $D_{\max} = m$. It must be stressed, however, that the parameter m for the OTFLinUCB and OTFLinTS is an integral part of the algorithm and, if it is not externally imposed on the algorithm for memory reasons, it should be tuned accurately depending on the expected delay distribution: we will see below that the performance of these two algorithms is heavily dependent on its value. On the other hand, BTS was found to be roughly independent from the value of $D_{\max}$.

We can thus recap the BTS algorithm at round n:

- The agent receives a dataset of n rows, where each row contains the observable information explained in subsection 4.3.1;
- The agent draws n_{prior} artificial data points, according to the procedure just described;
- The artificial and real data points are merged to form a unique dataset of size $(n + n_{\text{prior}})$;
- The agent samples the entire dataset once with replacement;
- The model described in section 4.3 is fit on this sampled dataset, yielding an estimate $\hat{S}(t)$ and $\hat{p}_1(x)$;
- The agent plays the action k such that, among $x_1, \dots, x_K$, the context x_k maximises the estimated probability $\hat{p}_1(x)$.

We must remark that the model described in 4.3 suffers from the identifiability issue explained in [45]. Namely, two problem instances can produce the same data but have strictly different parameters. As an example, consider problem instance $\mathcal{I}_1$ with, at round t_1, $S(t_1) = 80\%$ (i.e., 80% of conversions happen after t_1) and $p_1 = 90\%$. Consider then problem instance $\mathcal{I}_2$ with $S(t_1) = 10\%$ and $p_1 = 20\%$. At t_1, despite having very different parameters, these two instances produce exactly the same data, as the probability of observing a reward before t_1 is given by the product between p_1 and the CDF of delays $(1 - S(t_1))$.

Hence, the MLE estimator described above could either output $\mathcal{I}_1$, $\mathcal{I}_2$ or any other instance which is compatible with the observed data. Nevertheless, the *product* of the predicted conversion probability $\hat{p}_1(x)$ and the predicted CDF $(1 - \hat{S}(t))$ is the same for all these instances: the estimated probability of converting before a certain time is well-identified. This means that

- The next action should be selected on the basis of this product of probabilities. However, since we are dealing with distributions of delays which do not depend on context, maximising the product or just $p_1(x)$ yields the same result. Extra care should be taken if delays depend on the context.
- If one (as in [19]) is interested in the actual value of the conversion rate (besides its use for the optimisation algorithm), this should be intended as *conversion probability before a given time*.

The proposed algorithm is described in detail in Algorithm 4.

4.4.2 Simulation results

In what follows we will go through the results of the simulations. Two settings have been treated separately. In one, the windowing parameter m is externally imposed on the algorithm: if delay exceeds m rounds, the agent never receives

Algorithm 4 Bootstrapped Thompson Sampling for delayed conversions

Input: n_{prior}, $D_{\max}$, T, d, K.

Data $D_0 = ()$
for $n = 1, \dots, T$ **do**
  Update data D_n with observed conversions
  for $j = 1, \dots, n_{\text{prior}}$ **do**
    Sample prior ϑ_j and x_j uniformly over $[0, 1]^d$ and normalise them
    Sample prior reward from Bernoulli$(\vartheta_j \cdot x_j)$
    Sample delays uniformly over $[0, D_{\max}]$
  end for
  Concatenate n_{prior} times and rewards with dataset D_n
  Sample with replacement $n + n_{\text{prior}}$ data points from concatenated dataset
  Estimate $\hat{S}(t, x)$ and $\hat{p}_1(x)$ from sampled dataset via Algorithm 4
  Observe current contexts $x_1, \dots, x_K$
  for $i = 1, \dots, K$ **do**
    Calculate probability $\hat{S}(T, x_i)\hat{p}_1(x_i)$
  end for
  Select arm $\text{argmax}_i \hat{S}(T, x_i)\hat{p}_1(x_i)$
end for

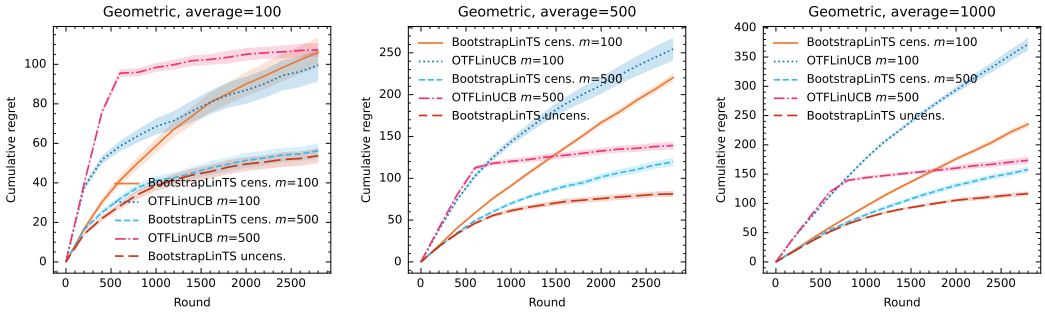
feedback; we call this setting *censored*. In the other setting, m is just a specific of the algorithm for OTFLinUCB and OTFLinTS, and the proposed BTS is free to use all past data: we call this setting *uncensored*. We will see that, as expected, censoring damages the performance of BTS and, among censored variants, the lower m is, the higher the regret. On the other hand, the effect of m on the algorithms of [18] is harder to predict.

