The CART Decision Tree for Mining Data Streams

Leszek Rutkowski[a,b], Maciej Jaworski[a], Lena Pietruczuk[a], Piotr Duda[a]

[a] *Institute of Computational Intelligence, Czestochowa University of Technology, ul. Armii Krajowej 36, 42-200 Czestochowa, Poland*
[b] *Information Technology Institute, Academy of Management, 90-113 Łódź, Poland*

Abstract

One of the most popular tools for mining data streams are decision trees. In this paper we propose a new algorithm, which is based on the commonly known CART algorithm. The most important task in constructing decision trees for data streams is to determine the best attribute to make a split in the considered node. To solve this problem we apply the Gaussian approximation. The presented algorithm allows to obtain high accuracy of classification, with a short processing time. The main result of this paper is the theorem 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 data stream.

Keywords: Data steam, decision trees, CART, Gini index, Gaussian approximation

1. Introduction

Among the plenty of techniques and methods used in machine learning or data mining, the classification seems to be one of the most important [14, 21, 31, 35]. Let A^i denotes the set of possible values of attribute a^i, for $i = 1, \ldots, D$. The aim of the classification task is to find a classifier $h : A^1 \times \ldots \times A^D \to \{1, \ldots, K\}$ based on the training dataset $\mathbf{S} \subset A^1 \times \ldots \times A^D \times \{1, \ldots, K\}$. The dataset $\mathbf{S}$ consists of n elements $s_m = (v_m, k_m) = ([v_m^1, \ldots, v_m^D], k_m)$, $m = 1, \ldots, n$, where

Email addresses: leszek.rutkowski@iisi.pcz.pl (Leszek Rutkowski), maciej.jaworski@iisi.pcz.pl (Maciej Jaworski), lena.pietruczuk@iisi.pcz.pl (Lena Pietruczuk), piotr.duda@iisi.pcz.pl (Piotr Duda)

- $v_m^i \in A^i$ is the value of attribute a^i for data element s_m,
- $k_m \in \{1, \ldots, K\}$ is a class of data element s_m.

The classifier h is used to assign a class $k \in \{1, \ldots, K\}$ to unlabeled data elements $v \in A^1 \times \ldots \times A^D$. For static datasets a variety of classification methods have been proposed in literature. The most popular are neural networks [29, 30], k-nearest neighbors [4] or decision trees [3, 26, 27], which are within the scope of this paper. The decision tree is a structure composed of nodes and branches. Terminal nodes are called leaves. To each node L_q, which is not a leaf, an appropriate splitting attribute a^i is assigned. The assignment of the attribute to the considered node is the crucial part of the decision tree construction algorithm. Usually the choice of the attribute is based on some impurity measure, calculated for the corresponding subset $\mathbf{S}_q$ of the training dataset $\mathbf{S}$. The impurity measure is used to calculate the split measure function for each attribute. According to the chosen attribute, the node is split into children nodes, which are connected with their parent nodes by branches. There exist two types of decision trees: binary and non-binary. In the case of non-binary tree, the node is split into as many children as the number of elements of set A^i. Each branch is labeled by a single value of attribute a^i. If the tree is binary, the node is split into two children nodes. The branches are labeled by some complementary subsets of A^i. According to the branches, the set $\mathbf{S}_q$ is partitioned into subsets, which then become the training subsets in the corresponding children nodes. Leaves serve to label unclassified data elements.

The existing algorithms for decision trees construction differ mainly in the two fields mentioned above: type of tree (binary or non-binary) and type of impurity measure. The ID3 algorithm [26], for example, produces non-binary trees. As the impurity measure the information entropy is applied. The split measure function, based on it, is called the information gain. An upgraded version of the ID3 algorithm, also based on the information entropy, is the C4.5 algorithm [27]. In this algorithm an additional function, called the split information, is proposed. It takes high values for attributes with large domains. As the split measure function in the C4.5 algorithm, the ratio of the information gain and the split information is used. In the CART algorithm [3] binary trees are constructed. The impurity measure is in the form of Gini index.

The algorithms mentioned above (ID3, C4.5 and CART) are designed for static datasets. They cannot be applied directly to data streams [1, 2, 7, 11,

12, 13, 24], which are of infinite size. Moreover, in case of data streams data elements income to the system continuously with very high rates. Additionally, the concept drift may occur [8, 10, 17, 20, 22, 34], which means that the concept of data evolve in time. In the literature there are various approaches to deal with data streams. In recent decade an appropriate tool to solve data streams problems is incremental learning [6, 15, 25]. Among few characterizations of incremental learning presented in the literature we cite [15] stating that 'incremental learning should be capable of learning a new information and retaining the previously acquired knowledge, without having access to the previously seen data'. It is easily seen that the approach based on the decision trees possesses main features of the incremental learning.

In this paper we present a method to adapt the CART algorithm to deal with data streams. The main problem is to determine the best attribute in each node. Since it is not possible to compute the values of split measure based on infinite dataset, they should be estimated using the sample of data in considered node. Then, with some probability, one can say whether the best attribute according to this sample is also the best with respect to the whole stream. In the literature there are few approaches to solve this problem:

a) The commonly known algorithm called 'Hoeffding's Tree' was introduced by P. Domingos and G. Hulten in [5]. The main mathematical tool used in this algorithm was the Hoeffding's bound [16] in the form:

Theorem 1. *If $X_1, X_2, \ldots, X_n$ are independent random variables and $a_i \leq X_i \leq b_i$ $(i = 1, 2, \ldots, n)$, then for $\epsilon > 0$*

$$P\{\overline{X} - E[\overline{X}] \geq \epsilon\} \leq e^{-2n^2\epsilon^2 / \sum_{i=1}^{n}(b_i - a_i)^2} \tag{1}$$

where $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $E[\overline{X}]$ is expected value of $\overline{X}$.

For $a_i = a$ and $b_i = b$, $(i = 1, 2, \ldots, n)$ it states that after n observations the true mean of the random variable of range $R = b - a$ does not differ from the estimated mean by more than

$$\epsilon_H = \sqrt{\frac{R^2 \ln 1/\alpha}{2n}} \tag{2}$$

with probability $1-\alpha$. However, we would like to emphasize that the Hoeffding's bound is wrong tool to solve the problem of choosing the best attribute to make a split in the node. This observation follows from the fact that the split measures, like information gain and Gini index, can not be presented as a sum of elements and they are using only frequency of elements. Moreover, Theorem 1 is applicable only for numerical data. Therefore the idea presented in [5] violates the assumptions of Theorem 1 and the concept of Hoeffding Trees has no theoretical justification.

b) In [33] the authors proposed new method in which they used the McDiarmid's inequality [23] instead of Hoeffding's bound as a tool for choosing the best attribute to make a split. First the function $f(\mathbf{S})$ was proposed as a difference between the values of Gini indices of two attributes

$$f(\mathbf{S}) = Gini_{a^x}(\mathbf{S}) - Gini_{a^y}(\mathbf{S}). \tag{3}$$

By applying the McDiarmid's inequality authors obtained the value of

$$\epsilon_M = 8\sqrt{\frac{\ln(1/\alpha)}{2n}}, \tag{4}$$

such that for any fixed α, if $f(\mathbf{S}) > \epsilon_M$, then with probability $1-\alpha$ attribute a^x is better to make a split than attribute a^y.

c) In [18] the authors proposed a method, based on the Multivariate Delta Method, for determining if the best attribute calculated from the data sample in considered node is also the best according to the whole stream. Even though the idea was valuable, the authors omitted the issue of calculating some of the necessary parameters what is not trivial. Therefore, the method does not have any practical application.

d) In [32] the authors presented the Gaussian Decision Tree (GDT) algorithm, which is based on the idea presented in [5]. The GDT algorithm was developed on the basis of the ID3 algorithm. Unfortunately, the GDT algorithm can be applied only for the two-class problem.

In this paper we propose a new algorithm, called *CART for data stream* (dsCART), inspired by [5]. The novelty is summarized as follows:

i. Following the idea of [18] we propose a new method to determine the best attribute to split the considered node. Contrary to [18], our method has a practical meaning and can be applied in a combination with many developed so far data stream mining algorithms, e.g. the Concept-adapting Very Fast Decision Trees (CVFDT) [17] or Hoeffding Option Trees (HOT) [24]. Our approach is based on the Taylor's Theorem and the properties of the normal distribution [19, 28].

ii. We prove the theorem (see Section 3) showing that the best attribute calculated from a finite data sample in the considered node is also the best according to the whole stream.

iii. We derive a formula (see section 3) allowing to determine a minimal number of data stream elements such that the tree node can be divided with respect to a considered attribute.

iv. We prove that for the two-class problem the number of elements n in a node, needed to determine if the split should be made, is smaller in the dsCART algorithm than in the McDiarmid Tree algorithm, with the same level of confidence $1-\alpha$. This is shown in section 5 and confirmed in experimental section.

v. The dsCART algorithm developed in this paper does not require any prepruning operations and is applicable for any number of classes, contrary to the GDT algorithm.

vi. Through computer simulations we show performance of our algorithm including a superiority of the dsCART over the GDT algorithm if computational time is taken into account.

