Random Forest for the Contextual Bandit Problem

Raphaël Féraud
raphael.feraud@orange.com
Orange Labs, France

Robin Allesiardo
robin.allesiardo@orange.com
Orange Labs, France

Tanguy Urvoy
tanguy.urvoy@orange.com
Orange Labs, France

Fabrice Clérot
fabrice.clerot@orange.com
Orange Labs, France

Abstract

To address the contextual bandit problem, we propose an online *random forest* algorithm. The analysis of the proposed algorithm is based on the sample complexity needed to find the optimal decision stump. Then, the decision stumps are recursively stacked in a random collection of decision trees, BANDIT FOREST. We show that the proposed algorithm is optimal up to logarithmic factors. The dependence of the sample complexity upon the number of contextual variables is logarithmic. The computational cost of the proposed algorithm with respect to the time horizon is linear. These analytical results allow the proposed algorithm to be efficient in real applications, where the number of events to process is huge, and where we expect that some contextual variables, chosen from a large set, have potentially non-linear dependencies with the rewards. In the experiments done to illustrate the theoretical analysis, BANDIT FOREST obtain promising results in comparison with state-of-the-art algorithms.

1 Introduction

By interacting with streams of events, machine learning algorithms are used for instance to optimize the choice of ads on a website, to choose the best human machine interface, to recommend products on a web shop, to insure self-care of set top boxes, to assign the best wireless network to mobile phones. With the now rising *internet of things*, the number of decisions (or actions) to be taken by more and more autonomous devices further increases. In order to control the cost and the potential risk to deploy a lot of machine learning algorithms in the long run, we need scalable algorithms which provide strong theoretical guarantees.

Most of these applications necessitate to take and optimize decisions with a partial feedback. Only the reward of the chosen decision is known. Does the user click on the proposed ad ? The most relevant ad is never revealed. Only the click on the proposed ad is known. This well known problem is called *multi-armed bandits* (MAB). In its most basic formulation, it can be stated as follows: there are K decisions, each having an unknown distribution of bounded rewards. At each step, one has to choose a decision and receives a reward. The performance of a MAB algorithm is assessed in terms of *regret* (or opportunity loss) with regards to the unknown optimal decision. Optimal solutions have been proposed to solve this problem using a stochastic formulation in Auer et al (2002), using a Bayesian formulation in Kaufman et al (2012), or using an adversarial formulation in Auer et al (2002). While these approaches focus on the minimization of the expected regret, the PAC setting (see Vailant (1984)) or the (ϵ, δ)-best-arm identification, focuses on the sample complexity (i.e. the number of time steps) needed to find an ϵ-approximation of the best arm with a failure probability of δ. This formulation has been studied for MAB problem in Even-Dar et al (2002); Bubeck et al (2009), for dueling bandit problem in Urvoy et al (2013), and for linear bandit problem in Soare et al (2014).

Several variations of the MAB problem have been introduced in order to fit practical constraints coming from ad serving or marketing optimization. These variations include for instance the death and birth of arms in Chakrabarti et al (2008), the availability of actions in Kleinberg et al (2008) or a drawing without replacement in Féraud and Urvoy (2012, 2013). But more importantly, in most of these applications, a rich contextual information is also available. For instance, in ad-serving optimization we know the web page, the position in the web page, or the profile of the user. This contextual information must be exploited in order to decide which ad is the most relevant to display. Following Langford and Zhang (2007); Dudik et al (2011), in order to analyze the proposed algorithms, we formalize below the *contextual bandit problem* (see Algorithm 1).

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

Let $\mathbf{x_t} \in \{0,1\}^M$ be a vector of binary values describing the environment at time t. Let $\mathbf{y_t} \in [0,1]^K$ be a vector of bounded rewards at time t, and $y_{k_t}(t)$ be the reward of the action (decision) k_t at time t. Let $D_{x,y}$ be a joint distribution on $(\mathbf{x}, \mathbf{y})$. Let $\mathcal{A}$ be a set of K actions. Let $\pi : \{0,1\}^M \to \mathcal{A}$ be a policy, and Π the set of policies.

Algorithm 1 The contextual bandit problem

repeat
  $(\mathbf{x_t}, \mathbf{y_t})$ is drawn according to $D_{x,y}$
  $\mathbf{x_t}$ is revealed to the player
  The player chooses an action $k_t = \pi_t(\mathbf{x_t})$
  The reward $y_{k_t}(t)$ is revealed
  The player updates its policy π_t
  $t = t + 1$
until $t = T$

Notice that this setting can easily be extended to categorical variables through a binary encoding. The optimal policy π^* maximizes the expected gain:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}_{D_{x,y}} \left[y_{\pi(\mathbf{x_t})} \right]$$

Let $k_t^* = \pi^*(\mathbf{x_t})$ be the action chosen by the optimal policy at time t. The performance of the player policy is assessed in terms of expectation of the accumulated regret against the optimal policy with respect to $D_{x,y}$:

$$\mathbb{E}_{D_{x,y}} \left[R(T) \right] = \sum_{t=1}^{T} \mathbb{E}_{D_{x,y}} \left[y_{k_t^*}(t) - y_{k_t}(t) \right] ,$$

where $R(T)$ is the accumulated regret at time horizon T.

2 Our contribution

Decision trees work by partitioning the input space in hyper-boxes. They can be seen as a combination of rules, where only one rule is selected for a given input vector. Finding the optimal tree structure (i.e. the optimal combination of rules) is NP-hard. For this reason, a greedy approach is used to build the decision trees offline (see Breiman et al (1984)) or online (see Domingos and Hulten (2000)). The key concept behind the greedy decision trees is the decision stump. While Monte-Carlo Tree Search approaches (see Kocsis and Szépesvari (2006)) focus on the regret minimization (i.e. maximization of gains), the analysis of proposed algorithms is based on the sample complexity needed to find the optimal decision stump with high probability. This formalization facilitates the analysis of decision tree algorithms. Indeed, to build a decision tree under limited resources, one needs to eliminate most of possible branches. The sample complexity is the stopping criterion we need to stop exploration of unpromising branches. In BANDIT FOREST, the decision stumps are recursively stacked in a random collection of L decision trees of maximum depth D. We show that BANDIT FOREST algorithm is optimal up to logarithmic factors with respect to a strong reference: a *random forest* built knowing the joint distribution of the contexts and rewards.