For every setting we will cover, BTS is among the best performing algorithms, while OTFLinTS is among the worst. This is a reminder that it is not just the act of sampling from a distribution, but also the details of how the distribution is built, that make TS an effective technique. For this reason and to avoid clutter, OTFLinTS will not be shown in the following, and BTS will be compared to OTFLinUCB alone.

In the plots, we will show the average regret suffered by each algorithm over the course of 20 simulations, together with the standard deviation of the mean.

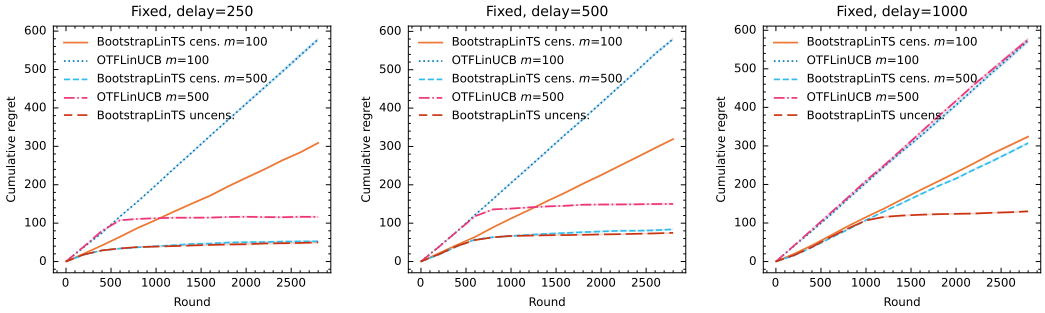
Geometric distribution

The first distribution we will consider is the geometric distribution with varying average delay. In all three cases, we see from figure 4.1 that BTS, either uncensored or censored with $m = 500$, performs best. On the other hand, due to the heavy censoring, BTS with $m = 100$ incurs higher regret. Nevertheless, when the average delay equals 100 rounds, its regret is lower with respect to both instances of OTFLinUCB for half of the rounds, and is comparable at the horizon $T = 3000$. In the other two cases, it behaves significantly better than the instance of OTFLinUCB



(a) *Average delay* = 100 *rounds* **(b)** *Average delay* = 500 *rounds* **(c)** *Average delay* = 1000 *rounds*

Figure 4.1: Average cumulative regret suffered by the examined algorithms when delays distribute according to a geometric distribution.



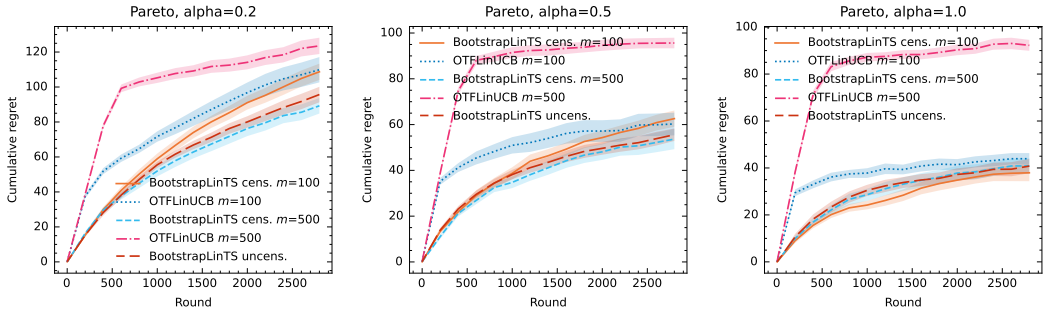
(a) *Fixed delay* = 100 *rounds* **(b)** *Fixed delay* = 500 *rounds* **(c)** *Fixed delay* = 1000 *rounds*

Figure 4.2: Average cumulative regret suffered by the examined algorithms when delays are fixed.

that has access to the same amount of information. Oddly enough, OTFLinUCB with $m = 100$ behaves better, when the average delay equals 100 rounds, with respect to OTFLinUCB with $m = 500$, despite having access to less information: this is due to the way m enters the algorithm. We can thus conclude that, if m is not externally imposed, its choice is non trivial for OTFLinUCB. On the other hand, it is straightforward for BTS: the higher, the better; if at all possible, it is even better not to censor the feedback.

