A Practical Method for Solving Contextual Bandit Problems Using Decision Trees

Adam N. Elmachtoub[*]
IEOR Department
Columbia University

Ryan McNellis
IEOR Department
Columbia University

Sechan Oh
Moloco
Palo Alto, CA 94301

Marek Petrik
Dept. of Computer Science
University of New Hampshire

Abstract

Many efficient algorithms with strong theoretical guarantees have been proposed for the contextual multi-armed bandit problem. However, applying these algorithms in practice can be difficult because they require domain expertise to build appropriate features and to tune their parameters. We propose a new method for the contextual bandit problem that is simple, practical, and can be applied with little or no domain expertise. Our algorithm relies on decision trees to model the context-reward relationship. Decision trees are non-parametric, interpretable, and work well without handcrafted features. To guide the exploration-exploitation trade-off, we use a bootstrapping approach which abstracts Thompson sampling to non-Bayesian settings. We also discuss several computational heuristics and demonstrate the performance of our method on several datasets.

1 INTRODUCTION

Personalized recommendation systems play a fundamental role in an ever-increasing array of settings. For example, many mobile and web services earn revenue primarily from targeted advertising (Yuan and Tsao, 2003; Dhar and Varshney, 2011). The effectiveness of this strategy is predicated upon intelligently assigning the right ads to users based on contextual information. Other examples include assigning treatments to medical patients (Kim et al., 2011) and recommending web-based content such as news articles to subscribers (Li et al., 2010).

In this paper, we tackle the problem where little or no prior data is available, and an algorithm is required to "learn" a good recommendation system in real time as users arrive and data is collected. This problem, known as the *contextual bandit problem* (or contextual multi-armed bandit problem), relies on an algorithm to navigate the *exploration-exploitation* trade-off when choosing recommendations. Specifically, it must simultaneously exploit knowledge from data accrued thus far to make high-impact recommendations, while also exploring recommendations which have a high degree of uncertainty in order to make better decisions in the future.

The contextual bandit problem we consider can be formalized as follows (Langford and Zhang, 2008). At each time point, a user arrives and we receive a vector of information, henceforth referred to as the *context*. We must then choose one of K distinct *actions* for this user. Finally, a random outcome or *reward* is observed, which is dependent on the user and action chosen by the algorithm. Note that the probability distribution governing rewards for each action is unknown and depends on the observed context. The objective is to maximize the total rewards accrued over time. In this paper, we focus on settings in which rewards are binary, observing a *success* or a *failure* in each time period. However, our algorithm does not rely on this assumption and may also be used in settings with continuous reward distributions.

A significant number of algorithms have been proposed for the contextual bandit problem, often with strong theoretical guarantees (Auer, 2002; Filippi et al., 2010; Chu et al., 2011; Agarwal et al., 2014). However, we believe that many of these algorithms cannot be straightforwardly and effectively applied in practice for personalized recommendation systems, as they tend to exhibit at least one of the following drawbacks.

1. *Parametric modeling assumptions*. The vast majority of bandit algorithms assume a parametric relationship between contexts and rewards (Li et al., 2010; Filippi et al., 2010; Abbasi-Yadkori et al., 2011; Agrawal and Goyal, 2013). To use such al-

[*]Authors are listed alphabetically.

gorithms in practice, one must first do some feature engineering and transform the data to satisfy the parametric modeling framework. However, in settings where very little prior data is available, it is often unclear how to do so in the right way.

2. *Unspecified constants in the algorithm*. Many bandit algorithms contain unspecified parameters which are meant to be tuned to control the level of exploration (Auer, 2002; Li et al., 2010; Filippi et al., 2010; Allesiardo et al., 2014). Choosing the wrong parameter values can negatively impact performance (Russo and Van Roy, 2014), yet choosing the right values is difficult since little prior data is available.

3. *Ill-suited learners for classification problems*. It is commonly assumed in the bandit literature that the relationship between contexts and rewards is governed by a linear model (Li et al., 2010; Abbasi-Yadkori et al., 2011; Agrawal and Goyal, 2013). Although such models work well in regression problems, they often face a number of issues when estimating probabilities from binary response data (Long, 1997).

In the hopes of addressing all of these issues, we propose a new algorithm for the contextual bandit problem which we believe can be more effectively applied in practice. Our approach uses decision tree learners to model the context-reward distribution for each action. Decision trees have a number of nice properties which make them effective, requiring no data modification or user input before being fit (Friedman et al., 2001). To navigate the exploration-exploitation tradeoff, we use a parameter-free bootstrapping technique that emulates the core principle behind Thompson sampling. We also provide a computational heuristic to improve the speed of our algorithm. Our simple method works surprisingly well on a wide array of simulated and real-world datasets compared to well-established algorithms for the contextual bandit problem.

2 LITERATURE REVIEW

Most contextual bandit algorithms in the existing literature can be categorized along two dimensions: (i) the base learner and (ii) the exploration algorithm. Our method uses decision tree learners in conjunction with bootstrapping to handle the exploration-exploitation trade-off. To the best of our knowledge, the only bandit algorithm which applies such learners is BanditForest (Féraud et al., 2016), using random forests as the base learner with decision trees as a special case. One limitation of the algorithm is that it depends on four problem-specific parameters requiring domain expertise to 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. By contrast, our algorithm requires no tunable parameters in its exploration and chooses the depth internally when building the tree. Further, BanditForest must sample actions uniformly-at-random until all trees are completely learned with respect to a particular context. As our numerical experiments show, this leads to excessive selection of low-reward actions, causing the algorithm's empirical performance to suffer. NeuralBandit (Allesiardo et al., 2014) is an algorithm which uses neural networks as the base learner. Using a probability specified by the user, it randomly decides whether to explore or exploit in each step. However, choosing the right probability is rather difficult in the absence of data.

