Stream data – state of the art and new results

Leszek Rutkowski
e–mail: *lrutko@kik.pcz.czest.pl*
Cooperation with Ph. D. Students:
Piotr Duda, Maciej Jaworski, Lena Pietruczuk,

Czestochowa University of Technology,
Department of Computer Engineering, Poland

Table of contents

Czym są dane strumieniowe?

strumień danych to potencjalnie nieskończony ciąg przychodzących danych

Cechy danych strumieniowych:

- Dane docierają do systemu nieustannie, teoretycznie ich rozmiar jest nieskończony
- Algorytm działa na paczkach danych lub po każdej danej (rekurencyjnie)
- Algorytm musi analizować daną (paczkę) na tyle szybko, aby zdążyć przed przybyciem kolejnej danej (paczki)
- Nie ma miejsca w pamięci na przechowywanie danych historycznych
- Informacja z przeszłości przechowywana w postaci *przybliżonych struktur podsumowujących*
- Rozkład danych może ulegać zmianie w czasie (concept drift)

Główne problemy jakie niesie ze sobą badanie danych strumieniowych to:

- ograniczenie pamięci
- możliwość wyłącznie jednorazowego odczytu danych
- duża szybkość przychodzenia danych
- zjawisko zwane „concept drift”

Co oznacza concept drift?

Concept drift jest zjawiskiem oznaczającym zmianę rozkładu danych, ich charakteru lub znaczenia w czasie. Może ono następować powoli albo zmiana taka może nastąpić nagle w pewnym momencie w czasie.

Przykłady strumieni danych:

-*monitorowanie sieci*
-*dane pochodzące z sensorów*
-*ruch uliczny*
-*informacje o transakcjach płatniczych*
-*sygnały radiowe pochodzące z przestrzeni kosmicznej*
-*rozmowy telefoniczne*

Review of the literature

1. Oza N. C., „*Online Ensemble Learning*”, Ph.D. Thesis, The University of California, Berkeley, CA, (2001)

Review of the literature

2. Tsymbal A., „*The Problem of Concept Drift: Definitions and Related Work*”, Techn. Report of Dept. of Comp. Sci, Trinity College Dublin, Ireland, (2004)

Review of the literature

3. Han J. and Kamber M., „*Data Mining: Concepts and Techniques*” The Morgan Kaufmann Series in Data Management Systems Series Elsevier Science \& Tech, (2006)

Review of the literature

4. Aggraval C. C., „*Data Streams: Models and Algorithms*”, Springer, (2007)

Review of the literature

5. Kirkby R., „*Improving Hoeffding trees*”, PhD thesis,. University of Waikato, November, (2007)

Review of the literature

6. Bifet A., and Kirkby R., „*Data Stream Mining - A Practical approach”*, University of WAIKATO, New Zeland, (2009)

Review of the literature

7. Franke C. „*Adaptivity in Data Stream Mining”,* PhD thesis, University of California at Davis, (2009)

Review of the literature

8. Bifet A., „*Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams*”, Frontiers in Artificial Intelligence and Applications Frontiers in Artificial Intelligence and Applications IOS Press, (2010)

Review of the literature

9. Brzeziński D., „*Mining Data Streams with Concept Drifts*”, Master's thesis supervised by J. Stefanowski, Poznan University of Technology, Poznan, Poland, (2010)

Review of the literature

10. Žliobaitė I., „*Adaptive Training Set Formation*”, Vilnius University, Lithuania, (2010)

