Dynamic allocation optimization in A/B tests using classification-based preprocessing

Emmanuelle Claeys, Pierre Gancarski, Myriam Maumy-Bertrand, Hubert Wassner

Dynamic allocation optimization in A/B tests using classification-based preprocessing

EMMANUELLE CLAEYS, University of Strasbourg
PIERRE GANÇARSKI, University of Strasbourg
MYRIAM MAUMY BERTRAND, University of Strasbourg
HUBERT WASSNER, AB Tasty, France

In traditional A/B testing, for instance on two webpages A and B, the objective is to decide which of these two pages is the best. To do that, a frequentist test can be used in which each page is randomly or alternatively chosen and applied to incoming website visitors for a given time. However, one problem with this approach is the non-adaptivity of the test. For example, if one page quickly appears as having a very stronger positive or negative impact than the other one, the test could be stopped earlier. One way to avoid this is to apply a bandit-based algorithm. Such an algorithm is able to automatically decides if a page should be chosen and applied more often than the other one. This approach, called dynamic allocation, allows to add adaptivity to the A/B test. However, bandit theory by traditional methods requires assumptions which are not always verified in reality. This is mainly due to the fact that the subjects tested are not homogeneous. We present our new method that finds the best variation for homogenous groups in a short period of time.

CCS Concepts: • **Transactions on Interactive Intelligent Systems**;

Additional Key Words and Phrases: Bandit strategies, Classification, A/B Testing, Conditional inference tree, UCB strategies, Non-linear bandit, Regret minimization

ACM Reference Format:
Emmanuelle Claeys, Pierre Gançarski, Myriam Maumy Bertrand, and Hubert Wassner. 2019. Dynamic allocation optimization in A/B tests using classification-based preprocessing. *ACM Trans. Interact. Intell. Syst.* , (2019), 33 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Evaluating the impact of a decision in regards to a quantifiable reward is a recurring problem in many industrial, medical or other applications. For instance, introducing new tools or new protocols in a production process often impacts the quality and/or quantity of the production. In marketing, changing a specific page on a website can influence the visitor's behaviour. So when the decision consists in choosing between two or more variations, a way

Authors' addresses: Emmanuelle Claeys, claeys@unistra.fr, University of Strasbourg, 7 Rue René Descartes, Strasbourg, 67084; Pierre Gançarski, University of Strasbourg, 7 Rue René Descartes, Strasbourg, 67084, gancarski@unistra.fr; Myriam Maumy Bertrand, University of Strasbourg, 7 Rue René Descartes, Strasbourg, 67084, mmaumy@math.unistra.fr; Hubert Wassner, AB Tasty, 3 Impasse de la Planchette, 75003 Paris, Paris, France, hubert@abtasty.com.

to evaluate the impact of the modification is to use *A/B tests*[1]. An A/B test begins with an *exploration phase* which aims to compare the performance of each variation in order to highlight the best ones (*i.e.*, the *optimal*) according to a given criterion.

The objective is to evaluate the performance of different alternatives (called *variations*), for example, of the same page of an e-commerce site. To do this, each visitor (called *item*) is assigned to a variation (*i.e.*, the web page that is presented to the visitor).

Then the result of the visit (called *reward*), for example a purchase, the amount of money spent or the number of clicks, is observed. At the end of the exploration phase, the *user* (e.g., the e-commerce company) takes a decision according to the rewards obtained using a given criterion depending on the objective (e.g., Increasing of the cumulative purchase value). The user can then proceed to the *exploitation phase*: setting the winning variation(s) as default into production, for every visitor. Tab 1 gives some examples of such A/B tests.

User	Item	Variation	Reward	Objective
E-commerce compagny	Visitor	Webpages	Purchase value	Global value
Car compagny	Car order	Product chain	Car quality	Quality average
Medical protocol	Patient	Treatment	Individual healing	Healing rate

Table 1. Examples of A/B tests

Detecting and quantifying differences between variations is the main challenge of A/B testing. One way to carry out the exploration phase is the frequentist approach. It is based on a *static allocation* and defines explicit ratios of allocation of the items to the different variations.

However, a real-world exploration phase implies eventually assigning an item to a non-best (referred as *non-optimal*) variation, automatically leading to a loss of earnings. The difference between the potential reward from the optimal variation and the actual reward is called *regret*. This means that each time an item is not assigned to the optimal variation, the *cumulative regret* increases. Handling the cumulative regret in order to limit it, while ensuring that the optimal solution is found, is the second challenge.

Experiments have shown that strategies based on *dynamic allocation* [36] that uses probabilistic criteria based on the comparison of the empirical reward distribution of each variation provide better results in terms of cumulative regret, as well as being faster at determining the best variation. In this context, methods implementing such strategies have been proposed and are mainly based on the bandit paradigm [17, 25, 33]. They have proved their ability to find optimal variation in the general case. But they often fail when the reward obtained by an item depends on both the variation and the item itself [28]. For instance, in web marketing, visitors (test subjects) naturally tend to click and buy differently, according to their own financial resources or their geographical localization. Consequently, contextual bandits have been developed to consider items characteristics and improve the strategy of dynamic allocation. Nevertheless, in a lot of cases, items belong to natural groups (e.g., social class, gender . . .). And the reward distribution of a variation often differs according to this belonging. For example, people belonging to a higher class (double income, home owners,...) can easier purchase an expensive thing than unemployed people. So, people from the first group can be impacted by the modification of the webpage whereas people from the second group not.

[1]For simplicity we define an A/B test as a test that can have more than two alternatives.

In [3] the author indicates that website visitors groups are generally persistent over the time, and the choice of variation displayed seems to have a low impact even if it exists, on the behavior.

In fact, sub populations may have different behavior, but when a sub population is too small the specific effect may be hard to measure. For example, a visitor with a low interest (or low financial resources) continues to no buy anything whatever version A or B of the displayed page. Thus, the ratio of people impacted by the modification can be very low. In this case, detect the very low decreasing or growing of cumulative reward appears as difficult.

It is therefore interesting, if not essential, for the user to study the impact of the modifications on each of these groups independently. Unfortunately, these groups are difficult to determine because they depend on the type of e-commerce: for example, "natural" groups related to sport may be completely different from those related to arts or politics. A solution to overcome this problem is to use data mining tools such as clustering or regression methods for example, on the items using their characteristics. Note that considering that A/B test users often perform several tests sequentially, extract these groups upstream seems more relevant than start from scratch at each test. Finally, this segmentation of items into groups can also help the user to define the modifications to be tested.

In this paper we propose an original A/B testing method, called Ctree-Ucb, which, instead of using a contextual bandit, is based on the use of several non-contextual bandits, each dedicated to a particular group of items. Our proposal consist to automatically create homogeneous groups by a conditional inference method in a pre-processing step to the A/B test (step 1).

These groups are created according to the objective of the test using information from previous items already subjected to the original variation (A) supposed implemented before the test. These information are obtained rewards, items characteristics, temporal information, . . .

In the A/B test itself (step 2), the method defines as many non-contextual bandits as there are groups. Each bandit aims to find the optimal variation associated to its group. To do that, each new item is classified in a group before be transmitted to the associated bandit.

The remainder of this paper is organized as follows. Section 2 presents the bandit model with an illustrative example. Based on this example, Section 3 gives a comprehensive review of the literature on the existing approaches and focuses on contextual strategies able to take into account the characteristics of the items. Section 4.2 details the proposed methods. Sections 5, 6 and 7 analyze and discuss the results obtained with this method on real data from the company AB Tasty [2]. Finally the conclusions of the study are done in Section 8.

2 A/B TEST AND BANDIT PARADIGM

2.1 A/B testing

Suppose that a marketing team has to decide whether the ‘picture A’ or ’picture B’ should be used on a webpage. Traditional techniques (such as frequentist test) trial each version by splitting incoming users into two groups during a given time. Each group only show a version of the page (P_A or P_B). At the end of the test, the team can decide which version is the best one. One problem with this approach is the non-adaptivity of the test [27]. For example, if a version quickly appears as strongly better than the other one or if the behaviors of the

[2]https://www.abtasty.com

visitors are the same independently of the shown version, the test could be concluded early. As introduced in Section 1, a bandit model is a good way to solve this problem: each time t a user c_t arrives on the webpage, the bandit algorithm chooses an arm a in $\mathcal{A} := \{P_A, P_B\}$ and the reward $X_{c_t,a=A_t}$ is reaped (for instance, it equals to 1 if the user purchases the product and 0 otherwise). The main characteristic of a bandit algorithm is that the ratio of allocation of the items to the different versions is automatically adjusted: the algorithm automatically decides when one version should be more often selected than the other one. It is a real gain because the ability to provide an automatical stop can be directly added to the A/B test. For instance, the test stops when the items are affected at 90% to a same version.

2.2 The bandit model

The notion of bandit was introduced by Lai and Robbins, [26] under the name of multi bandit but we use here only the term bandit to simplify the reading. It's defined as "*a problem in which a fixed limited set of resources must be allocated between competing (alternative) choices in a way that maximizes their expected gain, when each choice's properties are only partially known at the time of allocation, and may become better understood as time passes or by allocating resources to the choice*" [18].

In concrete terms, by analogy with casino slot machines, it is a matter of choosing, for a player, on a machine with several arms, the one presenting, for him, the best expectation of gain. Then, after playing and reaping the gain, the player updates the gain estimates on each arm of the machine. The player's goal is to find the best arm, called *optimal arm*, with as few tries as possible.

Let $\mathcal{A}$ be a set of possible arms (with $|\mathcal{A}| \in \mathbb{N}^+$) and a^* the optimal arm. The goal of a bandit is to find the a^* as quickly as possible in order to limit the cumulative regret. This regret is the sum, for each item, of the difference between the rewards obtained with a^* and those observed with the chosen arm. Let $X_{A_t,t}$ be the reward obtained by the arm A_t selected at the iteration t (corresponding to the t-th item). r_t is the simple regret at iteration t, defined by:

$$r_t = \max_{a \in \mathcal{A}}[X_{a,t}] - X_{A_t,t} \tag{1}$$

and R_n is the cumulative regret after n iterations, defined by:

$$R_n = \sum_{t=1}^{n} \max_{a \in \mathcal{A}}[X_{a,t}] - \sum_{t=1}^{n} X_{A_t,t} \tag{2}$$

.

The challenge is to find the best dynamic allocation strategy π. However, this depends on the application context, i.e. the potential rewards of each item on each arm. However, these rewards distributions are unknown and will be estimated. The better these estimates, the better the allocation. Concretely, with each new item c_t, only one arm $a_t \in \mathcal{A}$ is chosen, and therefore only its reward distribution $\nu_{a_t} = \mathbb{P}[X|a_t] : a_t \in \mathcal{A}$ is updated.

It is for this purpose that the bandit updates the distributions according to the items submitted to the arms and the rewards obtained. However, this update is tricky because it can only be done on the selected arm.

Algorithm 1 summarizes a bandit algorithm. It considers an infinite stopping time. In practice a stopping criterion is chosen according to the use case.

Recall that, since the best dynamic allocation strategy finds a^* the quickest (and gets a low cumulative convergent regret), the challenge here is to find it.

Algorithm 1 Bandit algorithm

```
Require: Assign at least one item to each arm $a \in \mathcal{A}$
1: loop
2:     Receive an item $c_t$
3:     Assign $c_t$ to an arm $A_t$ depending on its distribution $\nu_{A_t}$ and a strategy $\pi$
4:     Receive a $X_{c_t,A_t}$ reward
5:     Update $\nu_{A_t}$
Output: A sequence of arm choices and rewards
```

For the rest of the article the following notations will be used:

- Due to randomized reward from arm distribution, the regret R_n may be re-written as:

$$R_n = n\mu_{a^*} - \sum_{t=1}^{n}[X_{A_t,t}] \tag{3}$$

- $\mathbb{E}[R_n]$ is the average regret [5], defined by:

$$\begin{aligned}\mathbb{E}[R_n] &= \mathbb{E}[\sum_{t=1}^{n}\max_{a\in\mathcal{A}}[X_{a,t}] - \sum_{t=1}^{n}X_{A_t,t}] \\ &= \mu_{a^*} - \mathbb{E}[\sum_{t=1}^{n}X_{A_t,t}]\end{aligned} \tag{4}$$

- μ_a is the (unknown) actual reward average of a (under stationary hypothesis) and $\hat{\mu}_{a,t}$ its average estimate at time (or iteration) t corresponding to the t-th item tested.
- a^* the optimal arm (to be determined) and μ_{a^*} its reward average.
- $\Delta_a = \mu_{a^*} - \mu_a$ is the difference between the real average of a^* and the real average of a.
- $T_a(t)$ is the number of times where the arm a is chosen at t.

