A New Method for Data Stream Mining Based on the Misclassification Error

Leszek Rutkowski, *Fellow, IEEE*, Maciej Jaworski, Lena Pietruczuk, and Piotr Duda

***Abstract*— In this paper, a new method for constructing decision trees for stream data is proposed. First a new splitting criterion based on the misclassification error is derived. A theorem is proven showing that the best attribute computed in considered node according to the available data sample is the same, with some high probability, as the attribute derived from the whole infinite data stream. Next this result is combined with the splitting criterion based on the Gini index. It is shown that such combination provides the highest accuracy among all studied algorithms.**

***Index Terms*— Classification, data stream, decision trees, impurity measure, splitting criterion.**

I. Introduction

A. Motivation and Results

In recent years, the amount of data that needs to be analyzed is growing very fast. Potentially unlimited number of data is the cause of creation of a new field of research called data stream mining. This area is developing very fast and attracting a great number of authors who propose a variety of methods and algorithms [1]–[17]. By analyzing stream of data elements, one has to face new difficulties, therefore standard approach to the problem of data mining cannot be applied. One of the difficulties is the need of on-the-spot data analysis. This results from the fact that all data cannot be stored due to the limitation of memory. The second problem is the rate of incoming data. The algorithm used for data stream analysis should be fast to keep up with the incoming data. The next problem is the occurrence of concept drift [18]–[24]. Concept drift is the term that describes the change in data distribution or the meaning of data, e.g., previously relevant data in time can become irrelevant or unneeded. Therefore, the algorithm applied to this problem should evolve in time according to changing circumstances.

Manuscript received June 14, 2013; revised February 19, 2014 and June 16, 2014; accepted June 16, 2014. Date of publication July 16, 2014; date of current version April 15, 2015. This work was supported by the Foundation for Polish Science Team Programme through the European Union, European Regional Development Fund, Operational Programme Innovative Economy 2007-2013.

L. Rutkowski is with the Institute of Computational Intelligence, Czestochowa University of Technology, Czestochowa 42-200, Poland, and also with the Information Technology Institute, Academy of Social Sciences, Łódź 90-113, Poland (e-mail: leszek.rutkowski@iisi.pcz.pl).

M. Jaworski, L. Pietruczuk, and P. Duda are with the Institute of Computational Intelligence, Czestochowa University of Technology, Czestochowa 42-200, Poland, (e-mail: maciej.jaworski@iisi.pcz.pl; lena.pietruczuk@iisi.pcz.pl; piotr.duda@iisi.pcz.pl).

Digital Object Identifier 10.1109/TNNLS.2014.2333557

One of the most important techniques used in data mining is the data classification. Let us assume that the data elements are described by D attributes. Each attribute can be either nominal or numerical. If the ith attribute is nominal then the number of possible values is finite and equal to v_i. Moreover, to each data element, a class is assigned. The number of different classes is denoted by K. The aim of the classification is to construct a function called the classifier, based on the training data set of elements. The classifier maps the set of values of attributes into the set of classes. It is further used to classify unlabeled data elements. There exists a wide variety of methods used for data classification. The most popular are neural networks [25], k-nearest neighbors [26] and decision trees [27]–[29]. The last one is the main subject of this paper. Decision tree consists of nodes and leaves. Each node is split according to some attribute into its children (nodes or leaves). Each child corresponds to one value of the attribute (in case of nonbinary tree) or to the subset of possible values (if the tree is binary). Each split is associated with the partition of the training data set into its subsets. Splits should be made in such a way that the resulting subsets (partition) are as highly pure as possible. At the beginning of the decision tree construction, it is quite likely that the partition will be impure (e.g., it may contain a collection of data elements from different classes rather than from a single class) [30]. The degree of purity of the partition can be measured quantitatively, e.g., by the information entropy [31] or the Gini index [32]. In the high levels of the tree, the distribution of classes in corresponding data subsets is highly dominated by elements of only one class. In this case, the node does not need to be split and becomes a leaf. Leaves are used to assign a class to unlabeled data elements.

In the decision tree construction process, the main task is to determine, which attribute is the best to split the considered node. Let S denotes the set of data elements in this node, $n(S)$ is the cardinality of S and let $n^k(S)$ be the number of elements of set S from the kth class. The choice of the attribute to split the considered node proceeds as follows. For each attribute indexed by i, $i = 1, \ldots, D$, the data set S is divided into some disjoint Q subsets S_q^i, $q = 1, \ldots, Q$, where $Q = 2$ for binary trees and $Q = v_i$ for nonbinary trees. The quality of the split is evaluated according to a split measure function. In the majority of decision tree algorithms, this function is defined as the reduction of some impurity measure (or equivalently the gain of some information measure). Let as denote the impurity measure of set S by $g(S)$. Then, the split measure function can be

2162-237X © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

expressed as

$$\Delta g_i(S) = g(S) - \sum_{q=1}^{Q} \frac{n(S_q^i)}{n(S)} g(S_q^i). \quad (1)$$

The attribute that provides the highest value of the split measure function is used to split the considered node.

Any impurity measure $g(S)$ has to satisfy at least two of the following conditions.

(i) If the set S is dominated by elements of one class, i.e., $\exists_{k_0 \in \{1,\ldots,K\}} n^{k_0}(S) = n(S)$, then $g(S) = 0$ (the set S is maximally pure).

(ii) The function $g(S)$ is maximized if each class is represented by the same number of elements, i.e., $\forall_{k \in \{1,\ldots,K\}} n^k(S) = n(S)/K$ (the set S is maximally mixed).

For convenience, one can define the fraction of elements from the kth class in set S as a ratio between $n^k(S)$ and $n(S)$

$$p_k(S) = \frac{n^k(S)}{n(S)}. \quad (2)$$

In the literature, several forms of impurity measures g have been proposed, e.g., information entropy (in the ID3 algorithm [31]) or Gini index (in the CART algorithm [32]). Using the notation of fractions p_k, the information entropy is given by

$$g(S) = -\sum_{k=1}^{K} p_k(S) \log_2 p_k(S) \quad (3)$$

whereas the Gini index is expressed by the following:

$$g(S) = 1 - \sum_{k=1}^{K} (p_k(S))^2. \quad (4)$$

There exists another impurity measure, which is rarely mentioned in literature. It is called the misclassification error and is given in a very simple form

$$g(S) = 1 - \max_{k \in \{1,\ldots,K\}} \{p^k(S)\}. \quad (5)$$

It is easy to check that (3), (4), and (5) satisfy conditions (i) and (ii).

The challenge and research problem in stream data mining is to show that, with high probability, the attribute chosen using $n(S)$ samples and one of (3)–(5) impurity measures is the same as that chosen using infinite examples. In the last decade, the commonly used tool to tackle this problem was the Hoeffding bound [33]. In [34] and [35], we have shown that the Hoeffding bound is a wrong tool and we have proved several theorems showing a correct solution for impurity measures (3) and (4). To our best knowledge, impurity measure (5) has never been applied for stream data mining. Recently, we have observed that using impurity measure (5) is very beneficial in stream data mining at the beginning (lower level of the tree) of decision tree construction, and next, at higher levels, impurity measures (3) or (4) perform better. This observation inspired us to develop a hybrid algorithm presented in this paper. To implement this idea, we should derive a splitting criterion for impurity measure (5) and next to combine it with splitting criteria based on (3) or (4).

The main contribution and novelty of this paper can be summarized as follows.

1) A new splitting criterion based on the misclassification error (5) is proposed in Theorem 1. The criterion has strong mathematical foundations and allows determining the optimal attribute to split the node of the tree. The criterion guarantees, with high probability set by the user, that the attribute chosen on a basis of a finite number of data elements is the same as if it were for the case of infinite data set (e.g., data stream). The online tree construction procedure with the application of the proposed criterion is denoted further in the text as the mDT algorithm. The performance of the mDT algorithm was compared with the online tree for which the splitting criterion is based on the Gini index (the gDT algorithm). The justification for the gDT algorithm was formerly presented in [36]. It should be noted that in [36], the gDT algorithm was introduced under the name dsCART.
2) Combining advantages of both misclassification error (5) and Gini index (4), the new splitting criterion is proposed. The algorithm based on this hybrid criterion is called the hDT algorithm. The new algorithm provided the highest accuracy among all considered algorithms.
3) The experimental results are performed on 12 synthetic data concepts and two well known in literature real data sets. The mDT algorithm is compared with the static decision tree based on the misclassification error impurity measure (the smDT algorithm). Moreover, the performance of the mDT, gDT, and hDT algorithms is precisely examined.