DECISION TREES

for MINING DATA STREAMS

based on

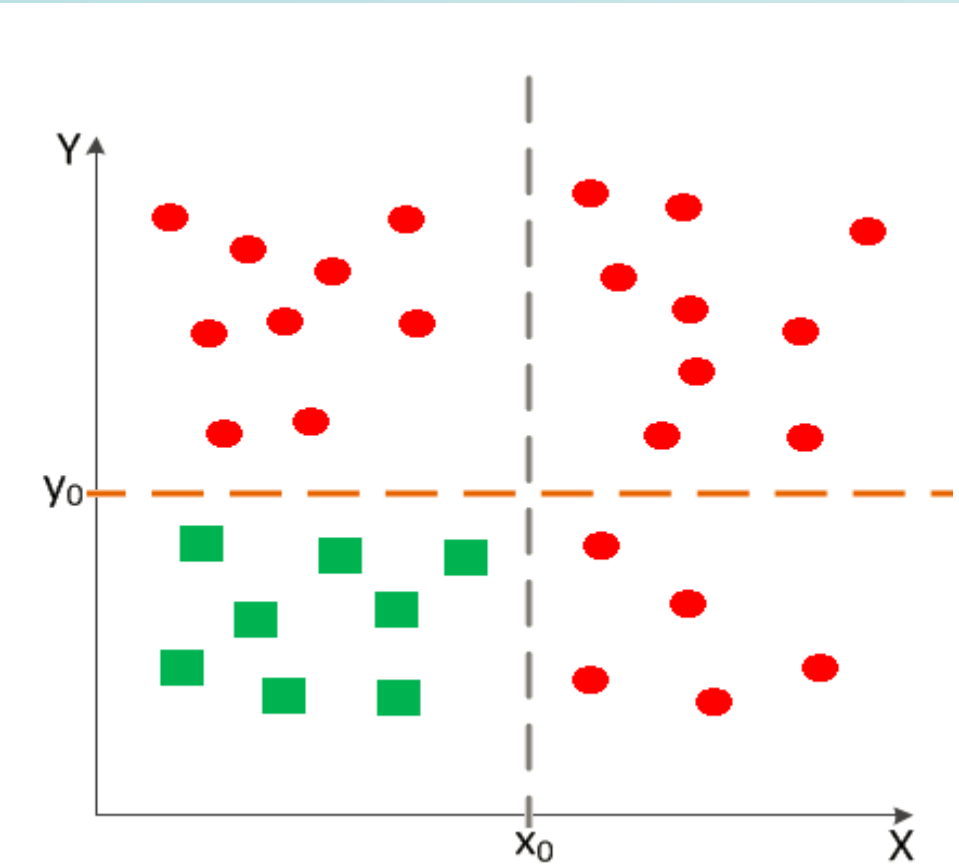
Leszek Rutkowski, Lena Pietruczuk, Piotr Duda, Maciej Jaworski, *Decision Trees for Mining Data Streams Based on the McDiarmid's Bound*, IEEE Transactions on Knowledge and Data Engineering, vol. 25, no. 6, pp. 1272–1279, 2013.

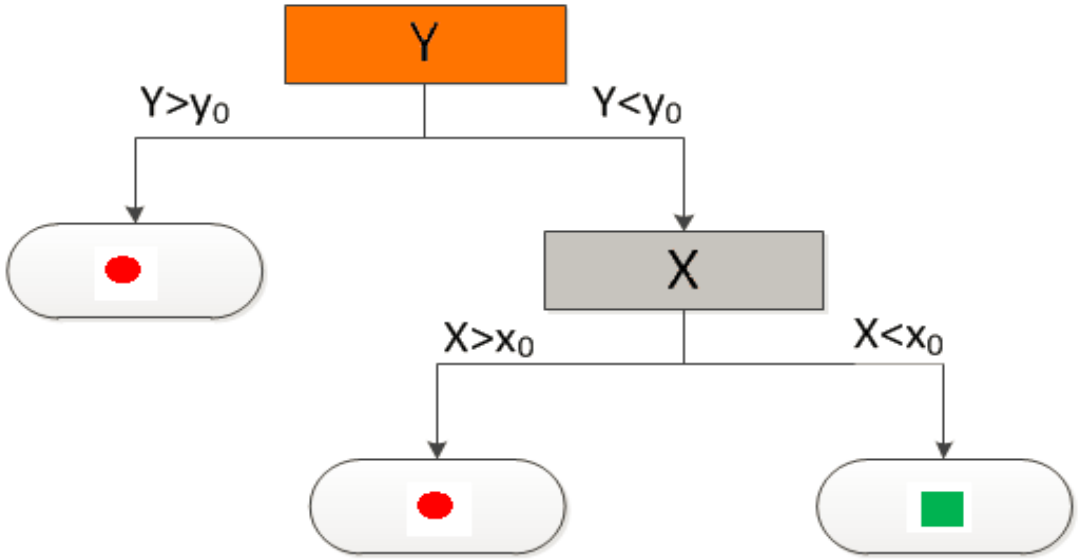
Leszek Rutkowski, Maciej Jaworski, Lena Pietruczuk, Piotr Duda, *Decision Trees for Mining Data Streams Based on the Gaussian Approximation*, IEEE Transactions on Knowledge and Data Engineering, vol. 99, no. to be published, 2013.