In comparison to algorithms based on search of the best policy from a finite set of policies (see Auer et al (2002); Dudik et al (2011); Agrawal et al (2014)) our approach has several advantages. First, we take advantage of the fact that we know the structure of the set of policies to obtain a linear computational cost with respect to the time horizon T. Second, as our approach does not need to store a weight for each possible tree, we can use deeper rules without exceeding the memory resources. In comparison to the approaches based on a linear model (see LINUCB in Li et al (2010)), our approach also has several advantages. First, it is better suited for the case where the dependence between the rewards and the contexts is not linear. Second, the dependence of regret bounds of the proposed algorithms on the number of contextual variables is in the order of $O(\log M)$ while the one of linear bandits is in $O(\sqrt{M})$ [1]. Third, its computational cost with respect to time horizon in $O(LMDT)$ allows to process large set of variables, while linear bandits are penalized by the update of a $M \times M$ matrix at each update, which leads to a computational cost in $O(KM^2T)$.

3 The decision stump

In this section, we consider a model which consists of a decision stump based on the values of a single contextual variable, chosen from at set of M binary variables.

3.1 A gentle start

In order to explain the principle and to introduce the notations, before describing the decision stump used to build BANDIT FOREST, we illustrate our approach on a toy problem. Let k_1 and k_2 be two actions. Let x_{i_1} and x_{i_2} be two binary random variables, describing the context. In this illustrative example, the contextual variables are assumed to be independent. Relevant probabilities and rewards are summarized in Table 1. $\mu_{k_2}^{i_1}|v$ denotes the conditional expected reward of the action k_2 given $x_{i_1} = v$, and $P(x_{i_1} = v)$ denotes the probability to observe $x_{i_1} = v$.

We compare the strategies of different players. Player 1 only uses uncontextual expected rewards, while Player 2 uses the knowledge of x_{i_1} to decide. According to Table 1, the best strategy for Player 1 is to always choose action k_2. His expected reward will be $\mu_{k_2} = 7/16$. Player 2 is able to adapt his strategy to the context: his best strategy is to choose k_2 when $x_{i_1} = v_0$ and k_1 when $x_{i_1} = v_1$.

[1] In fact the dependence on the number of contextual variables of the gap dependent regret bound is in $O(M^2)$ (see Theorem 5 in Abbasi-Yadkori and Szepesvári (2011)).

Table 1: The mean reward of actions k_1 and k_2 knowing each context value, and the probability to observe this context.

	v_0	v_1
$\mu^{i_1}_{k_1}\|v$	0	1
$\mu^{i_1}_{k_2}\|v$	3/5	1/6
$P(x_{i_1}=v)$	5/8	3/8
$\mu^{i_2}_{k_1}\|v$	1/4	3/4
$\mu^{i_2}_{k_2}\|v$	9/24	5/8
$P(x_{i_2}=v)$	3/4	1/4

According to Table 1, his expected reward will be:

$$\begin{aligned}\mu^{i_1} &= P(x_{i_1}=v_0)\cdot\mu^{i_1}_{k_2}|v_0 + P(x_{i_1}=v_1)\cdot\mu^{i_1}_{k_1}|v_1\\ &= \mu^{i_1}_{k_2,v_0} + \mu^{i_1}_{k_1,v_1} = 3/4\,,\end{aligned}$$

where $\mu^{i_1}_{k_2,v_0}$ and $\mu^{i_1}_{k_1,v_1}$ denote respectively the expected reward of the action k_2 and $x_{i_1}=v_0$, and the expected reward of the action k_1 and $x_{i_1}=v_1$. Whatever the expected rewards of each value, the player, who uses this knowledge, is the expected winner. Indeed, we have:

$$\mu^i = \max_k \mu^i_{k,v_0} + \max_k \mu^i_{k,v_1} \geq \max_k \mu_k$$

Now, if a third player uses the knowledge of the contextual variable x_{i_2}, his expected reward will be:

$$\mu^{i_2} = \mu^{i_2}_{k_2,v_0} + \mu^{i_2}_{k_1,v_1} = 15/32$$

Player 2 remains the expected winner, and therefore x_{i_1} is the best contextual variable to decide between k_1 and k_2.

The best contextual variable is the one which maximizes the expected reward of best actions for each of its values. We use this principle to build a reward-maximizing decision stump.

3.2 Variable selection

Let $\mathcal{V}$ be the set of variables, and $\mathcal{A}$ be the set of actions. Let $\mu^i_k|v = \mathbb{E}_{D_y}[y_k \cdot \mathbb{1}_{x_i=v}]$ be the expected reward of the action k conditioned to the observation of the value v of the variable x_i. Let $\mu^i_{k,v} = P(v).\mu^i_k|v = \mathbb{E}_{D_{x,y}}[y_k \cdot \mathbb{1}_{x_i=v}]$ be the expected reward of the action k and the value v of the binary variable x_i. The expected reward when using variable x_i to select the best action is the sum of expected rewards of the best actions for each of its possible values:

$$\mu^i = \sum_{v\in\{0,1\}} P(v).\max_k \mu^i_k|v = \sum_{v\in\{0,1\}} \max_k \mu^i_{k,v}$$

The optimal variable to be used for selecting the best action is: $i^* = \arg\max_{i\in\mathcal{V}} \mu^i$.

