Journal of Artificial Intelligence Research 60 (2017) 1031–1055 Submitted 03/2017; published 12/2017

Confidence Decision Trees via Online and Active Learning for Streaming Data

Rocco De Rosa ROCCO.DEROSA1982@GMAIL.COM
Dipartimento di Informatica
Università degli Studi di Milano
20135 Milano, Italy

Nicolò Cesa-Bianchi NICOLO.CESA-BIANCHI@UNIMI.IT
Dipartimento di Informatica
Università degli Studi di Milano
20135 Milano, Italy

Abstract

Decision tree classifiers are a widely used tool in data stream mining. The use of confidence intervals to estimate the gain associated with each split leads to very effective methods, like the popular Hoeffding tree algorithm. From a statistical viewpoint, the analysis of decision tree classifiers in a streaming setting requires knowing when enough new information has been collected to justify splitting a leaf. Although some of the issues in the statistical analysis of Hoeffding trees have been already clarified, a general and rigorous study of confidence intervals for splitting criteria is missing. We fill this gap by deriving accurate confidence intervals to estimate the splitting gain in decision tree learning with respect to three criteria: entropy, Gini index, and a third index proposed by Kearns and Mansour. We also extend our confidence analysis to a selective sampling setting, in which the decision tree learner adaptively decides which labels to query in the stream. We provide theoretical guarantees bounding the probability that the decision tree learned via our selective sampling strategy classifies suboptimally the next example in the stream. Experiments on real and synthetic data in a streaming setting show that our trees are indeed more accurate than trees with the same number of leaves generated by state-of-the-art techniques. In addition to that, our active learning module empirically uses fewer labels without significantly hurting the performance.

1. Introduction

Stream mining algorithms are becoming increasingly attractive due to the large number of applications generating large-volume data streams. These include: email, chats, click data, search queries, shopping history, user browsing patterns, financial transactions, electricity consumption, traffic records, telephony logs, and so on. In these domains, data are generated sequentially, and scalable predictive analysis methods must be able to process new data in a fully incremental fashion. Decision trees classifiers are one of the most widespread nonparametric classification methods. They are fast to evaluate and can naturally deal with mixed-type attributes; moreover, classifiers represented by small trees are fairly easy to interpret. Decision trees have been often applied to stream mining tasks —see, e.g., the survey by Ikonomovska, Gama, and Džeroski (2011). In such settings, the tree growth is motivated by the need of fitting the information brought by the newly observed examples. Starting from the pioneering work by Utgoff (1989), the incremental learning of decision trees has received a lot of attention in the past 25 years. Several papers build on the idea of using specific measures in order to evaluate the confidence when choosing a split (Musick, Catlett, & Russell, 1993). These works include Sequential ID3 (Gratch, 1995), VFDT (Domingos & Hulten, 2000), NIP-H and NIP-N (Jin & Agrawal, 2003). Sequential ID3 uses a sequential probability ratio test in order to minimize the number of examples sufficient to choose a good split. This approach guarantees that the incrementally learned tree is close to the one trained via standard batch learning. A similar yet stronger guarantee is achieved by the Hoeffding tree algorithm, which is at the core of the state-of-the-art VFDT system. Alternative approaches, such as NIP-H e NIP-N, use Gaussian approximations instead of Hoeffding bounds in order to compute confidence intervals. Several extensions of VFDT have been proposed, also taking into account nonstationary data sources (see, e.g., Gama, Rocha, & Medas, 2003; Gama, Medas, & Rocha, 2004; Bifet, Holmes, Pfahringer, Kirkby, & Gavaldà, 2009; Yang & Fong, 2012; Pfahringer, Holmes, & Kirkby, 2007; Hulten, Spencer, & Domingos, 2001; Kirkby, Bouckaert, Studen, Lin, Kibriya, Frank, Mayo, Mayo, Mutter, Pfahringer, et al., 2007; Liu, Li, & Zhong, 2009; Gama, Kosina, et al., 2011; Xu, Qin, Hu, & Zhao, 2011; Kosina & Gama, 2012; Salperwyck & Lemaire, 2013; Duda, Jaworski, Pietruczuk, & Rutkowski, 2014). All these methods are based on the classical Hoeffding bound (Hoeffding, 1963): after m independent observations of a random variable taking values in a real interval of size R, with probability at least $1-\delta$ the true mean does not differ from the sample mean by more than

$$\varepsilon_{\text{hof}}(m,\delta) = R\sqrt{\frac{1}{2m}\ln\frac{1}{\delta}}\ . \tag{1}$$

Confidence intervals help choosing the split function maximizing the expected gain G with respect to some given gain functional that upper bounds the classification risk. If we had access to the true data distribution, then we could grow the tree optimally. Namely, we could perfectly maximize the expected gain of each split. Since expected gain translates to risk reduction, we would achieve the best possible risk reduction per leaf added to the tree. However, because we ignore the data distribution, we must choose the splits by computing gain estimates based on the data observed so far. The main result of this paper (Theorem 4), bounds the probability that a data instance in the stream gets classified through a suboptimal split. Note that, similarly to previous analyses, our result can not be used to directly bound the risk of classification. It can be used to certify that an instance was classified through a path that is as good as the one we could have obtained having given access to the true data distribution.

The problem of computing the confidence interval for the split gain estimate can be phrased as follows: we are given a set of unknown numbers G (i.e., the true gains for the available splits) and want to find the largest of them. We do that by designing a sample-based estimator $\widehat{G}$ of each G, and then use an appropriate version of the Hoeffding bound to control the probability that $\left|\widehat{G} - G\right| > \varepsilon$ for any given $\varepsilon > 0$. It is easy to see that this allows to pick the best split at any given node: assume that $\widehat{G}_F$ is the highest empirical gain (achieved by the split function F) and $\widehat{G}_{F_2}$ is the second-best (achieved by the split function F_2). If $\widehat{G}_F - \widehat{G}_{F_2} > 2\varepsilon$ then with probability at least $1-\delta$ the split function F is optimal[1] —see Figure 1. Although all methods in the abovementioned literature use the

1. In the original work VFDT (Domingos & Hulten, 2000) $\widehat{G}_F - \widehat{G}_{F_2} > \varepsilon$ is erroneously used.

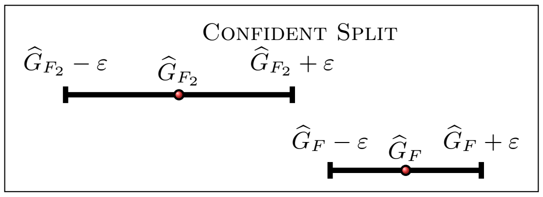

Confident Split
$\widehat{G}_{F_2} - \varepsilon$
$\widehat{G}_{F_2}$
$\widehat{G}_{F_2} + \varepsilon$
$\widehat{G}_F - \varepsilon$
$\widehat{G}_F$
$\widehat{G}_F + \varepsilon$

Figure 1: The condition $\widehat{G}_F - \widehat{G}_{F_2} > 2\varepsilon$ guarantees that the confidence intervals for the true gains G_F and G_{F_2} are non-overlapping.

Hoeffding bound (1) for computing the confidence interval associated with each split, we show here that the standard entropy-like criteria require a different approach.

The rest of the paper is organized as follows. Section 2 discusses related work. In Section 3 we state the basic decision tree learning concepts and introduce the notation used in the rest of the paper. In Section 4 we derive the new bounds for three splitting criteria. In Section 5 we apply the confidence bounds to the incremental learning of a decision tree and derive a formal guarantee (Theorem 4) on the probability that examples in the stream are classified using suboptimal splits based on any of the three splitting criteria. These theoretical guidelines are empirically tested in Section 7, where we show that our more refined bounds deliver better splits than the splits performed by the other techniques. In Section 6 we develop a selective sampling version of our algorithm based on the new confidence bounds. In selective sampling, the learner adaptively queries the labels of data points from the stream. In other words, everytime a new instance arrives from the stream, the learner may decide to save the cost of obtaining the corresponding label and thus skip a model update. The approach we propose is based on using the purity of a leaf to decide (at some confidence level) whether its classification is already optimal, thus making superfluous the acquisition of new labels. In Section 8 we compare our labeling strategy with a recent baseline. Section 9 concludes the paper.

A preliminary version of this work appeared in the Proceedings of the 2015 International Joint Conference on Neural Networks (IJCNN).

2. Related Work

The problem of computing the confidence interval for the splitting gain estimate $\widehat{G}$ can be attacked using large deviation bounds, which provide control on the *deviations* $\left|\widehat{G} - \mathbb{E}\widehat{G}\right|$. However, since splitting criteria (like entropy or Gini index) are typically nonlinear functions of the distribution at each node, further work is needed to control the *bias* $\left|\mathbb{E}\widehat{G} - G\right|$. The problem of properly controlling deviations for entropy and Gini criteria was solved by Rutkowski, Pietruczuk, Duda, and Jaworski (2013), where they used the McDiarmid bound (McDiarmid, 1989) for deriving estimates of confidence intervals $\varepsilon(m, d)$, as a function of sample size m and confidence level δ, such that $\left|\widehat{G} - \mathbb{E}\widehat{G}\right| \le \varepsilon(m, d)$ with probability at least $1 - \delta$. An example of their results is the following confidence bound for K classes and the entropy criterion,

$$\varepsilon_{\mathrm{md}}(m, \delta) = C(K, m)\sqrt{\frac{1}{2m}\ln\frac{1}{\delta}} \tag{2}$$

where $C(K, m) = 6\bigl(K \log_2 e + \log_2 2m\bigr) + 2\log_2 K$. While ignoring the bias of the estimate $\widehat{G}$, Rutkowski et al. (2013) correctly replace the Hoeffding bound (1), used in the VFDT algorithm and its successors, by the McDiarmid bound (2). In a more recent paper, Duda et al. (2014) focus on binary trees and binary classification, and apply the Hoeffding bound to the entropy splitting criterion. More specifically, they decompose the entropy gain calculation in three components, and apply the Hoeffding bound to each one of them, thus obtaining a confidence interval estimate for the deviations of the splitting gains. However, this still ignores the bias of the estimate. Matuszyk, Krempl, and Spiliopoulou (2013) directly use the classification error as a splitting criterion, rather than a concave approximation of it (like the entropy or the Gini index). Though this splitting criterion can be easily analyzed via the Hoeffding bound, its empirical performance is generally not very good —see Section 4 for more discussion on this. An improved bound for classification error is derived by Rutkowski, Jaworski, Pietruczuk, and Duda (2015), where combined splitting criteria were also considered.

