2015 IEEE Symposium Series on Computational Intelligence

Bayesian Credible Intervals for Online and Active Learning of Classification Trees

Timothé Collet
CentraleSupélec, MaLIS Research group, France
GeorgiaTech-CNRS UMI 2958, France
timothe.collet@centralesupelec.fr

Olivier Pietquin
Univ. Lille - CRIStAL Lab (UMR 9189), France
IUF (Institut Universitaire de France)
olivier.pietquin@univ-lille1.fr

***Abstract*—Classification trees have been extensively studied for decades. In the online learning scenario, a whole class of algorithms for decision trees has been introduced, called incremental decision trees. In the case where subtrees may not be discarded, an incremental decision tree can be seen as a sequential decision process, consisting in deciding to extend the existing tree or not. This problem involves a trade-off between exploration and exploitation, which is addressed in recent work with the use of Hoeffding's bounds. This paper proposes to use Bayesian Credible Intervals instead, in order to get the most out of the knowledge of the output's distribution's shape. It also studies the case of Active Learning in such a tree following the Optimism in the Face of Uncertainty paradigm. Two novel algorithms are introduced for the online and active learning problems. Evaluations on real-world datasets show that these algorithms compare positively to state-of-the-art.**

I. Introduction

We place this paper in the context of classification. This supervised learning framework consists in learning a mapping from an instance space to a finite discrete label set. For this purpose, it uses a training set containing a finite number of instances labeled by an expert. Two scenarios exist concerning the acquisition of data. In the batch learning scenario, the entire training set is given at once, and the classifier is computed considering all the instance/label pairs at the same time. In the online learning scenario, the data become available only sequentially, with the current classifier being required at every time step. This is the case, for example, in a system which learns while being used, with direct feedback. In this second scenario, a classifier capable of incremental updates is preferred to one being recomputed from scratch every time it receives new data, mainly for its lower computational time.

Our work focuses on classification trees. They are acyclic directed graphs whose nodes contain a test on instances, and branches represent the outcome of this test. Each leaf is associated to a label prediction based on the distribution of labels it received. In order to maximize the performance, the tree aims to partition the instance space into homogeneous label zones. Learning this classifier consists in reducing a given measure of heterogeneity, *e.g.* information entropy or standard deviation, estimated from received labels in the leaves, by performing recursive splits. An example is the c4.5 algorithm [1], which uses the information gain, or difference of entropy, to define the best splits.

Classification trees in the online learning scenario are called online or incremental classification trees [2] [3] [4]. We are only interested in the case where subtrees cannot be discarded and the existent tree can only be extended. This way, an online classification tree dynamically builds the tree, which corresponds to adaptively partitioning the instance space. It is a sequential decision process [5] which, for each newly received instance-label pair, considers changing the concerned leaf into one or more decision nodes. Once a test has been installed, it cannot be removed; it is thus critical to be sure that it is needed before installing it.

Learning an online classification tree involves an exploration/exploitation dilemma. Indeed, ideally, a split is needed if it decreases the true heterogeneities of the leaves instead of their estimates based on received labels. When considering to install a decision node leading to a decrease of the estimates, we can either decide to effectively split, which results in exploitation of the estimates. Or we can decide to wait for a better estimate, which results in exploration.

The exploration/exploitation dilemma has been successfully addressed in the case of search trees in [6], which introduces the UCT algorithm. In this problem, each leaf is associated to a distribution of rewards and the algorithm aims to find the best leaf through repeated playouts. Based on Upper Confidence Bounds [7], it builds confidence intervals on expected rewards and follows the branch for which the upper bound is highest. In the case of online classification trees, only one playout is performed (the tree is built once). Thus, only trustworthy actions must be taken by referring to lower bounds instead of upper bounds. In [4] and [8], the authors use Hoeffding's bounds on the gain of heterogeneities, and split if the lower bound is positive. Their experiments show good results on relatively large datasets. However, we show that for more limited datasets, those algorithms are not very efficient. Indeed, Hoeffding's bounds are known to be inaccurate with a small number of samples.

If the family of an output distribution is known a priori, it is possible to use a Bayesian approach to build confidence intervals instead of the usual frequentist ones. These intervals are known as Bayesian Credible Intervals. In [9], the authors show that using such bounds proves to be optimal for the multi-armed bandit. In addition, experiments show that their performances are better than when using Hoeffding's bounds.

In this paper, we introduce an online classification tree algorithm using Bayesian Credible Intervals. Indeed, in the context of classification, the family of the output distribution

978-1-4799-7560-0/15/$31 ©2015 IEEE

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.90

is Bernoulli. Bayesian Credible Intervals can therefore be used. As they come from concentration equalities instead of inequalities, they are the tightest confidence bounds possible whatever the number of samples, and are thus well fitted for online learning.

We also study the particular case of Active Learning [10]. An Active Learning algorithm is used to decide which instances to label. Indeed, the use of an expert for annotation is considered time- and/or money-consuming. Given a finite budget of annotations, an Active Learning algorithm aims to maximize the performance of the classifier. Although some one-shot Active Learning algorithms exist [11] [12], in which all the instances to be labeled are chosen at once, Active Learning is usually seen as a sequential process [5]. Indeed, the choice of each instance is based on the current predictions, depending on all the labels received so far. An online classifier is thus necessary.

At every moment, the online classification tree defines a partition of the instance space. An Active Learning strategy thus selects the leaf from which to draw the instance. In [13], the authors introduce an Active Learning algorithm for the case of single partitions. However, the partition used cannot change all along the process of the algorithm. Thus, the partition cannot adapt to the distribution of labels among the instance space. Moreover, the number of subsets in the partition does not depend on the number of labeled instances. This makes it impractical to use on real-world datasets.