Leszek Rutkowski, Maciej Jaworski, Lena Pietruczuk, Piotr Duda, *The CART Decision Tree for Mining Data Streams*, submitted for publication.

Leszek Rutkowski, Maciej Jaworski, Lena Pietruczuk, Piotr Duda, *A new method for data stream mining based on the misclassification error*, submitted for publication.

DECISION TREES





The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree, according to the first task in step 4 of the CLS algorithm. We would like to select the attribute that is most useful for classifying examples. What is a good quantitative measure of the worth of an attribute?

We will define a statistical property called *information gain* that measures how well a given attribute separates the training examples according to their target classification. ID3 uses this information gain measure to select among the candidate attributes at each step while growing the tree.

Entropy

The entropy of *S* is defined as follows

$$\text{Ent}(S) = -\sum_{i=1}^{M} p_i \log(p_i),$$

S – training set

M – number of classes

p_i - probability that element belongs to class i
(proportion of *S* belonging to class i)

When all data a set belong to a single class, there is no uncertainty and the entropy is zero. If the target attribute can take on M possible values, the entropy can be as large as $log_2 M$.

Given entropy as a measure of the impurity in a collection of training examples, we can now define a measure of the effectiveness of an attribute in classifying the training data. The measure we will use, called, *information gain*, is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.

Information gain

$$H(S|a) = Ent(S) - \sum_{i=1}^{N} w_i Ent(S_i),$$

where

a – an attribute with values in set $\{a_1, \ldots, a_N\}$

N – number of different values of attribute a

w_i - fraction of number of elements with value a_i
(probability that elements in S take value a_i)

S_i - subset of set S, with data elements for which the value of attribute a was equal a_i

Set *S*

married	age	uses computer at work	buy computer
Yes	young	Yes	Yes
Yes	young	Yes	Yes
No	young	Yes	Yes
Yes	middle	Yes	Yes
No	middle	Yes	No
No	old	Yes	Yes
Yes	old	No	No
Yes	old	No	No
Yes	young	No	Yes
Yes	middle	No	Yes

$$Ent(S) = -\left(\frac{7}{10} log_2(^7/_{10}) + \frac{3}{10} log_2(^3/_{10})\right) = 0{,}88129$$

Attribute a – use computer at work

S_{Yes}

is married	age	uses computer at work	by computer
Yes	young	Yes	Yes
Yes	young	Yes	Yes
No	young	Yes	Yes
Yes	middle	Yes	Yes
No	middle	Yes	No
No	old	Yes	Yes

S_{No}

is married	age	uses computer at work	by computer
Yes	old	No	No
Yes	old	No	No
Yes	young	No	Yes
Yes	middle	No	Yes

H(S|use computer at work)

S_{Yes}

is married	age	uses computer at work	by computer
Yes	young	Yes	Yes
Yes	young	Yes	Yes
No	young	Yes	Yes
Yes	middle	Yes	Yes
No	middle	Yes	Yes
No	old	Yes	Yes

S_{No}

is married	age	uses computer at work	by computer
Yes	old	No	No
Yes	old	No	No
Yes	young	No	Yes
Yes	middle	No	No