In this work, we significantly simplify the approach of Rutkowski et al. (2013) and extend it to a third splitting criterion. Moreover, we also solve the bias problem, controlling the deviations of $\widehat{G}$ from the real quantity of interest (i.e., G rather than $\mathbb{E}\widehat{G}$). In addition to that, unlike Matuszyk et al. (2013) and Duda et al. (2014), our bounds apply to the most popular splitting criteria. Our analysis shows that the confidence intervals associated with the choice of a suboptimal split not only depend on the number of leaf examples m —as in bounds (1) and (2)— but also on other problem dependent parameters, as the dimension of the feature space, the depth of the leaves, and the overall number of examples seen so far by the algorithm. As revealed by the experiments in Section 7.1, this allows a more cautious and accurate splitting in complex classification problems. Furthermore, because the extensions of VFDT (see Section 1) share the same confidence analysis as the Hoeffding tree algorithm, our technique can be easily applied to all these extensions yielding similar improvements.

Standard decision tree learning approaches assume that all training instances are labeled and available beforehand. In a true incremental learning setting, in which the classifier is asked to predict the label of each incoming instance, active learning techniques (see, e.g., Settles, 2012) allow us to model the interaction between the learning system and the labeler agent (typically, a human annotator). More specifically, such techniques help the learner adaptively select a small number of instances on which the annotator is invoked in order to obtain the true label. The overall goal is to maximize predictive accuracy at any given percentage of queried labels. In recent work, Zliobaite, Bifet, Pfahringer, and Holmes (2014) introduce a general active learning framework and apply it to Hoeffding trees. Various strategies to annotate the samples considering the output leaf class probabilities are presented. These techniques rely on the class probability estimates at the leaf level, without using confidence information; that is, whether the estimates are supported by a small or large number of sampled labels. In Section 6 we develop a selective sampling version of our algorithm based on the new confidence bounds. The approach we propose is based on using the purity of a leaf to decide (at some confidence level) whether its classification is optimal. Labels of such leaves are then queried at a small rate, dictated by the confidence level that increases as a function of time plus other relevant leaf statistics. The experimental results (Section 8) show that, when applied to Hoeffding trees, our confidence-based strategy is more robust than the strategy proposed by Zliobaite et al. (2014).

3. Batch Decision Tree Learning

For simplicity we only consider binary classification problems. The goal is to find a classifier $f : \mathbb{R}^d \to \{0,1\}$ assigning the correct category $Y = \{0,1\}$ to new instances $\boldsymbol{X}$. We consider binary decision trees based on a class $\mathcal{F}$ of binary split functions $F : \mathbb{R}^d \to \{0,1\}$ partitioning the feature space[2]. Training examples $(\boldsymbol{X}_1, Y_1), (\boldsymbol{X}_2, Y_2), \ldots \in \mathbb{R}^d \times \{0,1\}$ are drawn i.i.d. from a fixed but unknown probability distribution. Decision tree classifiers are typically constructed in an incremental way, starting from a tree consisting of a single node. The tree grows through a sequence of splitting operations applied to its leaves. If a split is decided for a leaf i, then the leaf is assigned some split function $F \in \mathcal{F}$ and two nodes i_0 and i_1 are added to the tree as children of the split node. Examples are recursively routed through the tree starting from the root as follows: when an example $(\boldsymbol{X}_t, Y_t)$ reaches an internal node i with split function F, then it is routed to child i_0 if $F(\boldsymbol{X}_t) = 0$ and to child i_1 otherwise. A decision tree T induces a classifier $f_T : \mathbb{R}^d \to \{0,1\}$. The prediction $f_T(\boldsymbol{X})$ of this classifier on an instance $\boldsymbol{X}$ is computed by routing the instance $\boldsymbol{X}$ through the tree until a leaf is reached. We use $\boldsymbol{X} \to i$ to indicate that $\boldsymbol{X}$ is routed to the leaf i. Then $f_T(\boldsymbol{X})$ is set to the most frequent label among the labels of all observed examples that reach that leaf. The goal of the learning process is to control the binary classification risk $\mathbb{P}\big(f_T(\boldsymbol{X}) \neq Y\big)$ of f_T. For any leaf i, let $Y_{|i}$ be the random variable denoting the label of an instance $\boldsymbol{X}$ routed to leaf i. Let $\mathcal{L}(T)$ be the leaves of T. The risk of f_T can then be upper bounded, with the standard bias-variance decomposition, as follows

$$\begin{aligned}
&\mathbb{P}\big(f_T(\boldsymbol{X}) \neq Y\big) \\
&\le \overbrace{\sum_{i \in \mathcal{L}(T)} \mathbb{P}\big(Y \neq y_i^* \mid \boldsymbol{X} \to i\big)\mathbb{P}(\boldsymbol{X} \to i)}^{\text{bias error}} + \overbrace{\sum_{i \in \mathcal{L}(T)} \mathbb{P}\big(f_T(\boldsymbol{X}) \neq y_i^* \mid \boldsymbol{X} \to i\big)\mathbb{P}(\boldsymbol{X} \to i)}^{\text{variance error}}
\end{aligned}$$

where $y_i^* = \mathbb{I}\big\{\mathbb{P}(Y = 1 \mid \boldsymbol{X} \to i) \ge \frac{1}{2}\big\}$ is the optimal label[3] for leaf i and $\mathbb{I}\{\cdot\}$ is the indicator function of the event at argument. The notation $\mathbb{P}\big(\cdot \mid \boldsymbol{X} \to i\big)$ is used to denote probabilities conditioned on the event that the instance $\boldsymbol{X}$ is routed to leaf i.

The *variance* terms are the easiest to control: $f_T(\boldsymbol{X})$ is determined by the most frequent label of the leaf i such that $\boldsymbol{X} \to i$. Hence, conditioned on $\boldsymbol{X} \to i$, the event $f_T(\boldsymbol{X}) = y_i^*$ holds with high probability whenever the confidence interval for the estimate of y_i^* does not cross the $\frac{1}{2}$ boundary[4]. The bias terms compute the Bayes error at each leaf. The error vanishes quickly when good splits for expanding the leaves are available. However,

2. Although we only considered binary classification and binary splits, our techniques can be potentially extended to multi-class classification and general splits.
3. The label assigned by the Bayes optimal classifier in the leaf partition.
4. This confidence interval shrinks relatively fast, as dictated by Hoeffding bound applied to the variable $y_i^* \in \{0,1\}$.

due to the large number of available split functions F, the confidence intervals for choosing such good splits shrink slower than the confidence interval associated with the bias error. Our Theorem 4 accurately quantifies the dependence of the split confidence on the various problem parameters. Let $\Psi(Y)$ be a shorthand for $\min\bigl\{\mathbb{P}(Y=0),\mathbb{P}(Y=1)\bigr\}$; similarly, let $\Psi(Y \mid A)$ be a shorthand for $\min\bigl\{\mathbb{P}(Y=0 \mid A),\mathbb{P}(Y=1 \mid A)\bigr\}$, where A is any event such that $\mathbb{P}(A)>0$. Every time a leaf i is split using F, the term $\Psi(Y_{|i})$ gets replaced by

$$\mathbb{P}\bigl(F=0 \mid \boldsymbol{X}\to i\bigr)\Psi\bigl(Y_{|i} \mid F=0\bigr)+\mathbb{P}\bigl(F=1 \mid \boldsymbol{X}\to i\bigr)\Psi\bigl(Y_{|i} \mid F=1\bigr)$$

corresponding to the newly added leaves (here and in what follows, F also stands for the random variable $F(\boldsymbol{X})$). The concavity of min ensures that no split of a leaf can ever make that sum bigger. Of course, we seek the split maximizing the risk decrease (or "gain"),

$$\Psi(Y_{|i})-\mathbb{P}\bigl(F=0 \mid \boldsymbol{X}\to i\bigr)\Psi\bigl(Y_{|i} \mid F=0\bigr)-\mathbb{P}\bigl(F=1 \mid \boldsymbol{X}\to i\bigr)\Psi\bigl(Y_{|i} \mid F=1\bigr)\ .$$

In practice, splits are chosen so to approximately maximize a gain functional defined in terms of a concave and symmetric function Φ, which bounds from the above the min function Ψ (used by Matuszyk et al., 2013 as splitting criterion). The curvature of Φ helps when comparing different splits, as opposed to Ψ which is piecewise linear. Indeed Ψ gives nonzero gain only to splits generating leaves with disagreeing majority labels (see, e.g., Dietterich, Kearns, & Mansour, 1996 for a more detailed explanation). Let Z be a Bernoulli random variable with parameter p. Three gain functions used in practice are: the scaled binary entropy $H_{1/2}(Z)=-\frac{p}{2}\ln p-\frac{1-p}{2}\ln(1-p)$ used in C4.5; the Gini index $J(Z)=2p(1-p)$ used in CART, and the function $Q(Z)=\sqrt{p(1-p)}$ introduced by Kearns and Mansour (1996) and empirically tested by Dietterich et al. (1996). Clearly, the binary classification risk can be upper bounded in terms of any upper bound Φ on the min function Ψ,

$$\mathbb{P}\bigl(f_T(\boldsymbol{X})\neq Y\bigr)\le\sum_{i\in\mathcal{L}(T)}\Phi(Y_{|i})\mathbb{P}(\boldsymbol{X}\to i)+\sum_{i\in\mathcal{L}(T)}\mathbb{P}\bigl(f_T(\boldsymbol{X})\neq y_i^* \mid \boldsymbol{X}\to i\bigr)\mathbb{P}(\boldsymbol{X}\to i)\ .$$