The algorithm VARIABLE SELECTION chooses the best variable. The Round-robin function sequentially explores the actions in $\mathcal{A}$ (see Algorithm 2 line 4). Each time t_k the reward of the selected action k is unveiled, the estimated expected rewards of the played action k and observed values $\hat{\mu}^i_{k,v}$ and all the estimated rewards of variables $\hat{\mu}^i$ are updated (see VE function lines 2-7). This *parallel exploration strategy* allows the algorithm to explore efficiently the variable set. When all the actions have been played once (see VE function line 8), irrelevant variables are eliminated if:

$$\hat{\mu}^{i'} - \hat{\mu}^i + \epsilon \geq 4\sqrt{\frac{1}{2t_k}\log\frac{4KMt_k^2}{\delta}}\,, \qquad (1)$$

where $i' = \arg\max_i \hat{\mu}^i$, and t_k is the number of times the action k has been played.

Algorithm 2 VARIABLE SELECTION

Inputs: $\epsilon \in [0,1)$, $\delta \in (0,1]$
Output: an ϵ-approximation of the best variable

1: $t=0, \forall k\ t_k=0, \forall(i,k,v)\ \hat{\mu}^i_{k,v}=0, \forall i\ \hat{\mu}^i=0$
2: **repeat**
3: Receive the context vector $\mathbf{x_t}$
4: Play $k = \text{Round-robin}\,(\mathcal{A})$
5: Receive the reward $y_k(t)$
6: $t_k = t_k + 1$
7: $\mathcal{V}$=VE$(t_k, k, \mathbf{x_t}, y_k, \mathcal{V}, \mathcal{A})$
8: $t = t+1$
9: **until** $|\mathcal{V}| = 1$

1: **Function** VE$(t, k, \mathbf{x_t}, y_k, \mathcal{V}, \mathcal{A})$
2: **for** each remaining variable $i \in \mathcal{V}$ **do**
3: **for** each value v **do**
4: $\hat{\mu}^i_{k,v} = \frac{y_k}{t}\mathbb{1}_{x_i=v} + \frac{t-1}{t}\hat{\mu}^i_{k,v}$
5: **end for**
6: $\hat{\mu}^i = \sum_{v\in\{0,1\}} \max_k \hat{\mu}^i_{k,v}$
7: **end for**
8: **if** $k = \text{LastAction}(\mathcal{A})$ **then**
9: Remove irrelevant variables from $\mathcal{V}$ according to equation 1, or 3
10: **end if**
11: **return** $\mathcal{V}$

The parameter $\delta \in (0,1]$ corresponds to the probability of failure. The use of the parameter ϵ comes from practical reasons. The parameter ϵ is used in order to tune the convergence speed of the algorithm. In particular, when two variables provide the highest expected reward, the use of $\epsilon > 0$ ensures that the algorithm stops. The value of ϵ has to be in the same order of magnitude as the best mean reward we want to select. In the analysis of algorithms, we will consider the case where $\epsilon = 0$. Lemma 1 analyzes the sample complexity (the number of iterations before stopping) of VARIABLE SELECTION.

Lemma 1: *when $K \geq 2$, $M \geq 2$, and $\epsilon = 0$, the sample complexity of* VARIABLE SELECTION *needed to obtain* $\mathbb{P}(i' \neq i^*) \leq \delta$ *is:*

$$t^* = \frac{64K}{\Delta_1^2} \log \frac{4KM}{\delta \Delta_1}, \text{ where } \quad \Delta_1 = \min_{i \neq i^*} (\mu^{i^*} - \mu^i)$$

Lemma 1 shows that the dependence of the sample complexity needed to select the optimal variable is in $O(\log M)$. This means that VARIABLE SELECTION can be used to process large set of contextual variables, and hence can be easily extended to categorical variables, through a binary encoding with only a logarithmic impact on the sample complexity. Finally, Lemma 2 shows that the variable selection algorithm is optimal up to logarithmic factors.

Lemma 2: *There exists a distribution $D_{x,y}$ such that any algorithm finding the optimal variable i^* has a sample complexity of at least:*

$$\Omega\left(\frac{K}{\Delta_1^2} \log \frac{1}{\delta}\right)$$

3.3 Action selection

To complete a decision stump, one needs to provide an algorithm which optimizes the choice of the best action knowing the selected variable. Any stochastic bandit algorithm such as UCB (see Auer et al (2002)), TS (see Thompson (1933); Kaufman et al (2012)) or BESA (see Baransi et al (2014)) can be used. For the consistency of the analysis, we choose SUCCESSIVE ELIMINATION in Even-Dar et al (2002, 2006) (see Algorithm 3), that we have renamed ACTION SELECTION. Let $\mu_k = \mathbb{E}_{D_y}[y_k]$ be the expected reward of the action k taken with respect to D_y. The estimated expected reward of the action is denoted by $\hat{\mu}_k$.

Algorithm 3 ACTION SELECTION

Inputs: $\epsilon \in [0, 1)$, $\delta \in (0, 1]$
Output: an ϵ-approximation of the best arm

$t = 0$, $\forall k\ \hat{\mu}_k = 0$ and $t_k = 0$
repeat
  Play $k = \text{Round-robin}(A)$
  Receive the reward $y_k(t)$
  $t_k = t_k + 1$
  $\mathcal{A}$=AE$(t_k, k, y_k(t), \mathcal{A})$
  $t = t + 1$
until $|\mathcal{A}| = 1$

The irrelevant actions in the set $\mathcal{A}$ are successively eliminated when:

$$\hat{\mu}_{k'} - \hat{\mu}_k + \epsilon \geq 2\sqrt{\frac{1}{2t_k} \log \frac{4Kt_k^2}{\delta}}\,, \tag{2}$$

Function AE$(t, k, \mathbf{x_t}, y_k, \mathcal{A})$
$\hat{\mu}_k = \frac{y_k}{t} + \frac{t-1}{t}\hat{\mu}_k$
if k=LastAction($\mathcal{A}$) **then**
  Remove irrelevant actions from $\mathcal{A}$ according to equation 2, or 4
end if
return $\mathcal{A}$

where $k' = \arg\max_k \hat{\mu}_k$, and t_k is the number of times the action k has been played.