It should be emphasized that our main result (Theorem 1 in Section II) is applicable to solve problems with concept drift. More specifically, our result should replace Hoeffding's bound used incorrectly in algorithms like the concept-adapting very fast decision trees (CVFDT) [19] or the Hoeffding option trees (HOT) [33], [37]. We stress that the idea of the CVFDT and HOT algorithms is correct, however, authors of both algorithms incorrectly used Hoeffding's bound in their papers. Our result can be combined with those algorithms, replacing Hoeffding's bound by appropriate splitting criteria presented in this paper. Moreover, decision trees based on the proposed splitting criteria can be used as classifiers in already known ensemble algorithms [2].

B. Background of the Problem

The ID3 and the CART algorithms mentioned previously are designed for static data, in which the training data set is of a fixed size. The same training data set (or its appropriate subsets) is used to determine splitting attribute in all nodes in the tree. This approach is not applicable to large data sets or data streams, for which special one-pass algorithms are needed. In particular, the choice of the splitting attribute in different nodes has to be made on the basis of different training data sets. However, in the case of large data sets (or data

streams without concept drift), these different training sets still belong to the same probability distribution. The problem is to decide if the number of data elements $n(S)$, collected so far in considered node, is enough to choose the attribute with the highest value of split measure function as the best attribute, i.e., as the splitting attribute. In general, the more data elements are collected, the more reliable the result is. The most trivial solution, taken directly from the static data set algorithms, is to fix some threshold number n_0.

A much better solution to the problem was proposed by the authors of the very fast decision tree (VFDT) algorithm in [33]. They claimed that in the ideal case the choice of the splitting attribute in a node should be made on basis of infinite set S^∞ of data examples ($n(S) = n(S^\infty) = \infty$). This means that for every attribute the split measure function (1) should be calculated as $\Delta g_i(S^\infty)$, where $i = 1, \ldots, D$ is the index of the attribute. Obviously it is not possible. Therefore, the number $n(S)$ should be large enough to say that

$$\arg\max_i\{\Delta g_i(S)\} = \arg\max_i\{\Delta g_i(S^\infty)\} \tag{6}$$

with high probability $1-\delta^* = (1-\delta)^{D-1}$ set by the user. Let x denotes the index of the attribute with the highest value of split measure function, and y with the second highest value. If the following inequality is satisfied:

$$\Delta g_x(S) - \Delta g_y(S) > \epsilon(n(S), \delta) \tag{7}$$

then $x = \arg\max_i\{\Delta g_i(S^\infty)\}$ with probability $1-\delta^*$ and the split is made with respect to the xth attribute. In [33], the function $\epsilon(n(S), \delta)$ was derived using the Hoeffding bound, obtaining the following general form:

$$\epsilon(n(S), \delta) = R\sqrt{\frac{\ln(1/\delta)}{2n(S)}} \tag{8}$$

where R is a constant. Therefore, if the difference between split measure function for xth and yth attributes is large, only a small amount of data $n(S)$ is needed to choose xth attribute as the splitting one. On the other hand, if the difference is small, a very large number $n(S)$ may be needed.

Although the VFDT algorithm based on the Hoeffding trees gives very satisfactory practical results, it was pointed out in [34] that it was wrongly mathematically justified. The Hoeffding bound is applicable only for the sum or average of random variables, whereas the information gain or Gini gain are nonlinear functions, which cannot be expressed in such a simple form. Rutkowski *et al.* [34] proposed to use the McDiarmid's bound instead of the Hoeffding's bound to derive the form of function $\epsilon(n(S), \delta)$. The McDiarmid's bound is applied to independent observations taken from data streams, however, unlike Hoeffding's bound, it is applicable for nonlinear functions. For Gini index, they obtained result similar to (8), with higher value of constant R. For information entropy, the result was much worse: the convergence of $\epsilon(n(S), \delta)$ to zero was in the form $O(\log_2 n(S)/\sqrt{n(S)})$.

Rutkowski *et al.* [35], [36] proposed another method of deriving the function $\epsilon(n(S), \delta)$ for information gain and Gini gain, respectively. In this method, the mentioned split measure functions were linearized with the application of Taylor's theorem and then the Gaussian approximation was used to obtain the final result. The convergence of $\epsilon(n(S), \delta)$ was, as in the case of the Hoeffding's bound, in the form $O(1/\sqrt{n(S)})$. Nevertheless, in case of information gain, the method was applicable only for two-class problem. For Gini gain, the proportionality coefficient was dependent on the number of classes K and reached large values.

The name of the Hoeffding tree algorithm proposed in [33], stems from the fact that authors tried to apply the Hoeffding's bound to derive a splitting criterion. However, as it was mentioned previously, the Hoeffding's bound is not applicable for nonlinear split measures. Nonetheless, the main idea of constructing such online trees [33] is reasonable if correct splitting criterion [34]–[36] replaces the Hoeffding's bound. To avoid any misunderstandings in this paper, the name online trees will denote the Hoeffding trees without any specified splitting criterion.

Summarizing, the forms of function $\epsilon(n(S), \delta)$ existing in literature either have wrong mathematical background or require impractically high number of $n(S)$ to reach relatively low value and allow to split the node quite fast. Both difficulties seem to arise from the nonlinearity of existing impurity measures. Therefore, searching for a new simple impurity measure could be a key to solve the problem and the challenge to be addressed in this paper.

The rest of this paper is organized as follows. In Section II, a new splitting criterion based on the misclassification error is proposed. The description of stream data mining algorithms is given in Section III. Section IV presents the results obtained in the simulations. In Section V, the conclusion of this paper is drawn.

II. New Splitting Criterion Based on the Misclassification Error

A. Main Result

The misclassification error impurity measure, according to (5), can be expressed as follows:

$$g(S) = 1 - \frac{\max_{k\in\{1,\ldots,K\}}\{n^k(S)\}}{n(S)}. \tag{9}$$

It is easy to check that (9) satisfies conditions (i) and (ii). The misclassification error is the basis of a split measure function, called further the accuracy gain.

Theorem 1: Let S be the finite set of data elements and let $g(S)$ denote the misclassification error of S. If, considering (1), the following condition is satisfied:

$$\Delta g_i(S) - \Delta g_j(S) > z_{(1-\delta)}\sqrt{\frac{1}{2n(S)}} \tag{10}$$

where $z_{(1-\delta)}$ is the $(1-\delta)$ quantile of the standard normal distribution $\mathcal{N}(0, 1)$, then with probability $(1-\delta)$ the ith attribute would give higher value of the accuracy gain than the jth attribute for the infinite data set S^∞ from the same probability distribution as S. Moreover, if the ith attribute is with the highest value of accuracy gain in set S and the jth attribute is the one with the second highest value, then with

probability at least $1-\delta^*$ the ith attribute would maximize the value of accuracy gain in set S^∞.