The gain for a split F at node i, written in terms of a generic entropy-like function Φ, takes the form

$$\begin{aligned}G_{i,F}&=\Phi(Y_{|i})-\Phi\bigl(Y_{|i} \mid F\bigr)\\&=\Phi(Y_{|i})-\mathbb{P}\bigl(F=0 \mid \boldsymbol{X}\to i\bigr)\Phi(Y_{|i} \mid F=0)-\mathbb{P}(F=1 \mid \boldsymbol{X}\to i)\Phi(Y_{|i} \mid F=1)\ . \qquad (3)\end{aligned}$$

Now, in order to choose splits with a high gain (implying a significant reduction of risk), we must show that $G_{i,F}$ (for the different choices of Φ) can be reliably estimated from the training examples. In this work we focus on estimates for choosing the best split F at any given leaf i. Since the term $\Phi(Y_{|i})$ in $G_{i,F}$ is invariant with respect to this choice, we may just ignore it when estimating the gain.[5]

4. Confidence Bound For Split Functions

In this section we compute estimates $\widehat{\Phi}_{i|F}$ of $\Phi(Y_{|i} \mid F)$ for different choices of Φ, and compute confidence intervals for these estimates. As mentioned in Section 1, we actually

5. Note that, for all functions Φ considered in this paper, the problem of estimating $\Phi(Y_{|i})$ can be solved by applying essentially the same techniques as the ones we used to estimate $\Phi\bigl(Y_{|i} \mid F\bigr)$.

bound the deviations of $\widehat{\Phi}_{i|F}$ from $\Phi(Y_{|i} \mid F)$, which is the real quantity of interest here. Due to the nonlinearity of Φ, this problem is generally harder than controlling the deviations of $\widehat{\Phi}_{i|F}$ from its expectation $\mathbb{E}\,\widehat{\Phi}_{i|F}$ (see, e.g., Rutkowski et al., 2013 for weaker results along these lines). In the rest of this section, for each node i and split F we write $p_k = \mathbb{P}(Y = 1, F = k)$ and $q_k = 1 - p_k$ for $k \in \{0,1\}$; moreover, we use $\widehat{p}_k$, $\widehat{q}_k$ to denote the empirical estimates of p_k, q_k.

4.1 Bound for the Entropy

Let $\Phi(Z)$ be the (scaled) binary entropy $H_{1/2}(Z) = -\frac{p}{2}\ln p - \frac{1-p}{2}\ln(1-p)$ for Z Bernoulli of parameter p. In the next result, we decompose the conditional entropy as a difference between entropies of the joint and the marginal distribution. Then, we apply standard results for plug-in estimates of entropy.

Theorem 1 *Pick a node i and route m i.i.d. examples $(\boldsymbol{X}_t, Y_t)$ to i. For any $F \in \mathcal{F}$, let*

$$\widehat{\Phi}_{i|F} = \widehat{H}_{1/2}(Y_{|i}, F) - \widehat{H}_{1/2}(F)$$

where $\widehat{H}_{1/2}$ denotes the empirical scaled entropy (i.e., the scaled entropy of the empirical measure defined by the i.i.d. sample). Then, for all $\delta > 0$,

$$\left|\widehat{\Phi}_{i|F} - H_{1/2}(Y_{|i} \mid F)\right| \le \varepsilon_{\mathrm{ent}}(m,\delta) \qquad \textit{where} \quad \varepsilon_{\mathrm{ent}}(m,\delta) = (\ln m)\sqrt{\frac{2}{m}\ln\frac{4}{\delta}} + \frac{2}{m}$$

with probability at least $1-\delta$ over the random draw of the m examples.

Proof 1 *In appendix A.*

4.2 Bound for the Gini Index

In the Bernoulli case, the Gini index takes the simple form $J(Z) = 2p(1-p)$ for Z Bernoulli of parameter p. First we observe that $J(Y_{|i} \mid F)$ is the sum of harmonic averages, then we use the McDiarmid inequality to control the variance of the plug-in estimate for these averages.

Theorem 2 *Pick a node i and route m i.i.d. examples $(\boldsymbol{X}_t, Y_t)$ to i. For any $F \in \mathcal{F}$, let*

$$\widehat{\Phi}_{i|F} = \mathrm{HM}(\widehat{p}_1, \widehat{q}_1) + \mathrm{HM}(\widehat{p}_0, \widehat{q}_0)$$

where HM denotes the harmonic mean $\mathrm{HM}(p,q) = \frac{2pq}{p+q}$. Then, for all $\delta > 0$

$$\left|\widehat{\Phi}_{i|F} - J(Y_{|i} \mid F)\right| \le \varepsilon_{\mathrm{Gini}}(m,\delta) \qquad \textit{where} \quad \varepsilon_{\mathrm{Gini}}(m,\delta) = \sqrt{\frac{8}{m}\ln\frac{2}{\delta}} + 4\sqrt{\frac{1}{m}}$$

with probability at least $1-\delta$ over the random draw of the m examples.

Proof 2 *In appendix B.*

4.3 Bound for the Kearns-Mansour Index

The third entropy-like function we analyze is $Q(Z) = \sqrt{p(1-p)}$ for Z Bernoulli of parameter p. The use of this function was motivated by a theoretical analysis of decision tree learning as a boosting procedure (Kearns & Mansour, 1996). See also the work of Takimoto and Maruoka (2003) for a simplified analysis and some extensions.

In this case McDiarmid inequality is not applicable because of the lack of good upper bounds on the constant c in (8). Hence we control $Q(Y_{|i} \mid F)$ using a direct argument based on classical large deviation results.

Theorem 3 *Pick a node i and route m i.i.d. examples $(\boldsymbol{X}_t, Y_t)$ to i. For any $F \in \mathcal{F}$, let*

$$\widehat{\Phi}_{i|F} = \sqrt{\widehat{p}_1 \widehat{q}_1} + \sqrt{\widehat{p}_0 \widehat{q}_0} \ .$$

Then, for all $\delta > 0$,

$$\left|\widehat{\Phi}_{i|F} - Q(Y_{|i} \mid F)\right| \le \varepsilon_{\mathrm{KM}}(m,\delta) \qquad \textit{where} \quad \varepsilon_{\mathrm{KM}}(m,\delta) = 4\sqrt{\frac{1}{m}\ln\frac{8}{\delta}}$$

with probability at least $1-\delta$ over the random draw of the m examples.

Proof 3 *In appendix C.*

5. Confidence Decision Tree Algorithm

A setting in which confidence intervals for splits are extremely useful is online or stream-based learning. In this setting, examples are received incrementally, and a confidence interval can be used to decide how much data should be collected at a certain leaf before a good split F can be safely identified. A well-known example of this approach are the so-called Hoeffding trees (Domingos & Hulten, 2000). In this section, we show how our confidence interval analysis can be used to extend and refine the current approaches to stream-based decision tree learning. For $t = 1, 2, \dots$ we assume the training example $(\boldsymbol{X}_t, Y_t)$ is received at time t. C-Tree (Algorithm 1) describes the online decision tree learning approach.

Algorithm 1 C-Tree

```
Input: Threshold $\tau > 0$
 1: Build a 1-node tree $T$
 2: for $t = 1, 2, \dots$ do
 3:   Route example $(\boldsymbol{X}_t, Y_t)$ through $T$ until a leaf $\ell_t$ is reached
 4:   if $\ell_t$ is not pure then
 5:     Let $\widehat{F} = \operatorname{argmax}_{F \in \mathcal{F}} \widehat{\Phi}_{\ell_t,F}$ and $\widehat{F}_2 = \operatorname{argmax}_{F \in \mathcal{F} : F \neq \widehat{F}} \widehat{\Phi}_{\ell_t,F}$
 6:     if $\widehat{\Phi}_{\ell_t,\widehat{F}} \le \widehat{\Phi}_{\ell_t,\widehat{F}_2} - 2\varepsilon_t$ or $\varepsilon_t \le \tau$ then
 7:       Let $F_{\ell_t} = \widehat{F}$ and expand $\ell_t$ using split $F_{\ell_t}$
 8:     end if
 9:   end if
10: end for
```

A stream of examples is fed to the algorithm, which initially uses a 1-node decision tree. At time t, example $(\boldsymbol{X}_t, Y_t)$ is routed to a leaf ℓ_t. If the leaf is not pure (both positive and negative examples have been routed to ℓ_t), then the empirically best $\widehat{F}$ and the second-best $\widehat{F}_2$ split for ℓ_t are computed. If the difference in gain between these two splits exceeds a value ε_t, computed via the confidence interval analysis, then the leaf is split using $\widehat{F}$. The leaf is also split when ε_t goes below a "tie-break" parameter τ, indicating that the difference between the gains of $\widehat{F}$ and $\widehat{F}_2$ is so tiny that waiting for more examples in order to find out the really best split is not worthwhile. Let $\varepsilon(m, \delta)$ be the size of the confidence interval computed via Theorem 1, 2 or 3. Fix any node i and let $m_{i,t}$ be the number of examples routed to that node in the first $t-1$ time steps. Clearly, for any $F, F' \in \mathcal{F}$ with $F \neq F'$, if $\widehat{\Phi}_{i|F} \leq \widehat{\Phi}_{i|F'} - 2\varepsilon(m_{i,t}, \delta)$ then $G_{i,F} \geq G_{i,F'}$ with probability at least $1-\delta$. Now, since the number of possible binary splits is at most $dm_{i,t}$, if we replace δ by $\delta/(dm_{i,t})$ the union bound guarantees that a node i is split using the function maximizing the gain.