Lemma 3: *when $K \geq 2$, and $\epsilon = 0$, the sample complexity of* ACTION SELECTION *needed to obtain* $\mathbb{P}(k' \neq k^*) \leq \delta$ *is:*

$$t^* = \frac{64K}{\Delta_2^2} \log \frac{4K}{\delta \Delta_2}, \text{ where } \quad \Delta_2 = \min_k (\mu_{k^*} - \mu_k).$$

The proof of Lemma 3 is the same than the one provided for SUCCESSIVE ELIMINATION in Even-Dar et al (2006). Finally, Lemma 4 states that the action selection algorithm is optimal up to logarithmic factors (see Mannor and Tsitsiklis (2004) Theorem 1 for the proof).

Lemma 4: *There exists a distribution $D_{x,y}$ such that any algorithm finding the optimal action k^* has a sample complexity of at least:*

$$\Omega\left(\frac{K}{\Delta_2^2} \log \frac{1}{\delta}\right)$$

3.4 Analysis of a decision stump

The decision stump uses the values of a contextual variable to choose the actions. The optimal decision stump uses the best variable to choose the best actions. It plays at time t: $k_t^* = \arg\max_k \mu_k^{i^*}|v$, where $i^* = \arg\max_i \mu^i$, and $v = x_{i^*}(t)$. The expected gain of the optimal policy is:

$$\mathbb{E}_{D_{x,y}}\left[y_{k_t^*}(t)\right] = \sum_v P(v)\mu_{k^*}^{i^*}|v = \sum_v \mu_{k^*,v}^{i^*} = \mu^{i^*}$$

To select the best variable, one needs to find the best action of each value of each variable. In DECISION STUMP algorithm (see Algorithm 4), an action selection task is allocated for each value of each contextual variable. When the reward is revealed, all the estimated rewards of variables $\hat{\mu}^i$ and the estimated rewards of the played action knowing the observed values of variables $\hat{\mu}_k^i|v$ are updated (respectively in VE and AE functions): the variables and the actions are simultaneously explored. However, the elimination of actions becomes effective only when the best variable is selected. Indeed, if an action k is eliminated for a value v_0 of a variable i, the estimation of $\hat{\mu}_{k,v_1}^i$, the mean expected reward of the action k for the value v_1, is biased. As a consequence, if an action is eliminated during the exploration

of variables, the estimation of the mean reward μ^i can be biased. That is why, the lower bound of decision stump problem is the sum of lower bound of variable and action selection problems (see Theorem 2). The only case where an action can be eliminated before the best variable be selected, is when this action is eliminated for all values of all variables. For simplicity of the exposition of the DECISION STUMP algorithm we did not handle this case here.

Algorithm 4 DECISION STUMP

$t = 0$, $\forall k\ t_k = 0$, $\forall i\ \hat{\mu}^i = 0$, $\forall (i,v,k)\ t_{i,v,k} = 0$, $\mathcal{A}_{i,v} = \mathcal{A}$, $\hat{\mu}^i_{k,v} = 0$ and $\hat{\mu}^i_k|v = 0$
repeat
  Receive the context vector $\mathbf{x_t}$
  if $|\mathcal{V}| > 1$ **then** Play $k = \text{Round-robin}\,(\mathcal{A})$
  else Play $k = \text{Round-Robin}\,(\mathcal{A}_{i,v})$
  Receive the reward $y_k(t)$
  $t_k = t_k + 1$
  $\mathcal{V}$=VE$(t_k, k, \mathbf{x_t}, y_k, \mathcal{V}, \mathcal{A})$
  for each variable $i \in \mathcal{V}$ **do**
    $v = x_i(t)$, $t_{i,v,k} = t_{i,v,k} + 1$
    $\mathcal{A}_{i,v} = \text{AE}(t_{i,v,k}, k, \mathbf{x_t}, y_k, \mathcal{A}_{i,v})$
  end for
  $t = t + 1$
until $t = T$

Theorem 1: *when $K \geq 2$, $M \geq 2$, and $\epsilon = 0$, the sample complexity needed by* DECISION STUMP *to obtain $\mathbb{P}\left(i' \neq i^*\right) \leq \delta$ and $\mathbb{P}\left(k'_t \neq k^*_t\right) \leq \delta$ is:*

$$t^* = \frac{64K}{\Delta_1^2} \log \frac{4KM}{\delta \Delta_1} + \frac{64K}{\Delta_2^2} \log \frac{4K}{\delta \Delta_2},$$

$$\textit{where } \Delta_1 = \min_{i \neq i^*} \left(\mu^{i^*} - \mu^i\right),$$
$$\textit{and } \Delta_2 = \min_{k \neq k^*, v \in \{0,1\}} \left(\mu^{i^*}_{k^*,v} - \mu^{i^*}_{k,v}\right).$$

Theorem 2 provides a lower bound for the sample complexity, showing that the factors $1/\Delta_1^2$ and $1/\Delta_2^2$ are inherent of the decision stump problem. Notice that for the linear bandit problem, the same factor $1/\Delta^2$ was obtained in the lower bound (see Lemma 2 in Soare et al (2014)).

Theorem 2: *It exists a distribution $D_{x,y}$ such that any algorithm finding the optimal decision stump has a sample complexity of at least:*

$$\Omega\left(\left(\frac{1}{\Delta_1^2} + \frac{1}{\Delta_2^2}\right) K \log \frac{1}{\delta}\right)$$

Remark 1: The factors in $\log 1/\Delta_1$, $\log 1/\Delta_2$ and $\log 1/\Delta$ could vanish in Theorem 1 following the same approach as that *Median Elimination* in Even-Dar et al (2006). Despite the optimality of this algorithm, we did not choose it for the consistency of the analysis. Indeed, as it suppresses $1/4$ of variables (or actions) at the end of each elimination phase, *Median Elimination* is not well suited when a few number of variables (or actions) provide lot of rewards and the others not. In this case this algorithm spends a lot of times to eliminate non-relevant variables. This case is precisely the one, where we would like to use a local model such as a decision tree.

Remark 2: The extension of the analytical results to an ϵ-approximation of the best decision stump is straightforward using $\Delta_\epsilon = \max(\epsilon, \Delta)$.