The previous notations is used in the rest of the paper conditionally, i.e. for a group k of observed items. For instance, the notation $R_{n,k} = n\mu_{a^*,k} - \sum_{t=1}^{n}[X_{A_t,t}|k]$ defines the regret for a group k of items.

Some proprieties. As the user is looking for the arm with the highest average a^*, if the probability distributions are identical it can lead to a simple non-zero regret although the bandit systematically assigns a^*. Cumulative regret increases over time, with the following theoretical guarantees:

Theorem 2.1 (**Agrawal et al (1989)**). *Let k and k' be some groups of item such as $k \cup k' = \mathcal{K}$ where $\mathcal{K}$ represent the whole population of item and $k \cap k' = \emptyset$. Assume k is the most represented group (can be considered as the true group) where $\nu_{a^*,k} = \nu_{a^*,k'}$ and $a^*_k \neq a^*_{k'}$.*

$$\lim_{n\to+\infty}\frac{R_n}{\log n} \geq \sum_{a\in\mathcal{A}}\frac{\mu_{a^*,k} - \mu_{a,k}}{KL(\nu_{a,k}||\nu_{a,k'})}$$

Where $KL(.||.)$ is the Kullback–Leibler divergence between two distributions.

In [37], Salomon and Audibert proposes an non asymptotic bound such as:

Theorem 2.2 (**Salomon and Audibert (2011)**). *Assume that $\nu_{a^*,k} \neq \nu_{a^*,k'}$ and $a^*_k \neq a^*_{k'}$ or $\exists a \in \mathcal{A} : \mathbb{P}[\frac{d\nu_{a,k}}{d\nu_{a,k'}}(X) > 0] = 0$. Then if $a^*_{k\cup k'}$ is the only unique best arm in the environment $\mathcal{K}$, then for any $\beta > 0$ it holds for some positive constants C and C'*

$$\forall n \forall a \neq a^*_{k\cup k'} \qquad \mathbb{P}[T_a(n) \geq \frac{C \log n}{\min_{a\in\mathcal{A}} \Delta^2_{a,k\cup k'}}] \leq C' n^{-\beta}$$

We present in the following section different bandit strategies. Two approaches are possible, the first one is a non-informative approach (only rewards are observed), the second is a contextual approach (item characteristics are considered before the assignment).

3 STATE OF THE ART

3.1 Non informative strategy

The quickest way to find the find the best arm is to assume that this arm is the best for the majority of items and so apply a *non informative strategy* which consists to choose the allocated arm independently of the characteristics of the item.

A bandit strategy is defined in two parts. The first one assumes that the distribution of all arm rewards follow the same law (like the Bernoulli distribution with Thompson sampling Algorithm [38, 43] or the Gaussian law Ucb [2]) or otherwise makes no assumptions. The other part of a bandit strategy is the mathematical model applied to limit regrets. Some of them are greedy where the best arm (based on previous observations) is chosen according to a predefined probability (such as Epsilon-Greedy Algorithm [42]) or adaptive (for example Softmax exploration [44]). According to these two parts, a bandit strategy is defined. The following section describes one of the most commonly used non-contextual approaches.

3.1.1 Ucb *strategy.* The Ucb method is a non informative method based on an optimistic Bayesian strategy (upper bounds of estimate averages): The principle of its π strategy is to use an overestimation of the empirical average $\hat{\mu}_a$ for each arm a, the total number of items and their allocation on the different arms, to assign a new item to an arm. Concretely, an arm is chosen if it is promising (because its estimated average is high) or/and few explored.

In fact, the $\hat{\mu}_{A_t}$ estimators may be no relevant at the beginning of the test, due to the small number of items considered [24]. To get around this difficulty, [2] proposes to calculate an overestimation of this average, called the *upper confidence bound*, as the value most probably higher than the real value of the average. The authors justify their proposition by a policy known as "optimistic in the face of uncertainty" and offers good results.

This upper bound is the sum of the empirical average rewards obtained so far and an exploration bonus (also known as the confidence interval). It depends on the number of items assigned and observed. The more observations an arm makes, the more the arm bonus decreases. If ν_a is Gaussian for all a, this bound always be upper than the real average. Thus, the authors define the upper bounds of each arm by:

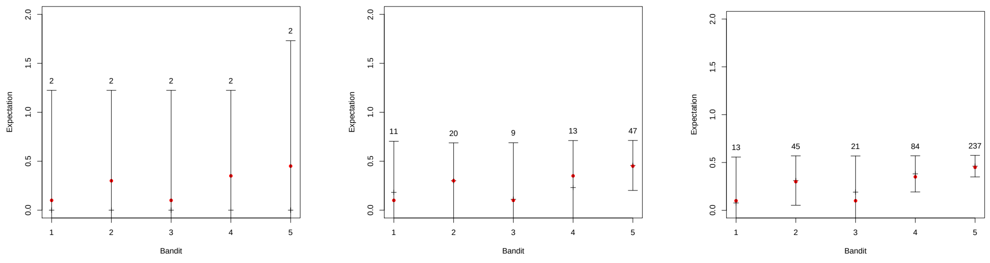
$$Upper_{\text{UCB}}(a,t) = \alpha\sqrt{\frac{2 * log(t)}{T_a(t)}} \tag{5}$$

where $T_a(t)$ is the number of times the arm a has been chosen and α is given by the user. In the initial version of Ucb, $\alpha = 1$ but in practice it has been shown that the optimal choice of this value depends on the arm distibutions [7].

The π algorithm consists in choosing the arm with the highest upper bound. After each assignation, $\hat{\mu}_{A_t}$ is updated and its bound is reduced (see equation 5). As the confidence interval depends of $T_a(t)$, the higher $T_a(t)$ the less the overestimation: the overestimation of the chosen arm's average decreases, to be equal to its real average. The upper bounds of the unchosen arms remain unchanged.

Figure 1 shows an example of evolution of the confidence bound for five arms according to the number of submitted items.

in Ucb how, over time, the adaptive arm bounds decrease with regard to the number of items tested.



(a) after 10 items tested (b) after 100 items tested (c) after 400 items tested

The red points are the real averages. Black crosses are the estimated averages.

Fig. 1. Confidence bound evolution according to number of times an arm has been chosen by Ucb

The Ucb algorithm is reported in Appendix A and achieves an average regret defined by:

$$\mathbb{E}[R_t] = \sum_{a \in \mathcal{A} \setminus \{a^*\}} (\mu_{a^*} - \mu_a)\mathbb{E}[T_a(t)] \tag{6}$$

To prove the logarithmic convergence of cumulative regret, in [2, 37], the authors define the upper bound of average regret as:

$$\mathbb{E}[R_t] \leq 12 \sum_{a \in \mathcal{A} \setminus \{a^*\}} \frac{\log(t)}{\Delta_k} \tag{7}$$

which can by reformulated, using Cauchy Schwarz inequality, as:

$$\mathbb{E}[R_t] \leq 12 \sum_{a \in \mathcal{A} \setminus \{a^*\}} \frac{\log(t)}{\Delta_k}$$

$$\mathbb{E}[R_t] \leq \sqrt{8|\mathcal{A}|t(\log t + \frac{\pi^2}{3})} \tag{8}$$

As refers in Equ. 8, for t the total number of items given, an asymptotic bound of regret is guaranteed over time. Requiring only the average of the arms and the number of items assigned to each of them, this algorithm offers a low complexity and allows the user to obtain a choice quickly. In [6], reader can found more details about properties of Ucb strategy. Variants of the Ucb algorithm have been proposed to improve the theoretical bound in [1, 15, 16].

3.1.2 Limit of Ucb *strategy.* The Ucb algorithm is really efficient when the actual distribution of rewards is Gaussian, assumption which can be checked retrospectively using a static allocation. Unfortunately, experiments have shown that this assumption is rarely validated. In fact, the upper bound is not reliable. Consequently, Ucb requires more items to find a^*. Moreover, if the reward can present extreme values, the convergence can be very long. But, it is proved that the UCB finally find a^*. And so, Despite these imperfections, as Ucb focus on the reward average that leads to a low complexity and as it is generally well understood by the user of an A/B test, it globally responds to the problems of interpretability and limited computation time.

The remain problem is that a^* can be not respond to the objective of the A/B test (find which and for whom, as quickly as possible, is the best variation). Indeed, suppose that it exists two sub-groups of items having different responses to the test. For instance, in the medical field, an alternative treatment may be efficient for the elderly and not for young people. In marketing, a page can be optimal only for smartphone users. In this case, the ν_a is very far from a Gaussian distribution. Thus, recent approaches assume that ν_a is a mixture of Gaussian especially in contextual-based strategies. The idea of the contextual strategy (presented below) is that reward X_{c_t,A_t} depends of both arm assigned and item's characteristics (covariates).

3.2 Contextual Strategy

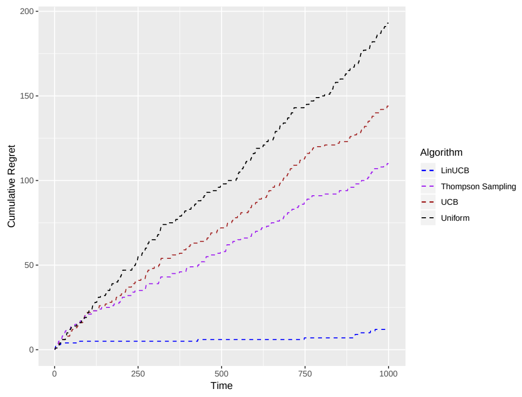
Contextual approaches assume that it exists sub-groups of items presenting each a different reward distribution. But, experiments show that it can be difficult to define such groups. Asking the user to define them is often unproductive, as he/she often does not have a clear vision of these different groups. To overcome this problem, it is assumed that there are unknown *a priori* link between on one hand, the *context* of the items (i.e., the characteristic vectors describing the items [47]) and the groups and on the other hand, the groups and the averages of the rewards obtained. To fit this link, two approaches can be considered:

- In *contextual bandit*, this link is modelled by a unique regression function: the groups and their associated average rewards are directly set by the bandit during the test (see Section 3.2.1).
- In two step-based approaches, the groups are set using a preprocessing step to the A/B test (see Section 3.2.2).

3.2.1 Contextual bandits. In this approach, rewards are supposed generated from a unknown function depending on the items covariates (characteristics) and the arm chosen. The objective is to fit this function during the test. Consequently, assumptions are made about the type of function, such as linearity property. Some methods like Lin-Ucb are based on this idea.

In bandits based on linear regression, the arms' parameters are mostly calculated by a matrix inversion, which can be very time consuming regarding to the number of item's covariates d [27]. These approaches have shown their theoretical and practical performances in reducing cumulative regrets [5]. In particular the Lin-Ucb algorithm is one of the most popular of such methods due to its performance and interpretability. Figure 2 shows the cumulative regret over time t with simulated data generated with a linear reward function. From this figure, one can see that the Lin-Ucb algorithm outperforms the Thompson Sampling, uniform and Ucb ones.

If the linearity assumption is not established, some techniques coming from statistics has been proposed. In [12] authors proposes a method based on Generalized Linear Model



The lower the cumulative regret, the better the method

Fig. 2. Cumulative regret of four linear regret-based bandit algorithms on simulated data

(GLM), called GLM-UCB. This method allows to consider a wider class of problems, and in particular cases where the rewards are counts or binary variables using, respectively, Poisson or logistic regression. Like Lin-Ucb, GLM-UCB requires an matrix inversion and can be very costly in terms of time response.

Recently, bandit algorithm based on tree regression was proposed with BanditForest algorithm [14]. This algorithm uniformly assigns visitors to each arm until a tree forest models the link function. This random forest is built with the joint distribution of contexts and rewards. Thus, all past observations (context and rewards) are stored. The uniformly assignment leads to excessive selection of sub optimal arms, causing the algorithm's performance suffers. Moreover, the main limitation of the algorithm is that it depends on four parameters requiring strong domain expertise to be set: two parameters directly influence the level of exploration, one controls the depth of the trees, and one determines the number of trees in the forest [10]. In [10, 11, 13] the authors propose the tree bootstrap algorithm, based on the similar approach but which requires no parameters and which automatically chooses the depth in the trees. However, these algorithm only considers binary rewards, which strongly limits its use.

The kernelized stochastic contextual bandit Kernel-Ucb [45] provides a non linear modelization of the link reward function (like GLM-UCB). It use a reproducing kernel Hilbert space (RKHS). Unfortunately, Kernel-Ucb requires a long latency to provide a choice.