The results of the paper are applicable to both categorical and numerical data, in the latter case we should properly choose a split point and generate a binary tree. This is commonly used procedure to deal with numerical data in decision trees. Moreover, it should be emphasized that our main result (Theorem 1 in section 3) is applicable to solve problems with concept drift. More specifically, our result should replace the Hoeffding's bound used incorrectly in algorithms like the Concept-adapting Very Fast Decision Trees (CVFDT) [17] or Hoeffding Option Trees (HOT) [24, 5] . We stress that the idea of the CVFDT and HOT algorithms is correct, however authors of both algorithms incorrectly used the Hoeffding's bound in their papers. Our result

can be combined with those algorithms, replacing the Hoeffding's bound by formulas (23) and (24).

The rest of the paper is organized as follows. In section 2 the CART algorithm is recalled. The main result of this paper is described in section 3. In section 4 the dsCART algorithm is introduced. In Section 5 the dsCART algorithm and the McDT algorithm are compared. Experimental results are shown in section 6 and conclusions are drawn in section 7.

2. The CART algorithm

Now we will briefly describe the CART algorithm. The introduced notation will be useful in the rest of the paper. The CART algorithm starts with a single node L_0 - the root. During the learning process, in each created node L_q a particular subset $\mathbf{S}_q$ of the training dataset $\mathbf{S}$ is processed (for the root $\mathbf{S}_0 = \mathbf{S}$). If all elements of set $\mathbf{S}_q$ are of the same class, the node is tagged as a leaf and the split is not made. Otherwise, according to the split measure function, the best attribute to split is chosen among the available attributes in considered node. For each available attribute a^i, the set of attribute values A^i is partitioned into two disjoint subsets A^i_L and A^i_R ($A^i = A^i_L \cup A^i_R$). The choice of A^i_L automatically determines the complementary subset A^i_R, therefore the partition is represented further only by A^i_L. The set of all possible partitions of set A^i is denoted by $\mathbf{V}_i$. Subsets A^i_L and A^i_R divide the dataset $\mathbf{S}_q$ into two disjoint subsets: left $\mathbf{L}_q(A^i_L)$ and right $\mathbf{R}_q(A^i_L)$

$$\mathbf{L}_q(A^i_L) = \{s_j \in \mathbf{S}_q | v^i_j \in A^i_L\}, \tag{5}$$

$$\mathbf{R}_q(A^i_L) = \{s_j \in \mathbf{S}_q | v^i_j \in A^i_R\}. \tag{6}$$

Sets $\mathbf{L}_q(A^i_L)$ and $\mathbf{R}_q(A^i_L)$ depend on the chosen attribute and partition of its values. Let $p_{L,q}(A^i_L)$ ($p_{R,q}(A^i_L)$) denote the fraction of data elements from $\mathbf{S}_q$, which belong to the subset $\mathbf{L}_q(A^i_L)$ ($\mathbf{R}_q(A^i_L)$). Since the fractions $p_{L,q}(A^i_L)$ and $p_{R,q}(A^i_L)$ are dependent

$$p_{R,q}(A^i_L) = 1 - p_{L,q}(A^i_L), \tag{7}$$

only one of these parameters is needed to be considered, e.g. $p_{L,q}(A^i_L)$. The fraction of elements from $\mathbf{L}_q(A^i_L)$ ($\mathbf{R}_q(A^i_L)$), from class k, is denoted by $p_{kL,q}(A^i_L)$ ($p_{kR,q}(A^i_L)$). The fraction of all data elements $\mathbf{S}_q$ in considered node L_q, from class k, is denoted by $p_{k,q}$. Note that $p_{k,q}$, $k = 1, \ldots, K$, are

not dependent on chosen attribute a^i and partition A_L^i. As it was mentioned previously, the impurity measure used in the CART algorithm is the Gini index. For any subset $\mathbf{S}_q$ of training dataset it is given by

$$Gini(\mathbf{S}_q) = 1 - \sum_{k=1}^{K} (p_{k,q})^2. \tag{8}$$

It is easily seen that the Gini index reaches its minimum (zero) when all cases fall into a single target category, and maximum is obtained when records are equally distributed among all classes. Furthermore, the weighted Gini index of subset $\mathbf{S}_q$, resulting from the choice of partition A_L^i, is defined as follows

$$wGini(\mathbf{S}_q, A_L^i) = \quad p_{L,q}(A_L^i) Gini(\mathbf{L}_q(A_L^i)) + (1 - p_{L,q}(A_L^i)) Gini(\mathbf{R}_q(A_L^i)), \tag{9}$$

where Gini indices of sets $\mathbf{L}_q(A_L^i)$ and $\mathbf{R}_q(A_L^i)$ are given analogously as in (8)

$$Gini(\mathbf{L}_q(A_L^i)) = 1 - \sum_{k=1}^{K} (p_{kL,q}(A_L^i))^2, \tag{10}$$

$$Gini(\mathbf{R}_q(A_L^i)) = 1 - \sum_{k=1}^{K} (p_{kR,q}(A_L^i))^2. \tag{11}$$

A split measure function in the CART algorithm is defined as a difference between Gini index (8) and the weighted Gini index (9). Analogously to the information gain used in the ID3 algorithm, this split measure function is called Gini gain. It is dependent on the chosen partition A_L^i of attribute a^i

$$g(\mathbf{S}_q, A_L^i) = Gini(\mathbf{S}_q) - wGini(\mathbf{S}_q, A_L^i). \tag{12}$$

Among all the possible partitions A_L^i of set A^i, the one which maximizes the value of Gini gain is chosen

$$\tilde{A}_{L,q}^i = \arg \max_{A_L^i \in \mathbf{V}_i} \{g(\mathbf{S}_q, A_L^i)\}. \tag{13}$$

The partition $\tilde{A}_{L,q}^i$ is called the optimal partition of set A^i for the subset $\mathbf{S}_q$ of training dataset. This optimal partition generates subsets $\mathbf{L}_q^i \equiv \mathbf{L}_q^i(\tilde{A}_{L,q}^i)$ and $\mathbf{R}_q^i \equiv \mathbf{R}_q^i(\tilde{A}_{L,q}^i)$. The value of $g_q^i = g(\mathbf{S}_q, \tilde{A}_{L,q}^i)$ is called the Gini gain of

subset $\mathbf{S}_q$ for attribute a^i. Among all the available attributes in the node L_q, the one with the highest value of Gini gain is chosen. The node L_q is split into two children nodes L_{last+1} and L_{last+2}, where $last$ is the index of the node created lately in the whole tree. Let us assume that the highest value of Gini gain is obtained for attribute a^x. Then all the calculations described above are performed in node L_{last+1}, using the subset $\mathbf{S}_{last+1} = \mathbf{L}_q^x$, and in node L_{last+2}, using the subset $\mathbf{S}_{last+2} = \mathbf{R}_q^x$. The list of available attributes in nodes L_{last+1} and L_{last+2} is taken from the node L_q, with the exception of the attribute a^x. The considered node L_q is not split if either the list of available attributes in the node contains only one element or all the elements from the subset $\mathbf{S}_q$ are from the same class.

3. Main Results

In section 2 all the fractions, e.g. $p_L(A_L^i)$, were computed based on the whole data set. In this section our discussion will concern a data stream problem. Since data streams are of infinite size, it is impossible to compute the fractions as in the CART algorithm. They can be only estimated based on available sample of data.

Now, we consider a situation in one particular node. Therefore, in all notations introduced before we omit $\mathbf{S}_q$ for clarity. Similarly to formula (7), the following dependencies are true

$$p_{KL}(A_L^i) = 1 - \sum_{j=1}^{K-1} p_{jL}(A_L^i), \tag{14}$$

$$p_{KR}(A_L^i) = 1 - \sum_{j=1}^{K-1} p_{jR}(A_L^i). \tag{15}$$

Therefore we consider further only $2(K-1)$ out of $2K$ parameters $p_{jL}(A_L^i)$ and $p_{jR}(A_L^i)$, $j \in \{1, \ldots, K-1\}$.