Lemma 1 *Assume a leaf i is expanded at time t only if there exists a split $\widehat{F}$ such that*

$$\widehat{\Phi}_{i|\widehat{F}} \leq \widehat{\Phi}_{i|F} - 2\varepsilon\big(m_{i,t}, \delta/(dm_{i,t})\big)$$

for all $F \in \mathcal{F}$ such that $F \neq \widehat{F}$. Then, $\widehat{F} = \operatorname*{argmax}_{F \in \mathcal{F}} G_{i,F}$ with probability at least $1-\delta$.

A reasonable goal in stream-based decision tree learning is ensuring that all splits are chosen optimally. That is, whenever a leaf gets replaced by an internal node, the split uses the best available function with respect to some underlying gain criterion defined in terms of the true data distribution. Because gains translate to drops in classification risk, leaves below nodes that were split optimally tend to be more accurate than leaves below nodes created with suboptimal splits. The next result provides a bound on the probability that, while running C-Tree, the t-th instance $\boldsymbol{X}_t$ is routed using a suboptimal split in the current tree (a similar result was proven by Domingos & Hulten, 2000 for Hoeffding trees). Here the probability is computed over the random draw of the first t examples. In other words, the theorem bounds the probability that two bad things simultaneously happen: first, the tree built using the first $t-1$ examples contains one or more splits which are not optimal; second, the t-th example is routed through one or more of these bad splits. Bad splits are formally defined using the following notion of suboptimality: a split F is suboptimal for node i when $G_{i,F} < \max_{F' \in \mathcal{F}} G_{i,F'}$.

Theorem 4 *Assume C-Tree (Algorithm 1) is run with*

$$\varepsilon_t = \varepsilon\left(m_{\ell_t,t}, \frac{\delta_t}{(h_t+1)(h_t+2)(t+1)^3 dm_{\ell_t,t}}\right) \tag{4}$$

where $\varepsilon(m,\delta)$ is the size of the confidence interval computed via Theorem 1, 2 or 3 and h_t is depth of ℓ_t. Then the probability that $\boldsymbol{X}_t$ is routed via one or more suboptimal splits is at most δ_t.

Proof 4 *Let T be the tree built by C-Tree in the first $t-1$ time steps and let $\mathcal{D}_h$ be the set of internal nodes at depth h. Clearly,*

$$\sum_{i \in \mathcal{D}_h} \mathbb{P}(\boldsymbol{X} \to i) \leq 1 \; . \tag{5}$$

For any internal node i of T, let $t_i < t$ be the time step when the node was split and let F_i be the function chosen by C-Tree at node i. Also, let F_i^* be any optimal split for node i. That is, $G_{i,F_i^*} = \max_{F \in \mathcal{F}} G_{i,F}$. Further let $\delta' = \frac{\delta}{(h+1)(h+2)d}$. We have

$$\begin{aligned}
&\mathbb{P}\big(\boldsymbol{X} \text{ routed via a suboptimal split}\big) = \mathbb{P}\big(\exists i \,:\, \boldsymbol{X} \to i,\ G_{i,F_i} < G_{i,F_i^*}\big) \\
&\quad\le \sum_{h=0}^{H} \sum_{i \in \mathcal{D}_h} \mathbb{P}\big(i \in \mathcal{D}_h,\ G_{i,F_i} < G_{i,F_i^*} \mid \boldsymbol{X} \to i\big)\mathbb{P}(\boldsymbol{X} \to i) \\
&\quad\le \sum_{h=0}^{H} \sum_{i \in \mathcal{D}_h} \mathbb{P}\left(i \in \mathcal{D}_h,\ G_{i,F_i} < G_{i,F_i^*},\ \widehat{\Phi}_{i|F_i} \le \widehat{\Phi}_{i|F_i^*} - 2\varepsilon\left(m_{i,t_i}, \frac{\delta'}{(t_i+1)^3 m_{i,t_i}}\right) \,\middle|\, \boldsymbol{X} \to i\right) \\
&\qquad\qquad \times \mathbb{P}(\boldsymbol{X} \to i) \\
&\quad\le \sum_{h=0}^{H} \sum_{i \in \mathcal{D}_h} \sum_{r=1}^{t-1} \sum_{s=1}^{r} \mathbb{P}\left(i \in \mathcal{D}_h,\ G_{i,F_i} < G_{i,F_i^*},\ \widehat{\Phi}_{i|F_i} \le \widehat{\Phi}_{i|F_i^*} - 2\varepsilon\left(s, \frac{\delta'}{(r+1)^3 s}\right) \,\middle|\, \boldsymbol{X} \to i\right) \\
&\qquad\qquad \times \mathbb{P}(\boldsymbol{X} \to i) \\
&\quad\le \sum_{h=0}^{H} \sum_{i \in \mathcal{D}_h} \mathbb{P}(\boldsymbol{X} \to i) \left(\sum_{r=1}^{t-1} \sum_{s=1}^{r} \frac{\delta}{(r+1)^3 (h+1)(h+2)s}\right) \qquad \textit{by Lemma 1} \\
&\quad\le \sum_{h=0}^{H} \frac{\delta}{(h+2)^2} \sum_{r=1}^{t-1} \sum_{s=1}^{r} \frac{1}{(r+1)^3} \qquad \textit{by (5)} \\
&\quad\le \delta\ .
\end{aligned}$$

The second inequality holds because

$$\widehat{\Phi}_{i|F_i} \le \widehat{\Phi}_{i|F_i^*} - 2\varepsilon\left(m_{i,t_i}, \frac{\delta'}{(t_i+1)^3 m_{i,t_i}}\right)$$

must be true when F_i is chosen at time t_i as the split at node i. The last inequality uses

$$\sum_{h=0}^{H} \frac{1}{(h+1)(h+2)} \sum_{r=1}^{t-1} \frac{r}{(r+1)^3} \le 1$$

which, in turn, relies on the facts $\sum_{h \ge 1} \big(h(h+1)\big)^{-1} = 1$ and $\sum_{k \ge 2} k^{-2} \le 1$.

Remark 1 *Choosing $\delta_t = \frac{1}{t}$ in Theorem 4 implies that only a logarithmic number of examples in the stream are routed via suboptimal splits. More formally, if N_T is the number of instances in that are routed via suboptimal splits in a stream of length T,*

$$\mathbb{E}\big[N_T\big] = \sum_{t=1}^{T} \mathbb{P}\big(\boldsymbol{X}_t \text{ routed via a suboptimal split}\big) \le \sum_{t=1}^{T} \frac{1}{t} = \mathcal{O}\big(\ln T\big)\ .$$

6. Selective Sampling for Decision Tree Learning

In a truly online learning setting, such as a surveillance system or a medical monitoring application, the classification system is asked to predict the label of each incoming data item.

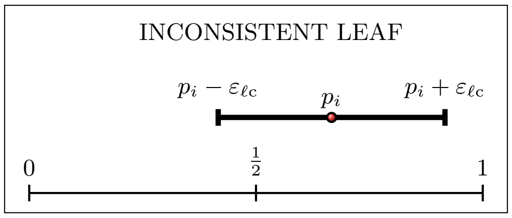


Figure 2: Example of a leaf which is not δ-consistent. The class probability confidence interval overlaps 0.5. In this case we are not sure, at the desired confidence level, if the prediction made by the leaf is the same as that of the Bayesian optimal classifier in the correspondent sub-region.

However, training labels can be only obtained through the help of a costly human annotator, who should be invoked only when the confidence in the classification of the current instance falls below a certain level. Selective sampling allows to model this interaction between the learning system and the labeler agent. In our online decision tree setting, we apply the selective sampling mechanism at the leaf level: whenever enough examples are routed to leaf i such that the event $f_T(\boldsymbol{X}) = y_i^*$ holds with the desired confidence level, the algorithm seldom asks the labels of additional examples that are routed to the same leaf.

Recall that ℓ_t is the leaf to which example $(\boldsymbol{X}_t, Y_t)$ is routed, and that $m_{\ell_t,t}$ is the number of queried data points routed to leaf ℓ_t in the first $t-1$ time steps. Let $\widehat{Y}_{\ell_t,t}$ be the fraction of positive examples among those points, so that $f_T(\boldsymbol{X}_t) = \mathbb{I}\Big\{\widehat{Y}_{\ell_t,t} \ge \frac{1}{2}\Big\}$. We say that ℓ_t is δ-consistent if

$$\big|\widehat{Y}_{\ell_t,t} - \tfrac{1}{2}\big| > \varepsilon_{\ell c}(m_{\ell_t,t}, t, \delta) \qquad \text{where} \quad \varepsilon_{\ell c}(m,t,\delta) = \sqrt{\frac{1}{2m}\ln\frac{2t}{\delta}}. \tag{6}$$

Let $p_i = \mathbb{P}(Y = 1 \mid \boldsymbol{X} \to i)$. If ℓ_t is δ-consistent but $f_T(\boldsymbol{X}_t) \neq y^*_{\ell_t}$, then it must be that $\big|\widehat{Y}_{\ell_t,t} - p_{\ell_t}\big| \ge \sqrt{\frac{1}{2m_{\ell_t,t}}\ln\frac{2t}{\delta}}$. Hence, when a leaf becomes δ-consistent we are confident that its classification is optimal at a certain confidence level —see Figure 2. We how show that the probability that the classification of a δ-consistent leaf is non-optimal is at most δ.

Theorem 5 $\mathbb{P}\big(f_T(\boldsymbol{X}_t) \neq y^*_{\ell_t},\ \ell_t \textit{ is } \delta\textit{-consistent}\big) \le \delta$.