4 BANDIT FOREST

A decision stump is a weak learner which uses only one variable to decide the best action. When one would like to combine D variables to choose the best action, a tree structure of $\binom{M}{D}.2^D$ multi-armed bandit problems has to be allocated. To explore and exploit this tree structure with limited memory resources, our approach consists in combining greedy decision trees. When decision stumps are stacked in a greedy tree, they combine variables in a greedy way to choose the action. When a set of randomized trees vote, they combine variables in a non-greedy way to choose the action.

As highlighted by empirical studies (see for instance Fernàndez-Delgado et al (2014)), *random forests* of Breiman (2001) have emerged as a serious competitors to state-of-the-art methods for classification tasks. In Biau (2012), the analysis of *random forests* shows that the reason of these good performances comes from the fact that the convergence of its learning procedure is consistent, and its rate of convergence depends only on the number of strong features, which is assumed to be low in comparison to the number of features. In this section, we propose to use a *random forest* built with the knowledge of $D_{x,y}$ as a reference for the proposed algorithm BANDIT FOREST.

Algorithm 5 θ-OGT $\left(c^*_\theta, d_\theta, \mathcal{V}_{c^*_\theta}\right)$

Inputs: $\theta \in [0,1]$
Output: the θ-optimal greedy tree when θ-OGT $((), 0, \mathcal{V})$ is called
if $d_\theta < D_\theta$ **then**
  $\mathcal{S}_{c^*_\theta} = \left\{i \in \mathcal{V}_{c^*_\theta} : f(\theta, c^*_\theta, i) = 1\right\}$
  $i^*_{d_\theta+1} = \arg\max_{i \in \mathcal{S}_{c^*_\theta}} \mu^i | c^*_\theta$
  θ-OGT $\left(c^*_\theta + (x_{i^*_{d_\theta+1}} = 0), d_\theta + 1, \mathcal{V}_{c^*_\theta} \setminus \{i^*_{d_\theta+1}\}\right)$
  θ-OGT $\left(c^*_\theta + (x_{i^*_{d_\theta+1}} = 1), d_\theta + 1, \mathcal{V}_{c^*_\theta} \setminus \{i^*_{d_\theta+1}\}\right)$
  else $k^* | c^*_\theta = \arg\max_k \mu_k | c^*_\theta$
end if

Let Θ be an iid random variable, which is independent of $\mathbf{x}$ and $\mathbf{y}$. Let $\theta \in [0,1]$ the value of Θ. Let D_θ be the maximal depth of the tree θ. Let $c_\theta = \{(x_{i_1} = v), ..., (x_{i_{d_\theta}} = v)\}$

be the index of a path of the tree θ. We use $c_\theta(\mathbf{x_t})$ to denote the path c_θ selected at time t by the tree θ. Let $f(\theta, i, j) : [0,1] \times \{0,1\}^{2^D} \times M \rightarrow \{0,1\}$ be a function which parametrizes $\mathcal{V}_{c_\theta}$ the sets of available variables for each of 2^{D_θ} splits. We call θ-optimal greedy tree, the greedy tree built with the knowledge of $D_{x,y}$ and conditioned to the value θ of the random variable Θ (see Algorithm 5). We call optimal *random forest* of size L, a *random forest*, which consists of a collection of L θ-optimal greedy trees. At each time step, the optimal *random forest* chooses the action k_t^*, which obtains the higher number of votes:

$$k_t^* = \arg\max_k \sum_{i=1}^{L} \mathbb{1}_{k^*_{\theta_i,t}=k} \, ,$$

where $k^*_{\theta_i,t}$ is the action chosen by the optimal greedy tree θ_i at time t.

Algorithm 6 BANDIT FOREST

1: $t = 0, \forall\theta \; c_\theta = ()$ and $d_\theta = 0, \forall\theta$ NewPath$(c_\theta, \mathcal{V})$
2: **repeat**
3: Receive the context vector $\mathbf{x_t}$
4: **for** each θ **do** select the context path $c_\theta(\mathbf{x_t})$
5: **if** $\forall\theta \; d_\theta = D_\theta$ and $|\mathcal{S}_{c_\theta}| = 1$, and $\forall(\theta, i, v)$ $|\mathcal{A}_{c_{\theta,i,v}}| = 1$ **then** $k = \arg\max_k \sum_{i=1}^{L} \mathbb{1}_{k_{\theta_i,t}=k}$
6: **else** $k =$ Round-robin$(\mathcal{A})$
7: **endif**
8: Receive the reward $y_k(t)$
9: **for** each θ **do**
10: $t_{c_\theta,k} = t_{c_\theta,k} + 1$
11: $\mathcal{S}_{c_\theta}$=VE$(t_{c_\theta,k}, k, \mathbf{x_t}, y_k, \mathcal{S}_{c_\theta}, \mathcal{A})$
12: **for** each remaining variable i **do**
13: $v = x_i(t), t_{c_\theta,i,v,k} = t_{c_\theta,i,v,k} + 1$
14: $\mathcal{A}_{c_\theta,i,v} =$ AE$(t_{c_\theta,i,v,k}, k, \mathbf{x_t}, y_k, \mathcal{A}_{c_\theta,i,v})$
15: **end for**
16: **if** $|\mathcal{S}_{c_\theta}| = 1$ **and** $d_\theta < D_\theta$ **then**
17: $\mathcal{V}_{c_\theta} = \mathcal{V}_{c_\theta} \setminus \{i\}$
18: NewPath$(c_\theta + (x_i = 0), \mathcal{V}_{c_\theta})$
19: NewPath$(c_\theta + (x_i = 1), \mathcal{V}_{c_\theta})$
20: **end if**
21: **end for**
22: $t = t + 1$
23: **until** $t = T$