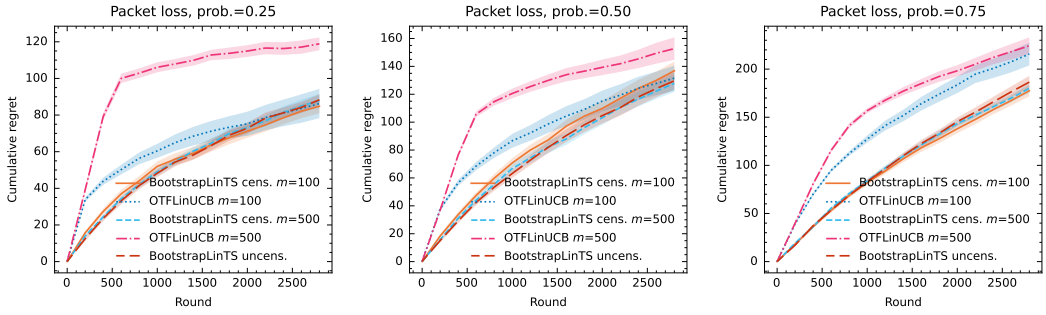
Fixed delays

Examining fixed delays in figure 4.2, we see that, whenever the censoring time is lower than the delay, the regret grows linearly: this is of course expected, as the agent is completely blind to feedback. Among the other algorithm instances, we see that BTS reaches significantly lower regret.



(a) $\alpha = 0.2$ **(b)** $\alpha = 0.5$ **(c)** $\alpha = 1.0$

Figure 4.3: Average cumulative regret suffered by the examined algorithms when delays are distributed according to an α-Pareto distribution, with varying α.



(a) $p_{\text{lost}} = 0.25$ **(b)** $p_{\text{lost}} = 0.50$ **(c)** $p_{\text{lost}} = 0.75$

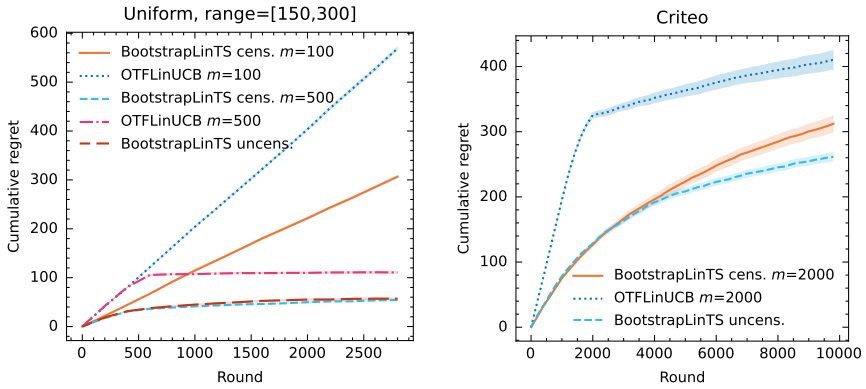
Figure 4.4: Average cumulative regret suffered by the examined algorithms when delays are distributed according to a packet loss distribution, with varying probability p of losing the packet (i.e. of having infinite delay).

α-Pareto distribution

The α-Pareto distribution presents polynomial tails: the smaller the parameter α is, the heavier is the tail. This is reflected in figure 4.3, where regret is generally higher for $\alpha = 0.2$: learning takes longer. Besides this, again we see that BTS performs best for all the examined values of α. For $\alpha = 0.2$ and $\alpha = 0.5$, the heavily censored instance of BTS is slightly worse than the others, as expected. On the other hand, OTFLinUCB with $m = 500$ incurs much higher regret with respect to the other algorithms (even if it suffers a lower degree of censorship with respect to $m = 100$).

Packet loss

By "packet loss", we refer to a scenario in which feedback can be either delivered immediately (i.e., with zero delay) or "lost" (i.e., it has infinite delay). In particular, it is lost with probability p (we employ a definition which is opposite to that in [50, 64], as our p is their $1 - p$). From figure 4.3, we see that again the instances of



(a) *Uniform distribution over* $[150, 300]$ **(b)** *Criteo*

Figure 4.5: Average cumulative regret suffered by the examined algorithms when delays are distributed according to a uniform distribution and according to the dataset [16] of the Criteo advertising platform.

BTS behave generally better than OTFLinUCB, and in particular the instance of OTFLinUCB with $m = 500$ performs significantly worse. We also note that, for high probability p of lost packet, all algorithms are still learning upon reaching the horizon $T = 3000$: at $p = 0.75$ only one in four events produce observable feedback. Likely, in this setting a different data generating mechanism with respect to that of section 4.3, that better captures this scenario, could be employed with better results.

Uniform distribution

In figure 4.5a, we can see the effect of uniformly distributed delays: the result is very similar to that of deterministic delays, as can be seen comparing it to figure 4.2a.

Criteo data

As a final setting, I have tested the algorithms on delays distributed according to the Criteo dataset [16], the same used in chapter 3 for testing the bid and budget optimisation strategy. As explained in detail at the beginning of the present chapter, in chapter 3 immediate reward was assumed. On the other hand, here the recorded delays between click and conversion are used. In particular, to provide a fair comparison, I used the same model for sampling these delays as in [18], and also the same value of m and of the horizon $T = 10000$. Also in this realistic setting, the result are strikingly in favour of the proposed BTS.

5 Conclusion and future directions

5.1 Summary of the contributions

This dissertation deals with the problem of maximising conversions in digital marketing campaigns, and on how a Multi-armed Bandit formulation represents a viable and efficient approach to this problem. In particular, it is focused on some limitations that affect state-of-the-art proposals and hinder their application in practical settings. We will here go through a summary of these limitations and the related contributions.