Proof 5 *We have*

$$\begin{aligned}
&\mathbb{P}\big(f_T(\boldsymbol{X}_t) \neq y^*_{\ell_t},\ \ell_t \textit{ is } \delta\textit{-consistent}\big)\\
&\quad\le \sum_{s=0}^{t-1} \mathbb{P}\left(\big|\widehat{Y}_{\ell_t,t} - p_{\ell_t}\big| \ge \sqrt{\frac{1}{2s}\ln\frac{2t}{\delta}}\right)\\
&\quad\le \sum_{s=0}^{t-1} \sum_{i\in\mathcal{L}(T)} \mathbb{P}\left(\big|\widehat{Y}_{i,t} - p_i\big| \ge \sqrt{\frac{1}{2s}\ln\frac{2t}{\delta}}\ \middle|\ \boldsymbol{X}_t \to i\right)\mathbb{P}(\boldsymbol{X}_t \to i)\\
&\quad\le \sum_{s=0}^{t-1} \sum_{i\in\mathcal{L}(T)} \frac{\delta}{t}\mathbb{P}(\boldsymbol{X}_t \to i) = \delta
\end{aligned}$$

where we used the standard Chernoff bound in the last step.

Similarly to Theorem 4, choosing $\delta = \frac{1}{t}$ and applying the union bound allows to conclude that at most a logarithmic number of examples in the stream are misclassified by δ-consistent leaves. We use this setting in all the experiments.

As labels are generally obtained via queries to human annotators, any practical active learning system for streaming settings should impose a bound on the query rate. The active framework we propose is taken from the work by Zliobaite et al. (2014) —see Algorithm 3 (ACTIVE setting). Whenever a new instance is presented to the model, the system makes a prediction and then invokes the active learning module in order to determine whether the label should be requested. If this is the case, then a query is issued to the annotator unless the query rate budget is violated. When the label is not requested, the model is not updated. Our labeling strategy, described in Algorithm 1, is the following: if the incoming sample is routed to a leaf which is not δ-consistent (see Figure 2), then the annotation is requested. However, in order to guarantee enough exploration, annotations are requested with some probability even on δ-consistent leaves. In this case the sampling probability is proportional to the budget rate B and to the class probability confidence interval $\varepsilon_{\ell \mathrm{c}}$ for the leaf. Moreover, the sampling probability is also inversely proportional to the "margin" of the class probability estimate, $\left|\widehat{Y}_{\ell_t,t} - \frac{1}{2}\right|$.

Algorithm 2 Query Strategy

```
1: Route instance $\boldsymbol{x}_t$ through $T$ until a leaf $\ell_t$ is reached
2: if $\ell_t$ is not $\delta$-consistent then
3:     return true
4: else
5:     return true with probability $\dfrac{B + \varepsilon_{\ell \mathrm{c}}}{B + \varepsilon_{\ell \mathrm{c}} + \left|\widehat{Y}_{\ell_t,t} - \frac{1}{2}\right|}$
6: end if
7: return false
```

7. Full Sampling Experiments

We ran experiments on synthetic datasets and popular benchmarks, comparing our C-Tree (Algorithm 1) against two baselines: H-Tree (VDFT algorithm in Domingos & Hulten, 2000) and CorrH-Tree (the method from Duda et al., 2014 using the classification error as splitting criterion).

The three methods (ours and the two baselines) share the same core, i.e., the HoeffdingTree (H-Tree) algorithm implemented in MOA[6]. In order to implement C-tree and the baseline CorrH-Tree, we directly modified the H-Tree code in MOA. The grace period parameter[7] was set to 100. In contrast to the typical experimental settings in the literature, we did not consider the tie-break parameter because in the experiments we observed that it caused the majority of the splits. Based on Theorem 4 and Remark 1 (which leads to the

6. `moa.cms.waikato.ac.nz/`
7. This is the parameter dictating how many new examples since the last evaluation should be routed to a leaf before revisiting the decision (see Domingos & Hulten, 2000).

choice $\delta_t = \frac{1}{t}$), we used the following version of our confidence bounds $\varepsilon_{\mathrm{KM}}$ and $\varepsilon_{\mathrm{Gini}}$ (the bound for $\varepsilon_{\mathrm{ent}}$ contains an extra $\ln m$ factor),

$$\widetilde{\varepsilon}_{\mathrm{KM}} = \widetilde{\varepsilon}_{\mathrm{Gini}} = c\sqrt{\frac{1}{m}\ln\bigl(m^2h^2td\bigr)} \tag{7}$$

where the parameter c is used to control the number of splits.

Remark 2 *The scaling factor c is introduced because the constants in large deviations bounds are typically suboptimal. Therefore, as the bound is known to be correct only up to scaling constants, choosing c empirically makes sense. Note that although our choice of c is dataset-specific, we use the same value of c in each streaming experiment, which implies that (7) is used with the same c and different values of t, m, and h.*

The logarithmic dependence on t in (7) is the result of setting $\delta_t = \frac{1}{t}$. In the experiments, we used $\ln t$ rather than $4\ln t$ dictated by (4) because we conjecture that the constant 4 is just an artifact of the proof of Theorem 4.

In a preliminary round of experiments, we found that the Gini index delivered a performance comparable to that of entropy and Kearns-Mansour, but —on average— produced trees that were more compact for all three algorithms (ours and the two baselines). Hence, we ran all remaining experiments using the Gini index.

In all experiments we measured the *online performance*. This is the average performance (either accuracy or F-measure) when each new example in the stream is predicted using the tree trained only over the past examples in the stream ("Interleaved Test-Then-Train" validation in MOA) —see Algorithm 3 (FULL setting).

Algorithm 3 Online Stream Validation Protocol

```
Input: labeling budget $B$
1: Initialize online accuracy $M_0 = 0$
2: for $i = 1, 2, \ldots$ do
3:    Receive instance $\boldsymbol{x}_i$
4:    Predict $\hat{y}_i$
5:    Update $M_i = \bigl(1 - \frac{1}{i}\bigr)M_{i-1} + \frac{1}{i}\mathbb{I}\{\hat{y}_i = y_i\}$
6:    if FULL setting then
7:       Receive true label $y_i$
8:       Update model using new example $(\boldsymbol{x}_i, y_i)$
9:    else (ACTIVE setting)
10:      if budget $B$ not exceeded and Query Strategy = true then
11:         Request true label $y_i$
12:         Update query rate and model using $(\boldsymbol{x}_i, y_i)$
13:      end if
14:   end if
15: end for
```

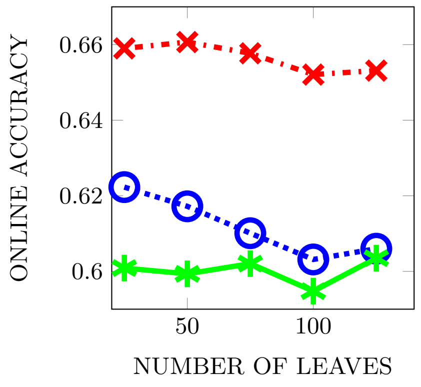


Figure 3: Online accuracy of C-Tree (cross with red line), H-Tree (circle with blue line), and CorrH-Tree (star with green line) against number of leaves. The parameters δ in H-Tree and CorrH-Tree, and c in C-Tree (in C-Tree we set $\delta_t = \frac{1}{t}$ according to Remark 1). were individually tuned on each dataset using a grid of 200 values, hence plots show the online performance of each algorithm when it is close to be optimally tuned. The ranges for the parameters are $c \in (0, 2)$ and $\delta \in (0, 1)$. The optimal values of c and δ generated in the tuning phase are, respectively, typically around $5 * 10^{-3}$ and 10^{-2}.

7.1 Controlled Experiments

These experiments show how the detailed form of our confidence bound, which —among other things— takes into account the number d of attributes and the structure of the tree (through the depth of the leaves to split), allows C-Tree to select splits that are generally better because they reduce risk more effectively than the splits selected by the baselines. In particular, we generated data streams from a random decision trees with 50 leaves and observed that C-Tree dominates across the entire range of parameters. Note that C-Tree achieves the best accuracy as a function of the number of generated leaves —see Figure 3.

The random binary trees were generated with fixed class distributions in each leaf according to Algorithm 7 (see Appendix D for a full description).

In Figure 3 we show online performances averaged over 1000 streams, each generated using a different random binary tree. The plots are obtained as follows: for each dataset and algorithm we logged the running average of the online performance —M_i in Algorithm 3— and the total number of leaves in the tree as the stream was being fed to the algorithm.

7.2 Experiments on Real-World Data

We constructed ten different streams from each dataset listed below here by taking a random permutation of the examples in it. A9A, COD-RNA and COVERTYPE are from the LIBSVM binary classification repository[8]. AIRLINES and ELECTRICITY are from the MOA collection[9]. In order to measure performance, we used F-measure of the smallest class

8. `www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`
9. `moa.cms.waikato.ac.nz/datasets/`

Dataset	Dimension	Examples	$\lvert + \rvert$	$\lvert - \rvert$
A9A*	123	48842	11687	37155
AIRLINES	7	539383	240264	299119
COD-RNA*	8	488565	162855	325710
COVERTYPE	54	581012	283301	297711
ELECTRICITY	8	45312	26075	19237

Table 1: Datasets used for benchmarking.

for the unbalanced datasets (marked with a star in Table 1) and accuracy for the remaining datasets. Even if the datasets are not particularly large, the plots show that trees generated by our algorithm compare favourably with respect to the baselines, especially in the first learning phases.