Rather than using non-parametric learners such as decision trees and neural nets, the vast majority of bandit algorithms assume a parametric relationship between contexts and rewards. Commonly, such learners assume a monotonic relationship in the form of a linear model (Li et al., 2010; Abbasi-Yadkori et al., 2011; Agrawal and Goyal, 2013) or a Generalized Linear Model (Filippi et al., 2010; Li et al., 2011). However, as with all methods that assume a parametric structure in the context-reward distribution, manual transformation of the data is often required in order to satisfy the modeling framework. In settings where little prior data is available, it is often quite difficult to "guess" what the right transformation might be. Further, using linear models in particular can be problematic when faced with binary response data, as such methods can yield poor probability estimates in this setting (Long, 1997).

Our method uses bootstrapping in a way which "approximates" the behavior of Thompson sampling, an exploration algorithm which has recently been applied in the contextual bandit literature (Agrawal and Goyal, 2013; Russo and Van Roy, 2014). Detailed in Section 4.2, Thompson sampling is a Bayesian framework requiring a parametric response model, and thus cannot be straightforwardly applied when using decision tree learners. The connection between bootstrapping and Thompson sampling has been previously explored in Eckles and Kaptein (2014), Osband and Van Roy (2015), and Tang et al. (2015), although these papers either focus on the context-free case or only consider using parametric learners with their bootstrapping framework. Baransi et al. (2014) propose a sub-sampling procedure which is related to Thompson sampling, although the authors restrict their attention to the context-free case.

Another popular exploration algorithm in the contextual bandit literature is Upper Confidence Bounding (UCB) (Auer, 2002; Li et al., 2010; Filippi et al., 2010). These methods compute confidence intervals around expected reward estimates and choose the action with the highest upper confidence bound. UCB-type algorithms often rely on a tunable parameter controlling the width of the confidence interval. Choosing the right parameter value can have a huge impact on performance, but with little problem-specific data available this can be quite challenging (Russo and Van Roy, 2014). Another general exploration algorithm which heavily relies on user input is Epoch-Greedy (Langford and Zhang, 2008), which depends on an unspecified (non-increasing) function to modulate between exploration and exploitation.

Finally, a separate class of contextual bandit algorithms are those which select the best policy from an exogenous finite set, such as Dudik et al. (2011) and Agarwal et al. (2014). The performance of such algorithms depends on the existence of a policy in the set which performs well empirically. In the absence of prior data, the size of the required set may be prohibitively large, resulting in poor empirical and computational performance. Furthermore, in a similar manner as UCB, these algorithms require a tunable parameter that influences the level of exploration.

3 PROBLEM FORMULATION

In every time step $t = 1, \ldots, T$, a user arrives with an M-dimensional context vector $x_t \in \mathbb{R}^M$. Using x_t as input, the algorithm chooses one of K possible actions for the user at time t. We let $a_t \in \{1, ..., K\}$ denote the action chosen by the algorithm. We then observe a random, binary reward $r_{t,a_t} \in \{0, 1\}$ associated with the chosen action a_t and context x_t. Our objective is to maximize the cumulative reward over the time horizon.

Let $p(a, x)$ denote the (unknown) probability of observing a positive reward, given that we have seen context vector x and offered action a. We define the *regret* incurred at time t as

$$\max_a E[r_{t,a}] - E[r_{t,a_t}] = \max_a p(a, x_t) - p(a_t, x_t)\,.$$

Intuitively, the regret measures the expected difference in reward between a candidate algorithm and the best possible assignment of actions. An equivalent objective to maximizing cumulative rewards is to minimize the T-period *cumulative regret*, $R(T)$, which is defined as

$$R(T) = \sum_{t=1}^{T} \max_a p(a, x_t) - p(a_t, x_t)\,.$$

In our empirical studies, we use cumulative regret as the performance metric for assessing our algorithm.

4 PRELIMINARIES

4.1 DECISION TREES

Our method uses decision tree learners in modeling the relationship between contexts and success probabilities for each action. Decision trees have a number of desirable properties: they handle both continuous and binary response data efficiently, are robust to outliers, and do not rely on any parametric assumptions about the response distribution. Thus, little user configuration is required in preparing the data before fitting a decision tree model. They also have the benefit of being highly interpretable, yielding an elegant visual representation of the relationship between contexts and rewards (Friedman et al., 2001). Figure 1 provides a diagram of a decision tree model for a particular sports advertisement. Observe that the tree partitions the context space into different regions, referred to as *terminal nodes* or *leaves*, and we assume that each of the users belonging to a certain leaf has the same success probability.

There are many efficient algorithms proposed in the literature for estimating decision tree models from training data. In our numerical experiments, we used the CART algorithm with pruning as described by Breiman et al. (1984). Note that this method does not rely on any tunable parameters, as the depth of the tree is selected internally using cross-validation.