- First, a state-of-the-art method for the bid/budget optimisation of Search Engine Marketing campaigns was extended to the multi-ad group domain, so that it can be applied to a widespread campaign format, and it can leverage all the freedom digital service platforms provide, namely differentiating the bids across the same marketing campaign.
- Exploiting domain knowledge, a parametric Bayesian regression model was introduced, to reduce the amount of data needed with respect to GPs and to naturally account for censoring, further freeing up resources for both exploration and exploitation. Parameters are interpretable, hence allowing for the easy elicitation of priors on them.
- Benefiting from the monotonicity properties of this model, the optimisation step in Thompson sampling can be carried out by local (as opposed to global) methods; the portion of the optimisation algorithm that cannot be performed with standard optimisation libraries is explained in detail.
- To reduce the computational footprint of the algorithm and study the effect of exact Bayesian inference on Thompson Sampling performance, a version of Bootstrapped TS was adapted to static contextual bandits.

- In order to test the performance of competing models, both a purpose built simulation environment and a public digital advertising dataset were used.

- A host of simulations show a clear improvement over the state of the art, especially over short times (implying a much faster convergence on average), when the budget constraint is very tight or the number of ad groups increases.

The above work assumes that the reward is *immediate*, i.e. that the agent is shown the reward of its past action before the next round occurs. In practical settings, this hypothesis works for optimising clicks; on the other hand, further steps of the marketing funnel (sales in particular) can occur many days after the first interaction of the user with an ad.

These considerations motivated, first and foremost, the need for an extensive review of the bandit literature in the presence of delays.

Rather than tackling the full bid/budget optimisation problem directly, from this review a setting emerged, which presents two traits that make it worthy of in depth study. On the one hand, this setting is interesting in itself, since it is representative of real-world scenarios in the online advertising scope, and the current state-of-the-art approach presented room for improvement. On the other, effectively managing this problem can pave the way to solutions to the bid/budget selection with delayed conversions, without having to deal with all its intricacies at once.

The aforementioned setting is that of linear contextual bandits with partially observable delayed rewards. The steps of the digital marketing funnel beyond the click, i.e., the conversions, are partially observable because the agent observes directly the conversions that do happen, but it cannot distinguish cases in which a conversion will not happen from those in which it is only delayed.

A technique with overall good practical performance, Thompson sampling, has thus been extended to effectively tackle this setting. This contribution is novel for several reasons:

- The problem of performing Bayesian inference on a coupled distribution of delays and rewards was circumvented without the need to be over-confident over the distribution of delays, i.e., without hampering exploration, as in current state-of-the-art approaches.

- The technique used, Bootstrapped Thompson Sampling, requires a Maximum Likelihood Estimator as input: the area of applicability was extended to include an Expectation Maximisation estimator. In fact, Expectation Maximisation effectively performs MLE without the need to having a closed form for the likelihood.

- In order to have a model of delays that is flexible enough to accommodate heavy-tailed distributions, an Expectation Maximisation model for partially observable rewards was extended to handle a widely adopted non-parametric estimator of time distributions, namely the Kaplan-Meier estimator. This

in turn means that this model can be used in conjunction with commonly available Machine Learning libraries.

- The resulting algorithm was applied to linear contextual bandits with partially observable rewards for the first time.

The proposed approach was compared to a state-of-the-art algorithm on a manifold of families of delay distributions, letting the parameters that characterise these distribution vary. These distributions cover a wide range of scenarios: some are bounded (even deterministic), while others have infinite expectation and even include $+\infty$ among possible realised values. The proposed approach performs significantly better than the competitor in the great majority of tested environments, and comparably in the remaining minority. Moreover, the competing algorithm requires some tuning of an hyperparameter, whose best value is affected by the distribution of delays (which is, however, unknown to the agent): on the other hand, the proposed approach was tested with the same configuration on all distributions, without any tuning.

5.2 Limitations of the proposed approaches

Besides the need to properly treat delays, which is the theme of chapter 4, some other limitations regarding the bid/budget optimisation algorithm of chapter 3 deserve being mentioned and refined in future work.

First, as noted in section 3.2, the *ansatz* formula (3.2) that simplifies learning is exact only in the deterministic case. In particular, this formula implies that, for a budget greater than the saturation cost of a campaign, the expected number of clicks does not depend on budget at all. This originates the abrupt saturation in figure 3.2b. The effect of this is especially visible when trying to split a total budget which is greater than the sum of the saturation costs of all campaigns: strictly speaking, all splittings which assure that every campaign reaches at least its saturation cost look the same to the algorithm, and one must resort to heuristics to split the remaining budget in a sensible way. At the cost of more involved calculations, a more principled way to treat the problem could be writing formula 3.2, rather than for expected values, for realisations of random variables: that formula would then be exact. From there, one could express the expected values via integrals, and then use techniques like perturbation theory to reach approximations that are smoother and more precise than (3.2).

Moreover, this works assumes that the set of campaigns is stable across the whole optimisation process, while it is often the case in practice that campaigns get added to or removed from the set of active campaigns. To solve this cold start problem, as seen in chapter 2, one could resort to associating campaigns with contexts, drawing inspiration from [40]. For instance, the context could include the features of the ad (derived from its text) and the demographic features of the target.