We propose to adapt their selection strategy to online classification trees. This requires that the instances were taken at random inside each leaf. By not allowing to remove any installed tests, the online classification tree respects this constraint.

This paper is structured as follows. In Section III, we introduce the Bayesian Online Classification Tree (BOCT) algorithm. We thus show how to compute Bayesian Credible Intervals and use them in place of Hoeffding bounds. In Section IV, we introduce an extension to BOCT including Active Learning. We derive our own selection strategy inspired by [13]. In Section V, we present the experiments used to evaluate the different algorithms. In Section VI, we describe the results of the evaluation and compare our algorithms to the state-of-the-art. We show that there is a significant improvement brought by using Bayesian Credible Intervals instead of Hoeffding bounds, and that Active Learning could further improve the performance. We conclude in Section VII.

II. Related Work

In [14], the author introduce an algorithm called Deep-MC-UCB. They study the problem of Active Learning in an adaptive partition, however with some differences. First, the purpose is to estimate uniformly well the mean values of the output distribution in each subset. Thus, it corresponds to regression rather than classification. In [13], the authors showed that using a loss specifically adapted for classification showed a significant improvement. Second, they use a predefined maximum partition, which is hard to use in a real-world problem. Yet, this motivated our toy problem. Last, they use Bernstein's bounds, where we can use Bayesian Credible Intervals, due to the fact that we study classification.

III. Online classification tree

In this section, we introduce an algorithm for learning a classification tree in an online fashion. Here, we only consider binary trees. In such a tree, each non-leaf node is associated to a test on an attribute, and branches represent the outcome of the test. Thus, the nodes correspond to subsets of the instance space conditioned by the successive tests. Each leaf is given a label which is the prediction for an instance belonging to the corresponding subset.

Learning this tree corresponds to determining which test is best to install, on which attribute, whether to install it or not, and which label to give to each leaf. In the online learning scenario, updating the tree corresponds to adding more decision nodes to the existing tree, without the possibility to remove any. Thus, a test must be carefully examined before being permanently installed. The solution is to first study in which case the split would be useful, then to study the probability of such a case, conditioned to the observations.

The algorithm is reported in Figure 2 and Figure 3. Apart from problem dependent parameters, it takes one parameter as input: δ, which is the probability of the true heterogeneity being inside the established intervals. The rest of the section describes the algorithm.

The labels drawn from one particular subset follow a Bernoulli distribution of unknown parameter. The true heterogeneity of a subset is a function of this parameter, usually information entropy, but others exist such as standard deviation or variance. A split is useful if it makes the heterogeneity decrease inside each obtained leaf, *i.e.*, the weighted sum of true heterogeneities of children nodes is lower than the true heterogeneity of the parent node. The true heterogeneity is not known but Bayesian Credible Intervals can be computed using labels received so far. Those intervals contain the true heterogeneity with a given probability. If the lower bound of the parent node is higher than the upper bound of the children's nodes, it is highly probable that the test must be installed. Fig. 1 shows the Bayesian Credible Intervals on the true heterogeneities of the parent node compared to ones on the weighted sum of true heterogeneities after several tests. In the following, we start by showing how to compute a Bayesian Credible Interval for the true heterogeneity.

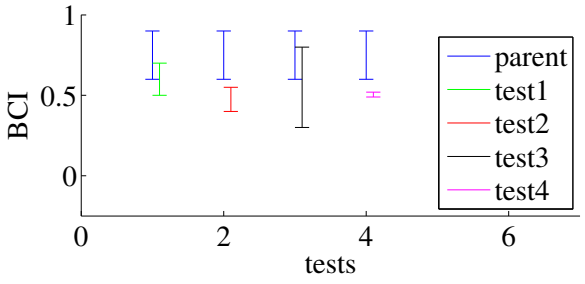


Figure 1: BCI on weighted sum of true heterogeneities after several tests compared to one of the parent node, here, test 4 would be installed.

In this paper, we use the following notations. Suppose that the instance space is X and the label set is Y. We are only interested in binary classification, in which the label set

contains two elements, in our case $Y = \{0, 1\}$. At every time step $t \in [\![1, n]\!]$, the algorithm receives a new instance $x_t \in X$ labeled $y_t \in Y$ and updates the classification tree.

Let us consider a generic subset $X_s \subset X$; we denote by $T_{s,t}$ the count of received instances that lie in that subset:

$$T_{s,t} = \sum_{i=1}^{t} \mathbb{1}_{x_i \in X_s}. \tag{1}$$

We denote by $\hat{\mu}_{s,t}$ the average of labels received in subset X_s:

$$\hat{\mu}_{s,t} = \frac{1}{T_{s,t}} \sum_{i=1}^{t} \mathbb{1}_{x_i \in X_s} y_i. \tag{2}$$

Let us denote μ_s the parameter of the Bernoulli distribution associated to subset X_s. Note that the family of conjugate priors for Bernoulli distributions is the Beta distributions. Thus, by Bayesian inference: $\forall \epsilon_s \in [0, 1]$

$$\mathbb{P}(\mu_s \geq \epsilon_s) = I_{\epsilon_s}(T_{s,t}\hat{\mu}_{s,t} + 1, T_{s,t}(1 - \hat{\mu}_{s,t}) + 1),$$

where $I(., \alpha, \beta)$ is the cumulative distribution function of a Beta distribution of parameters α and β.

Let $H(\mu_s)$ be the true heterogeneity of subset X_s. Usual true heterogeneity functions are:

- information entropy, with $H(\mu_s) = -\mu_s \log \mu_s - (1 - \mu_s) \log(1 - \mu_s)$,
- variance, with $H(\mu_s) = \mu_s(1 - \mu_s)$,
- standard deviation, with $H(\mu_s) = \sqrt{\mu_s(1 - \mu_s)}$.