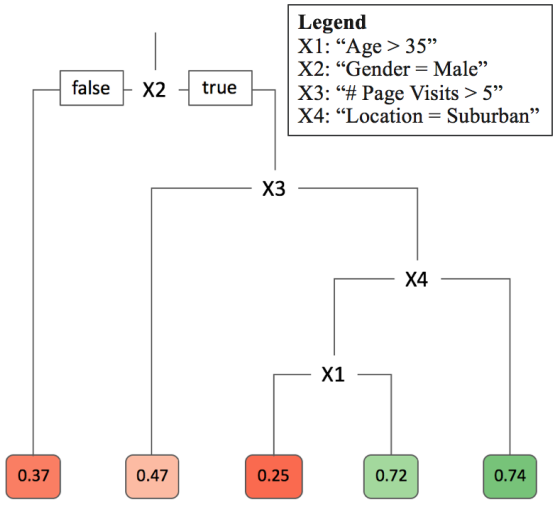


Figure 1: A decision tree modeling the distribution of rewards for a golf club advertisement. Terminal nodes display the success probability corresponding to each group of users.

4.2 THOMPSON SAMPLING FOR CONTEXTUAL BANDITS

Our algorithm was designed to mimic the behavior of Thompson sampling, a general algorithm for handling the exploration-exploitation trade-off in bandit problems. In addition to having strong performance guarantees relative to UCB (Russo and Van Roy, 2014), Thompson sampling does not contain unspecified constants which need to be tuned for proper exploration. Thus, there is sufficient motivation for using this method in our setting.

In order to provide intuition for our algorithm, we describe how Thompson sampling works in the contextual bandit case. To begin, assume that each action a has a set of unknown parameters θ_a governing its reward distribution, $P(r_a|\theta_a, x_t)$. For example, when using linear models with Thompson sampling, $E[r_a|\theta_a, x_t] = x_t'\theta_a$. We initially model our uncertainty about θ_a using a prespecified prior distribution, $P(\theta_a)$. As new rewards are observed for action a, we update our model accordingly to its so-called posterior distribution, $P(\theta_a|D_{t,a})$. Here, $D_{t,a}$ represents the set of context/reward pairs corresponding to times up to t that action a was offered, i.e., $D_{t,a} = \{(x_s, r_{s,a}) : s \leq t-1, a_s = a\}$.

Thompson sampling behaves as follows. During each time step, every action's parameters θ_a are first randomly sampled according to posterior distribution $P(\theta_a|D_{t,a})$. Then, Thompson sampling chooses the action which maximizes expected reward with respect to the sampled parameters. In practice, this can be implemented according to the pseudocode given in Algorithm 1.

Algorithm 1: ThompsonSampling()

for $t = 1, ..., T$ **do**
  Observe context vector x_t
  for $a = 1, ..., K$ **do**
    Sample $\tilde{\theta}_{t,a}$ from $P(\theta_a|D_{t,a})$
  end
  Choose action $a_t = \arg\max_a E[r_a|\tilde{\theta}_{t,a}, x_t]$
  Update D_{t,a_t} and $P(\theta_{a_t}|D_{t,a_t})$ with (x_t, r_{t,a_t})
end

5 THE BOOTSTRAPPING ALGORITHM

5.1 BOOTSTRAPPING TO CREATE A "THOMPSON SAMPLE"

As decision trees are inherently non-parametric, it is not straightforward to mathematically define priors and posteriors with respect to these learners, making Thompson sampling difficult to implement. However, one can use bootstrapping to simulate the behavior of sampling from a posterior distribution, the intuition for which is discussed below.

Recall that data $D_{t,a}$ is the set of observations for action a at time t. If we had access to many i.i.d. datasets of size $|D_{t,a}|$ for action a, we could fit a decision tree to each one and create a *distribution* of trees. A Thompson sample could then be generated by sampling a random tree from this collection. Although assuming access to these datasets is clearly impractical in our setting, we can use bootstrapping to approximate this behavior. We first create a large number of bootstrapped datasets, each one formed by sampling $|D_{t,a}|$ context-reward pairs from $D_{t,a}$ with replacement. Then, we fit a decision tree to each of these bootstrapped datasets. A "Thompson sample" is then a randomly selected tree from this collection. Of course, an equivalent and computationally more efficient procedure is to simply fit a decision tree to a *single* bootstrapped dataset. This intuition serves as the basis for our algorithm.

Following Tang et al. (2015), let $\tilde{D}_{t,a}$ denote the dataset obtained by bootstrapping $D_{t,a}$, i.e. sampling $|D_{t,a}|$ observations from $D_{t,a}$ with replacement. Denote the decision tree fit on $\tilde{D}_{t,a}$ by $\tilde{\theta}_{t,a}$. Finally, let $\hat{p}(\tilde{\theta}, x_t)$ denote the estimated probability of success from using decision tree $\tilde{\theta}$ on context x_t.

At each time point, the bootstrapping algorithm simply selects the action which maximizes $\hat{p}(\tilde{\theta}_{t,a}, x_t)$. See Algorithm 2 for the full pseudocode. Although our method is given with respect to decision tree models, note that this bootstrapping framework can be used with any base learner, such as logistic regression, random forests, and neural networks.