Finally, practitioners are often rightfully interested in leveraging the data of past campaigns to lower the cost of exploration at the beginning of the optimisation process. This task, however, is non-trivial: to learn from logged bandit feedback, one needs to take into account the counterfactual nature of the problem [43]. Usually this is done through propensity scoring techniques. However, the bid/budget problem is set apart by the continuous nature of bids and budget.

The approach proposed in this dissertation to treat delayed conversions is of course no silver bullet either: in what follows, some limiting aspects are discussed.

First, while reaching significantly lower regret than the state of the art in most studied settings, its execution is admittedly slower, as the Expectation Maximisation algorithm requires fitting two Maximum Likelihood estimators for several iterations before reaching convergence. Depending on the application, this may or may not constitute a problem: if this or a similar algorithm is used once a day to set the parameters of marketing campaigns for the following day, this increased execution time should not hinder its application. Nevertheless, it would be interesting to study an incremental variant of the proposed algorithm, adapting the *Ensemble sampling* technique of [71].

Moreover, the model was tested assuming that delays are independent of the context/arm. Note, however, that the derivation of the model itself makes no restricting assumption on the dependence of delays on context: it is only when specialising it to the Kaplan-Meier estimator that this choice is made. To take into account linear dependence of hazard on context, it would suffice to substitute the Kaplan-Meier estimator with the Cox proportional hazards model [34, 72], which is a semiparametric model: it makes a linear assumption on the effect of the features on the hazard function, but makes no assumption regarding the nature of the hazard function itself, like the Kaplan-Meier estimator.

This agnostic aspect of the Kaplan-Meier estimator was leveraged to accommodate delay distributions that are very different among each other. However, as seen above for Gaussian Processes, this expressiveness could prove detrimental for performance if the experimenter can place strong assumptions on the delay distribution. In these cases, using the given model with a parametric family of distributions could be more rewarding. Again, due to the general nature of the Expectation Maximisation model, this should entail no additional effort. Another way to incorporate stronger assumptions on the distributions involved is to increase the number of prior-like points in Bootstrapped Thompson Sampling.

Finally, this approach makes heavy use of the times between action and observed reward. If these times are not available, like in [18] and [64], this approach would require major modifications. However, this scenario in which the time between action and reward is unknown, but the reward can be linked to a specific action, is somewhat midway between (and probably less frequent than) knowing the delays as in the present work, and having aggregated, anonymous feedback as touched upon in the following section.

5.3 Future directions

Besides the extensions mentioned in the previous section, some promising avenues of research emerged, which will be addressed in future work.

An interesting step on the path to adapting the proposed Bootstrapped Thompson Sampling method for delayed conversions to the bid/budget optimisation problem would be extending the model to accommodate aggregated, anonymous feedback as in [57–59]. In a sense, this setting brings the partial observability of conversions to the extreme: an Expectation Maximisation approach, using and then eliminating unobserved variables, could prove beneficial also in this harder setting.

As seen above, non-stationarity of the distribution of rewards can be retroactively taken into account using a sliding window, beyond which old data points are discarded. However, in the presence of delays, this would effectively induce censoring on delays that exceed the window size, and as seen in simulations censoring can be detrimental for performances, if the window size is too small. This suggests several interesting streams of research. One is adapting to the partially observable setting the techniques introduced in [42] and [63] that account for the multiple touch points in the marketing funnel. Besides this, it would be interesting to study if the approach of [56], which takes into account a non-stationary baseline hazard, can be extended to the present setting. Moreover, as noted also in [42], it would be interesting to explore non-stationary bandit techniques besides the sliding window, like an adaptive window size, that takes into account how fast the environment changes.

The need for continuous retraining of models that predict the probability of conversion in the presence of delays has been studied also outside of the bandit literature. In particular, a recent stream of practitioner papers [73–76] focuses on conversion rate prediction for downstream tasks using so-called *importance sampling*. Since the model in [74] presents striking similarities to the one used in the present work, it would be interesting to study points of contact and whether some of the proposed ideas could be applied to the linear bandit setting with partially observable rewards.

As seen in the review of the literature, for the case finite-armed bandits with partially observable rewards a number of different approaches exists [19, 45, 56]. It would thus be interesting to employ the proposed approach in the finite-armed setting (rather than in the linear setting), as conduct extensive simulations comparing it to the other available methods.

Besides the area of applications of delayed conversions in online marketing, several research questions emerged from the review on bandits with delayed rewards.

First, as noted in [51], in the batched setting it can prove beneficial for UCB-like algorithms to plan actions in advance, rather than use heuristics to play each action in isolation. A natural question is whether such planning can be carried out also from the Bayesian point of view of Thompson sampling.

As noted commenting [64], the hallucination technique introduced in the

context of GP-bandits could actually be used also for finite-armed bandits. It would thus be interesting to compare it to available algorithms. Moreover, with a specific choice of kernel, linear functions can be seen as Gaussian Processes [30], so that GP-bandits become linear contextual bandits: it would be interesting to compare techniques born in the GP-bandit setting to those crafted for linear bandits.