Proof: The set S is split into subsets $S_q^i,\ q=1,\ldots,Q$, according to some attribute i. Combining (1) with (9), one obtains the form of the accuracy gain function

$$\Delta g_i(S) = 1 - \frac{\max_{k\in\{1,\ldots,K\}}\left\{n^k(S)\right\}}{n(S)} - \sum_{q=1}^{Q}\frac{n(S_q^i)}{n(S)}\left(1-\frac{\max_{k\in\{1,\ldots,K\}}\left\{n^k(S_q^i)\right\}}{n\left(S_q^i\right)}\right) \quad (11)$$

which can be further simplified to the following form:

$$\Delta g_i(S) = \frac{\left(\sum_{q=1}^{Q}\max_{k\in\{1,\ldots,K\}}\left\{n^k(S_q^i)\right\}\right)}{n(S)} - \frac{\max_{k\in\{1,\ldots,K\}}\left\{n^k(S)\right\}}{n(S)}. \quad (12)$$

Note that the term $\max_{k\in\{1,\ldots,K\}}\{n^k(S)\}$ is the same for all attributes. Since only the differences between split measure function values are important, the mentioned term can be neglected. Let us denote

$$\Delta\tilde{g}_i(S) = \frac{\sum_{q=1}^{Q}\max_{k\in\{1,\ldots,K\}}\left\{n^k(S_q^i)\right\}}{n(S)}. \quad (13)$$

Note that from (12) and (13), the following obvious equation can be derived:

$$\Delta\tilde{g}_i(S) - \Delta\tilde{g}_j(S) = \Delta g_i(S) - \Delta g_j(S). \quad (14)$$

The term $\max_{k\in\{1,\ldots,K\}}\{n^k(S_q^i)\}$ denotes the number of elements reaching the qth leaf belonging to the majority class in this leaf. On the other hand, the simplest method of assigning a class to an unlabeled data element is to assign the majority class of the leaf. Therefore, the term $\max_{k\in\{1,\ldots,K\}}\{n^k(S_q^i)\}$ is nothing else but the number of correctly classified data elements of set S, which would be sorted down to the qth leaf. Moreover, the sum in the nominator of (13) is the total number of correctly classified data elements of set S, if the split was made with respect to the ith attribute. As a result, (13) is the accuracy obtained for elements of set S, if the split was made with respect to the ith attribute. Hence, the choice of the attribute that maximizes the value of (13) provides not only the maximum loss of some impurity measure [defined by (9)], but it also guarantees the maximum gain of the accuracy of the tree. Let $s_j,\ j=1,\ldots,n(S)$, denote the elements of the set S. Let us define a function $P_i(s)$, which is equal to one if the element s would be correctly classified (after splitting the considered node with respect to the ith attribute) and zero otherwise. It is obvious that $P_i(s)$ is a random variable from the binomial distribution with some unknown expected value μ_i and variance $\mu_i(1-\mu_i)$. Now, (13) can be expressed in the following way:

$$\Delta\tilde{g}_i(S) = \frac{\sum_{j=1}^{n(S)}P_i(s_j)}{n(S)}. \quad (15)$$

Obviously, values of (15) tend to μ_i as size of set S tends to infinity. Under assumption that the set S is a random sample of the whole data stream (or the whole large data set), one can say that μ_i is the value of (15), which would be obtained for the infinite set of elements, i.e., $\mu_i = \Delta\tilde{g}_i(S^\infty)$. Hence, (15) [and (13)] is the estimator of $\Delta\tilde{g}_i(S^\infty)$. If $n(S)$ is big enough, the probability distribution of estimator (15) can be approximated by the following normal distribution:

$$\Delta\tilde{g}_i(S) \longrightarrow N\left(\Delta\tilde{g}_i(S^\infty), \frac{\mu_i(1-\mu_i)}{n(S)}\right). \quad (16)$$

Hence, the difference $\Delta\tilde{g}_i(S) - \Delta\tilde{g}_j(S)$ also belongs to the normal distribution

$$\Delta\tilde{g}_i(S) - \Delta\tilde{g}_j(S) \longrightarrow N\left(\Delta\tilde{g}_i(S^\infty) - \Delta\tilde{g}_j(S^\infty), \frac{\mu_i(1-\mu_i)+\mu_j(1-\mu_j)}{n(S)}\right). \quad (17)$$

From the properties of the normal distribution [38], one can conclude that with probability $1-\delta$, the following inequality is satisfied:

$$\Delta\tilde{g}_i(S) - \Delta\tilde{g}_j(S) < \Delta\tilde{g}_i(S^\infty) - \Delta\tilde{g}_j(S^\infty) + z_{(1-\delta)}\frac{\sqrt{\mu_i(1-\mu_i)+\mu_j(1-\mu_j)}}{\sqrt{n(S)}} \quad (18)$$

where $z_{(1-\delta)}$ is the $(1-\delta)$ quantile of the standard normal distribution $\mathcal{N}(0,1)$. From (18), the following statement can be derived:

If

$$\Delta\tilde{g}_i(S) - \Delta\tilde{g}_j(S) > z_{(1-\delta)}\frac{\sqrt{\mu_i(1-\mu_i)+\mu_j(1-\mu_j)}}{\sqrt{n(S)}} \quad (19)$$

then, with probability $1-\delta$, the following inequality is true:

$$\Delta\tilde{g}_i(S^\infty) - \Delta\tilde{g}_j(S^\infty) > 0 \quad (20)$$

i.e., with probability $1-\delta$, the ith attribute would provide higher value of the accuracy than the jth attribute, if the data set was of infinite size. The value of $\mu_i \equiv \Delta\tilde{g}_i(S^\infty)$ [and $\mu_j \equiv \Delta\tilde{g}_j(S^\infty)$] is not known, but it is within the interval [0; 1]. Hence, the term on the right side of (19) can be bounded by

$$z_{(1-\delta)}\frac{\sqrt{\mu_i(1-\mu_i)+\mu_j(1-\mu_j)}}{\sqrt{n(S)}} \le z_{(1-\delta)}\sqrt{\frac{1}{2n(S)}}. \quad (21)$$

Therefore, if (10) is satisfied then, in view of (14) and (21), (19) also holds. As a result, (20) is satisfied with probability $1-\delta$. Hence, one can say that (10) is enough to say that with probability $1-\delta$ the ith attribute would give higher value of the accuracy than the jth attribute, if the data set was of infinite size. Moreover, if i is the index of the attribute with the highest value of accuracy and j is the index of the one with the second highest value, then with probability $1-\delta^*$ the ith attribute would maximize the value of accuracy if the data set were of infinite size. Then, the considered node is split with respect to the ith attribute. ■

Example 1: Let us assume that there are $n(S) = 10\,000$ data elements in the considered tree node. Let x and y be the indices of attributes with the highest values of accuracy gain function. Let us assume that the value of $\Delta g_x(S) - \Delta g_y(S)$ is equal to 0.01161. If the level of confidence is set to $1 - \delta = 0.95$, then $z_{(1-\delta)}$, read from the mathematical table of quantiles of the standard normal distribution, is equal to 1.644854 and one obtains

$$z_{(1-\delta)} \frac{1}{\sqrt{2n(S)}} = 0.011631. \tag{22}$$

Therefore (10) is not satisfied and we cannot say that the attribute with index x is better than the one with index y. Let us assume now that new 100 data elements arrived to the considered tree node. The total number of elements $n(S)$ in this node equals 10 100. Let us assume that the new value of $\Delta g_x(S) - \Delta g_y(S)$ is 0.116. For the current value of n, we have

$$z_{(1-\delta)} \frac{1}{\sqrt{2n(S)}} = 0.011573. \tag{23}$$

This time (10) is satisfied. Therefore, with the 0.95 level of confidence, we are allowed to say that attribute with index x is better than attribute with index y. The considered tree node is divided with respect to the xth attribute.

Example 2: Inequality (10) can be transformed as follows:

$$n(S) > \frac{(z_{(1-\delta)})^2}{2(\Delta g_x(S) - \Delta g_y(S))^2}. \tag{24}$$

The above inequality is an alternative way for determining whether to split the considered node or not. If the current number of elements satisfies (24), the node is divided with respect to the xth attribute. Let us assume that $\Delta g_x(S) - \Delta g_y(S) = 0.0116$. The values of $z_{(1-\delta)} = 1.644854$ are the same as in Example 1. Then, the term on the right side of (24) equals

$$\frac{(z_{(1-\delta)})^2}{2(\Delta g_x(S) - \Delta g_y(S))^2} = 10053.3. \tag{25}$$

Therefore, if $n(S)$ is greater or equal to 10 054 elements, the tree node is divided with respect to attribute with index x.