Algorithm 2: TreeBootstrap()

for $t = 1, ..., T$ **do**
  Observe context vector x_t
  for $a = 1, ..., K$ **do**
    Sample bootstrapped dataset $\tilde{D}_{t,a}$ from $D_{t,a}$
    Fit decision tree $\tilde{\theta}_{t,a}$ to $\tilde{D}_{t,a}$
  end
  Choose action $a_t = \arg\max_a \hat{p}(\tilde{\theta}_{t,a}, x_t)$
  Update D_{t,a_t} with (x_t, r_{t,a_t})
end

Observe that TreeBootstrap may eliminate an action after a single observation if its first realized reward is a failure, as the tree constructed from the resampled dataset in subsequent iterations will always estimate a success proba-

bility of zero. There are multiple solutions to address this issue. First, one can force the algorithm to continue offering each action a until a success is observed (or an action a is eliminated after a certain threshold number of failures). This is the approach used in our numerical experiments. Second, one can add fabricated prior data of one success and one failure for each action, where the associated context for the prior is that of the first data point observed. The prior data prohibit the early termination of arms, and their impact on the prediction accuracy becomes negligible as the number of observations increases.

5.2 MEASURING THE SIMILARITY BETWEEN BOOTSTRAPPING AND THOMPSON SAMPLING

Here we provide a simple result that quantifies how close the Thompson sampling and bootstrapping algorithms are in terms of the actions chosen in each time step. Measuring the closeness of the two algorithms in the contextual bandit framework is quite challenging, and thus we focus on the standard (context-free) multi-armed bandit problem in this subsection. Suppose that the reward from choosing each action, i.e., $r_{t,a}$, follows a Bernoulli distribution with an unknown success probability. At a given time t, let n_a denote the total number of times that action a has been chosen, and let p_a denote the proportion of successes observed for action a.

In standard multi-armed bandits with Bernoulli rewards, Thompson sampling first draws a random sample of the true (unknown) success probability for each action a according to a Beta distribution with parameters $\alpha_a = n_a p_a$ and $\beta_a = n_a(1 - p_a)$. It then chooses the action with the highest sampled success probability. Conversely, the bootstrapping algorithm samples n_a observations with replacement from action a's observed rewards, and the generated success probability is then the proportion of successes observed in the bootstrapped dataset. Note that this procedure is equivalent to generating a binomial random variable with number of trials n_a and success rate p_a, divided by n_a.

In Theorem 1 below, we bound the difference in the probability of choosing action a when using bootstrapping versus Thompson sampling. For simplicity, we assume that we have observed at least one success and one failure for each action, i.e. $n_a \geq 2$ and $p_a \in (0, 1)$ for all a.

Theorem 1. *Let a_t^{TS} be the action chosen by the Thompson sampling algorithm, and let a_t^B be the action chosen by the bootstrapping algorithm given data (n_a, p_a) for each action $a \in \{1, 2, \ldots, K\}$. Then,*

$$|P(a_t^{TS} = a) - P(a_t^B = a)| \leq C_a(p_1, ..., p_K) \sum_{a=1}^{K} \frac{1}{\sqrt{n_a}}$$

holds for every $a \in \{1, 2, \ldots, K\}$, for some function $C_a(p_1, ..., p_K)$ of $p_1, ..., p_K$.

Note that both algorithms will sample each action infinitely often, i.e. $n_a \to \infty$ as $t \to \infty$ for all a. Hence, as the number of time steps increases, the two exploration algorithms choose actions according to increasingly similar probabilities. Theorem 1 thus sheds some light onto how quickly the two algorithms converge to the same action selection probabilities. A full proof is provided in the Supplementary Materials section.

5.3 EFFICIENT HEURISTICS

Note that TreeBootstrap requires fitting K decision trees from scratch at each time step. Depending on the method used to fit the decision trees as well as the size of M, K, and T, this can be quite computationally intensive. Various online algorithms have been proposed in the literature for training decision trees, referred to as Incremental Decision Trees (IDTs) (Crawford, 1989; Utgoff, 1989; Utgoff et al., 1997). Nevertheless, Algorithm 2 does not allow for efficient use of IDTs, as the bootstrapped dataset for an action significantly changes at each time step.

However, one could modify Algorithm 2 to instead use an *online* method of bootstrapping. Eckles and Kaptein (2014) propose a different bootstrapping framework for bandit problems which is more amenable to online learning algorithms. Under this framework, we begin by initializing B null datasets for every action, and new context-reward pairs are added in real time to each dataset with probability $1/2$. We then simply maintain $K \times B$ IDTs fit on each of the datasets, and a "Thompson sample" then corresponds to randomly selecting one of an action's B IDTs. Note that there is an inherent tradeoff with respect to the number of datasets per action, as larger values of B come with both higher approximation accuracy to the original bootstrapping framework and increased computational cost.

We now propose a heuristic which only requires maintaining one IDT per action, as opposed to $K \times B$ IDTs. Moreover, only one tree update is needed per time period. The key idea is to simply maintain an IDT $\hat{\theta}_{t,a}$ fit on each action's dataset $D_{t,a}$. Then, using the leaves of the trees to partition the context space into regions, we treat each region as a standard multi-arm bandit (MAB) problem. More specifically, let $N_1(\hat{\theta}_{t,a}, x_t)$ denote the number of successes in the leaf node of $\hat{\theta}_{t,a}$ corresponding to

x_t, and analogously define $N_0(\hat{\theta}_{t,a}, x_t)$ as the number of failures. Then, we simply feed this data into a standard MAB algorithm, where we assume action a has observed $N_1(\hat{\theta}_{t,a}, x_t)$ successes and $N_0(\hat{\theta}_{t,a}, x_t)$ failures thus far. Depending on the action we choose, we then update the corresponding IDT of that action and proceed to the next time step.