Finally, as noted commenting [50], it would be interesting to examine in greater depth what drives the relative performance of UCB and the Successive Elimination algorithm for varying delay distributions.

Bibliography

[1] PwC, "IAB Internet advertising revenue report, Full year 2021 results", (2022).

[2] PwC, "IAB Internet advertising revenue report, Full year 2022 results", (2023).

[3] E. M. Schwartz, E. T. Bradlow, and P. S. Fader, "Customer acquisition via display advertising using multi-armed bandit experiments", Marketing Science **36**, 500–522 (2017).

[4] A. Deng, J. Lu, and S. Chen, "Continuous monitoring of A/B tests without pain: optional stopping in Bayesian testing", in 2016 IEEE International Conference on Data Science and Advanced Analytics (Oct. 2016), pp. 243–252.

[5] A. Iacob, B. Cautis, and S. Maniu, "Contextual Bandits for Advertising Campaigns: A Diffusion-Model Independent Approach", in *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)* (), pp. 513–521.

[6] H. Robbins, "Some aspects of the sequential design of experiments", Bulletin of the American Mathematical Society **58**, 527–535 (1952).

[7] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples", Biometrika **25**, 285–294 (1933).

[8] A. Slivkins, "Introduction to Multi-Armed Bandits", Foundations and Trends® in Machine Learning **12**, 1–286 (2019).

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction* (MIT press, 2018).

[10] S. Agrawal, "Recent advances in Multiarmed Bandits for sequential decision making", Operations Research & Management Science in the Age of Analytics, 167–188 (2019).

[11] T. Lattimore and C. Szepesvári, *Bandit Algorithms* (Cambridge University Press, 2020).

[12] A. Nuara, F. Trovò, N. Gatti, and M. Restelli, "Online joint bid/daily budget optimization of Internet advertising campaigns", Artificial Intelligence **305**, 103663 (2022).

[13] *Google Ads Help*, see: `answer/1704396`, `answer/1722122`, `answer/2616012`, Google Ads (2021).

[14] M. Gigli and F. Stella, "Parametric Bandits for Search Engine Marketing Optimisation", in Pacific-Asia Conference on Knowledge Discovery and Data Mining (Springer, 2022), pp. 326–337.

[15] M. Gigli and F. Stella, "Multi-armed Bandits for Performance Marketing", submitted to International Journal of Data Science and Analytics (2023).

[16] Diemert Eustache, Meynet Julien, P. Galland, and D. Lefortier, "Attribution Modeling Increases Efficiency of Bidding in Display Advertising", in Proceedings of the ADKDD'17 (2017), pp. 1–6.

[17] I. Osband and B. Van Roy, "Bootstrapped Thompson sampling and deep exploration", arXiv preprint arXiv:1507.00300 (2015).

[18] C. Vernade, A. Carpentier, T. Lattimore, G. Zappella, B. Ermis, and M. Brückner, "Linear bandits with Stochastic Delayed Feedback", in Proceedings of the 37th International Conference on Machine Learning, Vol. 119, edited by H. D. III and A. Singh, Proceedings of Machine Learning Research (July 2020), pp. 9712–9721.

[19] Z. Wang, C. Carrion, X. Lin, F. Ji, Y. Bao, and W. Yan, "Adaptive Experimentation with Delayed Binary Feedback", in Proceedings of the ACM Web Conference 2022, WWW '22 (2022), pp. 2247–2255.

[20] C. Davidson-Pilon, "lifelines: survival analysis in Python", Journal of Open Source Software **4**, 1317 (2019).

[21] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A Contextual-Bandit Approach to Personalized News Article Recommendation", in Proceedings of the 19th International Conference on World Wide Web, WWW '10 (2010), pp. 661–670.

[22] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: no regret and experimental design", arXiv:0912.3995 (2009).

[23] T. Desautels, A. Krause, and J. W. Burdick, "Parallelizing exploration-exploitation tradeoffs in Gaussian Process bandit optimization", Journal of Machine Learning Research **15**, 3873–3923 (2014).

[24] O. Chapelle, "Modeling Delayed Feedback in Display Advertising", in Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14 (2014), pp. 1097–1105.

[25] Varian, Hal R. and Harris, Christopher, "The VCG auction in theory and practice", American Economic Review **104**, 442–45 (2014).

[26] S. Despotakis, R. Ravi, and A. Sayedi, "First-price auctions in online display advertising", Journal of Marketing Research **58**, 888–907 (2021).

[27] A. Nuara, F. Trovò, N. Gatti, and M. Restelli, "A Combinatorial-Bandit Algorithm for the Online Joint Bid/Budget Optimization of Pay-per-Click Advertising Campaigns", in Thirty-Second AAAI Conference on Artificial Intelligence (Apr. 2018).

[28] W. Chen, Y. Wang, Y. Yuan, and Q. Wang, "Combinatorial Multi-Armed Bandit and its extension to probabilistically triggered arms", JMLR **17**, 1–33 (2016).

[29] D. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A tutorial on Thompson sampling", arXiv:1707.02038 (2017).

[30] C. K. Williams and C. E. Rasmussen, *Gaussian Processes for Machine Learning*, Vol. 2, 3 (MIT press Cambridge, MA, 2006).