Moreover, fractions p_k ($k \in \{1, \ldots, K\}$) are dependent as well

$$p_K = 1 - \sum_{j=1}^{K-1} p_j. \tag{16}$$

Therefore, only $K-1$ of them, i.e $p_j, j \in \{1,\ldots,K-1\}$, are important. Note that the fraction p_j does not depend on a chosen attribute a^i and partition $A_L^i \cup A_R^i$, and it can be expressed using $p_L(A_L^i)$, $p_{jL}(A_L^i)$ and $p_{jR}(A_L^i)$ as follows

$$p_j(p_L, p_{jL}, p_{jR}) \quad = p_L(A_L^i)p_{jL}(A_L^i) + (1 - p_L(A_L^i))p_{jR}(A_L^i). \tag{17}$$

For any dataset $\mathbf{X}$, consisting of elements belonging to one of K classes, its Gini index can be calculated using the following formula

$$Gini(P_1,\ldots,P_{K-1}) = 1 - \sum_{j=1}^{K} P_j^2 = 1 - \sum_{j=1}^{K-1} P_j^2 - \left(1 - \sum_{j=1}^{K-1} P_j\right)^2, \tag{18}$$

where P_j is a fraction of elements from the set $\mathbf{X}$, belonging to the j-th class. Therefore, the Gini indices (8), (10) and (11) can be expressed as functions of $K-1$ variables, i.e. $Gini(p_1,\ldots,p_{K-1})$ for (8), $Gini(p_{1L}(A_L^i),\ldots,p_{(K-1)L}(A_L^i))$ for (10) and $Gini(p_{1R}(A_L^i),\ldots,p_{(K-1)R}(A_L^i))$ for (11), respectively.

The Gini gain function is a function of parameters mentioned above, i.e.

$$\begin{aligned} &g(p_L, p_{1L},\ldots,p_{(K-1)L}, p_{1R},\ldots,p_{(K-1)R}) = Gini(p_1,\ldots,p_{K-1}) \\ &-p_L Gini(p_{1L},\ldots,p_{(K-1)L}) - (1-p_L)Gini(p_{1R},\ldots,p_{(K-1)R}). \end{aligned} \tag{19}$$

The optimal partition of set A^i is defined analogously as in formula (13)

$$\tilde{A}_L^i = \arg\max_{A_L^i \in \mathbf{V}_i} \{g((p_L(A_L^i), p_{1L}(A_L^i),\ldots,p_{(K-1)L}(A_L^i), p_{1R}(A_L^i),\ldots,p_{(K-1)R}(A_L^i)\}. \tag{20}$$

We introduce the following notation for fractions associated with the optimal partition

$$\begin{aligned} &\left[p_L^i, p_{1L}^i,\ldots,,p_{(K-1)L}^i, p_{1R}^i,\ldots,p_{(K-1)R}^i\right] = \\ &= \left[p_L(\tilde{A}_L^i), p_{1L}(\tilde{A}_L^i),\ldots,p_{(K-1)L}(\tilde{A}_L^i),\ p_{1R}(\tilde{A}_L^i),\ldots,p_{1R}(\tilde{A}_L^i)\right] \end{aligned} \tag{21}$$

The value of $g^i = g(p_L^i, p_{1L}^i,\ldots,p_{(K-1)L}^i, p_{1R}^i,\ldots,p_{(K-1)R}^i)$ is called the Gini gain for attribute a^i. Parameters $\overline{p_L^i}, \overline{p_{1L}^i},\ldots,\overline{p_{(K-1)L}^i}$ and $\overline{p_{1R}^i},\ldots,\overline{p_{(K-1)R}^i}$ are

estimators of p_L^i, $p_{1L}^i, \ldots, p_{(K-1)L}^i$ and $p_{1R}^i, \ldots, p_{(K-1)R}^i$, respectively. They can be treated as arithmetic means of some random variables from binomial distributions. Let us consider the data elements s_m from data set $\mathbf{S}$, $m \in \{1, ..., n\}$. We define the random variable $\zeta_L^{i,m}$, which is equal to 1 if $s_m \in \mathbf{L}^i$ and 0 otherwise. Variable $\zeta_L^{i,m}$ is from the binomial distribution with mean $\mu_L^i = p_L^i$ and variance $\left(\sigma_L^i\right)^2 = p_L^i(1 - p_L^i)$. Similarly we define $\zeta_{kL}^{i,m}, k \in \{1, \ldots, K-1\}$ (for elements l_m from the set $\mathbf{L}^i$, $m \in \{1, ..., n_L^i\}$) and $\zeta_{kR}^{i,m}, k \in \{1, \ldots, K-1\}$ (for elements r_m from the set $\mathbf{R}^i$, $m \in \{1, ..., n_R^i\}$) random variables, from the binomial distributions with means $\mu_{kL}^i = p_{kL}^i$ and $\mu_{kR}^i = p_{kR}^i$ and variances $\left(\sigma_{kL}^i\right)^2 = p_{kL}^i(1 - p_{kL}^i)$ and $\left(\sigma_{kR}^i\right)^2 = p_{kR}^i(1 - p_{kR}^i)$, respectively. Variable $\zeta_{kL}^{i,m}$ is equal to 1 if l_m is from the k-th class and $\zeta_{kR}^{i,m}$ equals 1 if r_m is from the k-th class.

The main result of this paper is the following theorem stating that if the difference between the Gini gain estimates obtained for two attributes is greater than a specific value, given by (24), then with a fixed probability there is, roughly speaking, a statistical difference between the true Gini gains. This allows either to determine, from a recent fragment of data, the best attribute to split on or to say that the information to determine the split is statistically insufficient.

For convenience, let us denote

$$\overline{g^i} = g(\overline{p_L^i}, \overline{p_{1L}^i}, \ldots, \overline{p_{(K-1)L}^i}, \overline{p_{1R}^i}, \ldots, \overline{p_{(K-1)R}^i}) \tag{22}$$

Theorem 2. *Let us consider two attributes a^x and a^y, for which we have calculated the values of the Gini gain function. If the difference of these values satisfies the following condition*

$$\overline{g^x} - \overline{g^y} > \epsilon_{G,K}, \tag{23}$$

where

$$\epsilon_{G,K} = z_{(1-\alpha)} \frac{\sqrt{2Q(K)}}{\sqrt{n}}, \tag{24}$$

$z_{(1-\alpha)}$ is the $(1-\alpha)$-th quantile of the standard normal distribution $N(0,1)$ and

$$Q(K) = 5K^2 - 8K + 4, \tag{25}$$

then g^x *is greater than* g^y *with probability* $1 - \alpha$.

Proof: see Appendix A.

Remark 1

If a^x and a^y are attributes with the highest values of Gini gain, then a^x can be chosen to split the considered leaf node, with the level of confidence $(1 - \alpha)$.

Example 1

Let us assume that there are $n = 10000$ data elements in the considered tree node. The number of classes $K = 3$. According to formula (25) we have $Q(K) = 25$. Let a^x and a^y be the attributes with the highest values of Gini gain function. For convenience, let us denote

$$g^{x,y} = g(\overline{p_L^x}, \overline{p_{1L}^x}, \overline{p_{2L}^x}, \overline{p_{3L}^x}, \overline{p_{1R}^x}, \overline{p_{2R}^x}, \overline{p_{3R}^x}) - g(\overline{p_L^y}, \overline{p_{1L}^y}, \overline{p_{2L}^y}, \overline{p_{3L}^y}, \overline{p_{1R}^y}, \overline{p_{2R}^y}, \overline{p_{3R}^y}) \tag{26}$$

Let us assume, that the value of $g^{x,y}$ is equal to 0.1161. If the level of confidence is set to $1 - \alpha = 0.95$, then $z_{(1-\alpha)} = 1.644854$ and we have

$$z_{(1-\alpha)} \frac{\sqrt{2Q(K)}}{\sqrt{n}} = 0.11631. \tag{27}$$

Therefore, inequality (23) is not satisfied and we can not say that the attribute a^x is better than a^y. Let us assume now, that new 100 data elements arrived to the considered tree node. The total number of elements n in this node equals 10100. Let us assume that the new value of $g^{x,y}$ is 0.116. For the current value of n we have

$$z_{(1-\alpha)} \frac{\sqrt{2Q(K)}}{\sqrt{n}} = 0.11573. \tag{28}$$

This time inequality (23) is satisfied. Therefore, with the 0.95 level of confidence we are allowed to say that attribute a^x is better than attribute a^y. The considered tree node is divided with respect to attribute a^x.