Algorithm 3 provides the pseudocode for this heuristic using the standard Thompson sampling algorithm for multi-arm bandits. Note that this requires a prior number of successes and failures for each context region, S_0 and F_0. In the absence of any problem-specific information, we recommend using the uniform prior $S_0 = F_0 = 1$.

Algorithm 3: TreeHeuristic()

for $t = 1, ..., T$ **do**
  Observe context vector x_t
  for $a = 1, ..., K$ **do**
    Sample $TS_{t,a} \sim$
      Beta$(N_1(\hat{\theta}_{t,a}, x_t) + S_0, N_0(\hat{\theta}_{t,a}, x_t) + F_0)$
  end
  Choose action $a_t = \arg\max_a \ TS_{t,a}$
  Update tree $\hat{\theta}_{t,a_t}$ with (x_t, r_{t,a_t})
end

Both TreeBootstrap and TreeHeuristic are algorithms which aim to emulate the behavior of Thompson Sampling. TreeHeuristic is at least $O(K)$ times faster computationally than TreeBootstrap, as it only requires refitting one decision tree per time step – the tree corresponding to the sampled action. However, TreeHeuristic sacrifices some robustness in attaining these computational gains. In each iteration of TreeBootstrap, a new decision tree is resampled for every action to account for two sources of uncertainty: (a) global uncertainty in the tree structure (i.e. are we splitting on the right variables?) and (b) local uncertainty in each leaf node (i.e., are we predicting the correct probability of success in each leaf?). Conversely, TreeHeuristic keeps the tree structures fixed and only resamples the data in leaf nodes corresponding to the current context – thus, TreeHeuristic only accounts for uncertainty (b), not (a). Note that if both the tree structures *and* the leaf node probability estimates were kept fixed, this would amount to a pure exploitation policy.

6 EXPERIMENTAL RESULTS

We assessed the empirical performance of our algorithm using the following sources of data as input:

1. A simulated "sports ads" dataset with decision trees for each offer governing reward probabilities.
2. Four classification datasets obtained from the UCI Machine Learning Repository (Lichman, 2013): *Adult*, *Statlog (Shuttle)*, *Covertype*, and *US Census Data (1990)*.

We measured the cumulative regret incurred by TreeBootstrap on these datasets, and we compare its performance with that of TreeHeuristic as well as several benchmark algorithms proposed in the bandit literature.

6.1 BENCHMARK ALGORITHMS

We tested the following benchmarks in our computational experiments:

1. *Context-free MAB*. To demonstrate the value of using contexts in recommendation systems, we include the performance of a context-free multi-arm bandit algorithm as a benchmark. Specifically, we use context-free Thompson sampling in our experiments.
2. *BanditForest*. To the best of our knowledge, BanditForest is the only bandit algorithm in the literature which uses decision tree learners. Following the approach used in their numerical studies, we first recoded each continuous variable into five binary variables before calling the algorithm. Note this is a necessary preprocessing step, as the algorithm requires all contexts to be binary. The method contains two tunable parameters which control the level of exploration, δ and ϵ, which we set to the values tested in their paper: $\delta = 0.05$, and $\epsilon \sim$ Uniform$(0.4, 0.8)$. Additionally, two other tunable parameters can be optimized: the depth of each tree, D, and the number of trees in the random forest, L. We report the values of these parameters which attained the best cumulative regret with respect to our time horizon: $L = 3$ and $D = 4, 2, 4, 5$, and 2 corresponding to the simulated sports-advertising dataset, *Adult*, *Shuttle*, *Covertype*, and *Census*, respectively. Note that the optimal parameter set varies depending on the dataset used. In practice, one cannot know the optimal parameter set in advance without any prior data, and so the performance we report may be optimistic.
3. *LinUCB*. Developed by Li et al. (2010), LinUCB is one of the most cited contextual bandit algorithms in the recent literature. The method calls for fitting a ridge regression model on the context-reward data for each action (with regularization parameter $\lambda = 1$). We then choose the action with the highest upper confidence bound with respect to a new context's estimated probability of success. All contexts

were scaled to have mean 0 and variance 1 before calling the algorithm, and all categorical variables were binarized. Due to the high-dimensionality of our datasets (particularly after binarization), the predictive linear model requires sufficient regularization to prevent overfitting. In the hopes of a fairer comparison with our algorithm, we instead select the regularization parameter from a grid of candidate values using cross-validation. We report the best cumulative regret achieved when varying the UCB constant α among a grid of values from 0.0001 to 10. Similar to the BanditForest case, the optimal parameter depends on the dataset.