But any function could work as long as it is concave, symmetrical with respect to $\frac{1}{2}$, and null for $\mu_s = 0$ and $\mu_s = 1$. It is thus maximum for $\mu_s = \frac{1}{2}$. For any $\epsilon_s \in (0, H(\frac{1}{2}))$, let $\mu_{low}(\epsilon_s)$ and $\mu_{upp}(\epsilon_s)$ be such that $H(\mu_{low}) = H(\mu_{upp}) = \epsilon_s$ and $\mu_{low} < \mu_{upp}$.

For example, if the heterogeneity is measured by the standard deviation, $H(\mu_s) = \sqrt{\mu_s(1 - \mu_s)}$, then the solutions of $H(\mu_s) = \epsilon_s$ are $\mu_{low}(\epsilon_s) = \frac{1}{2} - \sqrt{\frac{1}{4} - \epsilon_s^2}$ and $\mu_{upp}(\epsilon_s) = \frac{1}{2} + \sqrt{\frac{1}{4} - \epsilon_s^2}$.

Let us define the function f such that, $\forall \epsilon_s \in (0, H(\frac{1}{2}))$,

$$\begin{aligned} f(\epsilon_s) &= \mathbb{P}(H(\mu_s) \geq \epsilon_s) \\ &= I_{\mu_{upp}(\epsilon_s)}(T_{s,t}\hat{\mu}_{s,t} + 1, T_{s,t}(1 - \hat{\mu}_{s,t}) + 1) \\ &\quad - I_{\mu_{low}(\epsilon_s)}(T_{s,t}\hat{\mu}_{s,t} + 1, T_{s,t}(1 - \hat{\mu}_{s,t}) + 1). \end{aligned} \tag{3}$$

The function f is strictly decreasing; it thus admits an inverse f^{-1}. The upper bound $\epsilon_s(\delta)$ of the true heterogeneity according to any given probability δ is $\forall \delta \in [0, 1]$,

$$\epsilon_s(\delta) = f^{-1}(\delta). \tag{4}$$

Note that the lower bound is $\epsilon_s(1 - \delta)$. Indeed, $\forall \delta \in [0, 1]$,

$$\mathbb{P}(H(\mu_s) \leq \epsilon_s(1 - \delta)) = 1 - \mathbb{P}(H(\mu_s) \geq \epsilon_s(1 - \delta)) = \delta.$$

Thus, the Bayesian Credible Interval is such that the true heterogeneity is in $[\epsilon_s(1 - \delta), \epsilon_s(\delta)]$ with probability $1 - 2\delta$. A low probability δ is usually chosen in order to get high probability confidence intervals.

Core algorithm

Require: X, $(x_i)_{i \in [\![1,n]\!]}$, δ
  Initialize $Tree$ as a one leaf tree.
  This leaf is associated to the whole instance space X.
  for $t = 1$ to n **do**
    Receive a label y_t for the instance x_t
    Let $node$ be the node of $Tree$ which x_t belongs to
    $SubTree = Update(node, (x_i)_{i \in [\![1,n]\!]}, (y_i)_{i \in [\![1,t]\!]}, \delta)$
    Replace $node$ by $SubTree$
  end for
Ensure: $Tree$

Figure 2: Core algorithm

UpdateTree

Require: $node$, $(x_i)_{i \in [\![1,n]\!]}$,$(y_i)_{i \in [\![1,t]\!]}$, δ
  Compute T_{node}, $\hat{\mu}_{node}$ and W_{node} using Eq. (1) (2) (5)
  Compute $\epsilon_{node}(1 - \delta)$ using Eq. (3) (4)
  for $a = 1$ to N_a **do**
    $(x_{i,a}^{sorted})_{i \in [\![1,t]\!]} = sort((x_{i,a})_{i \in [\![1,t]\!]})$
    for $i = 1$ to $t - 1$ **do**
      if $x_{i,a}^{sorted} \neq x_{i+1,a}^{sorted}$ **then**
        $cutValue_{i,a} = \frac{x_{i,a}^{sorted} + x_{i+1,a}^{sorted}}{2}$
        Let X_{c_1} and X_{c_2} be two subsets resulting from splitting X_{node} along attribute a with cut value $cutValue_{i,a}$
        Compute T_{c_1}, $\hat{\mu}_{c_1}$, W_{c_1}, T_{c_2}, $\hat{\mu}_{c_2}$, W_{c_2} (1) (2) (5)
        Compute $\epsilon_{c_1}(\delta)$ and $\epsilon_{c_2}(\delta)$ using Eq. (3) (4)
        $c_{i,a} = W_{c_1}\epsilon_{c_1}(\delta) + W_{c_2}\epsilon_{c_2}(\delta)$
      end if
    end for
  end for
  $\{i^{best}, a^{best}\} = \arg\min_{i,a} c_{i,a}$
  if $c_{i^{best},a^{best}} < W_{node}\epsilon_{node}(1 - \delta)$ **then**
    Create two branches from $node$ with test:
    $x_{.,a^{best}} > cutValue_{i^{best},a^{best}}$
    leading to leaves $child_1$ and $child_2$
    $child_1 = Update(child_1, (x_i)_{i \in [\![1,n]\!]}, (y_i)_{i \in [\![1,t]\!]}, \delta)$
    $child_2 = Update(child_2, (x_i)_{i \in [\![1,n]\!]}, (y_i)_{i \in [\![1,t]\!]}, \delta)$
  end if
Ensure: $Tree$

Figure 3: UpdateTree

Please note that ϵ_s should be a function of δ, $T_{s,t}$ and $\hat{\mu}_{s,t}$. In order not to overload the notations, it is left to the reader's understanding that the index s stands for $T_{s,t}$ and $\hat{\mu}_{s,t}$.