Moreover, in addition to having to make assumptions (e.g., Gaussian distribution, binary reward . . .)) to remain understandable and implementable, these methods suffer from many drawbacks identified in the literature:

- In [46] the authors explain that dynamic allocation does not provide more benefit than a frequentist allocation when the assumptions (e.g., linear dependence between characteristics and reward, independence between items, ...) have not been fulfilled.
- The choices made by the algorithm are not explicit (black box). While understanding choices is not always necessary in a recommendation system, it is important for A/B test as the user seeks to understand why and for whom one variation is better than another.
- These methods often require large data set.

- The CPU or memory resources required to carry out such algorithms can be significant, especially since the differences between the versions is small.

3.2.2 Two step-based approaches . In the approach, it is assumed that it exists some natural groups presenting each a Gaussian reward distribution and that these groups can be determinated before the A/B test itself. The idea is to build these groups a priori. Then, when a new item is submitted to the system, it classes it first into a group. The assignment to an arm is only done after and taking into an account the group the item belongs to.

In [30], the authors propose the SINGLE-K-UCB method and show that if groups are well defined, the cumulative regret early converges asymptotically and the average regret falls significantly. Indeed, intuitively, the cumulative regret is in this case, bounded by the sum of the cumulative gaps between best arm and sub-optimal arms for each group (which is higher than a gap on a non informative strategy). Authors assumes that reward distributions are clustered and the clusters are determined by some latent variables. They assume that there is a surjective function f which to links each item (with a c_t context) to a group k such as: $f(c_t) = k$ such that the reward distribution of a k group applied to an arm a : $\nu_{a,f(c)}$ is σ-Gaussian (where σ^2 is the variance, known). Unfortunately, they do not specify how to identify the f function and how to obtain different relevant groups. They only studied the problem in a context-free setting, and a very weak performance guarantee is provided when the reward distribution is unknown in those clusters [34].

To address this problem, we propose a new method called CTREE-UCB which is detailed in the next section.

4 CTREE-UCB: A CONTEXTUAL APPROACH TO A/B TEST

Our contribution aims at addressing constraints and needs experienced by users in real-world applications. Thus, in the first instance, we address cases where the system must respond in a time the shorter as possible (ideally in real-time): for instance, the delay of respond must be lower than the usual display time of a webpage. In the second instance, we consider that the items submitted to the system can be very heterogeneous but clusterisable according to a given criterion. Finally, we want to ensure that the results of the exploration phase are understandable, and possibly reusable, by the user.

In this context, we propose an approach that consists of defining groups based on the covariates (i.e., characteristics or features) of the items, each of these groups having the most homogeneous population as possible. The main idea is that in a such group, the items have a similar behavior regarding to the proposed version and thus that the distribution of groups' rewards can be modelled by a Gaussian distribution. Thus, each of these groups can be supported by a non-contextual bandit. The general procedure is that each time a new item is presented to the system, it is automatically assigned to a group according its own covariates and then, through the associated bandit, an alternative is assigned to it. As the complexity of this type of bandit is low, this ensure a satisfactory response time.

In summary, the proposed method CTREE-UCB consists of two main steps:

- an offline process for creating groups.
- an online A/B test.

4.1 Groups identification

To construct these groups, we suppose that there are information describing performance of the existing version i.e., a database (referred here as $\mathcal{L}_n$: $\mathcal{L}$) which contains, for each item, its context and the reward (conversion, number of clicks, survival, ...) produced on this "original" version.

Thus, if the test to be carried out concerns the same type of reward and on a variation of the original version (referred here as the arm A), using this information to build groups can only be beneficial. To do that, a model can be learned using learning and validation sets extracted from $\mathcal{L}_n$. This model is then used it to predict the group a new item belongs to according to its context.

A lot of supervised methods exist to produce such model.

For instance, The decision trees-based algorithms like C4.5 [35] or C.A.R.T. [4] have shown their effectiveness in finding such homogeneous groups according a numeric/binary rewards using an entropy measurement (C4.5) or with the Gini index (C.A.R.T.). Unfortunately, they present two fundamental issues: an over-fitting and selection bias towards continuous covariates [39]. In the same way, conditional inference trees approches have shown their robustness compared to these previous algorithms [31]. Moreover these trees have a high level of stability and robustness [29, 41]. More details will be given in the section 4.2. In fact, this approach seems very interesting because the pre-process of group construction at an A/B test and our method Ctree-Ucb based on its has shown good performance.

Moreover, we want the groups to be relevant for the user: they should be both fairly general (and therefore in small numbers) but nevertheless specific enough to provide real information to the user. However, too many groups slow down the exploration phase because too many items will be required to learn on each groups.

4.2 Ctree-UCB process

The global scheme of our framework is given by Fig.3.

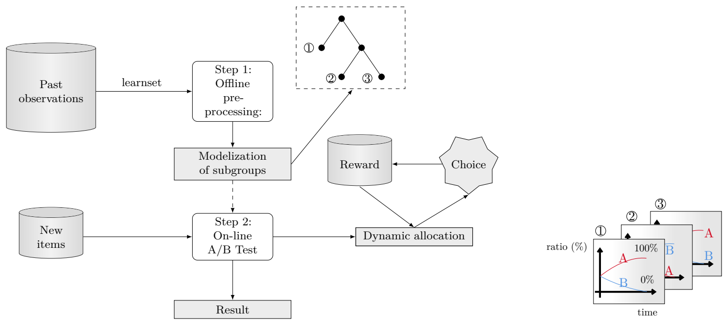


Fig. 3. Ctree-Ucb: a contextual approach to A/B test

4.2.1 Step 1. In the this first step, performed offline, a conditional inference tree algorithm (for instance the one described in [21, 22, 40]) called CTREE (Algo. 2) is applied to a learning data set (refered here as $\mathcal{L}_n$) of n items to identify homogeneous groups. It consists in initially creating one group containing all the items. This group is associated to the root node of the tree. Then, an recursive divisive process is applied from this node. An independence hypothesis H_0 between each $j \in \{1, \ldots, d\}$ covariate and the reward distribution is evaluated for all the items of the associated group. Then:

- If the hypothesis can be rejected, the group is split into two subgroups using the covariate j^* with the highest correlation with the reward, according to the value of this covariate that maximizes the difference in distribution between two groups. The algorithm is recursively applied on two new nodes associated to the two subgroups.

- If the hypothesis cannot be rejected, H_0, at the predetermined risk level ϵ, for any covariate, the recursion stops.

To verify the correlation hypothesis, numerous tests exist in the literature (for example A.N.O.V.A., Spearman test, Wilcoxon Man Witney, Chi2 ... [21, 40]). At the instantiation of the Ctree-Ucb scheme, such correlation tests must be defined according to the covariate types (continuous, binary, categorical ...) and reward type (continuous or binary) [23].

The conditional inference tree do not require a pruning process, which leads to avoid the overfitting. Moreover, the selection of the value to split in is based on the univariate p-values avoiding a variable selection bias towards caracteristics with many possible split value. If a statistically significant observation may have actually arisen by "chance" because of the size of the parameter space to be searched, a Bonneferoni correction can be apply [9]. However, tests integrating categorial covariates can require very long computation time when a Bonferroni correction is applied [20]. To reduce this time, it is possible to use a Bonferroni correction by the Monte Carlo method [32]. Such a correction includes a random part, which varies the tree structure.

At the end of Step 1, groups are described by a reward average and defined by one or more covariates. Using these information, a predictive function f is defined which link to each new item to a group.

This function actually predicts a group k defined by an expected reward according with A.

Thus, this function f can be also considered as a non-linear regression function.

Algorithm 2 CTREE algorithm

```
Require: • $\epsilon \in ]0,1[$
    • A dataset of covariates $Y$ and response $X$.
    • An influence function $h$ depending on the scale of $X$.
    • An appropriate function $g_j$, which depends on the scale of the covariate $Y_j$
 1: Calculate the the test statistics $s_{j0}$ for the observed data
 2: Permute the observation in the node
 3: Calculate $s$ for all permutations
 4: Calculate the p-values (number of test statistics $s$, where $|s| > |s_0|$)
 5: Correct p-values for multiples testing
 6: if $H_0$ not rejected (p-value $> \epsilon$ for all $Y_j$) then
 7:     return
 8: else
 9:     Select covariate $Y_j^*$ with the strongest association (smallest p-value)
10:     Search best split for $Y_j^*$ (maximize test statistic $s$) and partition data
11:     Apply CTREE for both of the new partitions
                    Output: An hierarchical partitioning
```

This regression is based on method described in [22] using the test statistic **T** which is derived from [40]. Appendix B gives more details about this method. This function is used during the A/B test (Step 2). As the step 1 is done offline, it does not delay the computational time of Ctree-Ucb.

Fig. 4 shows an example of obtained regression tree.

4.2.2 Step 2. The second step of section consist in the A/B test itself (online). It consists in associating a Ucb bandit to each subgroup to identify the best arms to this group. So, each time a new item is submitted, the f function decides the group the item belongs to.

Then the associated Ucb bandit assigns the item to a arm.As the bandits are independent, each of them can stops the exploration phase at any time and switch to the exploitation mode. The test can then end either when all the bandits are in exploitation mode, or after a given number of items, or a predefined duration. Algor. 3 refers Ctree-Ucb method.

In Appendix C the reader can find more details about theoretical guarantees of Ctree-Ucb.

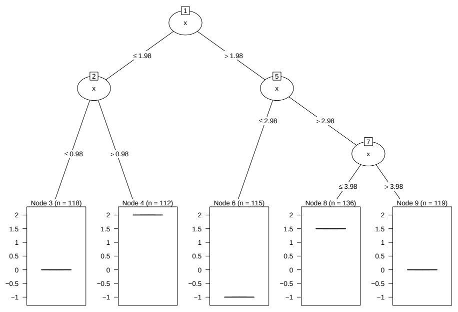
Algorithm 3 Ctree-Ucb algorithm

```
Require: $\alpha > 0$,$L_n$,$\epsilon \in [0, 1]$,
 1: Generate with $\mathcal{L}_n$ a conditional inference tree model : $f$ with an $\epsilon$ accepted error by
    CTREE algorithm.
 2: loop
 3:     $c_t \leftarrow$ a new item with a vector $Y_t$ of covariates
 4:     Assign $c_t$ to group $k$ such as $f(c_t) = k$
 5:     if $T_{a,k}(t) = 0$ then
 6:         $A_{t,k} = a$
 7:     else
 8:         $A_{t,k} = \text{argmax}_{a \in \mathcal{A}} \{\hat{\mu}_{a,k,t} + \alpha\sqrt{\frac{2*log(\sum_{a\in\mathcal{A}} T_{a,k}(t))}{T_{a,k}(t)}}\}$
 9: Assign arm $A_{t,y}$ to $c_t$
10: $X_{c_t,A_t,k} \leftarrow$ the arm $A_{t,k}$ reward
11: Update $\hat{\mu}_{A_t,k}$ and $T_{A_t,k}(t)$
        Output: A sequence of arm choices and rewards for each group $k$
```

4.3 Example on simulated data

Observations made on the data collected via A/B tests indicated that some of the link functions between the rewards and the covariate can be modelled by a continuous function per part. For example, the link between the price of a product and the quantity purchased. If the site offers one item for every 3 items purchased, the linear modelling between the covariate (quantity of item) and the reward no longer holds. In the medical field, if a treatment is effective for young children and elderly people, but not deficient for adults, linear modelling does not work either. However, by using a pairwise function, such cases can be represented. In such a function, the link is linear only over an interval of values taken by the covariate. When the link function between a covariate and a reward is linear or continuous per part, the above-mentioned bandit methods have an increasing cumulative regret. To validate our method, we first propose to simulate data from a pairwise function (2000 covariate x and 2000 rewards) and compare the results between Ctree-Ucb, Lin-Ucb, Ucb and uniform (see Appendix C.3 for more details).

4.3.1 Offline step. We set 30% of the data at $\mathcal{L}_n$ (so the past rewards of A are the only ones observed), and the remaining 70% are used for the test. The regression tree correctly



5 groups are identified: `Node 3`, `Node 4`, `Node 6`, `Node 8` and `Node 9`

Fig. 4. Groups identification on simulated data

identifies each group where the link between the covariate and the reward is identical (each final leaf is a group, represented by an estimated average, see Figure 4).

4.3.2 A/B test (Dynamic allocation). On the A/B test itself, a dynamic allocation is then made for each group. Figure 5 shows the cumulativce regret over time, of different algorithms. Figure 6 to 10 show the cumulative regret of Ctree-Ucb specific to each group. The following section does the same comparison on real datasets.