1: **Function** NewPath$(c_\theta, \mathcal{V}_{c_\theta})$
2: $\mathcal{S}_{c_\theta} = \{i \in \mathcal{V}_{c_\theta} : f(\theta, c_\theta, i) = 1\}$
3: $d_\theta = d_\theta + 1, \forall(i, v) \; \mathcal{A}_{c_\theta,i,v} = \mathcal{A}$
4: $\forall k \; t_{c_\theta,k} = 0, \forall(i, v, k) \; t_{c_\theta,i,v,k} = 0, \forall i \; \hat{\mu}^i|c_\theta = 0$
5: $\forall(i, k, v) \; \hat{\mu}^i_{k,v}|c_\theta = 0$ and $\hat{\mu}^i_k|(v, c_\theta) = 0$

BANDIT FOREST algorithm explores and exploits a set of L decision trees knowing $\theta_1, ..., \theta_L$ (see Algorithm 6). When a context $\mathbf{x_t}$ is received (line 3):

- For each tree θ, the path c_θ is selected (line 4).
- An action k is selected:
  - If the learning of all paths c_θ is finished, then each path vote for its best action (line 5).
  - Else the actions are sequentially played (line 6).
- The reward $y_k(t)$ is received (line 8).
- The decision stumps of each path c_θ are updated (lines 9-15).
- When the set of remaining variables of the decision stump corresponding to the path c_θ contains only one variable and the maximum depth D_θ is not reached (line 16), two new decision stumps corresponding to the values 0,1 of the selected variable are allocated (lines 18-20). The random set of remaining variables $\mathcal{V}_{c_\theta}$, the counts, and the estimated means are initialized in function NewPath.

To take into account the L decision trees of maximum depth D_θ, irrelevant variables are eliminated using a slight modification of inequality (1). A possible next variable x_i is eliminated when:

$$\hat{\mu}^{i'}|c_\theta - \hat{\mu}^i|c_\theta + \epsilon \geq 4\sqrt{\frac{1}{2t_{c_\theta,k}} \log \frac{4KMD_\theta Lt^2_{c_\theta,k}}{\delta}} \, , \quad (3)$$

where $i' = \arg\max_{i \in \mathcal{V}_{c_\theta}} \hat{\mu}^i|c_\theta$, and $t_{c_\theta,k}$ is the number of times the path c_θ and the action k have been observed. To take into account the L decision trees, irrelevant actions are eliminated using a slight modification of inequality (2):

$$\hat{\mu}_{k'|c_\theta} - \hat{\mu}_{k|c_\theta} + \epsilon \geq 2\sqrt{\frac{1}{2t_{c_\theta,i,v,k}} \log \frac{4KLt^2_{c_\theta,i,v,k}}{\delta}} \, , \quad (4)$$

where $k' = \arg\max_k \hat{\mu}_{k|c_\theta}$, and $t_{c_\theta,i,v,k}$ is the number of times the action k has been played when the path c_θ, and the value v of the variable i have been observed.

Theorem 3: *when $K \geq 2$, $M \geq 2$, and $\epsilon = 0$, the sample complexity needed by* BANDIT FOREST *learning to obtain the optimal random forest of size L with a probability at least $1 - \delta$ is:*

$$t^* = 2^D \left(\frac{64K}{\Delta_1^2} \log \frac{4KMDL}{\delta\Delta_1} + \frac{64K}{\Delta_2^2} \log \frac{4LK}{\delta\Delta_2} \right) \, ,$$

where $\Delta_1 = \min_{\theta, c^*_\theta, i \neq i^*} P(c^*_\theta) \left(\mu^{i^*}|c^*_\theta - \mu^i|c^*_\theta\right)$,
$\Delta_2 = \min_{\theta, c^*_\theta, k \neq k^*} P(c^*_\theta) \left(\mu_{k^*}|c^*_\theta - \mu_k|c^*_\theta\right)$, *and*
$D = \max_\theta D_\theta$.

The dependence of the sample complexity on the depth D is exponential. This means that like all decision trees, BANDIT FOREST is well suited for cases, where there is a

small subset of relevant variables belonging to a large set of variables ($D << M$). This usual restriction of local models is not a problem for a lot of applications, where one can build thousands of contextual variables, and where only a few of them are relevant.

Theorem 4: *There exists a distribution $D_{x,y}$ such that any algorithm finding the optimal random forest of size L has a sample complexity of at least:*

$$\Omega\left(2^D\left[\frac{1}{\Delta_1^2}+\frac{1}{\Delta_2^2}\right]K\log\frac{1}{\delta}\right)$$

Theorem 4 shows that BANDIT FOREST algorithm is optimal up to logarithmic factors. The result of this analysis is supported by empirical evidence in the next section.

Remark 4: We have chosen to analyze BANDIT FOREST algorithm in the case of $\epsilon = 0$ in order to simplify the concept of the optimal policy. Another way is to define the set of ϵ-optimal *random forests*, built with decision stumps which are optimal up to an ϵ approximation factor. In this case, the guarantees are given with respect to the worst policy of the set. When $\epsilon = 0$, this set contains only the optimal *random forest* of size L given the values of Θ.

5 Experimentation

In order to illustrate the theoretical analysis with reproducible results on large sets of contextual variables, we used three datasets from the *UCI Machine Learning Repository* (*Forest Cover Type*, *Adult*, and *Census1990*). We recoded each continuous variable using *equal frequencies* into 5 binary variables, and each categorical variable into disjunctive binary variables. We obtained 94, 82 and 255 binary variables, for *Forest Cover Type*, *Adult* and *Census1990* datasets respectively. For *Forest Cover Type*, we used the 7 target classes as the set of actions. For *Adult*, the categorical variable *occupation* is used as a set of 14 actions. For *Census1990*, the categorical variable *Yearsch* is used as a set of 18 actions. The gain of policies was evaluated using the class labels of the dataset with a reward of 1 when the chosen action corresponds to the class label and 0 otherwise. The datasets, respectively composed of 581000, 48840 and 2458285 instances, were shuffled and played in a loop to simulate streams. In order to introduce noise between loops, at each time step the value of each binary variable has a probability of 0.05 to be inverted. Hence, we can consider that each context-reward vector is generated by a stationary random process. We set the time horizon to 10 millions of iterations. The algorithms are assessed in terms of accumulated regret against the optimal *random forest* of size 100. The optimal *random forest* is obtained by training a *random forest* of size 100 not limited by depth on the whole dataset with full information feedback and without noise.