B. Hoeffding's Bound Analysis

As it was mentioned previously, Domingos and Hulten [33] claimed that the bound for the difference of split measure functions for two attributes should be given by (8). We will now show that such an approach is incorrect. Let us assume that the user demands that the best attribute computed in considered node according to the available data sample is the same, with probability $1 - \delta^*$, as the attribute derived from the whole infinite data stream (6). We will demonstrate that Hoeffding's bound (8) does not ensure obtaining the required probability $1 - \delta$. Domingos and Hulten [33] stated that the constant R in (8) is equal to the range of all possible values of the considered split measure function. For example, the information gain takes values in the interval $[0; \log_2 K]$, therefore $R = \log_2 K$. The authors did not consider the case of split measure based on the misclassifiation error. The set of possible values of this measure is $[0; 1 - 1/K]$, i.e., $R = 1 - 1/K$. Domingos and Hulten [33] claim that the bound (8) is valid for each possible probability distribution. Therefore, in the following analysis, a very simple two-class problem is considered ($K = 2$ and as a result the value of constant R is equal to 0.5). Data elements are characterized by two binary attributes ($D = 2$). Moreover, each combination of attributes values and class in data element is equally probable. Let S_N denote a set of N data elements taken randomly from the considered distribution. For this simple probability distribution, it is obvious that the difference between split measure values for the two attributes, i.e., $\Delta g_0(S_N) - \Delta g_1(S_N)$ (11), should be equal to 0 (no attribute should be chosen to split the considered node). The difference $\Delta g_0(S_N) - \Delta g_1(S_N)$ is allowed to exceed bound (8) only in the fraction δ of all cases of S_N [obviously the value of the bound depends on δ according to (8)]. To prove that the bound derived by the Hoeffding inequality is incorrect, it is enough to show that the actual probability, denoted by δ_1, that $\Delta g_0(S_N) - \Delta g_1(S_N)$ exceeds bound (8) is greater than δ at least for one particular value of δ and one particular number of data elements N. Let V_N denote the set of all possible sets S_N. Since there are two binary attributes and each element can belong to one of the two classes, the cardinality of set V_N is equal to 2^{3N}. For small N, the probability δ_1 can be calculated analytically

$$\delta_1 = \frac{|V'_N|}{|V_N|} \tag{26}$$

where $V'_N = \{S_N \in V_N : \Delta g_0(S_N) - \Delta g_1(S_N) > \epsilon(N, \delta)\}$ and $|\cdot|$ denotes the cardinality. For example for $N = 4$ and $\delta = 0.01$, the actual probability δ_1 is equal to 0.0176. The computations were performed for $N = 2, \ldots, 8$ and for two values of δ: 0.01 and 0.001. Unfortunately, for large numbers of data elements (such as $N \in \{100, 200, 1000\}$), the analytical calculations are impossible to carry out because of extremely large number of combinations 2^{3N} would have to be considered. Therefore, the Monte Carlo computer simulations were conducted to approximate the probability δ_1. For each value of N and δ, the set S_N was randomly generated 10^4 times. For each generated set, it was checked if the difference $\Delta g_0(S_N) - \Delta g_1(S_N)$ exceeds the bound. The fraction of positive cases is denoted by $\hat{\delta}_1$. The experiment was run for $N \in \{2, 3, 4, 5, 6, 7, 8, 100, 200, 1000\}$.

Analogously, the theoretical analysis and Monte Carlo simulations were performed for the approach proposed in this paper, for the same values of N and δ. Results are presented in Table I.

The results collected in the table were obtained for two different values of δ (first column) and for eleven different numbers of data elements N (second column). In the third column, the probability δ_1 of exceeding the Hoeffding bound, calculated by (26), is presented. The fourth column presents the values $\hat{\delta}_1$, which are the Monte Carlo approximations of δ_1. Analogously, the last two columns show the values of δ_1 and $\hat{\delta}_1$ with the Hoeffding bound replaced by the bound proposed in this paper (10)—based on the Gaussian approximation.

As can be seen, the actual probability δ_1 that the difference $\Delta g_0(S_N) - \Delta g_1(S_N)$ exceeds the Hoeffding bound is often

TABLE I
RESULTS OF THE HOEFFDING'S BOUND ANALYSIS

		Hoeffding's bound		Gaussian approximation bound	
	N	δ_1	$\hat{\delta}_1$	δ_1	$\hat{\delta}_1$
δ=1%	2	0.00%	0.00%	0.00%	0.00%
	3	0.00%	0.00%	0.00%	0.00%
	4	**1.76%**	**1.77%**	0.00%	0.00%
	5	**2.44%**	**2.3%**	0.00%	0.00%
	6	**3.74%**	**3.58%**	0.00%	0.00%
	7	0.47%	0.5%	0.00%	0.00%
	8	0.86%	0.9%	0.00%	0.00%
	100	-	**2.06%**	-	0.005%
	200	-	**2.04%**	-	0.005%
	1000	-	**2.08%**	-	0.001%
δ=0.1%	2	0.00%	0.00%	0.00%	0.00%
	3	0.00%	0.00%	0.00%	0.00%
	4	**1.76%**	**1.82%**	0.00%	0.00%
	5	0.00%	0.00%	0.00%	0.00%
	6	**0.3%**	**0.29%**	0.00%	0.00%
	7	**0.4%**	**0.41%**	0.00%	0.00%
	8	**0.8%**	**0.78%**	0.00%	0.00%
	100	-	**0.59%**	-	0.00%
	200	-	**0.68%**	-	0.00%
	1000	-	**0.81%**	-	0.00%

greater than the assumed probability δ. The theoretical results are confirmed by the computational simulations. It should be noted that the disagreement between δ and δ_1 occurs not only in the case of small N. It also holds for quite large values of N, what may have a practical meaning. On the other side, the bound presented in this paper, i.e., (10) in Theorem 1, guarantees that the wrong decision is never made in the case of $N = 2, \ldots, 8$. For $\delta = 0.01$ and $N \in \{100, 200, 1000\}$, the values of $\hat{\delta}_1$ are only a bit greater than 0, however, it is still much less than the assumed probability δ.

III. HYBRID ALGORITHM FOR DESIGNING DECISIONS TREES

As it was already mentioned in the introduction, all the algorithms considered in this paper (i.e., mDT, gDT, and hDT) are based on the idea presented in [33]. This idea is summarized as an online tree algorithm that pseudocode is shown in Fig. 1. The algorithm produces binary decision trees.

Let $n_{ij}^k(S)$ denote the number of data elements (in set S) from class k, with the jth value of ith attribute. The set of numbers $n_{ij}^k(S)$, $i = 1, \ldots, D$, $j = 1, \ldots, v_i$, $k = 1, \ldots, K$ is also called the sufficient statistics of set S. The algorithm starts with one single node—the root. The sufficient statistics in the root are all set to zero. Then, the tree is developed using subsequent data elements from the stream. Each data element s is sorted through the tree, according to the values of attributes and the current structure of the tree. Element s finally reaches a leaf L_p. The sufficient statistics in leaf L_p are updated. Let S_p denote the set of data elements collected so far in leaf L_p, $j_{i,s}$ denote the value of the ith attribute in data element s, and k_s denote the class of element s. The update of sufficient statistics is made in the following manner:

$$\forall_{i\in\{1,\ldots,D\}}\ n_{ij_{i,s}}^{k_s}(S_p) = n_{ij_{i,s}}^{k_s}(S_p) + 1. \tag{27}$$

Next, values of the split measure function $\Delta g_i(S_p)$ are calculated for each attribute, $i = 1, \ldots, D$. It should be noted

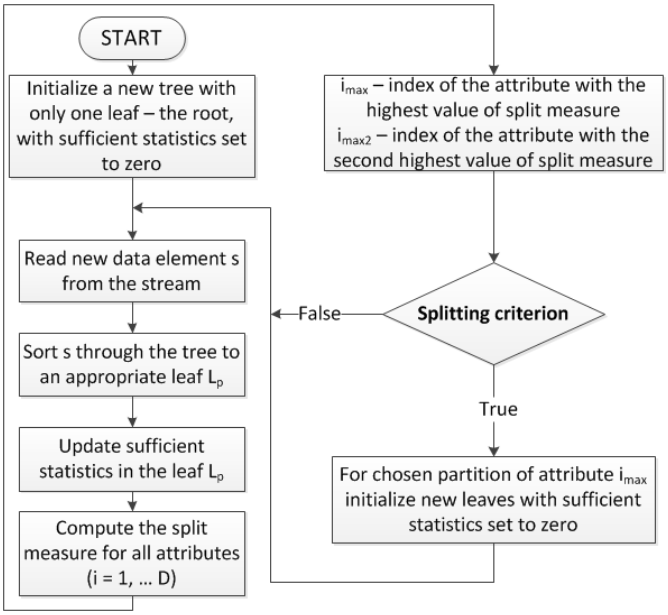

Fig. 1. Block diagram of the online tree algorithm.

that each value $\Delta g_i(S_p)$ is calculated with respect to the optimal partition of the set A_i of all possible values of the ith attribute into two disjoint subsets A_i^L and $A_i^R = A_i \backslash A_i^L$. The attributes with the highest ($i_{\max}$) and the second highest ($i_{\max 2}$) values of the split measure function are found. The difference $\Delta g_{i_{\max}}(S_p) - \Delta g_{i_{\max 2}}(S_p)$ is calculated and the splitting criterion is checked. If the result of the test is positive, the leaf L_p becomes a node and is split into two children nodes (leaves). One of them corresponds to the subset $A_{i_{\max}}^L$ of values of the ith attribute and the other one corresponds to the complement of set $A_{i_{\max}}^L$. The sufficient statistics in both children nodes are initially set to zero. Then, the whole procedure is performed for the next data element taken from the stream.