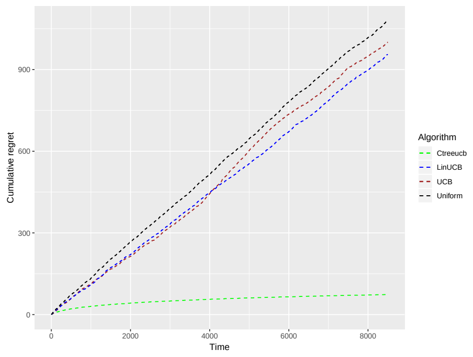


Fig. 5. Cumulative regret of Ctree-Ucb, Lin-Ucb, uniform and Ucb with a non linear reward function

5 MATERIALS AND EXPERIMENTS SETTING

To evaluate the performance of Ctree-Ucb, we compare it to existing A/B methods: a global-based bandit (Ucb), the Lin-Ucb and Kernel-Ucb bandits, as well as a uniform-based algorithm which chooses the variations alternatively. The main criteria for this evaluation are the cumulative and average regret. The experiments are carried out over three data sets. The first one is from the movieLens movie database (small MovieLens, [19]) while the second and third sets come from the company AB Tasty and correspond to an e-merchant A/B tests.

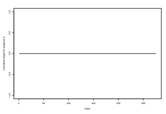


Fig. 6. Node 3

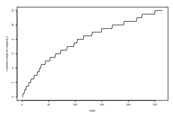


Fig. 7. Node 4

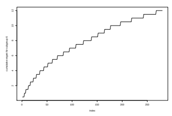


Fig. 8. Node 6

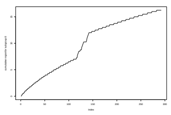


Fig. 9. Node 8

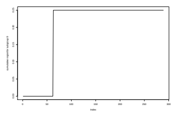


Fig. 10. Node 9

Fig. 11. Cumulative regret of Ctree-Ucb for each group

All experiments are carried out using the R free open language[3]

5.1 Data

5.1.1 Small MovieLens data set. This dataset comes from the MovieLens public database[4]. It contains movies described by 14 binary characteristics (Adventure, Action, Comedy, Drama, Thriller, Romance, Sci-Fi ...) and their associated rates (from 0 to 500) given by film reviewers.

To simulate an A/B test using this data, we define:

- movies as items: There are 9125 movies in the original database
- film reviewers as the variations: denoted by A, B, C, D and E: the 5 reviewers.
- rates as the rewards: the reward of a movie c_t is the rate $X|a, c_t$ given by the reviewer $a \in \{A, B, C, D, E\}$ associated to the variation. In case the film has not been rated by this reviewer (which may appear mainly with recent films), the missing value is evaluated by the global MovieLens average.

The objective is to obtain the best cumulative films evaluation.

5.1.2 AB Tasty datasets. These A/B tests are comparisons of a variation of a web-page with its original state (referred here as P_1 and P_2, respectively).

These tests were performed in 2018 on AB Tasty's servers for several e-merchant clients. It consisted in using a static allocation with an equal distribution of visitors between the two pages P_1 and P_2.

For each test, visitors who navigate on the tested web page (*i.e.*, a potential customer) are identified by an ID: the first time one visits the web page, a new visitor description (see Tab. 2) and its associated ID are generated.

This description may vary depending on the data available to the user. Then a variation (A or B) is allocated to it. A cookie memorizes the description, the ID and the allocated version.

[3]They have been done on a Intel® Core™ i5-8250U CPU 8-processors running at 1.60GHz with 7.5 GB of RAM on Ubuntu 17.10, 64 bits. All materials (including data, conditional tree regression framework CTREE [23] and Ctree-Ucb) are available at: https://github.com/R-workshop-strasbourg/bandit4abtest
[4]These datasets change over time, last update was done in October 2016 but the version of database used in out experiments is available at https://github.com/emmanuelleclaeys

Each time they come back on the tested page, the same variation is shown. So, we assume that no statistical association between visitors exist. We present the results of two A/B tests performed on two different websites

Table 2. Item covariates of A/B Test Dataset

Type	covariates (number of possible value or domain)
Integer	Visits ($\mathbb{N}$)
Categorial	Navigator's language (27), Navigator type (6), Device (3), Operating System (7)

All the actions of the visitors during their visits are stored. At the end of the test (*i.e.*, after T visitors) the reward is computed. According with the user's objective, the reward corresponds to a purchase value by the visitor during all of its visits regardeless the visit(s) the purchase(s) have happened after the assignation. It is defined by cumulative sum if the visitor t has purchased on the web page.

Notations used throughout the paper. For N variations $V_0, V_1, \dots, V_N$, S_i is the set of items to which variation V_i has been allocated, $T_i = |S_i|$ and $T = T_0 + T_1 + \dots + T_N$ where $|.|$ denotes the cardinalty of a set.

5.2 Existing methods to be compared

The performance of Ctree-Ucb is compared with four algorithms:

- Two algorithms with a non informative strategy which do not take into account the item's contexts:
  - uniform which no require any parameter.
  - The Ucb strategy described in Section 3.1.1.
- Two algorithms with a contextual strategy (see Section 3.2.1) :
  - The Lin-Ucb strategy algorithm using linear reliability assumption.
  - The Kernel-Ucb strategy algorithm without use of linear reliability assumption, these policy estimate each variation's reward as well as a kernel regression of characteristics

Each algorithm will provide a sequence of choice. These sequence are compared with a model who always chose the best variation (see section 2.2), trained with all data in the test set, so the cumulative and average regret at the end of the A/B test (time T) is evaluated.

Note that Lin-Ucb and Kernel-Ucb require to transform categorical characteristic into binary ones.

5.3 Experiment protocol

5.3.1 The A/B test parameters. All the algorithm (except uniform) are derived from the standard Ucb algorithm which requires setting the confidence interval parameter defined in Eq. 6, Section 3.1.1. To evaluate impact of the parameter on the results we have been carried out experiments with different values of α (from 0.25 to 2.5, values generally found in the literature).

To limit the CPU time consumed by the Kernel-Ucb algorithm, we have limited the numbers of items used in the kernel regression to 100.

5.3.2 A/B test simulation. The principle of simulation is to apply each algorithm on data sets and to compare their obtained cumulative regrets. We compare the results with a non-linear regression (CTREE) model that learns from all the data. Thus when assigning an item to a variation, regret is evaluated as the difference between the maximum prediction (between all possible variations) and the prediction of the chosen one. This assessment can be seen as a difference between conditional averages and is based on the theoretical definition presented in section 2.

For CTREE-UCB method, the offline step (see Section 5.3.3) consisting to learn a conditional regression tree is firstly done. Then each item is submitted to this tree in order to determinate the group it belongs to. Finally, the item is submitted to the classical UCB algorithm associated to this group.

For each data set, we have considered each variation as a potential original variation. So, for the Movie data set, five configurations have been tested corresponding each to a different choice of movie rate as original variation. Indeed, since the tree is built on the original, the choice of variation A assigns the rest of the process.

5.3.3 Ctree_ucb offline step. The CTREE-UCB offline step which consists in defining the item groups used in the contextual A/B test is crucial. It strongly impacts the A/B test process. To produce these groups, we used the `R` conditional tree regression framework `CTREE` [23] with ten-fold cross validation and different maximum error risk.

To assess this impact, we have carried out experiments with different configurations depending the ratio between the number of items used to learn the regression tree and the one used to simulate the A/B test

Additional notations. $L = |\mathcal{L}|$ where $\mathcal{L}$ is the set used to learn the conditional regression tree. $T_{A/B} = |S_{A/B}|$ where $S_{A/B}$ is the set used to simulate the A/B test. ϵ is the error risk parameter to `CTREE`

To respect the assumption that, before the A/B test, the user only knows the rewards obtained by the original variation (referred here as V_A), the regression tree can only build from the set of items S_A to which variation V_A has been allocated . So $\mathcal{L} \subset S_A$. Two configuration have been tested:

- $\text{Conf}_{30,70}$: $\mathcal{L}$ = 30% of V_0, $S_{A/B} = \sum_i \{70\% \text{ of } S_i\}$
- $\text{Conf}_{100,100}$: $\mathcal{L} = V_0$, $S_{A/B} = \sum_i S_i$

5.3.4 Experiment configurations. Tab. 3 summarizes all the parameters and their different potential values.

Data set	Algorithms	Parameters
MovieLens AB Tasty dataset 1 AB Tasty dataset 2	UCB, LIN-UCB, KERNEL-UCB	$\alpha \in \{0, 0.25, 0.5, 1, 1.5, 2, 2.5\}$
	CTREE-UCB	$V_A \in \{V_i\}$ Config. $\in \{\text{Conf}_{30,70}, \text{Conf}_{100,100}\}$ $\epsilon \in \{0.01, 0.05, 0.1\}$ $\alpha \in \{0, 0.25, 0.5, 1, 1.5, 2, 2.5\}$

Table 3. Algorithm parameters

There are 441 combinations : $3 \times 3 \times 7$ combinations for the 3 UCB-based algorithms and $(2 \times 2 \times 2 \times 3 \times 7) + (5 \times 2 \times 3 \times 7)$ for the CTREE-UCB method

We report only 88 combinations in our experiments for more readability.

For each experiment, a table summarizes the cumulative and average regret of each algorithm. The average regret allows to evaluate the evolution of the cumulative regret compared to the set of tested items (S). **The best performances (cumulative regret / average) appear in bold** in these table.

		CTREE-UCB					LIN-UCB	KERNEL-UCB	UCB	UNIFORM
		V_A : R$_1$	V_A : R$_2$	V_A : R$_3$	V_A : R$_4$	V_A : R$_5$				
Configuration		$\epsilon = 0.05$	$\epsilon = 0.05$	$\epsilon = 0.05$	$\epsilon = 0.05$	$\epsilon = 0.05$				
Config.$_{30,70}$	R_T	**19155**	21628	28458	27894	19976	22378	21239	20650	22808
	$\mathbb{E}[R_T]$	**3**	3.39	4.47	4.37	3.13	3.50	3.32	3.23	3.57
Config.$_{100,100}$	R_T	**27875**	34152	35679	35908	37679	68101	40848	31146	45643
	$\mathbb{E}[R_T]$	**3.05**	3.74	3.91	3.94	4.13	7.46	4.48	4.41	5

Table 4. Influence of segmentation's parameters on the cumulative regret R_T and average regret $\mathbb{E}[R_T]$ with movieLens dataset (with $\alpha = 1$)

6 EXPERIMENTS

6.1 Movie data set

6.1.1 Offline step. On Fig. 12, each leaf is associated the mean and the variance of a group (represented by a boxplot). There is 9 leafs.

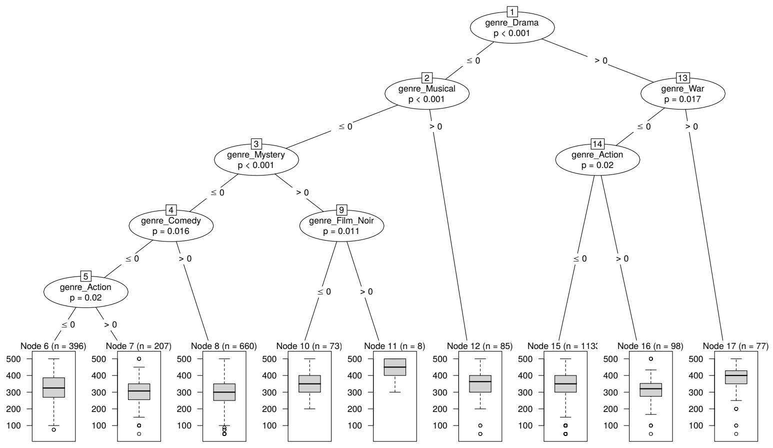


Fig. 12. Conditional inference tree on movieLens dataset (Config.$_{30,70}$, V_A : Reviewer 1, $\epsilon = 0.05$)

6.1.2 A/B test (Dynamic allocation). On MovieLens dataset, the last movies (with less ratings) of the dataset gets generally the same ratings for all reviewers. That is due to replacement of missing values by the average values. This can explain, for some movies, a regret equal to zero whatever the variation chosen. So find the best variation with dynamic allocation is thus difficult (see section 3.1.2). A such situation may decrease the performance of observed bandits and the UNIFORM algorithm gets equal performance in a such situation.