[31] D. Gammelli, I. Peled, F. Rodrigues, D. Pacino, H. A. Kurtaran, and F. C. Pereira, "Estimating latent demand of shared mobility through censored Gaussian Processes", Transportation Research Part C: Emerging Technologies **120**, 102775 (2020).

[32] O. Chapelle and L. Li, "An empirical evaluation of Thompson sampling", Advances in Neural Information Processing Systems **24**, 2249–2257 (2011).

[33] Stan Development Team, *Stan modeling language users guide and reference manual*, version 2.28 (2019).

[34] F. E. Harrell, *Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis*, Springer Series in Statistics (Springer International Publishing, 2015).

[35] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions", in AISTATS 2011 (PLMR, 2011), pp. 208–214.

[36] S. Agrawal and N. Goyal, "Thompson sampling for contextual bandits with linear payoffs", in ICML (PMLR, 2013), pp. 127–135.

[37] S. Filippi, O. Cappe, A. Garivier, and C. Szepesvári, "Parametric Bandits: The Generalized Linear Case", in NIPS, Vol. 23 (2010), pp. 586–594.

[38] M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini, "Finite-time analysis of kernelised contextual bandits", arXiv:1309.6869 (2013).

[39] C. Riquelme, G. Tucker, and J. Snoek, "Deep Bayesian bandits showdown: an empirical comparison of Bayesian deep networks for Thompson sampling", arXiv:1802.09127 (2018).

[40] B. Han and C. Arndt, "Budget Allocation as a Multi-Agent System of Contextual & Continuous Bandits", in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21 (2021), pp. 2937–2945.

[41] S. C. Geyik, A. Saxena, and A. Dasdan, "Multi-touch attribution based budget allocation in online advertising", in Proceedings of the Eighth International Workshop on Data Mining for Online Advertising (2014), pp. 1–9.

[42] C. Vernade, A. Gyorgy, and T. Mann, "Non-Stationary Delayed Bandits with Intermediate Observations", in Proceedings of the 37th International Conference on Machine Learning, Vol. 119, edited by H. D. III and A. Singh, Proceedings of Machine Learning Research (June 2020), pp. 9722–9732.

[43] A. Swaminathan and T. Joachims, "Counterfactual risk minimization: Learning from logged bandit feedback", in International Conference on Machine Learning (PMLR, 2015), pp. 814–823.

[44] O. Chapelle, "Offline evaluation of response prediction in online advertising auctions", in Proceedings of the 24th international conference on world wide web (2015), pp. 919–922.

[45] A. Gael Manegueu, C. Vernade, A. Carpentier, and M. Valko, "Stochastic bandits with arm-dependent delays", in Proceedings of the 37th International Conference on Machine Learning, Vol. 119, edited by H. D. III and A. Singh, Proceedings of Machine Learning Research (July 2020), pp. 3348–3356.

[46] X. Zhang, H. Jia, H. Su, W. Wang, J. Xu, and J.-R. Wen, "Counterfactual Reward Modification for Streaming Recommendation with Delayed Feedback", in Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21 (2021), pp. 41–50.

[47] M. Dudík, D. J. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang, "Efficient Optimal Learning for Contextual Bandits", CoRR **abs/1106.2369** (2011).

[48] P. Joulani, A. Gyorgy, and C. Szepesvari, "Online Learning under Delayed Feedback", in Proceedings of the 30th International Conference on Machine Learning, Vol. 28, edited by S. Dasgupta and D. McAllester, Proceedings of Machine Learning Research 3 (June 2013), pp. 1453–1461.

[49] T. Mandel, Y.-E. Liu, E. Brunskill, and Z. Popović, "The Queue Method: Handling Delay, Heuristics, Prior Data, and Evaluation in Bandits", Proceedings of the AAAI Conference on Artificial Intelligence **29**, 10.1609/aaai.v29i1.9604 (2015).

[50] H. Wu and S. Wager, "Thompson Sampling with Unrestricted Delays", in Proceedings of the 23rd ACM Conference on Economics and Computation, EC '22 (2022), pp. 937–955.

[51] E. A. Daxberger and B. K. H. Low, "Distributed batch Gaussian Process optimization", in Proceedings of the 34th International Conference on Machine Learning, Vol. 70, edited by D. Precup and Y. W. Teh, Proceedings of Machine Learning Research (Aug. 2017), pp. 951–960.

[52] K. Kandasamy, A. Krishnamurthy, J. Schneider, and B. Poczos, "Parallelised Bayesian Optimisation via Thompson Sampling", in Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, Vol. 84, edited by A. Storkey and F. Perez-Cruz, Proceedings of Machine Learning Research (Apr. 2018), pp. 133–142.

[53] S. R. Chowdhury and A. Gopalan, "On Batch Bayesian Optimization", CoRR **abs/1911.01032** (2019).

[54] A. Verma, Z. Dai, and B. K. H. Low, "Bayesian optimization under stochastic delayed feedback", in Proceedings of the 39th International Conference on Machine Learning, Vol. 162, edited by K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Proceedings of Machine Learning Research (June 2022), pp. 22145–22167.