Example 2

Inequality (23) can be transformed as follows

$$n > \frac{2Q(K)\left(z_{(1-\alpha)}\right)^2}{\left(g^{x,y}\right)^2}, \tag{29}$$

where $g^{x,y}$ is defined as in (26). The above inequality is an alternative way for determining whether to split the considered node or not. If the current number of elements satisfies inequality (29), the node is divided with respect to attribute a^x. Let us assume that $g^{x,y} = 0.116$. The values of $z_{(1-\alpha)} = 1.644854$ and $Q(K) = 25$ are the same as in Example 1. Then the term on the right side of inequality (29) equals

$$\frac{2Q(K)\left(z_{(1-\alpha)}\right)^2}{\left(g^{x,y}\right)^2} = 10053.3. \tag{30}$$

Therefore, if n is greater or equal to 10054 elements, the tree node is divided with respect to attribute a^x.

4. The dsCART Algorithm

Theorem 2 allows us to propose an algorithm called *CART for data streams (dsCART)*. This algorithm is a modification of the Very Fast Decision Tree algorithm proposed in [5]. Following the idea of the authors we introduce the tie breaking mechanism. It forces the split of the considered node after some fixed number of elements, even though the best and the second best attributes do not satisfy condition (23). Without this mechanism, splitting of the node can be blocked permanently if the two best attributes provide comparable values of Gini gain. The number of elements to force the split is the same in all nodes and depends on the tie breaking parameter θ.

For clarity of the pseudocode the following notation will be introduced:

- $\overline{g_q^i}$ is the Gini gain computed for the attribute a^i in the leaf L_q.
- $n_{i,\lambda,q}^k$ is a number of elements from the k-th class in the leaf L_q, for which the value of attribute a^i is equal to a_λ^i, $(a_\lambda^i \in A^i)$.
- n_q^k is a number of elements from the k-th class in the leaf L_q.

Algorithm 1: The dsCART

Inputs: $\mathbf{S}$ is a sequence of examples,
$\mathfrak{A}$ is a set of discrete attributes, of the one class
α is one minus the desired probability of choosing the correct attribute at any given node,
θ is the tie breaking parameter.
Output: $dsCART$ is a decision tree.

Procedure dsCART$(\mathbf{S}, \mathfrak{A}, \alpha)$
Let $dsCART$ be a tree with a single leaf L_0 (the root).
Let $\mathfrak{A}_0 = \mathfrak{A}$
For each attribute $a^i \in \mathfrak{A}$
    For each value a^i_λ of attribute a^i
        For each class k
            $n^k_{i,\lambda,0} = 0$
For each example s in $\mathbf{S}$
    Sort s into a leaf L_q using the current tree.
    For each attribute $a^i \in \mathfrak{A}_q$
        For each value a^i_λ of attribute a^i
            For each class k
                If value of example s for attribute a^i is equal to a^i_λ and s is from the k-th class then
                    Increment $n^k_{i,\lambda,q}$.
    Label L_q with the majority class among the examples seen so far at L_q.
    If the examples seen so far at L_q are not of the same class, then
        For each attribute $a^i \in \mathfrak{A}_l$
            For each partition of the set A^i into A^i_L, A^i_R
                Compute $\overline{g_q}(A^i_L)$ using the counts $n^k_{i,\lambda,q}$.
            $\overline{g^i_q} = \max\limits_{A^i_L \in \mathbf{V}_i} \{\overline{g_q}(A^i_L)\}$
        $a^x = \arg\max\limits_{a^i \in \mathfrak{A}_q} \{\overline{g^i_q}\}$
        $a^y = \arg\max\limits_{a^i \in \mathfrak{A}_q \setminus \{a^y\}} \{\overline{g^i_q}\}$
        Compute $\epsilon_{G,K}$ using formula (24)
        If $(\overline{g^x_q} - \overline{g^y_q} > \epsilon_{G,K})$ or $(\epsilon_{G,K} < \theta)$, then
            Replace L_q by an internal node that splits on a^x.

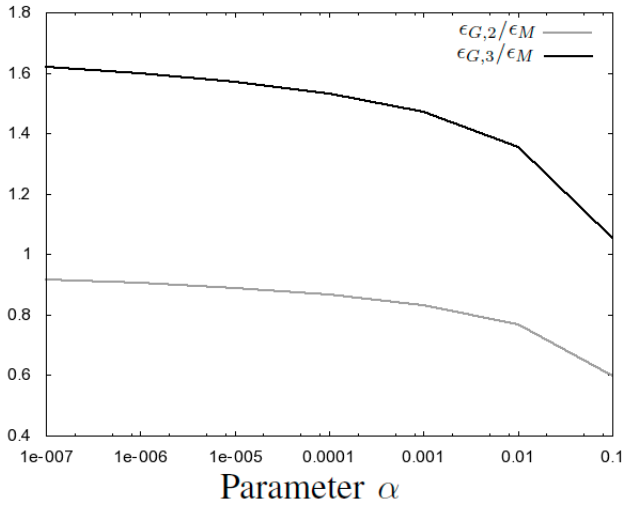

Figure 1: Ratio of $\epsilon_{G,K}$ to ϵ_M for $K = 2$ and $K = 3$.

For both branches of the split
Add a new leaf L_{last+1} and let $\mathfrak{A}_{last+1} = \mathfrak{A}_q \backslash \{a^x\}$ at L_{last+1}.
For each attribute $a^i \in \mathfrak{A}_{last+1}$
For each value a^i_λ of a^i
For each class k
$n^k_{i,\lambda,last+1} = 0$.
$last = last + 1$
Return $dsCART$.

5. Comparison with McDiarmid's bound result

The main difference between the dsCART and the McDT algorithms lies in the method of computing the bound for the difference between the true mean of random variables and the estimate value. The obtained bounds are $\epsilon_{G,K}$ and ϵ_M, given by (24) and (4), respectively.

For $\alpha \in (0; 0.5)$ the following condition is true for all $n > 0$ (see Fig. 1)

$$\begin{aligned} &\epsilon_{G,2} < \epsilon_M, \\ &\epsilon_{G,K} > \epsilon_M, \ K > 2. \end{aligned} \tag{31}$$

This means that, in the two-class problem, the dsCART algorithm needs less data elements to make a split than the McDT algorithm. Figure 1 shows

ratio of $\epsilon_{G,K}$ to ϵ_M for two-class and tree-class problem. One can see that when $\alpha \to 0$ then $\epsilon_{G,2} \to \epsilon_M$. If $\alpha \to 0.5$ then ratio of $\epsilon_{G,2}$ to ϵ_M tends to 0.

Example 3

Let us assume that a^x and a^y are attributes with the highest values of Gini gain. We want to use the McDiarmid's bound to determine if the value of Gini gain for a_x is higher then Gini gain for a^y with $(1-\alpha)$ level of confidence. Then, the difference $g^{x,y}$ (defined by (26)) should satisfy the following inequality (see [33])

$$g^{x,y} > 8\frac{\sqrt{\ln(1/\alpha)}}{\sqrt{2n}}. \tag{32}$$

The above inequality can be transformed to the form

$$n > 32\frac{\ln(1/\alpha)}{\left(g^{x,y}\right)^2}. \tag{33}$$

Hence, if the number of samples n satisfies inequality (33), the considered node in the tree is split with respect to the attribute a^x. Let us assume that $g^{x,y} = 0.224$ and the level of confidence $1-\alpha = 0.95$. Then the term on the right side of inequality (33) equals 1910.544.

Therefore, if the number of data elements n is greater or equal to 1911, the node is split. To compare the McDiarmid's bound with the method presented in this paper we have to know the number of classes K. We will consider two cases: first with $K = 2$ and second with $K = 3$. For $K = 2$, according to inequality (29), the required number n of elements in the node must be greater or equal to 863. In the second case ($K = 3$) n has to be greater or equal to 2697. Obtained results are consistent with those presented in Fig. 1.

6. Experimental results

6.1. Synthetic data

In this section the performance of the proposed method is examined and compared with the McDiarmid Tree [33] and the Gaussian Decision Tree [32] algorithms. Synthetic data were used, generated on a basis of synthetic decision trees. These synthetic trees were constructed in the same way as

described in [5]. At each level of the tree, after the first d_{min} levels, each node is replaced by a leaf with probability ω. To the rest of nodes a splitting attribute is randomly assigned; it has to be an attribute which 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 leave a class is randomly assigned. Each synthetic tree represents a different data concept. Data concept is a particular distribution of attributes values and classes. In this work twelve synthetic trees were generated (all of them with $\omega = 0.15$, $d_{min} = 3$ and $d_{max} = 18$) giving twelve different data concepts. Trees were generated with $D = 30$ binary attributes and $K = 2$ classes. Data elements for each synthetic tree are obtained in the following manner. The value of each attribute is chosen randomly, with equal probability for each possible value. Then the data element is sorted into a leaf using the synthetic tree, according to the values of attributes. The class assigned to this leaf is assigned to the considered data element. The accuracy of the decision tree is calculated every time a new leaf is generated based on new data set of 2000 elements. These testing sets are created the same way as training data set. In the following simulations, for any set of dsCART parameters (α, n), algorithm was run twelve times, once for each synthetic data concept. The final result was obtained as the average over all runs.