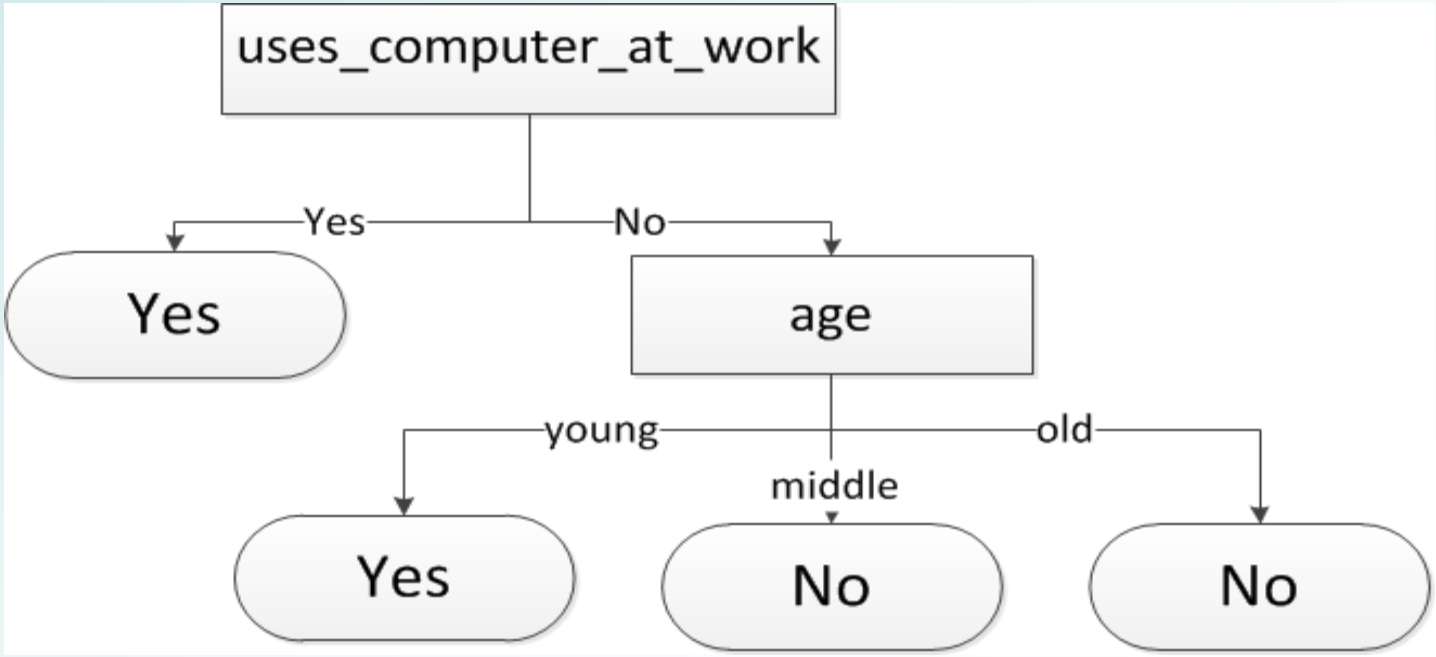
$$Pr(a = Yes) = {}^{6}/_{10}$$
$$Pr(a = No) = {}^{4}/_{10}$$

$$Ent(S_{Yes}) = 0$$
$$Ent(S_{No}) = 0{,}324511$$

$$H(S|a) = Ent(S) - {}^{6}/_{10} * 0 + {}^{4}/_{10} * 0{,}324511$$
$$= 0{,}55678$$

Decision tree

attribute a	*H(S|a)*
is married	0,191631
age	0,330313
uses computer at work	0,55678



Gini Index (the CART algorithm)

$$Gini(S) = \sum_{i=1}^{k} p_i(1 - p_i)$$

or equivalently

$$Gini(S) = 1 - \sum_{i=1}^{k} {p_i}^2$$

Misclassification error

$$MIS(S) = 1 - p_{max},$$

where

p_{max} - the fraction of data elements with majority class in set S

The Hoeffding's bound

The commonly known algorithm called ‘Hoeffdings Tree’ was introduced by P. Domingos and G. Hulten in [1]. The main mathematical tool used in this algorithm was the Hoeffding’s bound [2] in the form

Theorem 1:

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

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

where

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

[1] P. Domingos and G. Hulten, ”Mining high-speed data streams”, Proc. 6th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining, pp. 71-80, 2000.

[2] W. Hoeffding, ”Probability inequalities for sums of bounded random variables”, Journal of the American Statistical Association, vol. 58, issue 301, pp. 13-30, March 1963.