The results in terms of cumulative regret are little impacted by the value of the ϵ parameter, and its impact on results are treated on the next experiments. So, for sake of readability, in Tab. 4, we report only the classical value of the accepted risk of error, *i.e.* $\epsilon = 0.05$. CTREE-UCB presents the lowest cumulative regret (in bold in Tab. 4). The LIN-UCB algorithm presents low performance. One of the explications could that the linearity hypothesis required by this algorithm is not respected. KERNEL-UCB gets a cumulative regret comparable to a frequentist allocation. One explanation can be than its require more data to make its regression. CTREE-UCB gets a better result with reviewer 1 (in bold) with Config.$_{30,70}$ or Config.$_{100,100}$. The learning with reviewer 3 (on Config.$_{30,70}$) or 5 (Config.$_{100,100}$) decrease its performance. We suppose than they implies under or over fitting depending of the configuration.

To obtain good results with Ctree-Ucb, whatever the parameter V_0, the offline step must be done on a population representative of the one to be tested.

However, Fig. 14 shows how α impacts cumulative regret. Regardless the α value, Ctree-Ucb perform better then other methods and guarantees stable results. Opposite to other algorithms, Kernel-Ucb performs differently according to α: Its cumulative regret seems strongly dependent on this parameter. So choosing a sub optimal value for α can make it less efficient than uniform (Fig. 14).

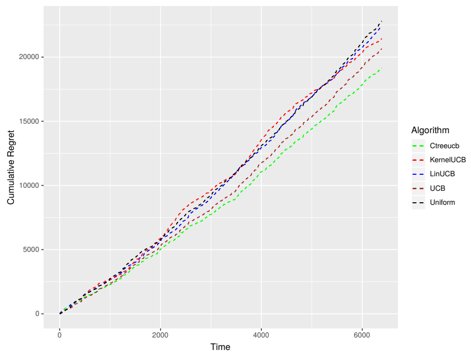


Fig. 13. Cumulative regret with movieLens dataset (V_A : R$_1$, Conf$_{30,70}$, $\epsilon = 0.05$,$\alpha = 0.25$)

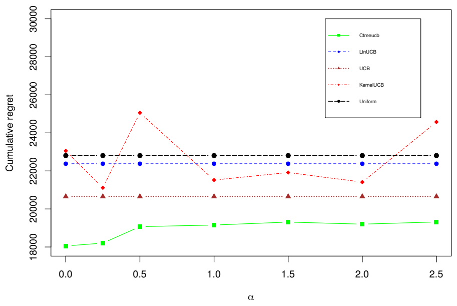


Fig. 14. Cumulative regret according to α with movieLens dataset (V_A : R$_1$, Conf$_{30,70}$, $\epsilon = 0.05$

6.2 AB tasty database 1

6.2.1 Offline step. The user have a clothing sales website and its objective is to increase the value of purchase. On Fig. 15, each leaf is associated an expected reward (a purchase value). On Config.$_{30,70}$ 2543 visitors are dedicated to Step 1 (learning step) and 5934 visitors are tested (Step 2). On Config.$_{100,100}$ steps 1 and 2 have 8477 visitors. There is 10 groups identifed by the first Step. Number of past visits before see the tested page has the strongest correlation with the purchase values. However, purchase value can increase or decrease according to the visitor's user agent or language.

		CTREE-UCB						LIN-UCB	KERNEL-UCB	UCB	UNIFORM
		$V_A : P_1$			$V_A : P_2$						
Configuration		$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$				
$\text{Conf}_{30,70}$	R_T	**100**	**100**	**100**	355	355	355	364	592	408	6610
	$\mathbb{E}[R_T]$	$\mathbf{1.68 * 10^{-2}}$	$\mathbf{1.68 * 10^{-}2}$	$\mathbf{1.68 * 10^{-2}}$	$\mathbf{5.59 * 10^{-}2}$	$\mathbf{5.59 * 10^{-}2}$	$\mathbf{5.59 * 10^{-}2}$	$\mathbf{6.11 * 10^{-}2}$	$\mathbf{9.97 * 10^{-}2}$	$6.87 * 10^{-}2$	1.11
$\text{Conf}_{100,100}$	R_T	152	152	152	67	67	67	24	448	241	8148
	$\mathbb{E}[R_T]$	$\mathbf{1.79 * 10^{-}2}$	$\mathbf{1.79 * 10^{-}2}$	$\mathbf{1.79 * 10^{-}2}$	$\mathbf{0.79 * 10^{-}2}$	$\mathbf{0.79 * 10^{-}2}$	$\mathbf{0.79 * 10^{-}2}$	$\mathbf{0.28 * 10^{-}2}$	$\mathbf{5.28 * 10^{-}2}$	$\mathbf{2.840.79 * 10^{-}2}$	0.96

Table 5. Influence of segmentation's parameters on the cumulative regret and average with AB Tasty dataset 1 ($\alpha = 1$)

6.2.2 A/B test (Dynamic allocation). On Config.$_{30,70}$, all results provided by CTREE-UCB (in bold) are the best. On AB tasty dataset 1, the Tabs 5 gives the cumulative regret according to the different parameters (ϵ and V_A). With all configuration the ϵ parameter doesn't modify the tree structure. However, the challenge in step 1 is to avoid overfitting (to many groups as in Config.$_{100,100}, V\_A : P\_1$ in Tabs. 5) or underfittig (to less group as in Config.$_{30,70}, V\_A : P\_2$ in Tabs. 5). In fact, too few groups lead to performance similar to that of a non-contextual strategy. On the other hand, too many groups slow down the exploration period, but CTREE-UCB results remain more effective than the UCB KERNEL-UCB and UNIFORM.

In this test, variation A was the best for all, therefore (see Fig 16) all bandit algorithms have a logarithmic cumulative regret It also appears that the exploration period would be stop early (after 2000 visitors) with our method than a frequentist approach (UNIFORM) and consequently save money to the user.

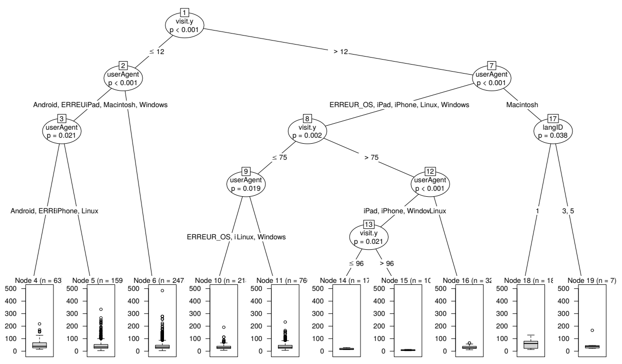


Figure 15. Tree on AB Tasty dataset 1 (Config.$_{30,70}, V\_A : P\_1, \epsilon = 0.05$)

6.3 A/B tasty database 2

6.3.1 Offline step. The user owns a media website, and wants to optimize the purchase values. 800 visitors are dedicated to Step 1 (learning step) and 1865 visitors are tested (Step 2). As the previous experiment, on Fig. 18, each leaf is associated an expected reward (a purchase value). There is 7 groups discovered in the first Step. We note that characteristics present in the previous experience (user agent, visits) have also an influence on groups. On the other hand, there is a new one that appears: the browser (called `name`). Our hypothesis is that since the sale concerns online videos, the user's browser is likely to influence the visual rendition displayed.

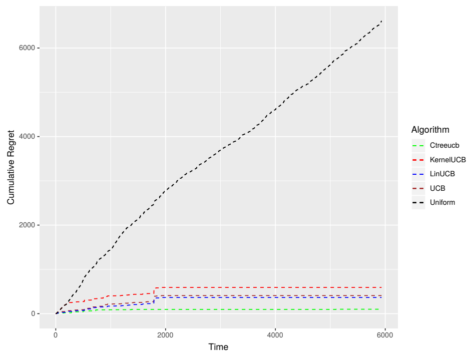


Fig. 16. Cumulative regret according to α with AB Tasty dataset 1 (Conf$_{30,70}$ $V_A = P_1$,$\epsilon = 0.05$)

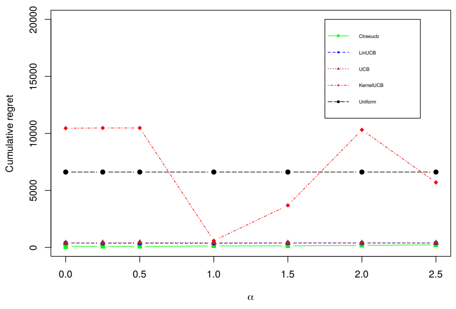


Fig. 17. Cumulative regret according to α with AB Tasty dataset 1 ($V_A : P_1$, Conf$_{30,70}$, $\epsilon = 0.05$)

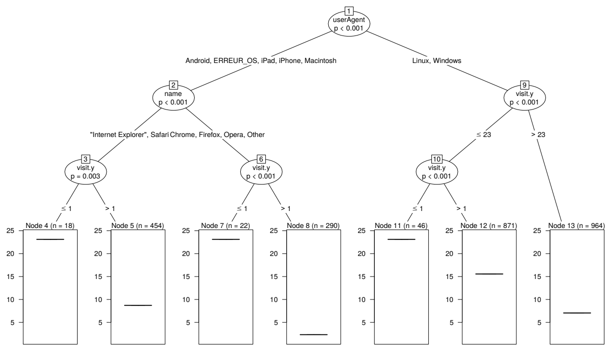


Fig. 18. Tree on abtasty dataset 2 (Config.$_{100,100}, V_A : P_1, \epsilon = 0.05$)

6.3.2 A/B test (Dynamic allocation). Tabs 6 gives the cumulative regret according to the different parameters (ϵ and V_A). On $V_A : P_1$, Conf$_{30,70}$ the tree structure is only slightly modified (occurrence or avoidance of one/two groups maximum). On Conf$_{100,100}$ with all configuration the ϵ parameter dosen't modify the tree structure, see Figure 20.

However, according to $V_A = P_1$ Ctree-Ucb gets the best results whatever the configuration. Nevertheless, on $V_A = P_2$, best results will given by Lin-Ucb. Fig 19 gives the gives the cumulative regret according to α and spots the robust performance of Ctree-Ucb

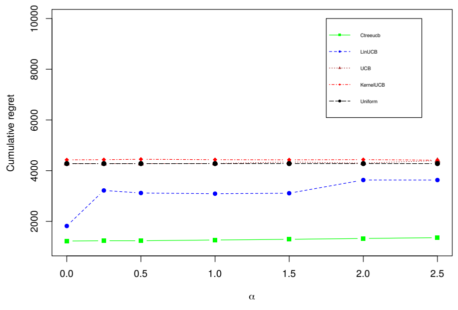


Fig. 19. Cumulative regret according to α with AB Tasty dataset 2 ($V_A : P_1$, Conf$_{30,70}$, $\epsilon = 0.05$)

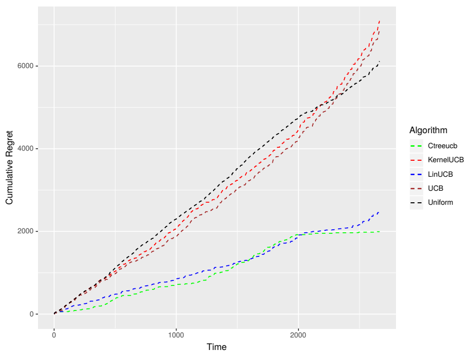


Fig. 20. Cumulative regret with A/B tasty dataset 2 (Conf$_{100,100}$ $V_A = P_1$,$\epsilon = 0.05$,$\alpha = 1$)

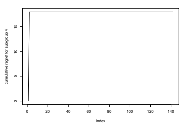
		Ctree-Ucb						Lin-Ucb	Kernel-Ucb	Ucb	uniform
		$V_A : P_1$			$V_A : P_2$						
Configuration		$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$				
Conf$_{30,70}$	R_T	**1251**	1263	1265	3110	3110	3110	3092	4469	4284	4279
	$\mathbb{E}[R_T]$	**0.67**	0.67	0.67	1.67	1.67	1.67	1.67	2.40	2.30	2.30
Conf$_{100,100}$	R_T	**1994**	**1994**	**1994**	3843	3843	3843	2457	7007	6114	6862
	$\mathbb{E}[R_T]$	**0.75**	**0.75**	**0.75**	1.44	1.44	1.44	0.92	2.63	2.30	2.57

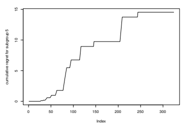
Table 6. Influence of segmentation's parameters on the cumulative regret and average with ab tasty dataset 2. ($\alpha = 1$)

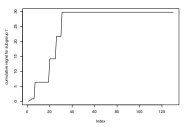
Groups analysis. The cumulative regret of Ctree-Ucb (see Fig. 20) is the sum of the cumulative regret of each group, we propose a more detailed analysis of Ctree-Ucb results by observing the cumulative regret of the 7 subgroups. The parameters of Ctree-Ucb are: (Conf$_{100,100}$ $V_A = P_1$,$\epsilon = 0.05$,$\alpha = 1$)

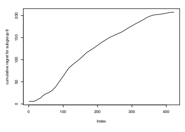
Note that the group names in Fig. 21 refer to the id of the leaf in the tree and not the n-th group.