4. *LogisticUCB*. As we are testing our algorithms using classification data, there is significant motivation to use a logistic model, as opposed to a linear model, in capturing the context-reward relationship. Filippi et al. (2010) describe a bandit algorithm using a generalized linear modeling framework, of which logistic regression is a special case. However, the authors tackle a problem formulation which is slightly different than our own. In their setting, each *action* has an associated, non-random context vector x_a, and the expected reward is a function of a single unknown parameter θ : $E[r_t|x_a] = \mu(x_a^T\theta)$ (here, μ is the so-called inverse link function). Extending their algorithm to our setting, we give the full pseudocode of LogisticUCB in the Supplementary Materials section. We take all of the same preprocessing steps as in LinUCB, and we report the cumulative regret corresponding to the *best* UCB constant. For the same reasons as above, we use regularized logistic regression in practice with cross-validation to tune the regularization parameter.

5. *OfflineTree*. Recall that the simulated sports-advertising dataset was constructed using decision tree truths. Thus, it is meaningful to compare TreeBootstrap with a regret of zero, as it is possible in theory for estimated decision trees to capture the truth exactly. However, as the four UCI datasets are all composed of real observations, there will always be part of the "truth" which decision trees cannot capture. Our benchmark OfflineTree measures this error, serving as a meaningful *lower bound* against which we can compare our algorithm. Described in detail in Algorithm 4, it can essentially be thought of as the offline classification error of decision trees with respect to a held-out test set. In our experimental results, we report the difference in cumulative regret on the UCI datasets between the candidate algorithms and OfflineTree.

Algorithm 4: OfflineTree()

For each observation (x, y), define $w_a(x, y)$ as follows:
$w_a(x, y) = 1$ if $y = a$, and
$w_a(x, y) = 0$ if $y \neq a$

Hold out T random observations, $\{x_t\}_{1 \leq t \leq T}$

for $a = 1, ..., K$ **do**
  Fit tree $\hat{\theta}_a$ on remaining data using $w_a(x, y)$ as the response variable
end
for $t = 1, ..., T$ **do**
  Observe context vector x_t
  Choose action $a_t = \arg\max_a \hat{p}(\hat{\theta}_a, x_t)$
end

6.2 SPORTS ADVERTISING DATASET

First, TreeBootstrap was tested under an idealistic setting – a simulated dataset where the context-reward model is a decision tree for each action. We frame this dataset in the context of sports web-advertising. Whenever a user visits the website, we must offer ads for one of $K = 4$ different products: golf clubs, basketball nets, tennis rackets, and soccer balls. We receive a reward of 1 if the user clicks the ad; otherwise, we receive no reward.

Figure 1 provides an example of the decision tree used to simulate the golf advertisement rewards, as well as information about the $M = 4$ (binary) contextual variables available for each user. Figure 2 provides the cumulative regret data for the tested methods. As expected, our algorithm outperforms the other benchmark methods which do not use decision tree learners, and the performance of TreeBootstrap and TreeHeuristic are very similar. Moreover, the regret seems to converge to zero for our decision tree algorithms as the number of observed users becomes large. Finally, note how BanditForest eventually achieves a regret comparable to the other algorithms, but nonetheless incurs a much higher cumulative regret. This is due to the fact that the algorithm takes most of the time horizon to exit its "pure exploration" phase, an observation which also holds across all the UCI datasets.

6.3 UCI REPOSITORY DATASETS

We next evaluated our algorithm on four classification datasets from the UCI Machine Learning Repository (Lichman, 2013): *Adult*, *Statlog (Shuttle)*, *Covertype*, and *US Census Data (1990)*. The response variables used were *occupation*, *Cover_Type*, and *dOccup* for *Adult*, *Covertype*, and *Census*, respectively, while for *Shuttle* we used the variable corresponding to the last column of the dataset. We first ran a preprocessing step remov-

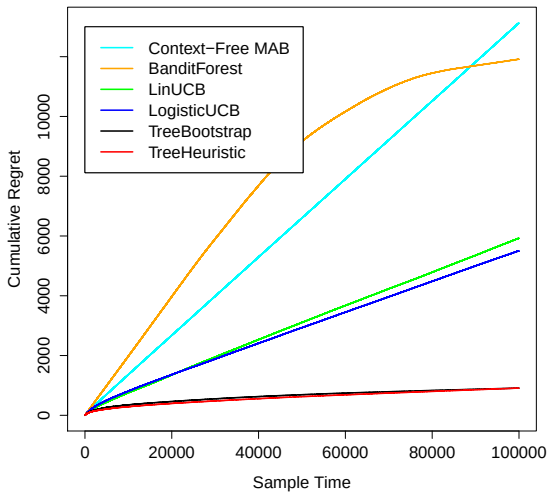


Figure 2: Cumulative regret incurred on the (simulated) sports advertising dataset.

ing classes which were significantly underrepresented in each dataset (i.e., less than 0.05% of the observations). After preprocessing, these datasets had $K = 12, 4, 7,$ and 9 classes, respectively, as well as $M = 14, 9, 54,$ and 67 contextual variables. We then constructed a bandit problem from the data in the following way: a regret of 0 (otherwise 1) is incurred if and only if the algorithm predicts the class of the data point correctly. This framework for adapting classification data for use in bandit problems is commonly used in the literature (Allesiardo et al., 2014; Agarwal et al., 2014; Féraud et al., 2016).

Figure 4 shows the cumulative regret of TreeBootstrap compared with the benchmarks on the UCI datasets. Recall that we plot regret relative to OfflineTree. In all cases, our heuristic achieves a performance equal to or better than that of TreeBootstrap. Note that LogisticUCB outperforms LinUCB on all datasets except *Covertype*, which demonstrates the value of using learners in our setting which handle binary response data effectively.