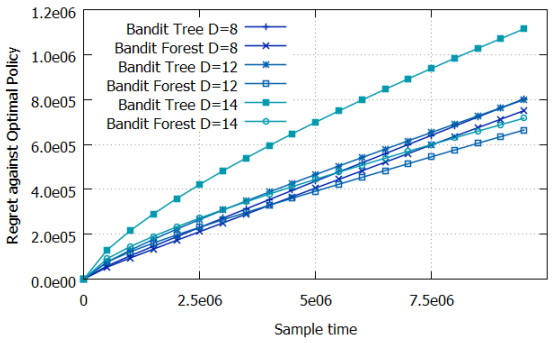

Figure 1: The accumulated regret of BANDIT FOREST and BANDIT TREE with different depths against the optimal policy averaged over ten trials on *Forest Cover Type* dataset played in a loop with noisy inputs.

To effectively implement BANDIT FOREST, we have done two modifications of the analyzed algorithms. Firstly, the Round-robin function is replaced by an uniform random draw from the union of the remaining actions of the paths. Secondly, the rewards are normalized using *Inverse Propensity Scoring* (see Horvitz and Thompson (1952)): the obtained reward is divided by the probability to draw the played action. First of all, notice that the regret curves of BANDIT FOREST algorithm are far from those of explore then exploit approaches: the regret is gradually reduced over time (see Figure 1). Indeed, BANDIT FOREST algorithm uses a *localized explore then exploit* approach: most of paths, which are unlikely, may remain in exploration state, while the most frequent ones already vote for their best actions. The behavior of the algorithm with regard to number of trees L is simple: the higher L, the greater the performances, and the higher the computational cost (see Figure 3). To analyze the sensitivity to depth of BANDIT FOREST algorithm, we set the maximum depth and we compared the performances of a single tree without randomization (BANDIT TREE) and of BANDIT FOREST with $L = 100$. The trees of the forest are randomized at each node with different values of ϵ (between 0.4 and 0.8), and with different sets of available splitting variables (random subset of 80% of remaining variables). For each tested depth, a significant improvement is observed thanks to the vote of randomized trees (see Figure 1). Moreover, BANDIT FOREST algorithm appears to be less sensible to depth than a single tree: the higher the difference in depth, the higher the difference in performance.

To compare with state-of-the-art, the trees in the forest are randomized with different values of the parameters D (between 10 and 18). On the three datasets, BANDITRON (Kakade et al (2008)) is clearly outperformed by the other tested algorithms (see Figures 2, 4, 5). NEURAL BANDIT (Allesiardo et al (2014)) is a Committee of Multi-Layer Perceptrons (MLP). Due to the non convexity of the error function, MLP trained with the back-propagation al-

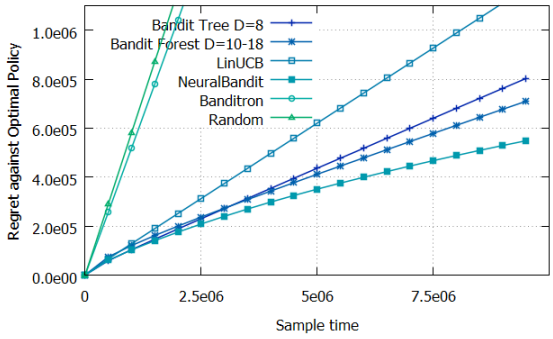

Figure 2: The accumulated regret against the optimal policy averaged over ten trials for the dataset *Forest Cover Type* played in a loop with noisy inputs.

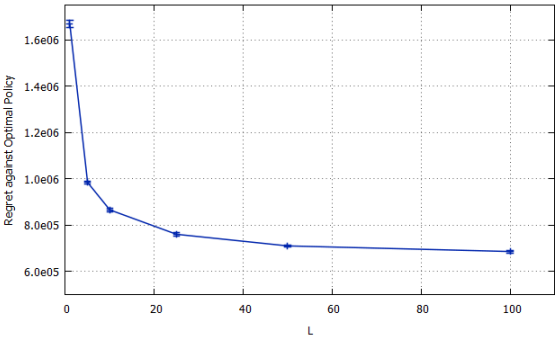

Figure 3: Sensitivity to the size of BANDIT FOREST on *Census 1990* dataset.

gorithm (Rumelhart et al (1986)) does not find the optimal solution. That is why finding the regret bound of such model is still an open problem. However, since MLPs are universal approximators (Hornik et al (1989)), NEURAL BANDIT is a very strong baseline. It outperforms LINUCB (Li et al (2010)) on *Forest Cover Type* and on *Adult*. BANDIT FOREST clearly outperforms LINUCB and BANDITRON on the three datasets. In comparison to NEURAL BANDIT, BANDIT FOREST obtains better results on *Census 1990* and *Adult*, and it is outperformed on *Forest Cover Type*, where the number of strong features seems to be high. Finally as shown by the worst case analysis, the risk to use BANDIT FOREST on a lot of optimization problems is controlled, while NEURAL BANDIT, which has no theoretical guaranty, can obtain poor performances on some problems, such as *Census 1990* where it is outperformed by the linear solution LINUCB. This uncontrolled risk increases with the number of actions, since the probability to obtain a non robust MLP linearly increases with the number of actions.

6 Conclusion

We have shown that the proposed algorithm is optimal up to logarithmic factors with respect to a strong reference:

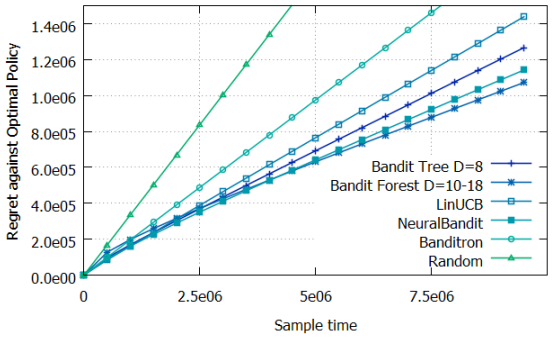

Figure 4: The accumulated regret against the optimal policy averaged over ten trials for the dataset *Adult* played in a loop with noisy inputs.