The only difference between the mDT, gDT, and hDT algorithms lies in the splitting criterion. The criterion for the mDT algorithm is justified by Theorem 1 and states as follows:

$$\Delta g_i(S) - \Delta g_j(S) > z_{(1-\delta)}\sqrt{\frac{1}{2n(S)}} \tag{28}$$

where $z_{(1-\delta)}$ is the $(1-\delta)$ quantile of the standard normal distribution $\mathcal{N}(0,1)$ and g(S) denotes the misclassification error for set S. The misclassification error is an impurity measure rarely mentioned in literature. For static decision trees, this measure provides much worse results than other impurity measures, e.g., Gini index that is used in the gDT algorithm. The splitting criterion for the gDT algorithm is introduced in [36] and states as follows:

$$\Delta g_i(S) - \Delta g_j(S) > z_{(1-\delta)}\sqrt{\frac{10K^2 - 16K + 8}{n(S)}} \tag{29}$$

where g(S) denotes the Gini index for set S. The main drawback of misclassification error is the fact that it allows a reasonable split only if it guarantees an increase of accuracy (in other cases it either makes a split according to a randomly

chosen attribute or it does not make a split at all—depending on particular implementation of the algorithm). In opposite, Gini index always provides a split that produces better organized subsets of data, even if this split does not improve the accuracy. As a result, it prepares data for reasonable splits of nodes in higher levels of the tree. Although the properties of misclassification error are undesirable in static decision trees, it has other advantages that can be suitable for online tree construction. In most cases, the mDT algorithm needs much less data to determine a split comparing with online trees based on other impurity measures.

As it is presented in the section of experimental results, the advantage of misclassification error discussed above is particularly beneficial at the beginning stages of online tree construction process. However, at higher levels of the tree data elements of one class predominate significantly over the rest of elements. In this case, no attribute provides the increase of accuracy. Then, the mDT algorithm cannot make a split and the further growth of the tree is inhibited. Therefore, for large number of processed data elements, the accuracy obtained for the gDT algorithm is higher than for the mDT algorithm.

The observation that the mDT algorithm gives higher accuracy at the beginning stages and the gDT algorithm performs better when the tree became more complex suggests combining these two methods. Therefore, the hDT algorithm has been proposed in this paper. The hDT algorithm produces the online tree with the splitting criterion defined as a disjunction of (28) and (29).

In summary, three algorithms are studied in this paper.

1) mDT—an online tree algorithm where splitting criterion is satisfied when (28) is satisfied.
2) gDT—an online tree algorithm where splitting criterion is satisfied when (29) is satisfied.
3) hDT—an online tree algorithm where splitting criterion is satisfied when at least one of (28) and (29) is satisfied.

The performance of these algorithms will be examined and discussed in the next section.

IV. Simulation Results

A. Data Preparation

In this section, the performance of the proposed method is discussed and compared with the gDT algorithm as well as with the static decision tree algorithm based on the misclassification error. Synthetic and real data were used.

Synthetic data were generated on a basis of synthetic decision trees. These synthetic trees were constructed in the same way, as described in [33]. At each level of the tree, after the first $d_{\min}$ levels, each node is replaced by a leaf with probability ω. The higher value of parameter ω implicates lower complexity of the tree. To the rest of nodes, a splitting attribute is randomly assigned; it has to be an attribute that has not already occurred in the path from the root to the considered node. The maximum depth of the synthetic tree is $d_{\max}$ (at this level all nodes are replaced by leaves). After the whole tree is constructed, to each leaf, a class is randomly assigned. Each synthetic tree represents a different data concept, which is a particular distribution of attributes values and classes. In brief, data concept determines the correlations between the attributes and classes, i.e., it is a particular classification problem. For the purpose of the following simulations, twelve synthetic trees were generated (all of them with $D = 30$ binary attributes, $d_{\min} = 3$ and $d_{\max} = 18$). Four different values of ω were considered: 0.1, 0.15, 0.2, and 0.25. For each of these values, three synthetic trees were generated. These twelve synthetic trees provided twelve different data concepts. For each concept, a testing data set consisting of 10 000 elements was generated using the corresponding synthetic tree. Hence, there are twelve testing data sets, one for each data concept. Each testing data set is generated only once and is used to evaluate the accuracy of decision trees at every stage of their development. In the simulations presented below, for any training data set size n and any value of parameter δ one obtains twelve different decision trees and, as a result, twelve values of classification accuracy (one for each synthetic data concept). The final result of accuracy for particular n and δ was calculated as the average over all twelve values.

In addition, two real data sets were used in the experiments, both taken from the UCI machine learning repository [39]. Although there are many different data sets available in the repository, only two of them seem to be suitable for the considered problem. An appropriate data set should contain as many data elements as possible to imitate the stream of data properly. One of the considered data sets is the KDD CUP 99, consisting of 4 898 431 data elements. Data are described by 41 attributes, seven of which are nominal and 34 are numerical. Each data element belongs to one of the two classes. One class represents a network attacks and the second one is identified with normal network connections (without attack). It should be noted that the data distribution is very uneven: the attack class constitutes about 80% of the whole data set. Although there exists an original testing data set in the repository, as its authors indicate, it is not from the same probability distribution as the training data set. Therefore, for the purposes of the presented simulations, the testing data sets were randomly selected from the training set. The data set was divided into two equinumerous parts, first one intended for learning and the second one for testing. Such procedure was performed five times, providing in total five training data sets and their complementary testing data sets. The accuracy evaluation was performed before every split of the node that occurred in the considered tree. The accuracy for one particular number of data elements is calculated as the average over five data sets described above. The second real set of data used in the following simulations is the Covertype data set, consisting of 581 012 data elements. Data are characterized by 10 numerical and 44 nominal attributes. Each data element belongs to one of seven classes, each representing forest cover type designation. The testing data sets were taken randomly from the training data set in the same way as for the KDD CUP 99 problem.

B. Performance of Decision Trees Based on Misclassification Error

First, we examine the performance of the mDT algorithm depending on the values of parameter δ. Simulation was

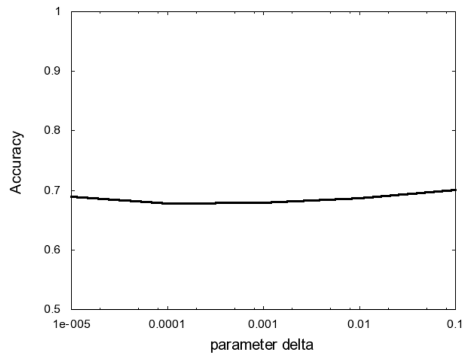

Fig. 2. Dependence between the value of parameter δ and the accuracy of the mDT algorithm averaged over all 12 concepts.

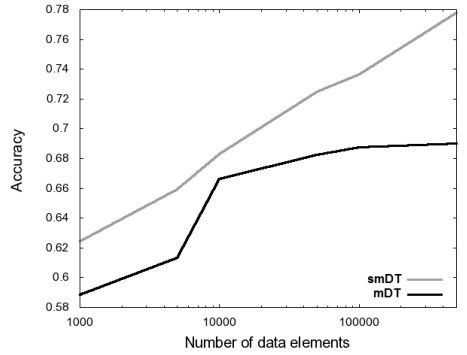

Fig. 3. Accuracies of online (black line) and static (gray line) decision trees with the splitting criterion based on the misclassification error averaged over all 12 concepts.