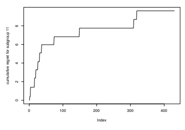
- Figures 21b, 21c and 21e shows the cumulative regret of Group #5, Group #7 and Group #11. In theses figures, the cumulative regret converges asymptotically. For example in Group #7, the first half of the visitors tested produced 100% of the total cumulative regret. For the last visitors their regret is always equal to zero. This result shows for these groups, that CTREE-UCB ends the exploitation in an optimal way. This also suggests that these groups are homogeneous (see section 3.1.2).
- Figures 21a, 21f and 21g shows groups' cumulative regret, almost equal throughout the A/B test. Only few visitors belonging to this group were impacted by the A/B test. These results shows that CTREE-UCB separate unaffected visitors correctly, see section 5.1.2).
- Figure 21d shows a case where the cumulative regret grows almost linearly throughout the A/B test. For this group, the variation chosen for exploration require time or is not the best for all visitors. There are different reasons may explain this: the gap between variations' average is really small, learning from the original page did not correctly identify all possible groups existing in the test dataset, or the characteristics used are not sufficient to give a reliable average for this group.

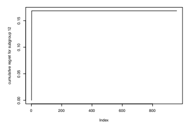

(a) Group #4


(b) Group #5


(c) Group #7


(d) Group #8


(e) Group #11


(f) Group #12

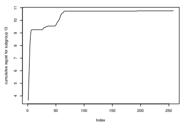

(g) Group #13

Fig. 21. Cumulative regret for some groups with AB tasty Database (Conf$_{100,100}$ $V_A = P_1$,$\epsilon = 0.05$, $\alpha = 1$)

Time response. When the user requires a fast response time (for example for an A/B test that chooses between two web pages, for each visitor the algorithm must give a choice in less than a half millisecond)

The step one of CTREE-UCB is computed before the ab test (offline), the user has the time to compute it. thus, time allocated to step one is therefore not considered. Only the computation time required for dynamic allocation, *i.e.* step 2 (online), is considered. We report at Tab.7 a computation time required for CTREE-UCB and compared it with LIN-UCB, KERNEL-UCB and UCB on Config.$_{100,100}$. We also include the longest calculation time among all groups (max group). As the visitor groups are tested separately, it is possible to distribute the computation time.

After several experiences, the response time per visitor for CTREE-UCB is always less than one millisecond. KERNEL-UCB, on the other hand, requires the longest computation time. This is mainly due to the regression calculation of the kernel. However, LIN-UCB and KERNEL-UCB time response increases with the number of covariates d. CTREE-UCB has a short execution time and respects the response time required. .

	Ctree-Ucb	Lin-Ucb	Kernel-Ucb	Ucb
Total time execution (second)	0.504 (max group : 0.226)	0.622	16.013	0.096
Time execution by visitor (millisecond)	0.18 (max group: 0.08)	0.23	6	0.03

Table 7. Total time calculation of each algorithm (Config.$_{100,100}$) on A/B Tasty dataset 2

7 DISCUSSION

Our experiences have indicated that Ctree-Ucb responds to different A/B test issues. Our results includes continuous or categorical characteristics, continuous reward and a number of possible variations higher than two (A/B/C/...). It shows the performance of Ctree-Ucb for different types of tests.

Ctree-Ucb requires 3 parameters : Conf., V_A, ϵ and α. Our experiences conclude that:

- A partial dataset (Conf$_{30,70}$) for step one is sufficient to obtain results comparable to the total dataset (Conf$_{100,100}$).
- By considering different original variation V_A, the results of Ctree-Ucb may be different. However, Ctree-Ucb remain good compared to result of Lin-Ucb, Kernel-Ucb and Ucb.
- ϵ parameter (associated with the accepted risk in the inference tree) get a lower influence on its result. A 0.05 default value can be set
- an incorrect α value can lead to a degradation of Ucb performance while Ctree-Ucb is less sensitive to the alpha parameter.

The quality result from Ucb and Lin-Ucb are very different according to data type. These algorithm can be equivalent than uniform when their assumptions (like linearity, ...) are not verified, or when the α value is not optimal.

Kernel-Ucb is difficult to use in practice. As Cesa-Bianch *et al.* [8] said "In particular, when the number of kernel evaluations is bounded, there are cases where no algorithm attains performance better than a trivial sub-sampling strategy, where most of the data is thrown away. Also, no algorithm can work well when the regularization parameter is sufficiently small or the norm constraint is sufficiently large."

On the other hand, furthermore, with AB Tasty's datasets, Ctree-Ucb gets the best results. Theses datasets corresponds to our main objective, the other one are mainly there to provide results on public datasets.

8 CONCLUSION

We presented a new contextual algorithm Ctree-Ucb and compared it to synthetic and real data. Ctree-Ucb gets a cumulative regret comparable to the best performance of other state of the heart algorithms. We showed that Ctree-Ucb provides good results whatever parameters are chosen and that a default setting is enough to obtain reliable results. We have also shown that the computation time required for Ctree-Ucb allows its integration in an industrial environment where a fast response time is necessary. Moreover, experiments have shown that the conditional inference tree is a powerful method for achieving group definition, leading to a decrease in cumulative/average regret. In practice, integrating Ctree-Ucb on an A/B test platform for e-commerce websites has presented the following advantages:

- Conditional inference trees are easily interpretable by a user (usually a marketing expert), which allows them to make assumptions for each group. The user identifies "personas": *i.e.* a type of visitor. This makes it easier to imagine variations.

- Conditional inference trees manage categorical data well and this is what is most common on the web in practice. These characteristics can be provided by the visitor (device, browser, etc...) or supplemented by external services (such as Data Management Platform) that provides "marketing" information (focus, gender, age category, etc...). Unlike comparative methods, it's not required to transform categorical data into binary one.
- Creating the tree on data from the original variation is reassuring for the user. There is no feeling of risk since it is the original variation, already in production, that is used. The variation (feared, for fear of losing money) will only be evaluated in a bandit context.
- Sometimes the variation changes are minor; in practice, the test will have an impact only on a few groups. Isolation of items producing so-called "extreme" or no possible rewards in a group. Like a robot that generates many clicks or no purchase . . . Such items can disrupt the exploration period. In fact, our method isolates extreme cases, limiting their negative impact on the whole experience. As the opposite a Lin-Ucb are strongly impacted by extreme values.
- The pre-process selection of characteristics correlated to the response excludes those that are of no interest. Then we also use a non-linear regression approach before the test. This is a trade-off between approaches that learn from scratch but can increase regret and a global approach that limits the interpretation of the A/B test (only one best variation is considered). That leads to a regret decreasing over the test.
- The group construction, performed offline, allows to obtain a response time comparable to a non-contextual method and can be decreased with a distributed computing (like one group per server). Thus, Ctree-Ucb has a very low response time compared to traditional contextual methods. It can be implemented for an A/B e-marketing test or any use case where the user needs a fast response.

The generation of groups by a conditional inference tree is based on statistical tests that accept missing values. Missing data can happen in many practical cases and the use of Ctree-Ucb is a solution to such situations.On the other hand, Ctree-Ucb does not require a large number of items to learn, e.g., in small (30 items) to moderately sized samples (100k items), conditional inference trees ensure that the right-sized tree is grown without additional (post-)pruning or cross-validation.

The good performance of Ctree-Ucb at identifying groups with different original variation suggests a correlation between the distribution of variations. In practice, most of the times, the changes provided by a variation are limited. So we can create groups on variation A and assume that they are similar on variation B.

Our experiment on partial data (first 30%) or on the all dataset shows that the groups are persistent over the time. Moreover, groups identification can help guide the user on the composition of the test itself. If the test was irrelevant for a group, another test can be done, more specific.

9 ACKNOWLEDGES

This research was supported by A/B Tasty company and CNRS institute. We thank Jérémie Mary for assistance with bandit techniques. Thanks to Gaëtan Philipot for proofreading and their support. We would also like to show our gratitude to the Rémi Aubert and Alix de Sagazan for sharing support to us during the course of this research.

REFERENCES

[1] Jean-Yves Audibert, Sébastien Bubeck, and Gábor Lugosi. 2014. Regret in Online Combinatorial Optimization. *Math. Oper. Res.* 39, 1 (Feb. 2014), 31–45. https://doi.org/10.1287/moor.2013.0598

[2] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time Analysis of the Multiarmed Bandit Problem. *Machine Learning* 47, 2 (01 May 2002), 235–256. https://doi.org/10.1023/A:1013689704352

[3] Léon Bottou, Jonas Peters, Joaquin Qui nonero Candela, Denis X. Charles, D. Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. 2013. Counterfactual Reasoning and Learning Systems: The Example of Computational Advertising. *Journal of Machine Learning Research* 14 (2013), 3207–3260. http://jmlr.org/papers/v14/bottou13a.html

[4] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. 1984. *Classification and Regression Trees*. Taylor & Francis.

[5] Sébastien Bubeck and Nicolò Cesa-Bianchi. 2012. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. *Foundations and Trends® in Machine Learning* 5, 1 (2012), 1–122. https://doi.org/10.1561/2200000024

[6] S. Bubeck, V. Perchet, and P. Rigollet. 2013. Bounded regret in stochastic multi-armed bandits. *ArXiv e-prints* (Feb. 2013). arXiv:math.ST/1302.1611

[7] Giuseppe Burtini, Jason Loeppky, and Ramon Lawrence. 2015. A Survey of Online Experiment Design with the Stochastic Multi-Armed Bandit. *CoRR* abs/1510.00757 (2015).

[8] Nicolò Cesa-Bianchi, Yishay Mansour, and Ohad Shamir. 2014. On the Complexity of Learning with Kernels. *CoRR* abs/1411.1158 (2014). arXiv:1411.1158

[9] Olive Jean Dunn. 1961. Multiple Comparisons Among Means. *J. Amer. Statist. Assoc.* 56, 293 (1961), 52–64.

[10] Adam N. Elmachtoub, Ryan McNellis, Sechan Oh, and Marek Petrik. 2017. A Practical Method for Solving Contextual Bandit Problems Using Decision Trees. In *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. http://auai.org/uai2017/proceedings/papers/171.pdf

[11] Adam N. Elmachtoub, Ryan McNellis, Sechan Oh, and Marek Petrik. 2017. A Practical Method for Solving Contextual Bandit Problems Using Decision Trees. *CoRR* abs/1706.04687 (2017). arXiv:1706.04687 http://arxiv.org/abs/1706.04687

[12] Sarah Filippi, Olivier Cappe, Aurélien Garivier, and Csaba Szepesvári. 2010. Parametric Bandits: The Generalized Linear Case. In *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta (Eds.). Curran Associates, Inc., 586–594. http://papers.nips.cc/paper/4166-parametric-bandits-the-generalized-linear-case.pdf

[13] Dylan J. Foster, Alekh Agarwal, Miroslav Dudík, Haipeng Luo, and Robert E. Schapire. 2018. Practical Contextual Bandits with Regression Oracles. In *ICML*.

[14] Raphaël Féraud, Robin Allesiardo, Tanguy Urvoy, and Fabrice Clérot. 2016. Random Forest for the Contextual Bandit Problem. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Arthur Gretton and Christian C. Robert (Eds.), Vol. 51. PMLR, Cadiz, Spain, 93–101. http://proceedings.mlr.press/v51/feraud16.html

[15] Aurélien Garivier and Olivier Cappé. 2011. The KL-UCB Algorithm for Bounded Stochastic Bandits and Beyond. In *Proceedings of the 24th Annual Conference on Learning Theory (Proceedings of Machine Learning Research)*, Sham M. Kakade and Ulrike von Luxburg (Eds.), Vol. 19. PMLR, Budapest, Hungary, 359–376.

[16] Aurélien Garivier, Hédi Hadiji, Pierre Menard, and Gilles Stoltz. 2018. KL-UCB-switch: optimal regret bounds for stochastic bandits from both a distribution-dependent and a distribution-free viewpoints. (May 2018). https://hal.archives-ouvertes.fr/hal-01785705 working paper or preprint.

[17] Author(s) J. C. Gittins and J. C. Gittins. 1979. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society, Series B* (1979), 148–177.