8. Selective Sampling Experiments

The validation protocol for the experiments with active strategies is described in Algorithm 3, where we used a different query strategy (invoked on line 10 of Algorithm 3) for each considered approach. The labeled instances are stored and used to update the model. The query rate is upper bounded by an input budget parameter $B \in [0,1]$. In these experiments, we calculated the query rate as the fraction of instances for which a label was requested among the ones observed so far (Zliobaite et al., 2014). We compared our Confidence Tree Strategy (Algorithm 1), which we call `ConfTree`, against a five baseline techniques proposed by Zliobaite et al. (2014):

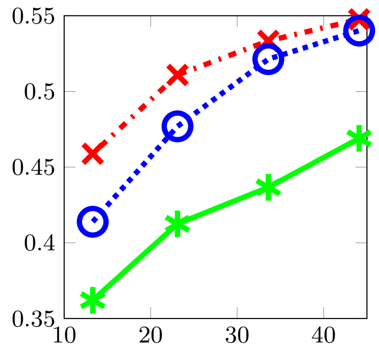
`Rnd.` The Random Strategy (Algorithm 4) is a naive method that queries the labels of incoming instances with probability equal to the query rate budget B without considering the actual incoming instance $\boldsymbol{x}_t$.

Algorithm 4 Random Strategy

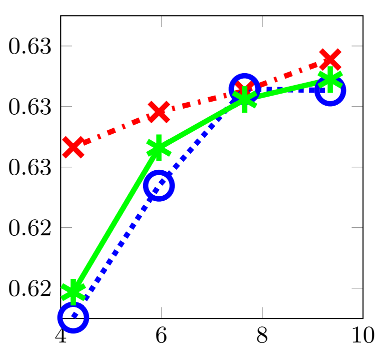
```
Input: labeling budget $B$
1: return true with probability $B$
2: else return false
```

`VarUn.` The Variable Uncertainty Strategy (Algorithm 5) uses a dynamically adjusted confidence threshold parameter Θ to determine requests for new labels. If the confidence of the leaf classification, measured by $\max\{p_{\ell_t}, 1 - p_{\ell_t}\}$, is above the current threshold Θ, then the threshold is increased by a fraction s in order to query only the most uncertain instances. In the opposite case, the threshold is reduced with the goal of acquiring more labels in regions where the estimator is less confident. As explained by Zliobaite et al. (2014), the parameter s can be safely set to a default value 0.01. We performed all the experiments with this setting.

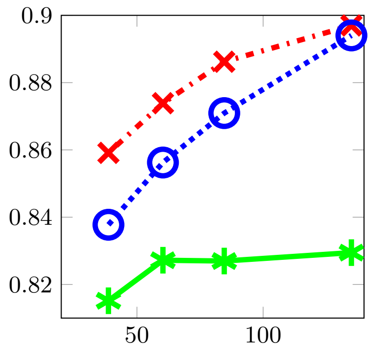
`RndVar.` This method is essentially the same as `VarUn` described above. `VarUn` always labels the instances that are close to the decision boundary. However, if the concept drifts (see, e.g., Widmer & Kubat, 1996) in areas away from the current decision boundary, the change



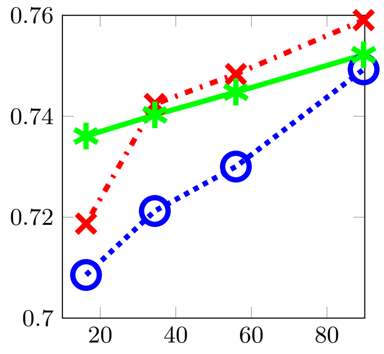
(a) A9A



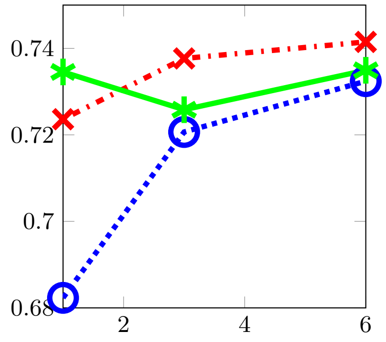
(b) AIRLINES



(c) COD-RNA



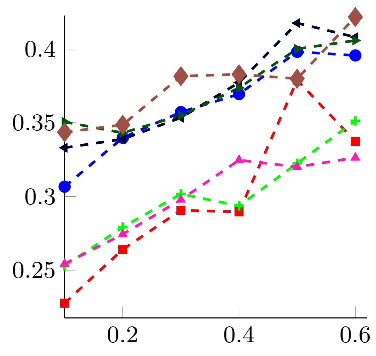
(d) COVERTYPE



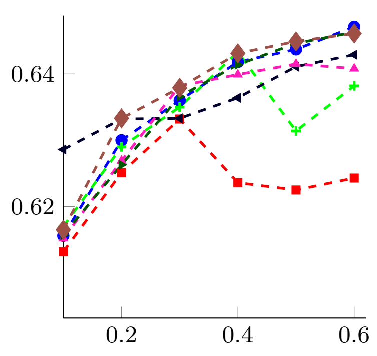
(e) ELECTRICITY

Figure 4: Online accuracy (F-measure for the unbalanced dataset) against number of leaves achieved by C-Tree (cross with red line), H-Tree (circle with blue line) and CorrH-Tree (star with green line).

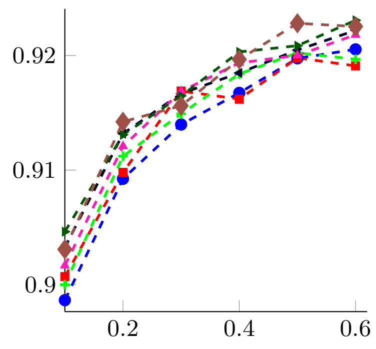
is missed unless the classifier issues extra queries on instances occurring in those areas. This technique addresses the issue by randomizing the query threshold through a rescaling with a normally distributed random variable. By doing that, the instances that are close to the decision boundary are labeled more often, but occasionally also some distant instances get queried —see the work by Zliobaite et al. (2014) for more details.



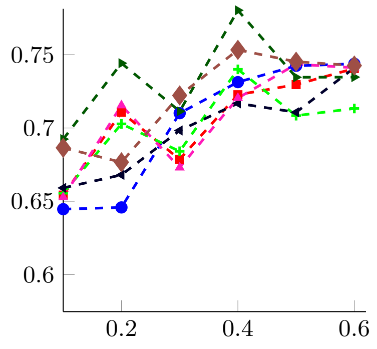
(a) A9A



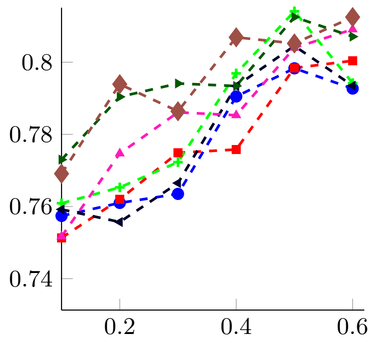
(b) AIRLINES



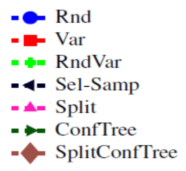
(c) COD-RNA



(d) COVERTYPE



(e) ELECTRICITY



(f) LEGEND

Figure 5: Online accuracy (F-measure for the unbalanced datasets) against label budget.

`Sel-Samp`. The Selective Sampling method is based on the work by Cesa-Bianchi, Gentile, and Zaniboni (2006), and uses a variable query threshold $\frac{B}{B+\left|\widehat{Y}_{\ell_t,t}-\frac{1}{2}\right|}$ similar to our random sampling mechanism for δ-consistent leaves.

`Split`. Many adaptive learning methods cope with concept drift using change-detection mechanisms. Change detectors (see, e.g., DDM in Gama, Medas, Castillo, & Rodrigues, 2004) are built with the implicit assumption that prediction errors are distributed uniformly in the stream unless the concept has drifted. However, the uniform distribution assumption may not hold over the substream of labels queried by an active learning method. Hence,

Algorithm 5 Variable Uncertainty Strategy

```
Input: threshold adjustment step $s \in (0, 1]$,
    confidence threshold $\Theta = 1$
1: Route instance $\boldsymbol{x}_t$ through $T$ until a leaf $\ell_t$ is reached
2: if $\max\{p_{\ell_t}, 1 - p_{\ell_t}\} < \Theta$ then
3:     decrease the confidence threshold $\Theta = (1 - s)\Theta$
4:     return true
5: else
6:     increase the confidence threshold $\Theta = (1 + s)\Theta$
7:     return false
8: end if
```

change detectors may have problems to distinguish a change in distribution due to active labeling from a change in distribution due to concept drift. To overcome this problem, the Split Strategy (Algorithm 6) randomly splits the stream. The first substream is labeled according to the Variable Uncertainty Strategy, while the second substream is labeled according to the Random Strategy. Both substreams are used to train the classifier, but only the random stream is used for detecting a change. In the experiments we set the parameter $\nu = 0.2$.

Algorithm 6 Split Strategy

```
Input: threshold adjustment step $s \in (0, 1]$
1: Route instance $\boldsymbol{x}_t$ through $T$ until a leaf is reached
2: Generate a uniform random variable rand $\in [0, 1]$;
3: if rand $\le \nu$ then
4:     ChangeDetectionMethod
5:     return Rnd
6: else
7:     return VarUn
8: end if
```

We also combined the Split Strategy (Algorithm 6) with our approach substituting the `VarUn` procedure with `ConfTree` method, we denote this approach by `SplitConfTree`.

All of our experiments were performed using the MOA data stream software suite. We added change detection to the base classifier in order to improve its performance (we chose DDM as suggested by Zliobaite et al., 2014). All the tested ACTIVE strategies used C-Tree (Algorithm 1) as the base learner with the same parameters setting as in Section 7. The algorithms had to predict the label of each new incoming instance. After each prediction, if the active learning system requested the true label, the instance —together with its label— was used as a new training example, see Algorithm 3. We ran all the competing algorithms using range $B \in \{.1, .2, .3, .4, .5, .6\}$[10] of budget values, and plotted in Figure 5 the resulting

10. Note that the budget is only an upper limit to the actual query rate: algorithms generally ask for a smaller number of annotations.

online accuracy as a function of the query budget. As for the FULL sampling experiments, we deactivated the tie-break mechanism and we tuned the tree learning parameters.