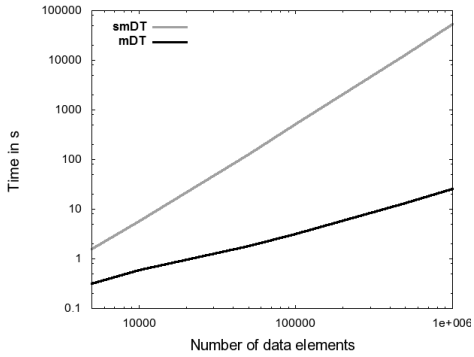

Fig. 4. Running time of online (black line) and static (gray line) decision trees with the splitting criterion based on the misclassification error averaged over all 12 concepts.

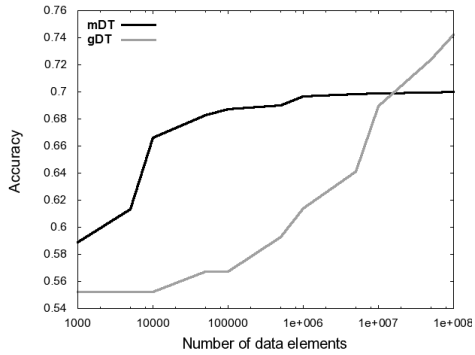

Fig. 5. Accuracies of the mDT algorithm (black line) and the gDT algorithm (gray line) averaged over all 12 concepts.

conducted for training data set of size $n = 10^8$, for five different values of δ, ranging from 10^{-5} to 10^{-1}. The results are shown in Fig. 2. As can be seen the accuracy does not depend on the value of δ. Owing to this fact, in the following experiments, the parameter δ is set to the value 0.1.

To compare the performance of the static and online decision trees based on the misclassification error impurity measure (the smDT and the mDT algorithms), two experiments were conducted. First, the accuracies of both algorithms were investigated. The averages of the values of accuracy obtained for 12 concepts are shown in Fig. 3. As can be seen the average performance of static tree is better than that obtained for the mDT. However, in half of investigated concepts, the accuracy of the mDT was slightly better, and for the other half, the accuracy of the static tree was much better. Due to this, the average result is in favor of the static tree. The second experiment compares the running time of the algorithms. The result shown in Fig. 4 shows that the average time needed for the static tree to process data is much higher than that obtained for online tree. The time necessary to process incoming data for the static tree is a power function depending on the number of data, whereas for the mDT, it is a linear function of n. Summarizing both former experiments, we draw the conclusion that with little decrease in accuracy, we gain significantly on processing time. Due to those results, we recommend to use the mDT for the purpose of mining data streams.

In the following simulations, the mDT and the gDT algorithms are compared. Algorithms were run for $n = 10^3$ data elements up to $n = 10^8$ and the results are shown in Fig. 5.

For the number of data elements n above 10^7, the gDT algorithm demonstrates higher accuracy; however, for smaller training data sets, the mDT dominates significantly. It is due to the fact that the mDT generally needs less data elements to split the node than the gDT algorithm. Therefore, at the beginning, the mDT produces more complex trees, what results in higher accuracy. Although Fig. 5 shows the accuracy averaged over 12 runs on 12 data concepts, the predominance of the mDT over the gDT at initial stages is observed for the majority of data concepts individually. This is shown in Fig. 6, where the situation for four different concepts is presented.

As can be seen the mDT is much more accurate at the beginning than the gDT for all four presented cases. Usually, when the gDT makes the first split in the tree (the split of the root), the mDT for the same number of data elements n already consists of a few nodes. Higher complexity of the tree ensures better accuracy.

The fact that the mDT guarantees higher accuracy for low numbers n may be of practical importance in some aspects of data stream mining. For example, the mDT may be used as a compound in some ensemble algorithms or other algorithms dealing with concept drift problem.

In the next experiments, the performance of the proposed method was tested on two real data sets, described

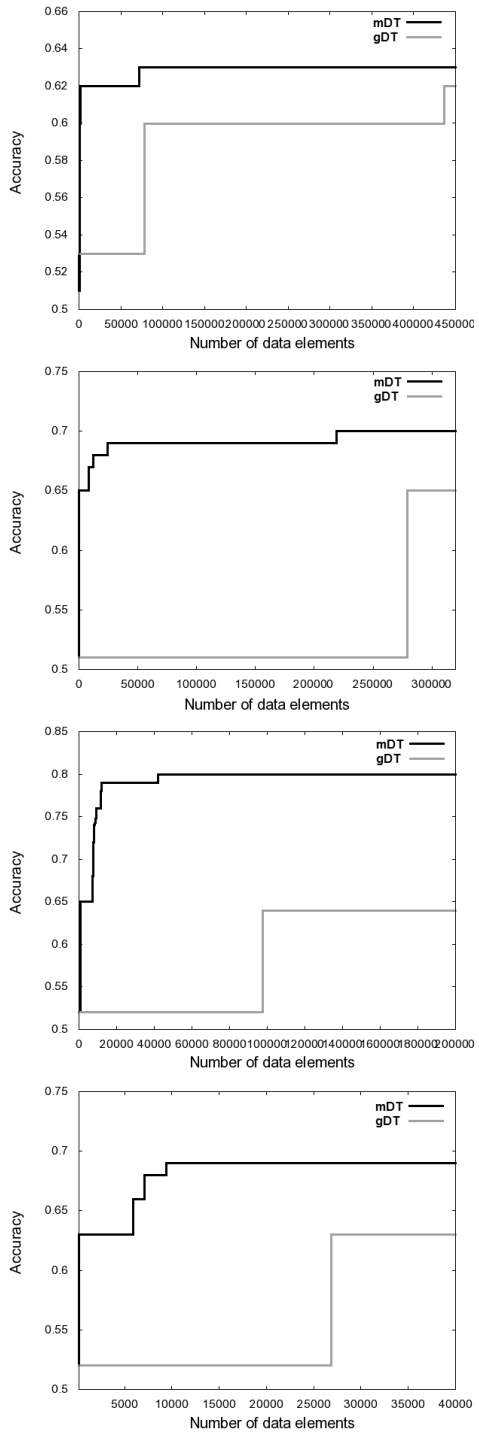

Fig. 6. Accuracies of the mDT algorithm (black line) and the gDT algorithm (gray line) for one data concept with $\omega = 0.1, 0.15, 0.2, 0.25$.

in Section IV-A. First, the KDD CUP 99 data set was considered. The final accuracy was almost the same for the mDT and the gDT algorithms (equal to 99.6% and 99.48%, respectively). However, the mDT algorithm achieves higher accuracy earlier than the gDT algorithm (i.e., processing fewer data elements). This result is shown in Fig. 7. The standard deviation of accuracy for 5000 data elements and above does not exceed the value $4 \cdot 10^{-3}\%$ for the gDT

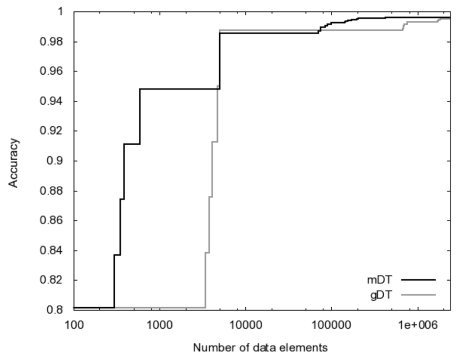

Fig. 7. Accuracies of the mDT algorithm (black line) and the gDT algorithm (gray line) for the KDD CUP 99 real data set.

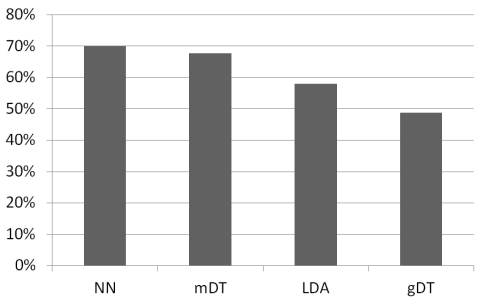

Fig. 8. Final accuracies of different classifiers obtained for the Covertype real data set.

algorithm and 0.6% for the mDT algorithm. For lower number of data elements, it is less than 10%.