LinUCB and LogisticUCB outperform our algorithm on two of the four UCI datasets (*Adult* and *Census*). However, there are several caveats to this result. First, recall that we only report the cumulative regret of LinUCB and LogisticUCB with respect to the best exploration parameter, which is impossible to know a priori. Figure 3 shows LinUCB's cumulative regret curves corresponding to each value of the exploration parameter α implemented on the *Adult* dataset. We overlay this on a plot of the cumulative regret curves associated with TreeBootstrap and our heuristic. Note that TreeBootstrap and TreeHeuristic outperformed LinUCB in at least half of the parameter settings attempted. Second, the difference in cumulative regret between TreeBootstrap and Linear/Logistic UCB on *Adult* and *Census* appears to approach a constant as the time horizon increases. This is due to the fact that decision trees will capture the truth eventually given enough training data. Conversely, in settings such as the sports ad dataset, *Covertype*, and *Shuttle*, it appears that the linear/logistic regression models have already converged and fail to capture the context-reward distribution accurately. This is most likely due to the fact that feature engineering is needed for the data to satisfy the GLM framework. Thus, the difference in cumulative regret between Linear/Logistic UCB and TreeBootstrap will become arbitrarily large as the time horizon increases. Finally, note that we introduce regularization into the linear and logistic regressions, tuned using cross-validation, which improve upon the original framework for LinUCB and Logistic UCB.

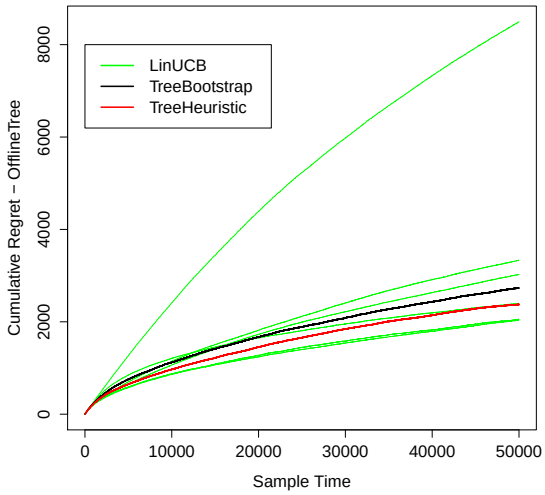
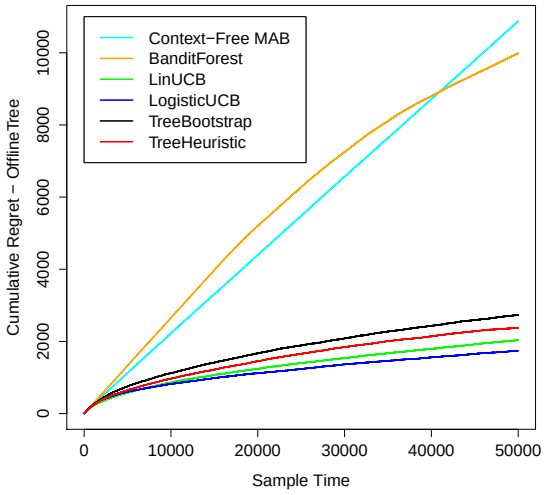


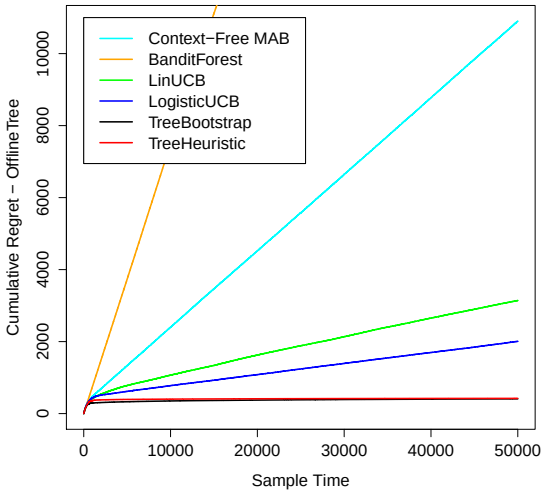
Figure 3: Cumulative regret incurred on the *Adult* dataset. The performance of LinUCB is given with respect to all values of the tuneable parameter attempted: α = 0.0001, 0.001, 0.01, 0.1, 1, 10

7 CONCLUSION

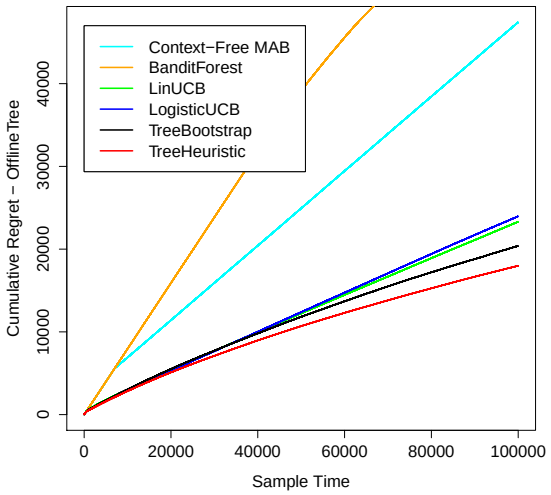
We propose a contextual bandit algorithm, TreeBootstrap, which can be easily and effectively applied in practice. We use decision trees as our base learner, and we handle the exploration-exploitation trade-off using bootstrapping in a way which approximates the behavior of Thompson sampling. As our algorithm requires fitting multiple trees at each time step, we provide a computational heuristic which works well in practice. Empirically, our methods' performance is quite competitive and robust compared to several well-known algorithms.