The first experiment examines the dependence between the accuracy of the algorithm and the size of the built tree. Data generated by various synthetic trees produced classification trees having different complexities. The number of leaves in the least complex tree was equal to 2971, therefore the accuracy is compared in the interval $[0, 2971]$. The size of dataset was $n = 10^9$, the value of parameter α was set to 10^{-5} and the value of parameter θ was 0.05. As it was expected the accuracy increases with the growth of the number of leaves (see Fig. 2). Therefore it is important to determine the best attribute to make a split using as few data as possible.

In the second experiment we compare the accuracy of the dsCART algorithm, the GDT algorithm and the McDT algorithm. For this simulation the value of parameter θ was set to 0.05 and the value of parameter α was 10^{-7}. The experiment was performed on different number of training data elements, from $n = 10^4$ to $n = 10^9$. As we can see in Fig. 3, the accuracy of these algorithms differs very little. We also calculated the corresponding values of standard deviation, which for clarity are collected in table 1.

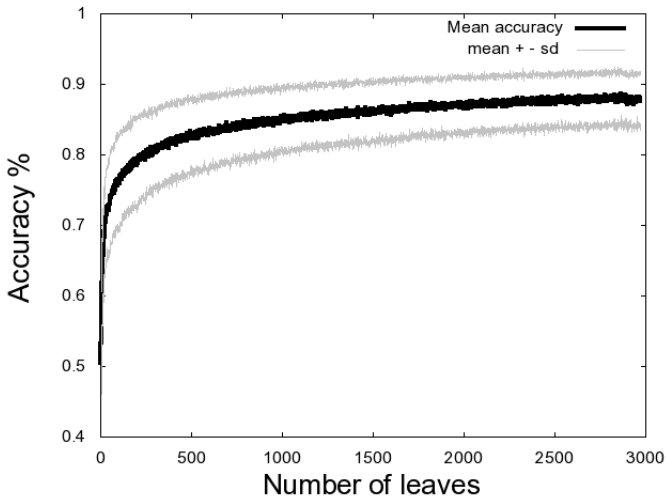

Figure 2: The dependence between the accuracy of the dsCART algorithm and the number of leaves (black line - mean value obtained for twelve trees, gray line - mean value ± standard deviation (sd)).

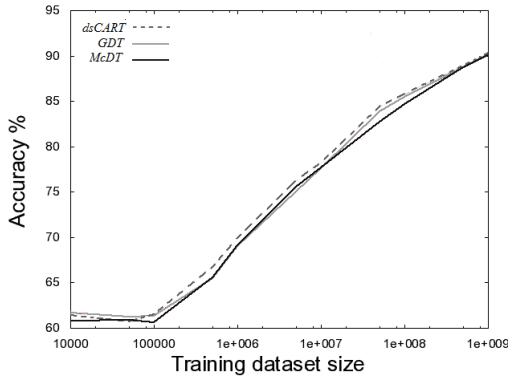

Figure 3: The dependence between the number of training data and the accuracy of the dsCART, the Gaussian Decision Tree and the McDiarmid Tree algorithms.

Table 1: Standard deviation of accuracy for different number of elements N and various classifiers.

N	10^4	10^5	10^6	10^7	10^8	10^9
dsCART	5,377	5,94	7,016	5,629	3,805	2,321
GDT	5,069	5,587	6,92	3,787	3,661	2,449
McDT	5,571	5,125	5,931	5,061	3,845	2,519

According to these values one can say that the three algorithms (McDT, GDT and dsCART) demonstrate comparable accuracies. This result comes from the fact that all three algorithms are based on the same mechanism of tree construction. Hence the corresponding nodes are split with respect to the same attribute with probability $1 - \alpha$. Therefore, with very high probability all three algorithms produce the same decision trees on a given data stream. The difference is only in the rate of splitting the nodes. For considered algorithms the accuracy is increasing with growing number of training data elements. For $n = 10^9$ the obtained accuracy was greater than 90%.

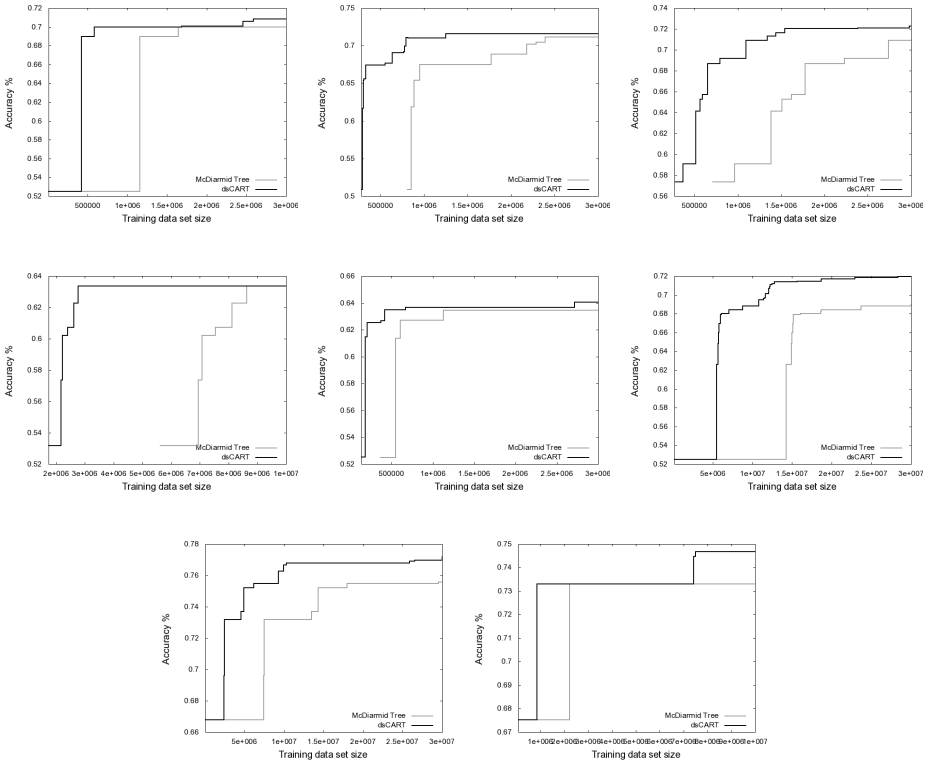

Figure 4: The dependence of the accuracies of the dsCART and the McDiarmid Tree algorithms on the number of elements obtained for 8 various concepts.

The third experiment was performed to compare the accuracy of dsCART

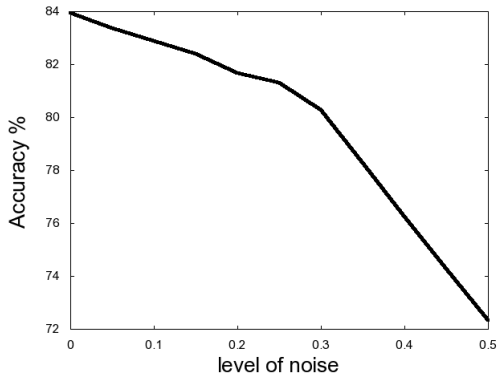

Figure 5: The dependence between the accuracy of the dsCART algorithm and the level of noise.

algorithm with McDiarmid Tree algorithm. The value of parameters α and θ was set to 0.1 and 0 respectively. In Fig. 4 we can see the results obtained for 8 various concepts. It shows that dsCART algorithm needs fewer data elements to make a split than McDiarmid Tree algorithm for every concept. The trees obtained by both algorithms are similar, therefore the final accuracy tends to the same value. The main advantage of dsCART is that a split is made based on fewer data elements what is especially important in some special cases. Particularly to deal with concept drift we can limit the maximal number of data elements considered in the tree (like in CVFDT [17]). Since the dsCART algorithm creates more complex trees than McDiarmid Tree algorithm, the former should provide higher accuracy. The most important split is generally a split of the root because it ensures the highest gain of accuracy. As one can see in Fig. 4 the dsCart algorithm always performed this first split faster than McDiarmid Tree algorithm.

In the next experiment we examined noisy data. The following mechanism was used to create the noisy data. Every time the data were generated, the value of each attribute and the class was changed with the probability φ to any possible value with the same probability. The value of φ vary from 0% to 50%. In Figure 5 we can see that with the growth of the value of noise the accuracy decreases. However, in the investigated range the accuracy decreases not more than 12%.