Although performances are oscillating, all the plots show that accuracies of all methods tend to increase as the budget grows. The plots also show how the performance of our approaches (`ConfTree` and `SplitConfTree`) is consistently good across the various dataset scenarios and with respect to the baseline methods. This suggests that including more accurate statistical information in order to decide whether to issue a query, as our methods do, may indeed improve performance. By comparing the plots of Figure 4 and Figure 5, we can also observe that a small fraction of the available labels is typically sufficient to achieve a performance close to that of the full sampling algorithms.

9. Conclusions and Future Works

The goal of this work was to provide a more rigorous statistical analysis of confidence intervals for splitting leaves in decision trees. Our confidence bounds take into account all the relevant variables of the problem. This improved analysis is reflected in the predictive ability of the learned decision trees, as we show in the experiments. It is important to note that the proposed bounds can be easily applied to the many existing variants of VFDT. Furthermore, we showed how these confidence bounds can lead to effective selective sampling techniques for saving labels in online decision tree learning. Note that both full and selective sampling settings are supported by our theoretical and empirical results.

Our confidence analysis applies to i.i.d. streams. Extending our results to more general processes remains an open problem. The selective sampling results raise interesting issues concerning the interplay between nonparametric learning models (such as decision trees) and subsampling techniques. For example, there are no theoretical bounds showing the extent to which labels can be saved without significantly hurting performance.

Acknowledgments

The authors would like to thank the editor and the anonymous reviewers for their careful reading and thoughtful comments.

Appendix A. Proof of Theorem 1

Let H be the standard (unscaled) entropy. Using the standard identity $H\big(Y_{|i} \mid F\big) = H\big(Y_{|i}, F\big) - H(F)$, we have $\widehat{\Phi}_{i|F} = \widehat{H}_{1/2}\big(Y_{|i}, F\big) - \widehat{H}_{1/2}(F)$. We now use part (iii) of the remark following (Antos & Kontoyiannis, 2001, Corollary 1), we have that

$$\left|\widehat{H}_{1/2}(F) - \mathbb{E}\,\widehat{H}_{1/2}(F)\right| \le \frac{\ln m}{2}\sqrt{\frac{2}{m}\ln\frac{4}{\delta}}$$

$$\left|\widehat{H}_{1/2}(Y_{|i}, F) - \mathbb{E}\,\widehat{H}_{1/2}(Y_{|i}, F)\right| \le \frac{\ln m}{2}\sqrt{\frac{2}{m}\ln\frac{4}{\delta}}$$

simultaneously hold with probability at least $1 - \delta$. These bounds hold irrespective to the size of the sets in which $Y_{|i}$ and F take their values.

Next, we apply (Paninski, 2003, Proposition 1), which states that

$$-\ln\left(1+\frac{N-1}{m}\right) \le \mathbb{E}\,\widehat{H}(Z) - H(Z) \le 0$$

for any random variable Z which takes N distinct values. In our case, $N = 2$ for $Z = F$ and $N = 4$ for $Z = \bigl(Y_{|i}, F\bigr)$. Hence, using $-a \le -\ln(1+a)$ for all a, we get

$$\Bigl|H_{1/2}(F) - \mathbb{E}\,\widehat{H}_{1/2}(F)\Bigr| \le \frac{1}{2m} \qquad \text{and} \qquad \Bigl|H_{1/2}(Y_{|i}, F) - \mathbb{E}\,\widehat{H}_{1/2}(Y_{|i}, F)\Bigr| \le \frac{3}{2m} \ .$$

Putting everything together gives the desired result.

Appendix B. Proof of Theorem 2

Lemma 2 (McDiarmid's inequality) *Let G be a real function of m independent random variables $X_1, \dots, X_m$ such that*

$$\Bigl|G(x_1, \dots, x_i, \dots, x_m) - G(x_1, \dots, x_i', \dots, x_m)\Bigr| \le c \tag{8}$$

for some constant $c \in \mathbb{R}$ and for all realizations $x_1, \dots, x_i, x_i', \dots, x_m$. Then

$$\mathbb{P}\bigl(\bigl|G - \mathbb{E}\,G\bigr| \ge \epsilon\bigr) \le 2\exp\left(\frac{-2\epsilon^2}{m\,c^2}\right) \ .$$

If we set the right-hand side equal to δ, then

$$\bigl|G - \mathbb{E}\,G\bigr| \le c\sqrt{\frac{m}{2}\ln\frac{2}{\delta}}$$

is true with probability at least $1-\delta$.

Note the following fact

$$\begin{aligned}
J(Y_{|i} \mid F) &= \mathbb{P}(F=1)\,2\,\frac{\mathbb{P}(Y_{|i}=1, F=1)}{\mathbb{P}(F=1)}\,\frac{\mathbb{P}(Y_{|i}=0, F=1)}{\mathbb{P}(F=1)} \\
&\quad + \mathbb{P}(F=0)\,2\,\frac{\mathbb{P}(Y_{|i}=1, F=0)}{\mathbb{P}(F=0)}\,\frac{\mathbb{P}(Y_{|i}=0, F=0)}{\mathbb{P}(F=0)} \\
&= 2\,\frac{\mathbb{P}(Y_{|i}=1, F=1)\,\mathbb{P}(Y_{|i}=0, F=1)}{\mathbb{P}(F=1)} \\
&\quad + 2\,\frac{\mathbb{P}(Y_{|i}=1, F=0)\,\mathbb{P}(Y_{|i}=0, F=0)}{\mathbb{P}(F=0)} \\
&= \mathrm{HM}(p_1, q_1) + \mathrm{HM}(p_0, q_0) \ .
\end{aligned} \tag{9}$$

In view of applying McDiarmid inequality, let $\widehat{p}_k = \frac{r}{m}$ and $\widehat{q}_k = \frac{s}{m}$. We can write the left-hand side of condition (8) in Lemma 2 for each term of (9) as

$$\frac{2}{m}\left|\frac{rs}{r+s} - \frac{r's'}{r'+s'}\right|$$

where $r, s = 1, \dots, m$ and r', s' may take the following forms: $(r+1, s-1)$ —when a label of an example in the current leaf is flipped, $(r+1, s)$ —when an example is moved from the sibling leaf to the current leaf, and $(r-1, s)$ —when an example is moved from the current leaf to the sibling leaf. Since the harmonic mean is symmetric in r and s, we can ignore the cases $(r-1, s+1)$, $(r, s+1)$, and $(r, s-1)$. A tedious but simple calculation shows that

$$\left| \frac{rs}{r+s} - \frac{r's'}{r'+s'} \right| \le 1 \ .$$

Therefore, we may apply Lemma 2 with $c = \frac{4}{m}$ and obtain that

$$\left| \widehat{\Phi}_{i,F} - \mathbb{E}\, \widehat{\Phi}_{i,F} \right| \le \sqrt{\frac{8}{m} \ln \frac{2}{\delta}} \tag{10}$$

holds with probability at least $1 - \delta$.

Next, we control the bias of $\textsc{hm}\big(\widehat{p}_k, \widehat{q}_k\big)$ as follows,

$$\begin{aligned}
0 &\le \textsc{hm}(p_k, q_k) - \mathbb{E}\Big[\textsc{hm}\big(\widehat{p}_k, \widehat{q}_k\big)\Big] \\
&= 2\frac{p_k q_K}{p_k + q_k} - 2\mathbb{E}\left[\frac{\widehat{p}_k \widehat{q}_k}{\widehat{p}_k + \widehat{q}_k}\right] && (11) \\
&= 2\mathbb{E}\left[\frac{p_k \widehat{p}_k (q_k - \widehat{q}_k) + q_k \widehat{q}_k (p_k - \widehat{p}_k)}{(p_k + q_k)\big(\widehat{p}_k + \widehat{q}_k\big)}\right] \\
&\le 2\mathbb{E}\left|q_k - \widehat{q}_k\right| + 2\mathbb{E}\big|p_k - \widehat{p}_k\big| \\
&\le 2\sqrt{\mathbb{E}\left[\big(q_k - \widehat{q}_k\big)^2\right]} + 2\sqrt{\mathbb{E}\left[\big(p_k - \widehat{p}_k\big)^2\right]} \\
&\le \frac{2}{\sqrt{m}} && (12)
\end{aligned}$$

where the first inequality is due to the concavity of $\textsc{hm}$. Combining (10) and (12) concludes the proof.

Appendix C. Proof of Theorem 3

Lemma 3 *Let B binomially distributed with parameters (m, p). Then*

$$\left| \sqrt{\frac{B}{m}} - \sqrt{p} \right| \le \sqrt{\frac{1}{m} \ln \frac{2}{\delta}}$$

is true with probability at least $1 - \delta$.

Proof 6 *The result is an immediate consequence of the Okamoto bounds (Okamoto, 1959) —see also (Boucheron, Lugosi, & Massart, 2013, Exercise 2.13). In particular,*

$$\mathbb{P}\left(\left| \frac{B}{m} - p \right| \ge \varepsilon \right) \le e^{-mD(p+\varepsilon \| p)} + e^{-mD(p-\varepsilon \| p)}$$

where $D(q\|p) = q \ln \frac{q}{p} + (1-q)\ln\frac{1-q}{1-p}$ *is the KL divergence, and*

$$D(p+\varepsilon\|p) \ge \left(\sqrt{p+\varepsilon} - \sqrt{p}\right)^2 \qquad \text{and} \qquad D(p-\varepsilon\|p) \ge 2\left(\sqrt{p-\varepsilon} - \sqrt{p}\right)^2 \ .$$

Simple algrebraic manipulation concludes the proof.