[18] J.C. Gittins and D.M. Jones. 1974. A Dynamic Allocation Index for the Sequential Design of Experiments. In *Progress in Statistics*, J. Gani (Ed.). North-Holland, Amsterdam, 241–266.

[19] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. https://doi.org/10.1145/2827872

[20] Torsten Hothorn, Kurt Hornik, Mark van de Wiel, and Achim Zeileis. 2008. Implementing a Class of Permutation Tests: The coin Package. *Journal of Statistical Software, Articles* 28, 8 (2008), 1–23. https://doi.org/10.18637/jss.v028.i08

[21] Torsten Hothorn, Kurt Hornik, Mark A van de Wiel, and Achim Zeileis. 2006. A Lego System for Conditional Inference. *The American Statistician* 60, 3 (2006), 257–263. https://doi.org/10.1198/

000313006X118430 arXiv:https://doi.org/10.1198/000313006X118430

[22] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. 2006. Unbiased Recursive Partitioning: A Conditional Inference Framework. *Journal of Computational and Graphical Statistics* 15, 3 (2006), 651–674. https://doi.org/10.1198/106186006X133933 arXiv:https://doi.org/10.1198/106186006X133933

[23] Torsten Hothorn, Universität München, Kurt Hornik, Wirtschaftsuniversität Wien, Achim Zeileis, and Wirtschaftsuniversität Wien. [n.d.]. party: A Laboratory for Recursive Partytioning.

[24] M N Katehakis and H Robbins. 1995. Sequential choice from several populations. *Proceedings of the National Academy of Sciences* 92, 19 (1995), 8584–8585. https://doi.org/10.1073/pnas.92.19.8584 arXiv:http://www.pnas.org/content/92/19/8584.full.pdf

[25] E. Kaufmann, O. Cappé, and A. Garivier. 2014. On the Complexity of A/B Testing. *ArXiv e-prints* (May 2014). arXiv:math.ST/1405.3224

[26] T.L Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics* 6, 1 (1985), 4 – 22. https://doi.org/10.1016/0196-8858(85)90002-8

[27] Tor Lattimore and Csaba Szepesvári. 2019. *Bandit Algorithms.* Cambridge University Press (preprint).

[28] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A Contextual-bandit Approach to Personalized News Article Recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 661–670. https://doi.org/10.1145/1772690.1772758

[29] Wei-Yin Loh. [n.d.]. Fifty Years of Classification and Regression Trees. *International Statistical Review* 82, 3 ([n. d.]), 329–348. https://doi.org/10.1111/insr.12016 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/insr.12016

[30] Odalric-Ambrym Maillard and Shie Mannor. 2014. Latent Bandits. Extended version of the paper accepted to ICML 2014 (paper and supplementary material).

[31] John Mingers. 1987. Expert Systems-Rule Induction with Statistical Data. *The Journal of the Operational Research Society* 38, 1 (1987), 39–47.

[32] Toshihiko Morikawa, Akira Terao, and Masakazu Iwasaki. 1996. Power evaluation of various modified bonferroni procedures by a monte carlo study. *Journal of Biopharmaceutical Statistics* 6, 3 (1996), 343–359. https://doi.org/10.1080/10543409608835148 PMID: 8854237.

[33] O. Nicol, J. Mary, and Ph. Preux. 2012. ICML Exploration and Exploitation challenge: Keep it Simple !. In *Journal of Machine Learning Research (JMLR)*.

[34] Yi Qi, Qingyun Wu, Hongning Wang, Jie Tang, and Maosong Sun. 2018. Bandit Learning with Implicit Feedback. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., 7276–7286. http://papers.nips.cc/paper/7958-bandit-learning-with-implicit-feedback.pdf

[35] J. Ross Quinlan. 1993. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[36] Herbert Robbins. 1952. Some aspects of the sequential design of experiments. *Bull. Amer. Math. Soc.* 58, 5 (09 1952), 527–535.

[37] A. Salomon, J.-Y. Audibert, and I. El Alaoui. 2011. Regret lower bounds and extended Upper Confidence Bounds policies in stochastic multi-armed bandit problem. *ArXiv e-prints* (Dec. 2011). arXiv:stat.ML/1112.3827

[38] Steven L. Scott. 2015. Multi-armed Bandit Experiments in the Online Service Economy. *Appl. Stoch. Model. Bus. Ind.* 31, 1 (Jan. 2015), 37–45. https://doi.org/10.1002/asmb.2104

[39] Y.-S Shih. 2004. A note on split selection bias in classification trees. *Computational Statistics & Data Analysis* 45, 3 (2004), 457–466. https://doi.org/10.1016/S0167-9473(03)00064-1

[40] Helmut Strasser and Christian Weber. 1999. On the asymptotic theory of permutation statistics.

[41] Carolin Strobl, James C. Malley, and Gerhard Tutz. 2009. An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods* 14 4 (2009), 323–48.

[42] Richard S. Sutton and Andrew G. Barto. 1998. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks* 16 (1998), 285–286.

[43] William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25, 3-4 (1933), 285–294. https://doi.org/10.1093/biomet/25.3-4.285 arXiv:/oup/backfile/content$_public/journal/biomet/25/3-4/10.1093/biomet/25.3-4.285/2/25-3-4-285.pdf$

[44] Michel Tokic and Günther Palm. 2011. Value-Difference Based Exploration: Adaptive Control between Epsilon-Greedy and Softmax. In *KI 2011: Advances in Artificial Intelligence*, Joscha Bach and Stefan

Edelkamp (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 335–346.

[45] Michal Valko, Nathan Korda, Rémi Munos, Ilias Flaounas, and Nello Cristianini. 2013. Finite-time Analysis of Kernelised Contextual Bandits. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence (UAI'13)*. AUAI Press, Arlington, Virginia, United States, 654–663.

[46] Joannès Vermorel and Mehryar Mohri. 2005. Multi-armed Bandit Algorithms and Empirical Evaluation. In *Machine Learning: ECML 2005*, João Gama, Rui Camacho, Pavel B. Brazdil, Alípio Mário Jorge, and Luís Torgo (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 437–448.

[47] Li Zhou. 2015. A Survey on Contextual Multi-armed Bandits. *CoRR* abs/1508.03326 (2015). arXiv:1508.03326

A UCB ALGORITHM

Algorithm 4 Ucb algorithm

```
Require: $\alpha > 0$
Require: Assign at least one item to each arm $a$
1: loop
2:     $c_t \leftarrow$ a new item
3:     $A_t = \text{argmax}_{a \in \mathcal{A}} \{\hat{\mu}_{a,t} + \alpha\sqrt{\frac{2*log(t)}{T_a(t)}}\}$
4:     Assign arm $A_t$ to $c_t$
5:     $X_{c_t,A_t} \leftarrow$ the arm $A_t$ reward
6:     Update $\hat{\mu}_{A_t}$ and $T_{A_t}(t)$
Output: A sequence of arm choices and rewards
```

B TEST STATISTIC

Ctree-Ucb receives a set of d-dimensional covariates vector Y and associated responses X. The f regression predicts an average representing a single group k. It is assumed that the conditional mean of the response X is related to the linear function of Y through a link function h where $h\{E(X|Y, \mathbb{I}\{f(Y) = k\})\} = \alpha_k + \beta_k^\top Y$, with $\mathbb{I}\{f(Y) = k\}$, the membership variable that returns one if the item belongs to the k-th group and zero otherwise.

Theorem B.1 (**Horthon et al, (2006)**).

$$\boldsymbol{T}_j(\mathcal{L}_n, w) = vec\Big(\sum_{i=1}^{n} w_i g_i (Y_{ij} h(X_i, (X_1, ..., X_n))^\top\Big) \in \mathbb{R}$$

The core of the test statistic can be a matrix. In that case, $vec()$ operator vectorize the matrix. Observation with are not in the current partition will get the weight $w = 0$ and otherwise $w = 1$. g_j is a transformation of the $j-th$ covariate Y_j. h is the influence function: transformation of the response X. The function h and g_j stays the same for each partition, so it is enough to choose them once before the first split.

Conditional expectation $\mu_j \in \mathbb{R}$ and and covariance $\Sigma_j \in \mathbb{R}$ of the $\mathbf{T}_j$ under H_0 given all permutations $\sigma \in S(\mathcal{L}_n, w)$ of the responses are also derived by the framework from [40]:

$$
\begin{aligned}
n_{\text{node}} &= \sum_{i=1}^{n} w_i \\
\mu_j &= \mathbb{E}(\mathbf{T}_j(\mathcal{L}_n, w)|S(\mathcal{L}_n, w)) = \text{vec}\Big(\big(\sum_{i=1}^{n} w_i g_i(Y_{ji})\big)\mathbb{E}[h|S(\mathcal{L}_n, w)]^\top\Big) \\
\Sigma_j &= \mathbb{V}(\mathbf{T}_j(\mathcal{L}_n, w)|S(\mathcal{L}_n, w)) \\
&= \frac{n_{\text{node}}}{n_{\text{node}} - 1}\mathbb{V}[h|S(\mathcal{L}_n, w)] \otimes \big(\sum_{i=1}^{n} w_i g_i(Y_{ji}) \otimes w_i g_i(Y_{ji})^\top\big) \\
&- \frac{n_{\text{node}}}{n_{\text{node}} - 1}\mathbb{V}[h|S(\mathcal{L}_n, w)] \otimes \big(\sum_{i=1}^{n} w_i g_i(Y_{ji})\big) \otimes \big(w_i g_i(Y_{ji})^\top\big)
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}(h|S(\mathcal{L}_n, w)) &= n_{\text{node}}^{-1} \sum_{i=1}^{n} w_i h(X_i, X_1 \ldots X_n)) \in \mathbb{R}^q \\
\mathbb{V}(h|S(\mathcal{L}_n, w)) &= n_{\text{node}}^{-1} \sum_{i=1}^{n} w_i (h(X_i, (X_1 \ldots X_n)) - \mathbb{E}(h|S(\mathcal{L}_n, w)))(h(X_i, (X_1 \ldots X_n)) - \\
&\quad \mathbb{E}(h|S(\mathcal{L}_n, w)))^\top
\end{aligned}
$$

C THEORICAL BOUND OF CTREE-UCB

C.1 Variance reduction in Step 1

We firstly set a continuous regression model in step 1, where both the response X and the covariates Y_j (where $j = 1, ..., d$) are measured on a numeric scale. We define n as the total size of observations. We assumes for simplicity than the influence function h and the transformation function g_j is the identity function, which means the variable will not be transformed at all. Thus $h = X_i$ and $g_j = Y_j$. The formulation $\sum_{i \in \text{node}}$ means sum over all observation in the partition. This is the same as the sum over all observations with additional weights, because only observation in the current node have weight $w = 1$ the other have weights $w = 0$. This yields following the standardized test statistic:

$$
\mathbf{T}_j(\mathcal{L}_n, w) = \sum_{i=1}^{n} w_i X_{j,i} Y_i = \sum_{i \in \text{node}} X_{j,i} Y_i
$$

$n_{\text{node}} := \sum_{i=1}^{n} w_i$ is the number of observation in a node. $\hat{X}_{\text{node}}$ is the mean of in node node. $\hat{Y}_{j,\text{node}}$ is the mean of Y_j in a node.