Figure 6 shows the difference between the processing time of the GDT algorithm and the dsCART algorithm according to the size of training dataset n. As we can see the dsCART algorithm is much faster than the GDT

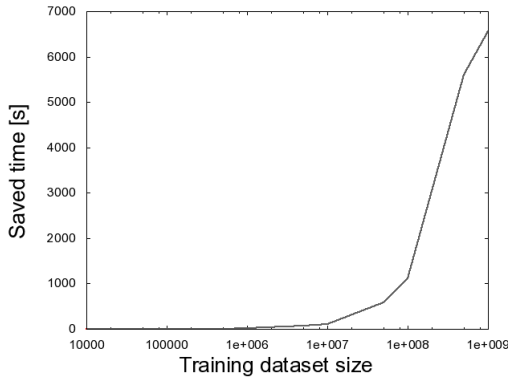

Figure 6: The difference of the processing time of the Gauss Decision Tree algorithm and the dsCART algorithm depending on the size of training dataset n.

algorithm. For $n = 10^4$ the difference was equal to $0.16s$ and for $n = 10^9$ it increased up to $6563s$.

6.2. Real data

The performance of the dsCART algorithm was also examined on real data. We decided to choose a dataset from the UCI repository [9]. Although there is a variety of different datasets available in the repository, only several of them can imitate a data stream for our purposes. The most suitable seems to be the 'KDD CUP 99' dataset, consisting of 4898431 data elements. Data are described by 41 attributes, 7 of which are nominal and 34 are numerical. To adapt the node splitting procedure to the numerical attributes we applied the standard method of dividing the range of the attribute into bins. In the experiment we set the number of bins to 20 for all numerical attributes. Each data element belongs to one of the five classes. Classes represent four types of network attacks ('dos', 'u2r', 'r2l' and 'probe') and the fifth class is reserved for 'normal' network connections (without attack). We performed two simulations: first for the original five-class problem, and the second for the two-class problem, in which all four types of network attacks are merged into one class labeled as 'attack'. The whole dataset was divided randomly into two parts: the training set, consisting of 4896431 elements, and the testing set, consisting of 2000 elements. The parameter α was set to 0.00001 and the tie breaking mechanism was turned off ($\theta = 0$). Obtained results are presented in Fig. 7.

As we can see we had to deal with very specific data. The fractions of

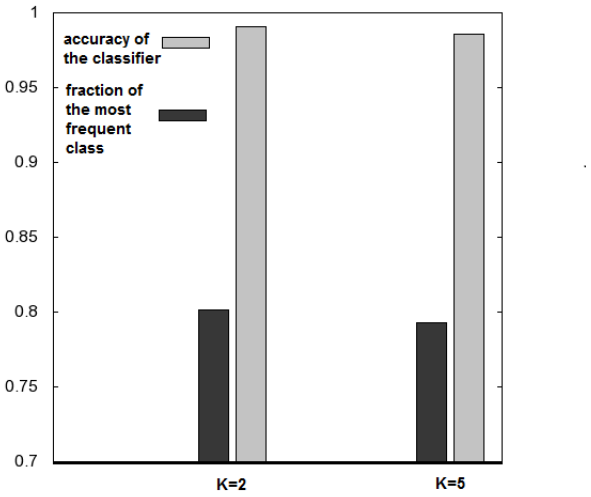

Figure 7: The fraction of the most frequent class and the obtained accuracy of the classifier, for two-class problem ($K = 2$) and five-class problem ($K = 5$).

the most frequent class were equal to 0.801 for $K = 2$ and 0.799 for $K = 5$. It shows that the dataset is strongly unbalanced. However, the obtained accuracies (99.1% for $K = 2$ and 98.6% for $K = 5$) seem to be satisfactory.

7. Conclusions

In this paper we discussed the problem of creating a decision tree for data stream classification. We propose a new method of deciding, if the best attribute to split the considered node obtained according to a finite data sample is also the best attribute for the whole data stream. We propose a modification of the CART algorithm called the dsCART algorithm. To show the mathematical foundations of this algorithm we use the properties of the normal distribution and the Taylor's Theorem. In the experimental results we show that this new algorithm is accurate and fast at the same time. Therefore we proved that it is proper tool for solving the problem of data stream classification.

Acknowledgments

This paper was prepared under project operated within the Foundation for Polish Science Team Programme co-financed by the EU European Re-

gional Development Fund, Operational Program Innovative Economy 2007-2013, and also supported by National Science Center NCN.

The authors would like to thank the reviewers for helpful comments.

Appendix A.

In order to simplify a description, the following notations are introduced:

$$g^i = g(\mu_L^i, \mu_{1L}^i, \ldots, \mu_{(K-1)L}^i, \mu_{1R}^i, \ldots, \mu_{(K-1)R}^i), \tag{A.1}$$

$$\left(\tau_L^i\right)^2 = \left(\frac{\partial g}{\partial p_L}\right)^2 \left(\sigma_L^i\right)^2, \tag{A.2}$$

$$\left(\tau_{jL}^i\right)^2 = \left(\frac{\partial g}{\partial p_{jL}}\right)^2 \left(\sigma_{jL}^i\right)^2,\ j \in \{1, \ldots, K-1\}, \tag{A.3}$$

$$\tau_{jkL}^i = \frac{\partial g}{\partial p_{jL}} \frac{\partial g}{\partial p_{kL}} cov_{jkL}^i, \qquad j, k \in \{1, \ldots, K-1\},\ k \neq j, \tag{A.4}$$

$$\left(\tau_{jR}^i\right)^2 = \left(\frac{\partial g}{\partial p_{jR}}\right)^2 \left(\sigma_{jR}^i\right)^2,\ j \in \{1, \ldots, K-1\}, \tag{A.5}$$

$$\tau_{jkR}^i = \ \frac{\partial g}{\partial p_{jR}} \frac{\partial g}{\partial p_{kR}} cov_{jkR}^i,\ j, k \in \{1, \ldots, K-1\},\ k \neq j. \tag{A.6}$$

where $cov_{jkL}^i = -p_{jL}^i p_{kL}^i$ is the covariance of $\zeta_{jL}^{i,m}$ and $\zeta_{kL}^{i,m}$, and $cov_{jkR}^i = -p_{jR}^i p_{kR}^i$ is the covariance of $\zeta_{jR}^{i,m}$ and $\zeta_{kR}^{i,m}$.
To prove the Theorem 1 we will introduce and prove the following lemma

Lemma 1.
If function $g(p_L, p_{1L}, \ldots, p_{(K-1)L}, p_{1R}, \ldots, p_{(K-1)R})$ is given by formula (19), then the value $g(\overline{p_L^i}, \overline{p_{1L}^i}, \ldots, \overline{p_{(K-1)L}^i}, \overline{p_{1R}^i}, \ldots, \overline{p_{(K-1)R}^i})$ can be approximated by the normal distribution

$$g(\overline{p_L^i}, \overline{p_{1L}^i}, \ldots, \overline{p_{(K-1)L}^i}, \overline{p_{1R}^i}, \ldots, \overline{p_{(K-1)R}^i}) \longrightarrow \tag{A.7}$$
$$N\Big(g^i, \frac{(\tau_L^i)^2}{n} + \sum_{j=1}^{K-1} \frac{(\tau_{jL}^i)^2}{n_L^i} + \sum_{j=1}^{K-1} \sum_{k=1, k\neq j}^{K-1} \frac{\tau_{jkL}^i}{n_L^i} + \sum_{j=1}^{K-1} \frac{(\tau_{jR}^i)^2}{n_R^i} + \sum_{j=1}^{K-1} \sum_{k=1, k\neq j}^{K-1} \frac{\tau_{jkR}^i}{n_R^i}\Big),$$

Proof. Parameters p_L^i, p_{kL}^i and p_{kR}^i are estimated from the data sample in considered leaf node as

$$\overline{p_L^i} = \frac{\sum_{m=1}^{n} \zeta_L^{i,m}}{n}, \tag{A.8}$$

$$\overline{p_{kL}^i} = \frac{\sum_{j=m}^{n_L^i} \zeta_{kL}^{i,m}}{n_L^i}, k \in \{1, \ldots, K-1\} \tag{A.9}$$

$$\overline{p_{kR}^i} = \frac{\sum_{j=m}^{n_R^i} \zeta_{kR}^{i,m}}{n_R^i}, k \in \{1, \ldots, K-1\} \tag{A.10}$$

For big values of n, n_L^i and n_R^i distributions of $\overline{p_L^i}$, $\overline{p_{kL}^i}$ and $\overline{p_{kR}^i}$, according to the Central Limit Theorem, can be approximated by appropriate normal distributions

$$\overline{p_L^i} \longrightarrow N\left(\mu_L^i, \frac{(\sigma_L^i)^2}{n}\right), \tag{A.11}$$

$$\overline{p_{kL}^i} \longrightarrow N\left(\mu_{kL}^i, \frac{(\sigma_{kL}^i)^2}{n_L^i}\right),\ k \in \{1, \ldots, K-1\}, \tag{A.12}$$

$$\overline{p_{kR}^i} \longrightarrow N\left(\mu_{kR}^i, \frac{(\sigma_{kR}^i)^2}{n_R^i}\right),\ k \in \{1, \ldots, K-1\}. \tag{A.13}$$