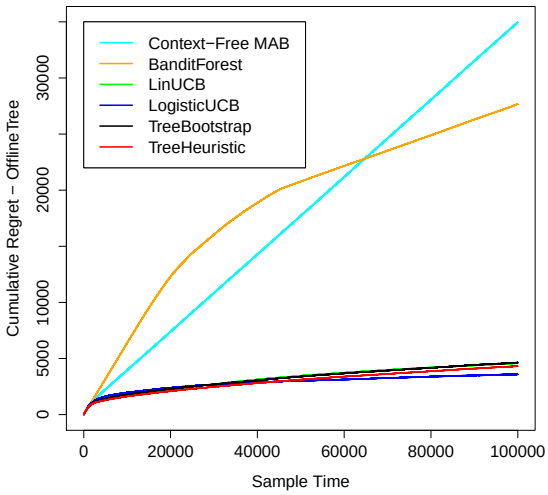
(a) *Adult* Dataset Results



(b) *Statlog (Shuttle)* Dataset Results



(c) *Covertype* Dataset Results



(d) *Census* Dataset Results

Figure 4: Cumulative regret incurred on various classification datasets from the UCI Machine Learning Repository. A regret of 0 (otherwise 1) is incurred iff a candidate algorithm predicts the class of the data point correctly.

References

Abbasi-Yadkori, Y., Pál, D., and Szepesvári, C. (2011). Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320.

Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., and Schapire, R. (2014). Taming the monster: A fast and simple algorithm for contextual bandits. In *Proceedings of The 31st International Conference on Machine Learning*, pages 1638–1646.

Agrawal, S. and Goyal, N. (2013). Thompson sampling for contextual bandits with linear payoffs. In *ICML (3)*, pages 127–135.

Allesiardo, R., Féraud, R., and Bouneffouf, D. (2014). A neural networks committee for the contextual bandit problem. In *International Conference on Neural Information Processing*, pages 374–381. Springer.

Auer, P. (2002). Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422.

Baransi, A., Maillard, O.-A., and Mannor, S. (2014). Sub-sampling for multi-armed bandits. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 115–131. Springer.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. (1984). *Classification and regression trees*. CRC press.

Chu, W., Li, L., Reyzin, L., and Schapire, R. E. (2011). Contextual bandits with linear payoff functions. In *AISTATS*, volume 15, pages 208–214.

Crawford, S. L. (1989). Extensions to the cart algorithm. *International Journal of Man-Machine Studies*, 31(2):197–217.

Dhar, S. and Varshney, U. (2011). Challenges and business models for mobile location-based services and advertising. *Communications of the ACM*, 54(5):121–128.

Dudik, M., Hsu, D., Kale, S., Karampatziakis, N., Langford, J., Reyzin, L., and Zhang, T. (2011). Efficient optimal learning for contextual bandits. *arXiv preprint arXiv:1106.2369*.

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

Féraud, R., Allesiardo, R., Urvoy, T., and Clérot, F. (2016). Random forest for the contextual bandit problem. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 93–101.

Filippi, S., Cappe, O., Garivier, A., and Szepesvári, C. (2010). Parametric bandits: The generalized linear case. In *Advances in Neural Information Processing Systems*, pages 586–594.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin.

Kim, E. S., Herbst, R. S., Wistuba, I. I., Lee, J. J., Blumenschein, G. R., Tsao, A., Stewart, D. J., Hicks, M. E., Erasmus, J., Gupta, S., et al. (2011). The battle trial: personalizing therapy for lung cancer. *Cancer discovery*, 1(1):44–53.

Langford, J. and Zhang, T. (2008). The epoch-greedy algorithm for multi-armed bandits with side information. In *Advances in neural information processing systems*, pages 817–824.

Li, L., Chu, W., Langford, J., Moon, T., and Wang, X. (2011). An unbiased offline evaluation of contextual bandit algorithms with generalized linear models.

Li, L., Chu, W., Langford, J., and Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM.

Lichman, M. (2013). UCI machine learning repository.

Long, J. S. (1997). Regression models for categorical and limited dependent variables. *Advanced quantitative techniques in the social sciences*, 7.

Osband, I. and Van Roy, B. (2015). Bootstrapped thompson sampling and deep exploration. *arXiv preprint arXiv:1507.00300*.

Russo, D. and Van Roy, B. (2014). Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243.

Tang, L., Jiang, Y., Li, L., Zeng, C., and Li, T. (2015). Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 323–332. ACM.

Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine learning*, 4(2):161–186.

Utgoff, P. E., Berkman, N. C., and Clouse, J. A. (1997). Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29(1):5–44.

Yuan, S.-T. and Tsao, Y. W. (2003). A recommendation mechanism for contextualized mobile advertising. *Expert Systems with Applications*, 24(4):399–414.