Now that we know how to compute Bayesian Credible Intervals, let us define upon which condition a test is installed. Let us consider the installation of a test node which splits a current leaf's subset $X_p \subset X$ into two subsets $X_{c_1} \subset X$ and $X_{c_2} = X \backslash X_{c_1}$.

The heterogeneities of each node must be weighted by the size of the node's subset. We denote by W_s the size of a generic subset X_s. This size could be the volume of the subset relatively to some metric. It could be easily computed by starting with a delimited instance space and knowing the

splitting criterion. However, this does not accounts for the distribution of instances among the instance space. In order to estimate it, we could count the instances received in the subset. In this case, $W_s = T_{s,t}$. However, in order to get more precise estimates that does not change over the time, we place ourselves in a semi-supervised context. In this context, all the unlabeled instances are known from the beginning. The counts of instances now become:

$$W_s = \sum_{i=1}^{n} \mathbb{1}_{x_i \in X_s}. \tag{5}$$

Nota Bene: the difference is that the sum ends at n whereas before it ended at the current time t.

In the case where the true heterogeneities of the different subsets are known, the test should be installed if:

$$W_p H(\mu_p) > W_{c_1} H(\mu_{c_1}) + W_{c_2} H(\mu_{c_2}). \tag{6}$$

However, the true heterogeneities are not known. But, we have access to Bayesian Credible Intervals which give an information about their value. We now show a second condition which guarantees a certain probability that the previous condition is respected. We first give the upper bound of the Bayesian Credible Interval for the right side of Eq. (6).

Combining the two following upper bounds:

$$\mathbb{P}(H(\mu_{c_1}) \geq \epsilon_{c_1}(\delta)) = \delta \quad \text{and} \quad \mathbb{P}(H(\mu_{c_2}) \geq \epsilon_{c_2}(\delta)) = \delta,$$

we obtain

$$\mathbb{P}(W_{c_1} H(\mu_{c_1}) + W_{c_2} H(\mu_{c_2}) \geq W_{c_1}\epsilon_{c_1}(\delta) + W_{c_2}\epsilon_{c_2}(\delta)) \leq 2\delta.$$

We remind that we have the following lower bound of the Bayesian Credible Interval for the left side of Eq. (6):

$$\mathbb{P}(H(\mu_p) \leq \epsilon_p(1-\delta)) = \delta.$$

Finally, if the lower bound of the left side of Eq. (6) is higher than the upper bound of the right side, we can assume that there is a high probability that the condition is met. Formally,

$$\text{if} \qquad W_p\epsilon_p(\delta) > W_{c_1}\epsilon_{c_1}(\delta) + W_{c_2}\epsilon_{c_2}(\delta),$$
$$\text{then} \qquad \mathbb{P}(W_p H(\mu_p) \leq W_{c_1} H(\mu_{c_1}) + W_{c_2} H(\mu_{c_2})) \leq 3\delta.$$

This criterion is used to decide if a test has to be installed or not. However, when considering the split of a leaf, it happens that several tests may be valid. In this case, the best test to install is the one leading to the best decrease in true heterogeneity. Thus, we consider that the test for which the upper bound of children subsets' heterogeneity is lowest is best. Note that for a given parent node, the choice of the test is exactly the choice of $\{c_1, c_2\}$:

$$test^{best} = \arg\min_{test} W_{c_1}\epsilon_{c_1}(\delta) + W_{c_2}\epsilon_{c_2}(\delta).$$

At time step t, this process of choosing a test is carried out on the leaf of the current tree in which the newly received instance lies. Indeed, it is the only leaf affected by the new labeled instance. It is repeated for each newly created leaf every time a test is successfully installed.

In this section, the Bayesian Online Tree algorithm was introduced. It allows a classification tree to be updated when facing an online scenario, which avoids to recompute the whole tree at each time step. This decreases the computational time because it does not reconsider to split leaves other than the one currently receiving an instance. Here, we suppose that the function f^{-1} can be stored in a table. Indeed, the inputs are discrete and $T_{s,t}$ is bounded by n, and the same value is accessed many times while executing the algorithm.

One other advantage of this method is that the installed tests are never removed. This is particularly interesting in Active Learning, which is the focus of the next section.

IV. Active Learning of classification trees

In this section, we show how the learning rate of the previous classifier can be improved by actively selecting the instances to be labeled. At each time step t, the current classification tree partitions the instance space into N_t subsets. As it makes no distinction between the instances belonging to the same subset, the selection strategy only chooses at each time step the subset k_t in which to pick an instance at random. The goal is here to design a criterion based on labels received so far, that represents the need for receiving more labels in a subset.

The algorithm is described in Fig. 4. The input parameter δ is the probability on which are built all the confidence intervals. The rest of the section describes the operation of the algorithm.

Let us start with an intuition: if one subset only received labels of the same class, it is likely to receive the same label again. Thus, the predicted label will not change for this subset. On the contrary, if it received a mix of labels, getting another one not only refines the prediction but may also allow the subset to be split.

The aim of Active Learning is to get the best performance of the classifier given a finite budget of labeled instances. This performance is measured by the true risk based on the 0/1 loss, it thus represents the misclassification rate on unseen instances. Let us define a selection strategy which aims at decreasing the true risk. Let R_t be the true risk of a classification tree at time step t; it is the sum of the true risk in each subset:

$$R_t = \sum_{s=1}^{N_t} R_{s,t},$$

with

$$R_{s,t} = \begin{cases} W_s\mu_s & \text{if } y_s = 0 \\ W_s(1-\mu_s) & \text{if } y_s = 1, \end{cases}$$

where y_s is the label predicted by the classifier for subset s and W_s is the size of this subset. Here, the predicted label in each subset is the majority of received labels in this subset: $y_s = [\hat{\mu}_{s,t}]$, where $[.]$ is the rounding operator. Indeed, the labels received in subset s follow a Bernoulli distribution of parameter μ_s. The true risk is thus bounded by