$(\Sigma)_{ll}$ is the l-th diagonal entry of the co-variance matrix. Univariate test statistics s mapping an observed multivariate linear statistic $\mathbf{t} \in \mathbb{R}^d$ into the real line. It can be done by the maximum of the absolute values of the standardized linear statistics:

$$
s_{\max(\mathbf{t},\mu,\Sigma)} = \max_{l=1,\ldots,d} \Big| \frac{(\mathbf{t} - \mu)_l}{\sqrt{(\Sigma)_{ll}}} \Big| = \Big| \frac{\mathbf{t} - \mu}{\sqrt{\Sigma}} \Big|
$$

$$
\begin{aligned}
\mu_j &= \big(\sum_{i=1}^{n} w_i Y_{ji}\big)\mathbb{E}_{node}[h] = n_{node}\hat{Y}_{j,node}\hat{X}_{node} \\
\Sigma &= \frac{n_{node}}{n_{node}-1}\mathbb{V}_{node}(h)\sum_{i=1}^{n} w_i Y_{ji}^2 - \frac{1}{n_{node}-1}\mathbb{V}_{node}(h) n_{node}^2 \hat{Y}_{j,node}^2 \\
&= \frac{1}{n_{node}-1}\sum_{i\in node}(X_i - \hat{X}_{node})^2 \sum_{i=1}^{n} Y_{ji}^2 \\
&- \frac{1}{n_{node}-1}\frac{1}{n_{node}}\sum_{i\in node}(X_i - \hat{X}_{node})^2 n_{\text{node}}^2 \hat{Y}_{j,\text{node}}^2 \\
&= \frac{1}{n_{node}-1}\sum_{i\in node}(X_i - \hat{X}_{node})^2)(\sum_{i\in node} Y_{j,i}^2 - n_{\text{node}}\hat{Y}_{j,node}^2) \\
&= \frac{1}{n_{node}-1}(\sum_{i\in node}(X_i - \hat{X}_{node})^2)(\sum_{i\in node}(Y_{j,i} - \hat{Y}_{j,i})^2))
\end{aligned}
$$

Therefore:

$$
s \propto \left| \frac{\sum_{i\in node} Y_{ji}X_i - n_{node}\hat{Y}_{j,node}\hat{X}_{node}}{\sqrt{(\sum_{i\in node}(X_{i,j} - \hat{X}_{node})^2)(\sum_{i\in node}(Y_{i,j} - \hat{Y}_{j,node})^2)}} \right| \tag{9}
$$

The linear test statistic is proportional the the linear coefficient of correlation. The next step is to calculate the test statistics (correlation coefficient multiplied by a constant). This will be done with permutation test where if coefficient in eq 9 is very extreme compare to coefficients from randomized permutations, the p-value will be very low.
The splitting critriea is the value where a partition E or $\overline{E}$ is made (inferior or equal for numerical covariates, a set for categorical covariates) with selected covariate j^*. It leads to two different set of n_E (resp: $n_{\overline{E}}$) items for partition E (resp: $\overline{E}$). The statistics used to find the best split is the following:

$$
\mathbf{T}_{j*}^{E}(\mathcal{L}_n, w) = \sum_{i=1}^{n} w_i \mathbb{I}\{Y_{j^*,i} \in E\} X_i = \sum_{i:Y_{j^*,i}\in E} X_i = n_E X_i
$$

$$
\mu_{j^*}^{E} = \sum_{i=1}^{n} w_i \mathbb{I}\{Y_{j^*,i} \in E\}\frac{1}{n_{node}}\sum_{i=1}^{n} w_i X_i = n_E \hat{X}_{node} \tag{10}
$$

$$
\Sigma_{j^*}^{E} = \frac{n_{node}}{n_{node}-1}\frac{1}{n_{node}}\sum_{i=1}^{n} w_i (X_i - \hat{X}_{node})^2 \sum_{i=1}^{n} w_i \mathbb{I}\{Y_{j^*,i} \in E\}^2 \tag{11}
$$

$$
\begin{aligned}
&- \frac{n_{node}}{n_{node}-1}\frac{1}{n_{node}}\sum_{i=1}^{n} w_i (X_i - \hat{X}_{node})^2 (\sum_{i=1}^{n} w_i \mathbb{I}\{Y_{j^*,i} \in E\})^2 \\
&= \frac{1}{n_{node}-1}\sum_{i\in node}(X_i - \hat{X}_{node})^2 n_E (1 - \frac{n_E}{n_{node}})
\end{aligned}
$$

Note that $(1 - \frac{n_E}{n_{node}})$ is actually the probability to be assign to E. Let Z a randomized value according to a binomial distribution such as $Z \sim \mathcal{B}(n_{node}, \frac{n_E}{n_{node}})$

Equation 11 can be rewritten as

$$\begin{aligned}\Sigma_{j^*}^E &= \frac{1}{n_{node} - 1} \sum_{i \in node} (X_i - \hat{X}_{node})^2 n_{node} \frac{n_E}{n_{node}} (1 - \frac{n_E}{n_{node}}) \\ &= \sigma^2_{X_{node}} \sigma^2_Z\end{aligned} \tag{12}$$

Thus, the standardized test statistics is:

$$s_{\max}(\mathbf{t}_{j^*}^E, \mu_{j^*}^E, \Sigma_{j^*}^E) = \max_l \left| \frac{(\mathbf{t}^E - \mu)_l}{\sqrt{(\Sigma)_{ll}}} \right| = n_E \left| \frac{\hat{X}_E - \hat{X}_{node}}{\sqrt{\sigma^2_{X_{node}} \sigma^2_Z}} \right|$$

C.2 Final bound

If a group of items k have a σ_k-subgaussian distribution of reward for a s number of trials with $s \leq n$ and a given $1 - \delta$ probability, the real average reward of an arm a is bounded by:

$$\frac{1}{s} \sum_{t=1}^{s} X_{a,t,k} \leq \mu_{a,k} + \sqrt{\frac{2\sigma^2_{a,k} \log \frac{1}{\delta}}{s}} \tag{13}$$

with traditionally $\delta = \frac{1}{n_y^2}$ [2].

We assume the case where a prediction function assign items to a group k where it's reward follows a σ_k-gaussian distribution regardless of the arm applied. The experiences in the next section indicate that this hypothesis can be confirmed in practice. Define a reward at time t:$X_t = X_{A_t,k}$ and $T_{A_t,k}(t)$ the number of time when the arm A_t in group k have been played. By definition of regret:

$$R_{n,k} = \sum_{a \in \mathcal{A}} \Delta_{a,k} \mathbb{E}[T_{a,k}(n)] \tag{14}$$

.

The regret increase of $\Delta_{a,k}$ when

(a) U.C.B$_{a \neq a^*}(k,t) > \mu_{a^*,k}$

(b) U.C.B$_{a = a^*}(k,t) < \mu_{a^*,k}$

Let's $G_{a,k}$ the event of (a) or (b) define by a constant $u_{a,y}$:

$$G_{a,k} = \{\mu_{a^*,k} < \min_{t \in [n]} \text{U.C.B}_{a=a^*}(k,t)\} \cap \{\hat{\mu}_{a,k,u_i} + \sigma_k \sqrt{\frac{1}{u_a} \log \frac{1}{\delta}} < \mu_{a^*,k}\}$$

where if $G_{a,k}$ happens, the regret converge to a finite value and so if $G_{a,k}$ occur, then $T_{a,k}(n) \leq u_{i,k}$. Let $G^C_{a,k}$ the complementary event.

$$\begin{aligned}\mathbb{E}[T_{a,k}(n)] &= \mathbb{E}[\mathbb{I}\{G_{a,k}\} T_{a,k}(n)] + \mathbb{E}[\mathbb{I}\{G^C_{a,k}\} T_{a,k}(n)] \\ &\leq u_{a,k} + \mathbb{P}[G^C_{a,k}] n\end{aligned}$$

Bound of complementary event $G^C_{a,k}$ is done in two parts:

$$G^C_{a,k} = \{\mu_{a^*,k} \geq \min_{t \in [n]} \text{U.C.B}_{a=a^*}(k,t)\} \cup \{\hat{\mu}_{a,k,u_a} + \sigma_k \sqrt{\frac{2}{u_a} \log \frac{1}{\delta}} \geq \mu_{a^*,k}\}$$

$$\{\mu_{a^*,k} \geq \min_{t\in[n]} \text{U.C.B}(k,t)\} \subset \cup_{s\in[n]}\{\mu_{a^*,k} \geq \hat{\mu}_{a^*,k,s} + \sigma_k\sqrt{\frac{2}{s}\log\frac{1}{\delta}}\}$$

$$\mathbb{P}[\mu_{a^*,k} \geq \min_{t\in[n]} \text{U.C.B}_{a=a^*}(k,t)] \leq \sum_{s=1}^{n}\mathbb{P}[\mu_{a^*,k} \geq \hat{\mu}_{a^*,k,s} + \sigma_k\sqrt{\frac{2}{s}\log\frac{1}{\delta}}] \\ \leq n\delta$$

And so $\mathbb{P}[G^C_{a,k}] \leq n\delta + \mathbb{P}[\hat{\mu}_{a,k,u_a} + \sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geq \mu_{a^*,k}]$. We remind that $\Delta_{a,k} = \mu_{a^*,k} - \mu_{a,k}$ Assume a positive constant c where $\Delta_{a,k} - \sigma_k\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geq c\Delta_{a,k}$:

$$\begin{aligned}\mathbb{P}[\hat{\mu}_{a,k,u_a} + \sigma_k\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geq \mu_{a^*,k}] &= \mathbb{P}[\hat{\mu}_{a,k,u_a} + \sigma_k\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geq \Delta_{a,k} + \mu_{a,k}] \\ &= \mathbb{P}[\hat{\mu}_{a,k,u_a} - \mu_{a,k} \geq \Delta_{a,k} - \sigma_k\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}}] \\ &\leq \mathbb{P}[\hat{\mu}_{a,k,u_a} - \mu_{a,k} \geq c\Delta_{a,k}] \\ &\leq \exp(-\frac{u_a c^2 \Delta^2_{a,k}}{2\sigma^2_k})\end{aligned} \tag{15}$$

The last line of the equation 15 refers to a bounded error for a random variable following a Gaussian distribution. So $\mathbb{P}[G^C_a] \leq n\delta + \exp(-\frac{u_a c^2\Delta^2_{a,k}}{2\sigma^2_k})$. We looking for u_a that satisfy:

$$\Delta_{a,k} - \sigma_y\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geq c\Delta_{a,k} \\ u_a \geq \frac{2\sigma^2_k \log\frac{1}{\delta}}{\Delta^2_{a,k}(1-c)^2}$$

If $\delta = \frac{1}{n^2}$

$$\begin{aligned}\mathbb{E}[T_{a,k}(n)] &\leq \frac{2\sigma^2_k\log\frac{1}{\delta}}{\Delta^2_{a,k}(1-c)^2} + 1 + n\exp(-\frac{u_a c^2\Delta^2_{a,k}}{2\sigma^2_k}) \\ &\leq \frac{2\sigma^2_k\log\frac{1}{\delta}}{\Delta^2_{a,k}(1-c)^2} + 1 + n^{-1+\frac{c^2}{(1+c)^2}}\end{aligned} \tag{16}$$

The function $-1 + \frac{c^2}{(1+c)^2}$ is strictly negative and for any $c < \frac{1}{2}$. The equation 16 can be re written as

$$\mathbb{E}[T_{a,k}(n)] \leq \frac{2\sigma^2_k \log n^2}{\Delta^2_{a,k}\frac{1}{2}} + 2$$

For any group k the regret is bounded by:

$$\begin{aligned} R_{n,k} &= \sum_{a\in\mathcal{A}} \Delta_{a,k}\mathbb{E}[T_{a,k}(n)] \\ &= \sum_{a\in\mathcal{A}} \frac{16\sigma_k^2 \log n^2}{\Delta_{a,k}} + 2\sum_{a\in\mathcal{A}} \Delta_{a,k} \end{aligned}$$

This finally define the cumulative regret for any group k based on past data from $a = A$ (the original variation to improve). If the conditional inference tree define a set of $\mathcal{K}$ admissible groups where $\forall k \in \mathcal{K}$, $\mathcal{N}(\nu_{k,a=A}, \sigma_{k,a=A})$ is statistically different (with an accepted risk of ϵ) and $|\mathcal{K}| = K$, the cumulative regret during the A/B test is

$$\begin{aligned} R_n^{\textsc{Ctree-Ucb}} = \sum_{k\in\mathcal{K}} R_{n,k} &= \sum_{k\in\mathcal{K}} (\sum_{a\in\mathcal{A}} \frac{16\sigma_k^2 \log n^2}{\Delta_{a,k}} + 2\sum_{a\in\mathcal{A}} \Delta_{a,k}) \\ &\le K \max_{k\in\mathcal{K}}[\sigma_k^2](\sum_{a\in\mathcal{A}} \frac{16\log n^2}{\Delta_{a,k}} + 2\sum_{a\in\mathcal{A}} \Delta_{a,y}) \end{aligned}$$

C.3 Simulation

We report an example of a simulation to test the performance of Ctree-Ucb under real assumptions. For each variation (A or B), 10000 rewards are generated by the following function, related to a covariate x:

$$\begin{aligned} \theta_A &= (2, -1, 1.5, 0) \\ \theta_B &= (1.5, -0.5, 1.25, 0) \end{aligned}$$

$$X = \left\{ \begin{array}{ccc} \text{if } 1 \le x_1 < 2 & X_A = \theta_A[1] & X_B = \theta_B[1] \\ \text{if } 3 \le x_1 < 4 & X_A = \theta_A[2] & X_B = \theta_B[2] \\ \text{if } x_1 \ge 4 & X_A = \theta_A[3] & X_B = \theta_B[3] \\ \text{if } x_1 < 1 & X_A = \theta_A[4] & X_B = \theta_B[4] \end{array} \right\}$$