For big values of n, n_L^i and n_R^i one can assume that the values of $\overline{p_L^i}$, $\overline{p_{kL}^i}$ and $\overline{p_{kR}^i}$ are very close to their expected values μ_L^i, μ_{kL}^i and μ_{kR}^i. Therefore one can apply the Taylor's Theorem for Gini gain in neighborhood of point $(\overline{p_L^i}, \overline{p_{1L}^i}, \ldots, \overline{p_{(K-1)L}^i}, \overline{p_{1R}^i}, \ldots, \overline{p_{(K-1)R}^i}) = (\mu_L^i, \mu_{1L}^i, \ldots, \mu_{(K-1)L}^i, \mu_{1R}^i, \ldots, \mu_{1R}^i)$, which is given by formula (A.14).

$$\begin{aligned}
&g(\overline{p_L^i}, \overline{p_{1L}^i}, \ldots, \overline{p_{(K-1)L}^i}, \overline{p_{1R}^i}, \ldots, \overline{p_{(K-1)R}^i}) \approx \\
&g(\mu_L^i, \mu_{1L}^i, \ldots, \mu_{(K-1)L}^i, \mu_{1R}^i, \ldots, \mu_{(K-1)R}^i) \\
&+\frac{\partial g(\mu_L^i, \mu_{1L}^i, \ldots, \mu_{(K-1)L}^i, \mu_{1R}^i, \ldots, \mu_{(K-1)R}^i)}{\partial p_L}(\overline{p_L^i} - \mu_L^i) \\
&+\sum_{j=1}^{K-1} \frac{\partial g(\mu_L^i, \mu_{1L}^i, \ldots, \mu_{(K-1)L}^i, \mu_{1R}^i, \ldots, \mu_{(K-1)R}^i)}{\partial p_{jL}}(\overline{p_{jL}^i} - \mu_{jL}^i) \\
&+\sum_{j=1}^{K-1} \frac{\partial g(\mu_L^i, \mu_{1L}^i, \ldots, \mu_{(K-1)L}^i, \mu_{1R}^i, \ldots, \mu_{(K-1)R}^i)}{\partial p_{jR}}(\overline{p_{jR}^i} - \mu_{jR}^i).
\end{aligned} \tag{A.14}$$

Therefore, according to (A.11) - (A.14), the distribution of the random variable $g(\overline{p_L^i}, \overline{p_{1L}^i}, \ldots, \overline{p_{(K-1)L}^i}, \overline{p_{1R}^i}, \ldots, \overline{p_{(K-1)R}^i})$ can be approximated by (A.7). This completes the proof. □

Observing that $n_L^i = n\overline{p_L}$ and $n_R^i = n(1-\overline{p_L})$, the variance of normal distribution (A.7) can be further simplified as follows

$$\frac{\left(\tau_L^i\right)^2}{n} + \sum_{j=1}^{K-1} \frac{\left(\tau_{jL}^i\right)^2}{n_L^i} + \sum_{j=1}^{K-1} \sum_{k=1,k\neq j}^{K-1} \frac{\tau_{jkL}^i}{n_L^i} + \sum_{j=1}^{K-1} \frac{\left(\tau_{jR}^i\right)^2}{n_R^i} + \sum_{j=1}^{K-1} \sum_{k=1,k\neq j}^{K-1} \frac{\tau_{jkR}^i}{n_R^i} =$$

$$\frac{\left(\tau_L^i\right)^2 + \sum_{j=1}^{K-1} \frac{\left(\tau_{jL}^i\right)^2}{\overline{p_L^i}} + \sum_{j=1}^{K-1} \sum_{k=1,k\neq j}^{K-1} \frac{\tau_{jkL}^i}{\overline{p_L^i}} + \sum_{j=1}^{K-1} \frac{\left(\tau_{jR}^i\right)^2}{1-\overline{p_L^i}} + \sum_{j=1}^{K-1} \sum_{k=1,k\neq j}^{K-1} \frac{\tau_{jkR}^i}{1-\overline{p_L^i}}}{n}$$

$$= \frac{\left(\tau^i\right)^2}{n}. \tag{A.15}$$

Distribution (A.7) is then given by

$$g(\overline{p_L^i}, \overline{p_{1L}^i}, \ldots, \overline{p_{(K-1)L}^i}, \overline{p_{1R}^i}, \ldots, \overline{p_{(K-1)R}^i}) \longrightarrow N\left(g^i, \frac{\left(\tau^i\right)^2}{n}\right). \tag{A.16}$$

Now we will introduce the proof of the Theorem 1.

Proof. The difference of values $\overline{g^x} \equiv g(\overline{p_L^x}, \overline{p_{1L}^x}, \ldots, \overline{p_{(K-1)L}^x}, \overline{p_{1R}^x}, \ldots, \overline{p_{(K-1)R}^x})$ and $\overline{g^y} \equiv g(\overline{p_L^y}, \overline{p_{1L}^y}, \ldots, \overline{p_{(K-1)L}^y}, \overline{p_{1R}^y}, \ldots, \overline{p_{(K-1)R}^y})$ has the normal distribution

$$\overline{g^x} - \overline{g^y} \longrightarrow N\left(g^x - g^y, \frac{\left(\tau^x\right)^2 + \left(\tau^y\right)^2}{n}\right). \tag{A.17}$$

We do not know the true value of the mean $g^x - g^y$. Properties of the normal distribution [19] ensure that the following inequality is satisfied with probability $1-\alpha$

$$\overline{g^x} - \overline{g^y} < \epsilon_{(1-\alpha)}^{x,y} + (g^x - g^y), \tag{A.18}$$

where

$$\epsilon_{(1-\alpha)}^{x,y} = z_{(1-\alpha)} \frac{\sqrt{\left(\tau^x\right)^2 + \left(\tau^y\right)^2}}{\sqrt{n}}. \tag{A.19}$$

Obviously, inequality (A.18) is equivalent to the following one

$$g^x - g^y > (\overline{g^x} - \overline{g^y}) - \epsilon^{x,y}_{(1-\alpha)}. \tag{A.20}$$

It means that the attribute a^x is the better than a^y according to the whole stream, i.e.

$$g^x - g^y > 0, \tag{A.21}$$

with probability $1 - \alpha$, only if the following inequality is satisfied

$$\epsilon^{x,y}_{(1-\alpha)} < \overline{g^x} - \overline{g^y}. \tag{A.22}$$

Moreover, if we would show, that the $\epsilon_{G,K}$ given by (24) is the upper bound of $\epsilon^{x,y}_{(1-\alpha)}$, i.e.

$$\epsilon^{x,y}_{(1-\alpha)} < \epsilon_{G,K} \tag{A.23}$$

then the following condition is enough to satisfy inequality (A.21) with probability $1 - \alpha$

$$\overline{g^x} - \overline{g^y} > \epsilon_{G,K}. \tag{A.24}$$

Then one can say that if inequality (A.24) is true, then $g^x > g^y$ with probability $1 - \alpha$. Now inequality (A.23) will be proved.

Partial derivatives of Gini index (18) are given by

$$\frac{\partial Gini(P_1, \ldots, P_{K-1})}{\partial P_i} = -2P_i + 2\left(1 - \sum_{i=1}^{K-1} P_j\right) = 2(P_K - P_i). \tag{A.25}$$

Observing that $\frac{\partial p_i}{\partial p_{iL}} = p_L$ and $\frac{\partial p_i}{\partial p_{iR}} = (1 - p_L)$, we have

$$\begin{aligned} \frac{\partial g}{\partial p_{iL}} &= \frac{\partial Gini(p_1,\ldots,p_{K-1})}{\partial p_i}\frac{\partial p_i}{\partial p_{iL}} - p_L\frac{\partial Gini(p_{1L},\ldots,p_{(K-1)L})}{\partial p_{iL}} \\ &= 2p_L(p_K - p_i - p_{KL} + p_{iL}) \leq 4p_L \end{aligned} \tag{A.26}$$

and

$$\begin{aligned}\frac{\partial g}{\partial p_{iR}} &= \frac{\partial Gini(p_1,\ldots,p_{K-1})}{\partial p_i}\frac{\partial p_i}{\partial p_{iR}} - (1-p_L)\frac{\partial Gini(p_{1R},\ldots,p_{(K-1)R})}{\partial p_{iR}} \\ &= 2(1-p_L)(p_K - p_i - p_{KR} + p_{iR}) \leq 4(1-p_L),\end{aligned} \tag{A.27}$$