$$R_t \leq N_t \max_{s \in [\![1, N_t]\!]} R_{s,t}.$$

Thus, in order to decrease the true risk, we aim to decrease the maximum true risk among subsets. For this purpose, the solution is to sample the subset for which the true risk is currently maximum:

$$k_t = \arg\max_{s \in [\![1, N_t]\!]} R_{s,t}.$$

Active Learning algorithm

Require: X, $(x_i)_{i\in[\![1,n]\!]}$, δ_1, δ_2
  Initialize $Tree$ as a one leaf tree.
  This leaf is associated to the whole instance space X.
  for $t = 1$ to n **do**
    for $s = 1$ to N_t **do**
      Compute $\hat{\mu}_{s,t}$, $T_{s,t}$ and W_k
      Compute $\epsilon_k(\delta_2)$ using Eq. (7)
    end for
    Compute $k_t = \arg\max_k \epsilon_k$
    Receive a label y_t for a random instance $x_t \in X_{k_t}$
    Let $node$ be the node of $Tree$ which x_t belongs to
    $SubTree = Update(node, (x_i)_{i\in[\![1,n]\!]}, (y_i)_{i\in[\![1,t]\!]}, \delta)$
    Replace $node$ by $SubTree$
  end for
Ensure: $Tree$

Figure 4: Active Learning algorithm

Note that, even though this optimal selection strategy is inspired from [13], it is not exactly identical. In their work, the authors chose to sample according to the maximum difference between the current true risk and the optimal true risk. Indeed, in the case of a fixed partition, it is unnecessary to spend effort to improve the prediction if the risk cannot be decreased. Thus, a heterogeneous subset should not receive so many labeled instances. In our case, a heterogeneous subset may be split and the risk improved, it is thus necessary to get more labeled instances in those subsets.

The value of the parameter and thus the true risk is unknown. As the selection strategy takes the form of a maximum of an unknown value, it falls in the scope of Optimism in the Face of Uncertainty. Under this paradigm, confidence intervals must be computed for each criterion, and the subset to sample is chosen according to the maximum upper bound of the intervals. In our problem, as we study classification, the family of the label distribution is known. We can therefore use Bayesian Credible Intervals for the true risk.

The family of the label distribution being Bernoulli, the family of conjugate prior is Beta distributions. By Bayesian inference: $\forall \epsilon_k \in [0,1]$

$$\mathbb{P}(\mu_k \leq \epsilon_k) = \mathcal{I}_{\epsilon_k}(T_{s,t}\hat{\mu}_{s,t} + 1, T_{s,t}(1-\hat{\mu}_{s,t}) + 1),$$

with $T_{s,t}$ the count of labeled instances in subset s and $\mathcal{I}_{\cdot}(\alpha,\beta)$ the incomplete Beta function of parameter α and β.

Thus, the cumulative distribution function of the true risk is: $\forall \epsilon_k \in [0,1]$, $\mathbb{P}(R_{s,t} \leq \epsilon_k) =$

$$\begin{cases} \mathcal{I}_{\frac{\epsilon_k}{W_k}}(T_{s,t}\hat{\mu}_{s,t} + 1, T_{s,t}(1-\hat{\mu}_{s,t}) + 1) & \text{if } [\hat{\mu}_{s,t}] = 0 \\ 1 - \mathcal{I}_{\frac{1-\epsilon_k}{W_k}}(T_{s,t}\hat{\mu}_{s,t} + 1, T_{s,t}(1-\hat{\mu}_{s,t}) + 1) & \text{if } [\hat{\mu}_{s,t}] = 1. \end{cases}$$

Suppose given $\delta_2 \in [0,1]$, we compute the upper bound $\epsilon_k(\delta_2)$ of the Bayesian Credible Interval as the value for which the probability of the true risk being higher is equal to δ_2:

$$\mathbb{P}(R_{s,t} \geq \epsilon_k(\delta_2)) = \delta_2 \iff \epsilon_k(\delta_2) = W_k \times \ldots \tag{7}$$
$$\begin{cases} \mathcal{I}^{-1}(1-\delta_2, T_{s,t}\hat{\mu}_{s,t} + 1, T_{s,t}(1-\hat{\mu}_{s,t}) + 1) & \text{if } [\hat{\mu}_{s,t}] = 0 \\ (1 - \mathcal{I}^{-1}(\delta_2, T_{s,t}\hat{\mu}_{s,t} + 1, T_{s,t}(1-\hat{\mu}_{s,t}) + 1)) & \text{if } [\hat{\mu}_{s,t}] = 1, \end{cases}$$

where $\mathcal{I}^{-1}(.,\alpha,\beta)$ is the inverse of the incomplete Beta function of parameters α and β.

The selection strategy is thus to sample the subset k_t for which the upper bound of the Bayesian Credible Interval for the true risk is maximum:

$$k_t = \arg\max_k \epsilon_k(\delta_2).$$

In this section, we introduced an Active Learning algorithm based on online classification trees. Considering an adaptive partitioning of the instance space, it allocates more effort to regions needing most. The online classification tree is well fitted to the Active Learning of an adaptive partition because it does not allow to remove splits. This is of major importance in the Active Learning algorithm in order to get meaningful estimates of the parameter. Thus, at any time step, the distribution of labeled instances inside a subset must follow the distribution of unlabeled instance. This is not the case if a split is removed and one of the children was favored over the other during the Active Learning process.

V. Experimental setup

In this section, we design experiments to evaluate the online classification tree algorithm introduced in the previous sections and its extension to the Active Learning problem. We design two kinds of experiments. The first one is a toy problem, with the true distribution parameters fixed by the user. The second is on real-world problems, where the number of available instances is limited.

a) Toy problem: This provides a better understanding of the behavior of the algorithm, but also comes with several advantages which we explain here. This problem consists in a predefined binary tree; to each leaf is given the parameter of a Bernoulli distribution and a size. The parameters and sizes are propagated back up in the tree, in order to know the parameters of all the non-leaf nodes.