[55] C. Vernade, O. Cappé, and V. Perchet, "Stochastic Bandit Models for Delayed Conversions", in Conference on Uncertainty in Artificial Intelligence (2017).

[56] H. Wu and S. Wager, "Partial likelihood Thompson sampling", in Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence, Vol. 180, edited by J. Cussens and K. Zhang, Proceedings of Machine Learning Research (Aug. 2022), pp. 2138–2147.

[57] C. Pike-Burke, S. Agrawal, C. Szepesvari, and S. Grunewalder, "Bandits with Delayed, Aggregated Anonymous Feedback", in Proceedings of the 35th International Conference on Machine Learning, Vol. 80, edited by J. Dy and A. Krause, Proceedings of Machine Learning Research (June 2018), pp. 4105–4113.

[58] S. Garg and A. K. Akash, "Stochastic Bandits with Delayed Composite Anonymous Feedback", CoRR **abs/1910.01161** (2019).

[59] S. Wang, H. Wang, and L. Huang, "Adaptive Algorithms for Multi-armed Bandit with Composite and Anonymous Feedback", Proceedings of the AAAI Conference on Artificial Intelligence **35**, 10210–10217 (2021).

[60] Z. Zhou, R. Xu, and J. Blanchet, "Learning in Generalized Linear Contextual Bandits with Stochastic Delays", in Advances in Neural Information Processing Systems, Vol. 32, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (2019).

[61] B. Howson, C. Pike-Burke, and S. Filippi, "Delayed Feedback in Generalised Linear Bandits Revisited", in Proceedings of The 26th International Conference on Artificial Intelligence and Statistics, Vol. 206, edited by F. Ruiz, J. Dy, and J.-W. van de Meent, Proceedings of Machine Learning Research (Apr. 2023), pp. 6095–6119.

[62] T. A. Mann, S. Gowal, A. Gyorgy, H. Hu, R. Jiang, B. Lakshminarayanan, and P. Srinivasan, "Learning from Delayed Outcomes via Proxies with Applications to Recommender Systems", in Proceedings of the 36th International Conference on Machine Learning, Vol. 97, edited by K. Chaudhuri and R. Salakhutdinov, Proceedings of Machine Learning Research (June 2019), pp. 4324–4332.

[63] T. M. McDonald, L. Maystre, M. Lalmas, D. Russo, and K. Ciosek, "Impatient Bandits: Optimizing Recommendations for the Long-Term Without Delay", in Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD '23 (2023), pp. 1687–1697.

[64] T. Lancewicki, S. Segal, T. Koren, and Y. Mansour, "Stochastic Multi-Armed Bandits with Unrestricted Delay Distributions", in Proceedings of the 38th International Conference on Machine Learning, Vol. 139, edited by M. Meila and T. Zhang, Proceedings of Machine Learning Research (July 2021), pp. 5969–5978.

[65] K. Suresh, C. Severn, and D. Ghosh, "Survival prediction models: an introduction to discrete-time modeling", BMC medical research methodology **22**, 207 (2022).

[66] E. L. Kaplan and P. Meier, "Nonparametric Estimation from Incomplete Observations", Journal of the American Statistical Association **53**, 457–481 (1958).

[67] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm", Journal of the royal statistical society: series B (methodological) **39**, 1–22 (1977).

[68] D. Eckles and M. Kaptein, "Thompson sampling with the online bootstrap", arXiv preprint arXiv:1410.4009 (2014).

[69] B. Efron, "Bootstrap Methods: Another Look at the Jackknife", The Annals of Statistics **7**, 1–26 (1979).

[70] D. B. Rubin, "The Bayesian Bootstrap", The Annals of Statistics **9**, 130–134 (1981).

[71] X. Lu and B. Van Roy, "Ensemble Sampling", in Advances in Neural Information Processing Systems, Vol. 30, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (2017).

[72] D. R. Cox, "Regression models and life-tables", Journal of the Royal Statistical Society: Series B (Methodological) **34**, 187–202 (1972).

[73] S. I. Ktena, A. Tejani, L. Theis, P. K. Myana, D. Dilipkumar, F. Huszár, S. Yoo, and W. Shi, "Addressing Delayed Feedback for Continuous Training with Neural Networks in CTR Prediction", in Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19 (2019), pp. 187–195.

[74] J.-Q. Yang, X. Li, S. Han, T. Zhuang, D.-C. Zhan, X. Zeng, and B. Tong, "Capturing Delayed Feedback in Conversion Rate Prediction via Elapsed-Time Sampling", Proceedings of the AAAI Conference on Artificial Intelligence **35**, 4582–4589 (2021).

[75] S. Gu, X.-R. Sheng, Y. Fan, G. Zhou, and X. Zhu, "Real Negatives Matter: Continuous Training with Real Negatives for Delayed Feedback Modeling", in Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21 (2021), pp. 2890–2898.

[76] Y. Chen, J. Jin, H. Zhao, P. Wang, G. Liu, J. Xu, and B. Zheng, "Asymptotically Unbiased Estimation for Delayed Feedback Modeling via Label Correction", in Proceedings of the ACM Web Conference 2022, WWW '22 (2022), pp. 369–379.