The results obtained for the Covertype data set (which is a seven-class problem) are shown in Fig. 8. The values of accuracy for the mDT and the gDT algorithms are equal to 67.78% $\pm$ 0.84 and 48.79% $\pm$ 0.1, respectively. As the authors of the data set claim in its description [39], the accuracy obtained using neural network with backpropagation learning was 70%. The linear discriminant analysis method provided the accuracy equal to 58%. In view of these results, the accuracy obtained by the proposed method seems to be quite satisfactory. Moreover, it should be noted that, unlike neural networks and linear discriminant analysis, the mDT algorithm is performed in online manner, i.e., each single data element is processed only once. It should be noted that the size of training data set occurred to be too small to enable the gDT algorithm to make a split. Therefore, additional experiment was performed, using 579 012 data elements as a training set and 2000 as a testing set. The obtained accuracies were 67.1% for the mDT algorithm and 63.65% for the gDT algorithm.

C. Performance of Decision Trees Based on the Hybrid Split Measure

The major benefit of using the hDT algorithm is higher accuracy than for the mDT and the gDT algorithms, it is shown in Fig. 9. In most respects, initial splits were identical as in the mDT algorithm. Split of the root, using Gini index as a split measure, was made only once, for instance when

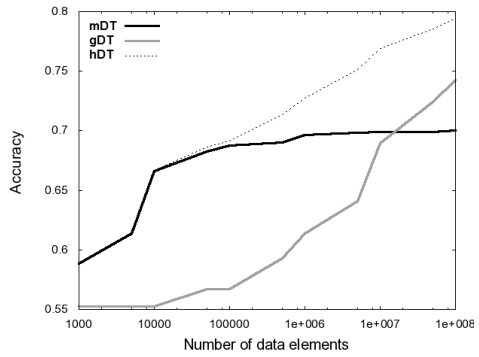

Fig. 9. Accuracies of the mDT algorithm (solid black line), the gDT algorithm (gray line), and the hDT algorithm (dotted black line) averaged over all 12 concepts.

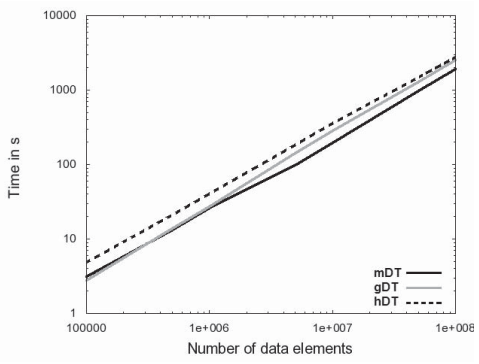

Fig. 10. Running time of the mDT algorithm (solid black line), the gDT algorithm (gray line), and the hDT algorithm (dotted black line) averaged over all 12 concepts.

application of misclassification error split measure does not allow making a split at all. The advantages of the misclassification error over the Gini index were already shown in Fig. 6. At some stage of splitting tree, the arrangement of data caused Gini index to be the impurity measure that provides split faster. However, in most cases, only one split was made using this measure and further growth of tree was obtained using misclassification error impurity measure again. In line with such procedure obtained accuracy is significantly higher for all concepts. On average over all 12 data concepts, the misclassification error impurity measure was used in about 92% cases. The maximum and minimum participation of Gini index in one concept was 12.1% and 6.24%, respectively.

The drawback of the hDT algorithm is double calculation and verification of splitting criterion. It is obvious that it has to prolong the running time. Moreover, more complex tree also extends time of processing data. However, as Fig. 10 shows, the running time is only slightly higher.

The experiments on real data sets were also conducted for the hDT algorithm. However, the results are exactly the same as in the case of the mDT algorithm. Due to properties of the data sets, all splits in the tree were executed according to the misclassification error impurity measure. The number of data elements occurred to be insufficient to made any split using the Gini index and consequently insufficient to reveal the advantages of the hDT algorithm.

TABLE II
A COMPARISON OF AVERAGE ACCURACIES OBTAINED FOR DIFFERENT ALGORITHMS AND VARIOUS DATASETS

	average accuracy				
Dataset	HTe	HTG	HTm	mDT	hDT
KDD CUP '99	0.9977	0.9973	0.9971	0.9960	0.9960
Covertype	0.6620	0.6720	0.6863	0.6777	0.6777
Adults	0.7512	0.7490	0.7975	0.7910	0.7910
Electricity	0.7053	0.6957	0.7309	0.6957	0.6957

TABLE III
STANDARD DEVIATIONS OF AVERAGE ACCURACIES FOR DIFFERENT ALGORITHMS AND VARIOUS DATASETS

	standard deviation of accuracy				
Dataset	HTe	HTG	HTm	mDT	hDT
KDD CUP '99	0.0008	0.0014	0.0004	0.0019	0.0019
Covertype	0.0095	0.0004	0.0168	0.0084	0.0084
Adults	0.0074	0.0030	0.0087	0.0023	0.0023
Electricity	0.0149	0.0014	0.0041	0.0014	0.0014

D. Comparison With the Hoeffding Tree Algorithm

In Section II-B, it was shown analytically and using Monte Carlo simulations that the bound obtained in [33] cannot be justified using Hoeffding's theorem. However, it still can be considered as a heuristic method, which gives satisfactory practical results. Therefore, the comparison between the mDT and the hDT algorithms with the Hoeffding tree algorithm was carried out. The simulations were performed on two previously mentioned data sets KDD CUP 99, Covertype and additionally two small data sets Adults and Electricity. The Adults data set consists of 30 161 elements and was taken from the UCI repository [39]. The Electricity data set with 45 312 elements can be found in [40]. Data sets were divided into training and testing subsets in the same way, as described in Ssection IV-A. The Hoeffding tree algorithm was run with the application of three different impurity measures: information entropy, Gini index, and misclassification error. The average accuracies obtained for each data set are presented in Table II and their standard deviations are collected in Table III. The abbreviations HTe, HTG, and HTm stand for the Hoeffding tree with information entropy, Gini index, and misclassification error, respectively.

As can be seen in majority of cases, the differences of accuracies for compared methods are within the range of standard deviation. Only for the Electricity data set, the HTm algorithm seems to provide noticeably higher accuracy than other methods. However, the HTm method produces decision trees with significantly higher number of leaves. The average number of leaves for different methods and different data sets are presented in Table IV.

The mDT and the hDT algorithms provide significantly less complex trees than the HTm algorithm. Moreover, as it was already mentioned, the basis of the mDT algorithm is mathematically justified unlike the HTm algorithm, which is heuristic. The conducted experiments demonstrate the high usability of the misclassification error impurity measure in the task of data stream mining.

TABLE IV
AVERAGE NUMBER OF LEAVES OBTAINED FOR DIFFERENT ALGORITHMS AND VARIOUS DATASETS

	average number of leaves				
Dataset	HTe	HTG	HTm	mDT	hDT
KDD CUP '99	19	8.2	6.8	4.8	4.8
Covertype	8.6	6.8	36.6	7.2	7.2
Adults	2.0	1.0	3.2	2.0	2.0
Electricity	54.0	2.4	77.6	2.2	2.2

V. CONCLUSION

In this paper, the application of decision trees in the task of data stream classification was considered. A new criterion for splitting the tree nodes was proposed, which is based on the impurity measure called misclassification error. The criterion allows to decide if the best attribute determined for the current set of data elements in the node is also the best according to the whole data stream. Numerical simulations proved that the presented algorithm (mDT) is able to give satisfactory accuracies of classification, especially at the beginning stages of decision tree development. In addition, the mentioned criterion was combined with the criterion based on Gini index. The resultant hybrid algorithm (hDT) provided satisfactory accuracies at any time of data stream processing, what was demonstrated in numerical experiments.

ACKNOWLEDGMENT

The authors would like to thank the associate editor and reviewers for their helpful comments.

REFERENCES

[1] J. Gama, "A survey on learning from data streams: Current and future trends," *Prog. Artif. Intell.*, vol. 1, no. 1, pp. 45–55, Apr. 2012.

[2] V. Grossi and F. Turini, "Stream mining: A novel architecture for ensemble-based classification," *Knowl. Inform. Syst.*, vol. 30, no. 2, pp. 247–281, Feb. 2012.