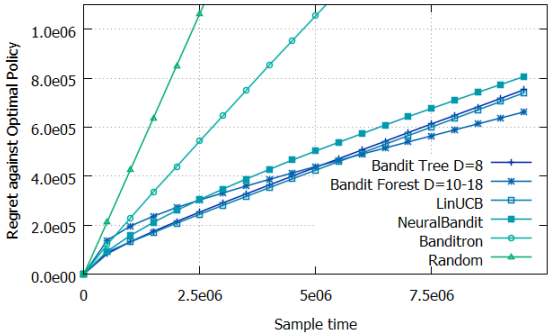

Figure 5: The accumulated regret against the optimal policy averaged over ten trials for the dataset *Census1990* played in a loop with noisy inputs.

a *random forest* built knowing the joint distribution of the contexts and rewards. In the experiments, BANDIT FOREST clearly outperforms LINUCB, which is a strong baseline with known regret bound, and performs as well as NEURAL BANDIT, for which we do not have theoretical guaranty. Finally, for applications where the number of strong features is low in comparison to the number of possible features, which is often the case, BANDIT FOREST shows valuable properties:

- its sample complexities have a logarithmic dependence on the number of contextual variables, which means that it can process a large amount of contextual variables with a low impact on regret,
- its low computational cost allows to process efficiently infinite data streams,
- like all decision tree algorithms, it is well suited to deal with non linear dependencies between contexts and rewards.

References

Abbasi-Yadkori Y., Pál D., and Szepesvári, C.: Improved Algorithms for Linear Stochastic Bandits, NIPS, 2011.

Agrawal, A., Hsu, D., Kale, S., Langford, J., Li, L., Schapire, R. E.: Taming the Monster: A Fast and Simple Algorithm for Contextual Bandits, ICML, 2014.

Allesiardo, R., Féraud, R., Bouneffouf, D.: A Neural Networks Committee for the Contextual Bandit Problem, ICONIP, 2014.

Auer, P., Cesa Bianchi, N., Fischer, P.: Finite-time Analysis of the Multiarmed Bandit Problem, Machine Learning,47, 235-256, 2002.

Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R. E.: The nonstochastic multiarmed bandit problem, SIAM J. COMPUT., 32 48-77, 2002.

Baransi, Akram and Maillard, Odalric-Ambrym and Mannor, Shie: Sub-sampling for Multi-armed Bandits, Machine Learning and Knowledge Discovery in Databases, 8724, 115-131, 2014.

Biau, G.: Analysis of Random Forest Model, JMLR, 13:1063-1095, 2012.

Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J.: Classification and regression trees, Monterey, CA: Wadsworth & Brooks Cole Advanced Books & Software, 1984.

Breiman, L.: Random Forest, Machine Learning, 45:5-32, 2001.

Bubeck, S., Wang, T., Stoltz: Pure exploration in multi-armed bandits problems, COLT, 2009.

Chakrabarti, D., Kumar, R., Radlinski, F., Upfal, E.: Mortal multi-armed bandits, NIPS, pp. 273-280, 2008.

Dudík, M., Hsu, D., Kale, S., Karampatziakis, N., Langford, J., Reyzin, L., Zhang, T.:Efficient Optimal Learning for Contextual Bandits, UAI, 169-178, 2011.

Domingos, P., Hulten, G.: Mining high-speed data streams, In: KDD, 71-80, 2000.

Even-Dar, E., Mannor, S., Mansour Y.: PAC Bounds for Multi-armed Bandit and Markov Decision Processes, COLT, 255-270, 2002.

Even-Dar, E., Mannor, Mansour, Y.: Action Elimination Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems, JMLR 7:1079-1105, 2006.

Féraud, R., Urvoy, T.: A stochastic bandit algorithm for scratch games, ACML, 25:129-145, 2012.

Féraud, R., Urvoy, T.: Exploration and Exploitation of Scratch Games, Machine Learning, 92:377-401, 2013.

Fernàndez-Delgado, M., Cernadas, E., Barro, S.: Do we Need Hundreds of Classifiers to Solve Real World Classification Problems ?, JMLR 15:3133-3181, 2014.

Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Amer. Statist. Assoc. 58:13-30, 1963.

Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators, Neural Networks. 2:359-366, 1989.

Horvitz, D. G., Thompson, D. J.: A generalization of sampling without replacement from a finite universe. Journal of the American Statistical Association, 47(260): 663-685, 1952.

Kakade, S. M., Shalev-Shwartz, S., Tewari, A.: Efficient Bandit Algorithms for Online Multiclass Prediction, ICML, 2008.

Kaufman, E., Korda, N., Munos, R.: Thomson sampling: An asymptotically optimal finite time analysis, COLT, 2012.

Kleinberg, R. D., Niculescu-Mizil, Sharma, T.: Regrets bounds for sleeping experts and bandits, COLT, 2008.

Kocsis, L., Szépesvari, C.: Bandit based monte-carlo planning, ECML, 2006.

Langford, J., Zhang, T.: The epoch-greedy algorithm for contextual multi-armed bandits, NIPS, 2007.

Li, L., Chu, W., Langford, J., Schapire, R. E.: A Contextual-Bandit Approach to Personalized News Article Recommendation, Nineteenth International Conference on World Wide Web, 2010.

Mannor, S., and Tsitsiklis, J. N.: The Sample Complexity of Exploration in the Multi-Armed Bandit Problem, JMLR, 2004

Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1, pp. 318-362. MIT Press, Cambridge, 1986.

Soare, M., Lazaric, L., Munos, R.: Best-Arm Identification in Linear Bandits, NIPS, 2014.

Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika 25:285-294, 1933.

Urvoy, T., Clérot, F., Féraud, R., Naamane, S.: Generic Exploration and K-armed Voting Bandits, ICML, 2013.

Valiant, L.: A theory of the learnable, Communications of the ACM, 27, 1984.