Similarly to the proof of Theorem 2, note that

$$\begin{aligned}
&Q(Y_{|i} \mid F) \\
&\quad = \mathbb{P}(F=1)\sqrt{\frac{\mathbb{P}(Y_{|i}=1, F=1)}{\mathbb{P}(F=1)}\,\frac{\mathbb{P}(Y_{|i}=0, F=1)}{\mathbb{P}(F=1)}} \\
&\qquad\quad + \mathbb{P}(F=0)\sqrt{\frac{\mathbb{P}(Y_{|i}=1, F=0)}{\mathbb{P}(F=0)}\,\frac{\mathbb{P}(Y_{|i}=0, F=0)}{\mathbb{P}(F=0)}} \\
&\quad = \sqrt{\mathbb{P}(Y_{|i}=1, F=1)\,\mathbb{P}(Y_{|i}=0, F=1)} + \sqrt{\mathbb{P}(Y_{|i}=1, F=0)\,\mathbb{P}(Y_{|i}=0, F=0)} \\
&\quad = \sqrt{p_1 q_1} + \sqrt{p_0 q_0}
\end{aligned}$$

Then

$$\left|\sqrt{\widehat{p}_1\widehat{q}_1} + \sqrt{\widehat{p}_0\widehat{q}_0} - \sqrt{p_1 q_1} - \sqrt{p_0 q_0}\right| \le \left|\sqrt{\widehat{p}_1\widehat{q}_1} - \sqrt{p_1 q_1}\right| + \left|\sqrt{\widehat{p}_0\widehat{q}_0} - \sqrt{p_0 q_0}\right| \le 4\varepsilon$$

whenever $\left|\sqrt{\widehat{p}_k} - \sqrt{p_k}\right| \le \varepsilon$ and $\left|\sqrt{\widehat{q}_k} - \sqrt{q_k}\right| \le \varepsilon$ for $k \in \{0,1\}$. Using the union bound and Lemma 3 we immediately get

$$\left|\sqrt{\widehat{p}_1\widehat{q}_1} + \sqrt{\widehat{p}_0\widehat{q}_0} - \sqrt{p_1 q_1} - \sqrt{p_0 q_0}\right| \le 4\sqrt{\frac{1}{m}\ln\frac{8}{\delta}}$$

thus concluding the proof.

Appendix D. Recursive Algorithm for Generating Random Binary Trees

The random binary trees are constructed through recursive random splits according to Algorithm 7. More precisely, we start at the root with a budget of n leaves. Then we assign to the left and right sub-trees $\lfloor nX \rfloor$ and $n - 1 - \lfloor nX \rfloor$ leaves respectively, where X is uniformly distributed in the unit interval. This splitting continues with i.i.d. draws of X until the left and right sub-trees are left with one leaf each. Whenever a split is generated, we assign it a uniformly random attribute and a random threshold value. In the experiment, we generated 1000 random binary trees with $n = 50$ leaves. The random splits are performed choosing among $d = 5$ attributes. For simplicity, we only considered numerical attributes and thresholds in the $[0,1]$ interval. A random binary tree is then used to generate a stream as follows: for each leaf of the tree, 10,000 examples are uniformly drawn from the subregion of $[0,1]^5$ defined by the leaf, obtaining 500,000 examples. Each of these examples is given label 1 with probability 0.7 for a left leaf and with probability 0.3 for a right leaf.

Algorithm 7 RandCBT

```
Input: tree $T$, total number of leaves num-leaves, number of attributes $d$, leaf class conditional probability $q$
Output: complete binary tree $T$
    current-node $i$ = CreateNode()
 2: if num-leaves == 1 then
      mark $i$ as leaf
 4:   if $i$ is a left child then
        $\mathbb{P}(Y = 1 \mid \boldsymbol{X} \to i) = q$
 6:   else
        $\mathbb{P}(Y = 1 \mid \boldsymbol{X} \to i) = 1 - q$
 8:   end if
    else
10:   Generate a uniform random variable rand $\in [0, 1]$;
      left-leaves = $\max\{1, \lfloor$num-leaves*rand$\rfloor\}$
12:   right-leaves = num-leaves − left-leaves
      $i$ = RandomAttribute$(1, \ldots, d)$
14:   $v$ = RandomValueInSubRegion$(i)$
      add split test $(i, v)$ to $i$
16:   l-child = RandCBT($i$, left-leaves, $d$, $q$)
      r-child = RandCBT($i$, right-leaves, $d$, $q$)
18:   add l-child and r-child as a descendent of $i$
    end if
20: return current-node $i$
```

References

Antos, A., & Kontoyiannis, I. (2001). Convergence properties of functional estimates for discrete distributions. *Random Structures & Algorithms*, *19*(3–4), 163–193.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 139–148, New York, NY, USA. ACM Press.

Boucheron, S., Lugosi, G., & Massart, P. (2013). *Concentration Inequalities.* Oxford University Press.

Cesa-Bianchi, N., Gentile, C., & Zaniboni, L. (2006). Worst-case analysis of selective sampling for linear classification. *The Journal of Machine Learning Research*, *7*, 1205–1230.

Dietterich, T., Kearns, M., & Mansour, Y. (1996). Applying the weak learning framework to understand and improve C4.5. In *International Conference on Machine Learning*, pp. 96–104. Morgan Kaufmann.

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data*

Mining, pp. 71–80. ACM Press.

Duda, P., Jaworski, M., Pietruczuk, L., & Rutkowski, L. (2014). A novel application of Hoeffding's inequality to decision trees construction for data streams. In *International Joint Conference on Neural Networks*. IEEE Press.

Gama, J. a., Medas, P., & Rocha, R. (2004). Forest trees for on-line data. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, pp. 632–636, New York, NY, USA. ACM Press.

Gama, J. a., Rocha, R., & Medas, P. (2003). Accurate decision trees for mining high-speed data streams. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge discovery and data mining*, pp. 523–528. ACM Press.

Gama, J., Kosina, P., et al. (2011). Learning decision rules from data streams. In *Proceedings of the International Joint Conference on Artificial Intelligence*, Vol. 22, p. 1255.

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Advances in Artificial Intelligence – SBIA 2004*, pp. 286–295. Springer.

Gratch, J. (1995). Sequential inductive learning. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 779–786. AAAI Press.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, *58*(301), 13–30.

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 97–106. ACM Press.

Ikonomovska, E., Gama, J., & Džeroski, S. (2011). Learning model trees from evolving data streams. *Data mining and knowledge discovery*, *23*(1), 128–168.

Jin, R., & Agrawal, G. (2003). Efficient decision tree construction on streaming data. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 571–576. ACM Press.

Kearns, M., & Mansour, Y. (1996). On the boosting ability of top-down decision tree learning algorithms. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pp. 459–468. ACM Press.

Kirkby, R., Bouckaert, R. R., Studen, M., Lin, C.-S. A., Kibriya, A. M., Frank, E., Mayo, M., Mayo, M., Mutter, S., Pfahringer, B., et al. (2007). Improving Hoeffding trees. *Int. J. Approx. Reasoning*, *45*, 39–48.

Kosina, P., & Gama, J. (2012). Very fast decision rules for multi-class problems. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pp. 795–800. ACM Press.

Liu, J., Li, X., & Zhong, W. (2009). Ambiguous decision trees for mining concept-drifting data streams. *Pattern Recognition Letters*, *30*(15).

Matuszyk, P., Krempl, G., & Spiliopoulou, M. (2013). Correcting the usage of the Hoeffding inequality in stream mining. In *Advances in Intelligent Data Analysis XII*, pp. 298–309. Springer.

McDiarmid, C. (1989). On the method of bounded differences. *Surveys in combinatorics*, *141*(1), 148–188.

Musick, R., Catlett, J., & Russell, S. J. (1993). Decision theoretic subsampling for induction on large databases.. In *International Conference on Machine Learning*, pp. 212–219. Morgan Kaufmann.

Okamoto, M. (1959). Some inequalities relating to the partial sum of binomial probabilities. *Annals of the Institute of Statistical Mathematics*, *10*(1), 29–35.

Paninski, L. (2003). Estimation of entropy and mutual information. *Neural Computation*, *15*(6), 1191–1253.

Pfahringer, B., Holmes, G., & Kirkby, R. (2007). New options for Hoeffding trees. In *Proceedings of the 20th Australian joint conference on Advances in Artificial Intelligence*, pp. 90–99. Springer.

Rutkowski, L., Jaworski, M., Pietruczuk, L., & Duda, P. (2015). A new method for data stream mining based on the misclassification error. *IEEE Transactions on Neural Networks and Learning Systems*, *26*(5), 1048–1059.

Rutkowski, L., Pietruczuk, L., Duda, P., & Jaworski, M. (2013). Decision trees for mining data streams based on the McDiarmid's bound. *IEEE Transactions on Knowledge and Data Engineering*, *25*(6), 1272–1279.

Salperwyck, C., & Lemaire, V. (2013). Incremental decision tree based on order statistics. In *International Joint Conference on Neural Networks*, pp. 1–8. IEEE Press.

Settles, B. (2012). Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *6*(1), 1–114.

Takimoto, E., & Maruoka, A. (2003). Top-down decision tree learning as information based boosting. *Theoretical Computer Science*, *292*(2), 447–464.

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

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning*, *23*(1), 69–101.

Xu, W., Qin, Z., Hu, H., & Zhao, N. (2011). Mining uncertain data streams using clustering feature decision trees. In *Advanced Data Mining and Applications*, pp. 195–208. Springer.

Yang, H., & Fong, S. (2012). Incrementally optimized decision tree for noisy big data. In *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pp. 36–44, New York, NY, USA. ACM Press.

Zliobaite, I., Bifet, A., Pfahringer, B., & Holmes, G. (2014). Active learning with drifting streaming data. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(1), 27–39.