While learning the online classification tree on this problem, the only tests available are those of the predefined tree. Thus, only one test is considered for each node. The purpose of the algorithm is thus only to decide if the test must be installed or not, and not to choose which one to install. One limitation is that the most refined tree that can be outputted by the algorithm is the whole predefined tree, but this one is tried to be taken sufficiently refined.

At each time step, the next sample comes from a leaf of the predefined tree, randomly drawn from a set of possible leaves with a probability proportional to its size. In the online learning scenario without Active Learning, the set of possible leaves always contains all the leaves of the predefined tree. In the Active Learning scenario, the set of possible leaves contains only the leaves descending from the subset selected by the Active Learning algorithm. The new label is then drawn from the corresponding Bernoulli distribution for this subset.

The use of known distribution parameters allows avoiding using a test set for evaluation. Indeed, the true risk is entirely known by the evaluation procedure (but not by the algorithm). Also, the number of samples that the algorithm can receive is not limited. There is no need of instances, as the labels are

given to subsets of the predefined tree, and the sizes of all subsets are given. Consequently, this problem is independent of the number of attributes. Indeed, it is only defined by the predefined tree, which can adapt to any instance space.

In our experiments, the toy problem has been fitted to an instance space which is $[0,1]^2$ being split in two along alternating dimensions, until the number of leaves reaches 2^6. The parameters of the leaves were set to

$$\mu_k = \sigma(5(x_{k,2} - \sqrt{x_{k,1}})),$$

where $x_{k,1}$ and $x_{k,2}$ represent the coordinates of the center of subset k, and $\forall x \in \mathbb{R}, \sigma(x) = \frac{1}{1+\exp^{-x}}$ is the sigmoid function. All leaves were of equal size.

b) Real-world problems: In our experiments, we use several datasets from the UCI Machine Learning repository [15]. Before each run of an experiment, the dataset is divided into two equal sets. The first is used as a pool from which the instances and their corresponding labels used for training are drawn. The unlabeled instances in the pool also serve to compute the size of each subset. The second is the test set, used for evaluation. At each time step, the algorithm makes predictions for instances in the test set, and the number of misclassified instances is recorded. The labels in the test set are totally hidden from the algorithm, which does not get any feedback from its prediction.

Experiments consisted into a certain number of runs of the algorithms. The performance in each run were averaged. The toy problem experiment consisted in 100 runs of the algorithms and a budget of 1,000 labeled instances. The real-world experiments consisted in 1,000 runs of the algorithms and were not stopped until all the instances in the training pool were labeled.

Our algorithms are compared to the following methods:

- c4.5 [1]: This is the most common batch learning classification tree algorithm. It uses information entropy as a measure of heterogeneity.
- Hoeffding: This is a modified version of VFDT [4] where the best split is not compared to the second best but only to no-split. Indeed, VFDT needs many instances before splitting in order to be sure that the installed test is the one that would be installed under perfect information. This is impractical for datasets with a limited number of instances. Hence, the only difference with BOCT is the use of Hoeffding bounds, which we want to compare.
- mbinomial [13]: This is an Active Learning algorithm which requires a fixed partition. The choice of the partition is non trivial for a real-world dataset. For this reason, we only test it on the toy problem. We instantiate this algorithm with two different partitions. The first partition we use is the whole predefined tree; we denominate the resulting algorithm by: mbinomial_64. The second partition we use is a subtree of depth 3 of the predefined tree; we denominate the resulting algorithm by: mbinomial_8.

For all the algorithms used for comparison and for each dataset, the parameter δ (δ_1 and δ_2 in the case of BOCT-AL) has been tuned using a grid search.

VI. Evaluation

In this section, we focus on comparing the algorithms introduced in the previous sections to state-of-the-art algorithms. For this, we use benchmarks described in Section V. In order to evaluate the performance, we plot the true risk, averaged over the runs, as a function of the number of labeled examples.

c) Toy problem: We start by depicting the results for the toy problem; results are displayed on Fig. 5. The goal of this experiment is to show the effect of using an adaptive partitioning of the instance space rather than a fixed partition. The mbinomial algorithm requires a fixed partition which cannot change during the process. The more refined the partition, the best optimal performance. We can see that with 64 subsets of equal size in the partition, mbinomial achieves a better performance at the end than with only 8 subsets. However, having a great number of subsets affects the learning rate. Indeed, in order to get an equivalent precision about the estimates of the parameters in each subsets, the number of points needed is proportional to the number of subsets. This is confirmed by the fact that the performances with a few labeled instances is better for mbinomial with 8 subsets than with 64.

By using an adaptive partition, the algorithm is allowed to start with a small number of subsets, and thus get a good quality of prediction for each one of them with few samples. When more samples are received, the algorithm may use a more refined partition, while keeping the same quality of prediction. BOCT and BOCT-AL succeed in splitting at the right time, by splitting a subset as soon as the gain in true risk resulting from predicting different labels in each children subset counters the loss in quality of the predictions due to getting fewer labeled instances in each one of them. We see that, this way, BOCT-AL gets the same performance as the best algorithm between mbinomial_8 and mbinomial_64, which means it succeeded in choosing the best partition at any time.