[3] C. Aggarwal, *Data Streams: Models and Algorithms*. New York, NY, USA: Springer-Verlag, 2007.

[4] M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "Mining data streams: A review," *Sigmod Rec.*, vol. 34, no. 2, pp. 18–26, Jun. 2005.

[5] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 81–94, Jan. 2014.

[6] R. Bose, W. van der Aalst, I. Zliobaite, and M. Pechenizkiy, "Dealing with concept drifts in process mining," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 154–171, Jan. 2014.

[7] K. Dyer, R. Capo, and R. Polikar, "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 12–26, Jan. 2014.

[8] J. Gomes, M. Gaber, P. Sousa, and E. Menasalvas, "Mining recurring concepts in a dynamic feature space," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 95–110, Jan. 2014.

[9] L. Kuncheva and W. Faithfull, "PCA feature extraction for change detection in multidimensional unlabeled data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 69–80, Jan. 2014.

[10] M. Pratama, S. Anavatti, P. Angelov, and E. Lughofer, "PANFIS: A novel incremental learning machine," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 55–68, Jan. 2014.

[11] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 27–39, Jan. 2014.

[12] E. Lughofer and P. Angelov, "Handling drifts and shifts in on-line data streams with evolving fuzzy systems," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 2057–2068, Mar. 2011.

[13] N. Kasabov, "Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 31, no. 6, pp. 902–918, Dec. 2001.

[14] P. Angelov, E. Lughofer, and X. Zhou, "Evolving fuzzy classifiers using different model architectures," *Fuzzy Sets Syst.*, vol. 159, no. 23, pp. 3160–3182, Dec. 2008.

[15] P. Angelov and N. Kasabov, "Evolving computational intelligence systems," in *Proc. 1st Int. Workshop Genet. Fuzzy Syst.*, Granada, Spain, 2005, pp. 76–82.

[16] D. Dovzan, V. Logar, and I. Skrjanc, "Solving the sales prediction problem with fuzzy evolving methods," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2012, pp. 1–8.

[17] P. Angelov, D. Filev, and N. Kasabov, *Evolving Intelligent Systems: Methodology and Applications*. Piscataway, NJ, USA: IEEE Press, 2010.

[18] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 81–94, Jan. 2014.

[19] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2001, pp. 97–106.

[20] W. Fan, Y. Huang, and P. Yu, "Decision tree evolution using limited number of labeled data items from drifting data streams," in *Proc. 4th IEEE Int. Conf. Data Mining*, 2004, pp. 379–382.

[21] C. Franke, "Adaptivity in data stream mining," Ph.D. dissertation, Dept. Elect. Eng., Univ. California, Berkeley, CA, USA, 2009.

[22] J. Liu, X. Li, and W. Hong, "Ambiguous decision trees for mining concept-drifting data streams," *Pattern Recognit. Lett., Elsevier*, vol. 30, no. 15, pp. 1347–1355, Nov. 2009.

[23] A. Tsymbal, "The problem of concept drift: Definitions and related work," Dept. Comput. Sci., Trinity College Dublin, Dublin, Ireland, Tech. Rep. TCD-CS-2004-15, Apr. 2004.

[24] P. Vivekanandan and R. Nedunchezhian, "Mining rules of concept drift using genetic algorithm," *J. Artif. Intell. Soft Comput. Res.*, vol. 1, no. 2, pp. 135–145, 2011.

[25] A. Martin and P. Bartlett, *Neural network learning: Theoretical Foundations*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[26] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. Inf. Theory*, vol. 13, no. 1, pp. 21–27, Jan. 1967.

[27] J. Gama, R. Fernandes, and R. Rocha, "Decision trees for mining data streams," *Intell. Data Anal.*, vol. 10, no. 1, pp. 23–45, Mar. 2006.

[28] A. Bifet and R. Kirkby. (2009). *Data Stream Mining a Practical Approach* [Online]. Available: http://www.bibsonomy.org/bibtex/23644381aa9a9bcb54a5db059600221c7/gerds0n

[29] M. Wozniak, "A hybrid decision tree training method using data streams," *Knowl. Inform. Syst.*, vol. 29, no. 2, pp. 335–347, 2011.

[30] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Amsterdam, The Netherlands: Elsevier, 2006.

[31] J. Quinlan, *Learning Efficient Classification Procedures and Their Application to Chess End Games*. San Francisco, CA, USA: Morgan Kaufmann, 1983, pp. 463–482.

[32] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. London, U.K.: Chapman and Hall, 1993.

[33] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2000, pp. 71–80.

[34] L. Rutkowski, L. Pietruczuk, P. Duda, and M. Jaworski, "Decision trees for mining data streams based on the McDiarmid's bound," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1272–1279, Jun. 2013.

[35] L. Rutkowski, L. Pietruczuk, P. Duda, and M. Jaworski, "Decision trees for mining data streams based on the Gaussian approximation," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 1, pp. 108–119, Jan. 2014.

[36] L. Rutkowski, M. Jaworski, L. Pietruczuk, and P. Duda, "The CART decision tree for mining data streams," *Int. J. Inform. Sci.*, vol. 266, pp. 1–15, May 2014.

[37] B. Pfahringer, G. Holmes, and R. Kirkby, "New options for Hoeffding trees," in *AI*, M. A. Orgun and J. Thornton, Eds. New York, NY, USA: Springer-Verlag, 2007, pp. 90–99.

[38] O. Kardaun, *Classical Methods of Statistics*. New York, NY, USA: Springer-Verlag, 2005.

[39] A. Frank and A. Asuncion. (2010). *UCI Machine Learning Repository* [Online]. Available: http://archive.ics.uci.edu/ml/

[40] (2014). *Massive Online Analysis* [Online]. Available: http://moa.cms.waikato.ac.nz/datasets/

Leszek Rutkowski (F'05) received the M.Sc. and Ph.D. degrees from the Wroclaw University of Technology, Wroclaw, Poland, in 1977 and 1980, respectively.

He has been with the Czestochowa University of Technology, Czestochowa, Poland, since 1980, where he is currently a Professor and the Director of the Institute of Computational Intelligence. From 1987 to 1990, he held a visiting position with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA. He has authored over 200 publications, including 20 papers in various series of IEEE Transactions. His current research interests include stream data mining, computational intelligence, pattern classification, and expert systems.

Dr. Rutkowski served in the IEEE Computational Intelligence Society as the Chair of the Distinguished Lecturer Program from 2008 to 2009 and the Chair of the Standards Committee from 2006 to 2007. He is the Founding Chair of the Polish Chapter of the IEEE Computational Intelligence Society, which received the 2008 Outstanding Chapter Award. In 2004, he was elected as a member of the Polish Academy of Sciences. He was awarded by the IEEE Fellow Membership Grade for contributions to neurocomputing and flexible fuzzy systems in 2004. He was a recipient of the IEEE TRANSACTIONS ON NEURAL NETWORKS Outstanding Paper Award in 2005.

Maciej Jaworski was born in Czestochowa, Poland, in 1985. He received the M.Sc. (Hons.) degree in theoretical physics from Jagiellonian University, Krakow, Poland, in 2009, and the M.Sc. degree in applied computer science from the University of Science and Technology, Krakow, in 2011. He is currently pursuing the Ph.D. degree in computer science with the Institute of Computational Intelligence, Czestochowa University of Technology, Czestochowa.

His current research interests include computational intelligence and data stream mining.

Lena Pietruczuk was born in Blachownia, Poland, in 1986. She received the M.S. degree in mathematics (genetic algorithms and their use in forecasting demand) from the Department of Mathematics and Computer Science, University of Wroclaw, Wroclaw, Poland, in 2010. She is currently pursuing the Ph.D. degree in computer science with the Institute of Computational Intelligence, Czestochowa University of Technology, Czestochowa, Poland.

Her current research interests include data stream mining, neural networks, and evolutionary algorithms.

Piotr Duda received the M.Sc. degree in mathematics from the Department of Mathematics, Physics, and Chemistry, University of Silesia, Katowice, Poland, in 2009. He is currently pursuing the Ph.D. degree in computer science with the Institute of Computational Intelligence, Czestochowa University of Technology, Czestochowa, Poland.

His current research interests include statistics and data stream mining, in particular, classification problems.