Those inequalities are satisfied for every attribute. Therefore, for any chosen attribute a^i, we obtain the following bounds:

$$\frac{(\tau^i_{jL})^2}{p^i_L} = \frac{\left(\frac{\partial g}{\partial p^i_{jL}}\right)^2 p^i_{jL}(1-p^i_{jL})}{p^i_L} \leq 16 p^i_L \frac{1}{4} = 4p^i_L, \tag{A.28}$$

$$\frac{(\tau^i_{jR})^2}{p^i_R} = \frac{\left(\frac{\partial g}{\partial p^i_{jR}}\right)^2 p^i_{jR}(1-p^i_{jR})}{(1-p^i_L)} \leq 16(1-p^i_L)\frac{1}{4} = 4(1-p^i_L), \tag{A.29}$$

$$\frac{(\tau^i_{jkL})^2}{p^i_L} = -\frac{\frac{\partial g}{\partial p^i_{jL}}\frac{\partial g}{\partial p^i_{kL}} p^i_{jL} p^i_{kL}}{p^i_L} \leq 16 p^i_L \frac{1}{4} = 4p^i_L, \tag{A.30}$$

$$\frac{(\tau^i_{jkR})^2}{p^i_R} = -\frac{\frac{\partial g}{\partial p^i_{jR}}\frac{\partial g}{\partial p^i_{kR}} p^i_{jR} p^i_{kR}}{(1-p^i_L)} \leq 16(1-p^i_L)\frac{1}{4} = 4(1-p^i_L), \tag{A.31}$$

Since that $\frac{\partial p^i_j}{\partial p^i_L} = p^i_{jL} - p^i_{jR}$, the derivative of g with respect to p^i_L is given by

$$\begin{aligned}&\frac{\partial g}{\partial p^i_L} = \sum_{j=1}^{K-1} \frac{\partial Gini}{\partial p^i_j}\frac{\partial p^i_j}{\partial p^i_L} - Gini(p^i_{1L},\ldots,p^i_{(K-1)L}) + Gini(p^i_{1R},\ldots,p^i_{(K-1)R}) \\ &= \sum_{j=1}^{K-1} 2(p^i_K - p^i_j)(p^i_{jL} - p^i_{jR}) - Gini(p^i_{1L},\ldots,p^i_{(K-1)L}) + Gini(p^i_{1R},\ldots,p^i_{(K-1)R}) \\ &\leq 2(K-1) - Gini(p^i_{1L},\ldots,p^i_{(K-1)L}) + Gini(p^i_{1R},\ldots,p^i_{(K-1)R}).\end{aligned} \tag{A.32}$$

Since the Gini index takes values in the interval $[0;1)$, the following bound is true

$$(\tau_L^i)^2 = \left(\frac{\partial g}{\partial p_L^i}\right)^2 p_L^i(1-p_L^i) \leq (2(K-1)+1)^2\frac{1}{4} < K^2. \qquad \text{(A.33)}$$

Finally, according to bounds (A.28) - (A.31) and (A.33), the following bound for $(\tau^i)^2$ (defined in (A.15)) is satisfied

$$\begin{aligned}(\tau^i)^2 \leq K^2 + (K-1)4p_L + (K-1)(K-2)4p_L + (K-1)4(1-p_L) \\ +(K-1)(K-2)4(1-p_L) \\ = K^2 + 4(K-1) + 4(K-1)(K-2) = 5K^2 - 8K + 4 = Q(K). \qquad \text{(A.34)}\end{aligned}$$

The inequality (A.34) holds for any attribute a^i, in particular for $a^i = a^x$ and $a^i = a^y$. Therefore, back to formula (A.19), the pessimistic value of $\epsilon_{1-\alpha}^{x,y}$ can be expressed in the form

$$\epsilon_{1-\alpha}^{x,y} = z_{1-\alpha}\frac{\sqrt{2Q(K)}}{\sqrt{n}}. \qquad \text{(A.35)}$$

Therefore, in general case $\epsilon_{1-\alpha}^{x,y} \leq \epsilon_{G,K}$. □

References

[1] C. Aggarwal, Data Streams. Models and Algorithms, Springer, LLC, New York, 2007.

[2] A. Bifet, R. Kirkby, Data Stream Mining a Practical Approach, Technical Report, University of WAIKATO, 2009.

[3] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Chapman and Hall, New York, 1993.

[4] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Transactions on Information Theory 13 (1967) 2127.

[5] P. Domingos, G. Hulten, Mining high-speed data streams, in: Proc. 6th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining, pp. 71–80.

[6] S. Ertekin, L. Bottou, C.L. Giles, Nonconvex online support vector machines, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (2011) 368–381.

[7] W. Fan, Y. Huang, H. Wang, P. Yu, Active mining of data streams, in: Proc. SDM.

[8] W. Fan, Y. Huang, P. Yu, Decision tree evolution using limited number of labeled data items from drifting data streams, in: Proc. 4th IEEE Internat. Conf. on Data Mining, pp. 379–382.

[9] A. Frank, A. Asuncion, UCI machine learning repository, 2010. URL: `http://archive.ics.uci.edu/ml`.

[10] C. Franke, Adaptivity in Data Stream Mining, Ph.D. thesis, University of California, DAVIS, 2009.

[11] M. Gaber, A. Zaslavsky, S. Krishnaswamy, Mining data streams: A review, Sigmod Record 34 (2005) 18–26.

[12] J. Gamaa, R. Fernandes, R. Rocha, Decision trees for mining data streams, Intelligent Data Analysis 10 (2006) 23–45.

[13] J. Gao, W. Fan, J. Hang, On appropriate assumptions to mine data streams: Analysis and practice, in: 2007 IEEE International Conference on Data Mining (ICDM'07), Omaha, NE.

[14] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Elsevier, 2006.

[15] H. He, S. Chen, K. Li, X. Xu, Incremental learning from stream data, IEEE Transactions on Neural Networks (2011) 1901–1914.

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

[17] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proc. 7th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining, KDD'01, pp. 97–106.

[18] R. Jin, G. Agrawal, Efficient decision tree construction on streaming data, in: Proc. 9th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24-27, 2003.

[19] O. Kardaun, Classical Methods of Statistics, Springer, 2005.

[20] L.I. Kuncheva, Classifier ensembles for changing environments, in: Multiple Classifier Systems, Springer, 2004, pp. 1–15.

[21] D. Larose, Discovering Knowledge in Data. An Introduction to Data Mining, Wiley & Sons, 2005.

[22] J. Liu, X. Li, W. Hong, Ambiguous decision trees for mining concept-drifting data streams, Pattern Recognition Letters, Elsevier 30 (2009) 1347–1355.

[23] C. McDiarmid, On the method of bounded differences, On the method of bounded differences, number 141 in Surveys in Combinatorics, Math. Soc. Lecture, London, 1989, pp. 148–188.

[24] B. Pfahringer, G. Holmes, R. Kirkby, New options for Hoeffding trees, in: M.A.Orgun, J. Thornton (Eds.), AI 2007, number 4830 in LNAI, Springer, 2007, pp. 90–99.

[25] R. Polikar, L. Udpa, S. Udpa, V. Honavar, Learn++: An incremental learning algorithm for supervised neural networks, IEEE Transactions on System, Man and Cybernetics (C), Special Issue on Knowledge Management 31 (2001) 497–508.

[26] J. Quinlan, Learning efficient classification procedures and their application to chess end games, Learning efficient classification procedures and their application to chess end games, Morgan Kaufmann, San Francisco, CA, 1983, pp. 463–482.

[27] J. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, San Francisco, CA, 1993.

[28] J. Rice, Mathematical Statistics and Data Analysis, Duxbury Press, 2007.

[29] R. Rojas, Neural Networks: A Systematic Introduction, Springer, Berlin, 1996.

[30] L. Rutkowski, Adaptive probabilistic neural-networks for pattern classification in time-varying environment, IEEE Trans. Neural Networks 15 (2004) 811–827.

[31] L. Rutkowski, New Soft Computing Techniques for System Modeling, Pattern Classification and Image Processing, Springer-Verlag, 2004.

[32] L. Rutkowski, L. Pietruczuk, P. Duda, M. Jaworski, Decision trees for mining data streams based on the Gaussian approximation, IEEE Transactions on Knowledge and Data Engineering PP (2013).

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

[34] A. Tsymbal, The problem of concept drift: definitions and related work, Technical Report TCD-CS-2004-15, Computer Science Department, Trinity College Dublin, Ireland, 2004.

[35] I. Witten, E. Frank, G. Holmes, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufman, Amsterdam, Boston, 2005.