Finally, we can observe that BOCT-AL shows better performance than BOCT. Thus, the Active Learning strategy used in BOCT-AL is effective. It manages to select subsets for which either the prediction can be improved or which may be split

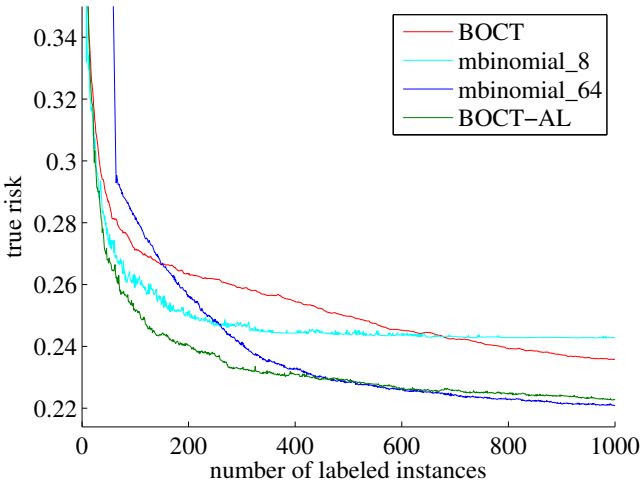


Figure 5: Toy problem

when acquiring more labeled instances.

To conclude on this experiment, we designed an adaptive partition algorithm which extends the mbinomial algorithm, by letting the partition be automatically chosen by the algorithm. It succeeds in using at any time the best partition, which leads to best performance. Now that the toy problem experiment demonstrated that our adaptive partition algorithms can effectively replace fixed partition ones, the goal is now to test it on real-world datasets.

d) Real-world datasets: In this experiment, we use four different datasets coming from the UCI Machine Learning repository [15]. The results for each of these datasets are displayed on Fig. 6[a-d]. The objective of this experiment is twofold. First, we want to test the utility of using Bayesian Credible Intervals rather than using much simpler and general bounds such as Hoeffding's bounds. Second, we want to measure the effect of actively selecting the labeled instances on the performances.

We can see that the algorithm using Hoeffding's bounds fails to achieve performances comparable to either BOCT or c4.5 on three out of four datasets. On the contrary, when comparing BOCT to c4.5, we observe that the final risks are equal on two datasets and higher for BOCT on two datasets. In addition, on Diabetes, where the Hoeffding's version is as good as c4.5, BOCT is better. Thus, BOCT is always better than the algorithm using Hoeffding's bounds. Indeed, Bayesian Credible Intervals are much tighter than Hoeffding's bounds, particularly with a small number of samples where Hoeffding's bounds are known to lack tightness. Moreover, for an online classification tree, we may want to split as soon as possible, which means that the number of received samples while having to take the right decision will always be small. It is thus of major importance to use bounds that are tight with only few samples.

One other observation that we can have on these figures is that, even though BOCT attains final performances comparable to c4.5 and even better, the initial performances are always better for c4.5. The reason for this relies on the batch nature of c4.5. This way it does not have to plan ahead the splits of the instance space. At each time step, it considers the current best splits. For example, when having received only two labeled instances, one of each label, it may split the instance space to separate them, with a good chance that this increases the performance. On the contrary, in the online scenario, as no split nodes can be removed from the tree, an early bad split affects the quality of the prediction as it requires more labeled instances to get an equal accuracy. The algorithm may want to wait before splitting, as it may lead to a better split. During this waiting period, the performance is necessarily lower than in the batch scenario, but this enables a comparable long term performance.

With all the instances being labeled, we can see that the batch algorithm performs worse than an online algorithm. The reason for this is that in c4.5, the measure of heterogeneity is totally confident on the observations. Whereas in BOCT, an uncertainty is considered through the confidence intervals. Thus, c4.5 splits too much noisy subsets (parameter of Bernoulli close to 0.5) containing few labeled instances.

One may notice that the c4.5 curves are noisier than other algorithms' curves. In fact, this noise is also present on the other algorithms but only over the runs. In the batch scenario, at each time step, the classifier is recomputed, which makes the noise appear on the curve. In the online learning scenario, the classifier is only updated, thus, the noise is the same all along one specific run.

We see that BOCT-AL shows slightly better performance than BOCT. We can notice that BOCT-AL has exactly the same performance as BOCT at the beginning of the process. This is because before the first test has been installed, there is only one subset, and the Active Learning is useless since it serves to select one subset. We also observe that at the end, the performances of BOCT-AL and BOCT are the same. Indeed, even though the priority of one cluster among another is different, the decisions in a particular subset are not affected by the state of other subsets. This means that Active Learning only changes the order in which subsets are studied. When all the instances in the pool have been labeled, the Active Learning has no effect.

To conclude on this experiment, we have seen that using Bayesian Confidence Intervals instead of Hoeffding's bounds greatly improves the performance of the algorithm. In the case of a large number of instances to label, when anticipating on the future is necessary, an online classification tree is preferred to a batch one. And the BOCT performances can be further improved by using the Active Learning criterion designed in this paper.

VII. Conclusion

In this paper, we show that Bayesian Credible Intervals are well suited for learning an online classification tree, as the family of the output distribution is known. This enables using confidence intervals which are perfectly tight, both with few and many samples. We introduce BOCT and BOCT-AL, two novel algorithms for Online and Active Learning of a classification tree. The first one uses Bayesian Credible Intervals on the heterogeneity of labels to decide when to split. The second one adds the use of Bayesian Credible Intervals on the risk in order to select the instances to be labeled. Both algorithms are evaluated on a toy problem, as well as on several real-world datasets, and compare positively to the state-of-the-art. Following work may focus on a theoretical analysis of these algorithms; this could lead to an automatic choice of the parameter. This works focuses on binary classification, and could be easily adapted to the multi-class problem. Finally, such trees could be used in a forest of trees [16].

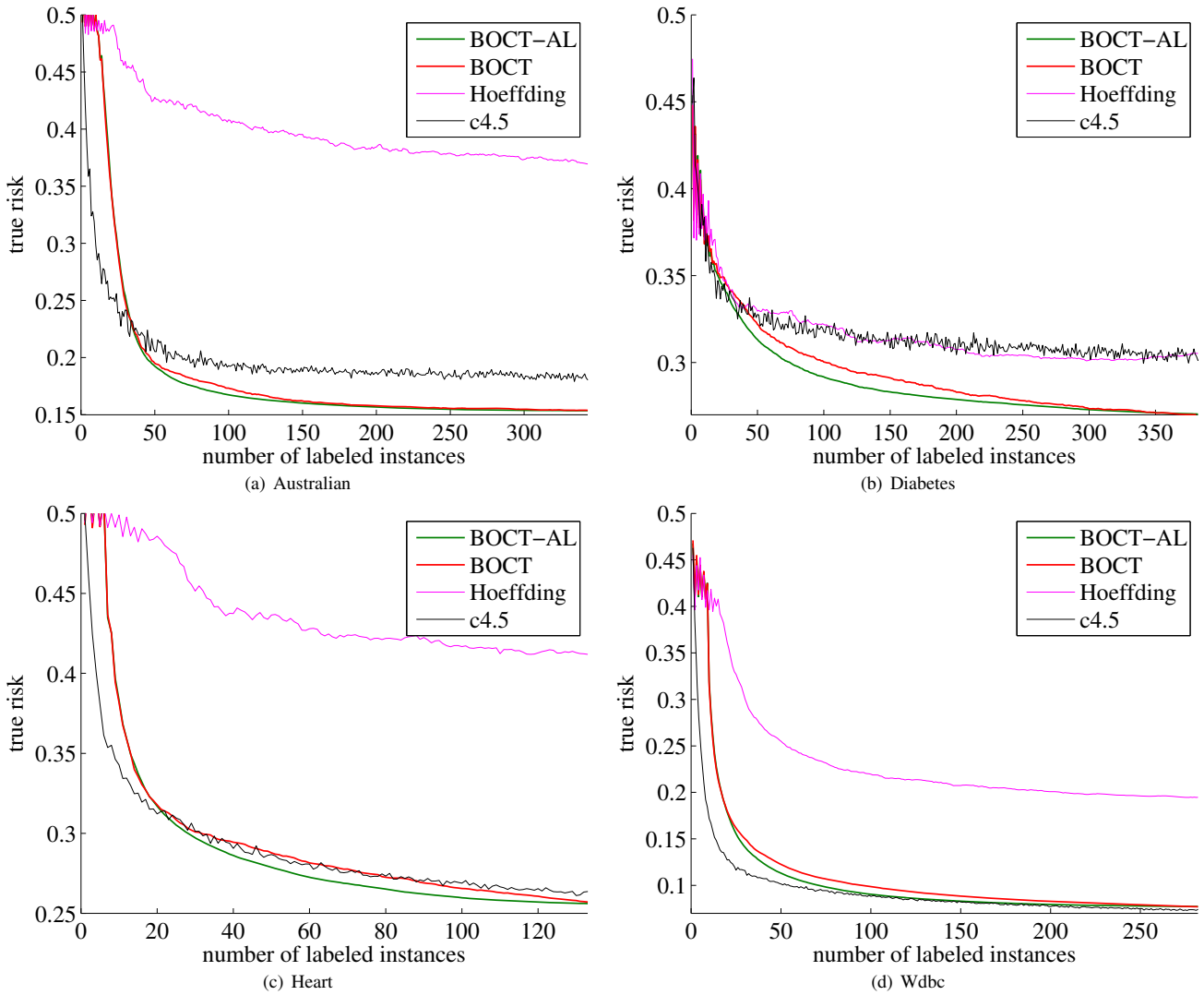
Acknowledgment

The InterCell MISN program of the French State to Lorraine region 2007-2013 aims at setting up a cluster (256 PCs) for interactive fine grain computation. It is granted by INRIA and the Region Lorraine (CPER 2007). The experiments reported in this paper where lead on this hardware.

References

[1] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[2] S. L. Crawford, "Extensions to the CART algorithm," *International Journal of Man-Machine Studies*, vol. 31, no. 2, pp. 197–217, 1989.



(a) Australian (b) Diabetes

(c) Heart (d) Wdbc

Figure 6

[3] P. E. Utgoff, N. C. Berkman, and J. A. Clouse, "Decision tree induction based on efficient tree restructuring," *Machine Learning*, vol. 29, no. 1, pp. 5–44, 1997.

[4] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the 6th SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2000, pp. 71–80.

[5] M. H. DeGroot, *Optimal statistical decisions*. John Wiley & Sons, 2005, vol. 82.

[6] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *ECML*. Springer, 2006, pp. 282–293.

[7] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, 2002.

[8] J. Gama, P. Medas, and R. Rocha, "Forest trees for on-line data," in *Proceedings of the 19th Symposium on Applied Computing*. ACM, 2004, pp. 632–636.

[9] E. Kaufmann, O. Cappé, and A. Garivier, "On Bayesian Upper Confidence Bounds for Bandit Problems," in *International Conference on Artificial Intelligence and Statistics*, 2012, pp. 592–600.

[10] B. Settles, "Active Learning Literature Survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[11] K. Yu, J. Bi, and V. Tresp, "Active learning via transductive experimental design," in *Proceedings of the 23rd International Conference on Machine Learning*. ACM, 2006, pp. 1081–1088.

[12] Q. Gu, T. Zhang, J. Han, and C. H. Ding, "Selective labeling via error bound minimization," in *Advances in Neural Information Processing Systems*, 2012, pp. 323–331.

[13] T. Collet and O. Pietquin, "Active learning for classification: An optimistic approach," in *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 2014 IEEE Symposium on*. IEEE, 2014, pp. 1–8.

[14] A. Carpentier, "De l'échantillonage optimal en grande et petite dimension," Ph.D. dissertation, Lille 1, 2012.

[15] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[16] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.