The Hoeffding's bound

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

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

with probability $1 - \alpha$.

[1] P. Domingos and G. Hulten, ”Mining high-speed data streams”, Proc. 6th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining, pp. 71-80, 2000.
[2] W. Hoeffding, ”Probability inequalities for sums of bounded random variables”, Journal of the American Statistical Association, vol. 58, issue 301, pp. 13-30, March 1963.

The Hoeffding's bound

For independent random variables $X_1,\ X_2, \ldots,\ X_n$, $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$ and $E[\bar{X}]$ is expected value of $\bar{X}$

$$P\left\{\bar{X} - E[\bar{X}] \geq \sqrt{\frac{R^2 \ln 1/\alpha}{2n}}\right\} \leq 1 - \alpha$$

The Hoeffding's bound

The Hoeffding's bound is wrong tool to solve the problem of choosing the best attribute to make a split in the node (see [3]). 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
- they are using only frequency of elements.
- Theorem 1 is applicable only for numerical data.

Therefore the idea presented in [1] violates the assumptions of Theorem 1 (see [2]) and the concept of Hoeffding Trees has no theoretical justification.

[1] P. Domingos and G. Hulten, ”Mining high-speed data streams”, Proc. 6th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining, pp. 71-80, 2000.

[2] W. Hoeffding, ”Probability inequalities for sums of bounded random variables”, Journal of the American Statistical Association, vol. 58, issue 301, pp. 13-30, March 1963.

[3] 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, vol. 99, no. PrePrints, 2012

Algorithms based on the Hoeffding bound

a) Hoeffding Tree Algorithm

The Hoeffding tree algorithm is a decision tree learning method for stream data classification. It uses Hoeffding trees and the Hoeffding bound, which exploit the idea that a small sample can often be enough to choose an optimal splitting attribute.

Algorithms based on the Hoeffding bound

b) Very Fast Decision Tree (VFDT)

The VFDT (Very Fast Decision Tree) algorithm makes several modifications to the Hoeffding tree algorithm to improve both speed and memory utilization. The modifications include:

- breaking near-ties during attribute selection more aggressively,
- computing the *G* function after a number of training examples,
- deactivating the least promising leaves whenever memory running low,
- dropping poor splitting attributes,
- improving the initialization method.

Algorithms based on the Hoeffding bound

c) Concept-adapting Very Fast Decision Tree (CVFDT)

CVFDT uses a sliding window approach, the main features are the following:

- it does not construct a new model from scatch each time,
- it updates statistics at the modes by incrementing the counts associated with new examples and decrementing the counts associated with old ones,
- if there is a concept drift, some nodes may no longer pass the Hoeffding bound. When this happens, an alternate subtree will be grown, with the new best splitting attribute at the root. As new examples stream in, the alternate subtree will continue to develop, without yet being used for classification. Once the alternate subtree becomes more accurate than the existing one, the old subtree is replaced.

Challenges in Stream Data Mining

a) Develop a theoretical tool to deal with data streams, in particular replace the Hoeffding bound (wrong technique) by another approach.

Challenges in Stream Data Mining

b) Design a decision tree learning system for stream data such that its output is nearly identical to that of a conventional learner.

Challenges in Stream Data Mining

c) Find (analytically) a number of samples of the infinite stream data such that a split in decision tree learning system can be made.

Decision Trees for Mining Data Streams

Suppose that attribute a can take one of $|a|$ different values from the set $A = \{a_1, \ldots, a_{|a|}\}$ (analogously for any other attribute) and $\Lambda = \{k_1, \ldots, k_K\}$ is a set of different classes. Let

$$Z = \{X_1, \ldots, X_i, \ldots, X_N\}$$

be the training set (set of examples) of size N, where $X_1, \ldots, X_N$ are independent random variables defined as follows

$$X_i = (a_{j_i}, b_{l_i}, \ldots, k_{q_i}) \in A \times B \times \cdots \times \Lambda$$

for $i = 1, \ldots, N$, $j_i \in \{1, \ldots, |a|\}$, $l_i \in \{1, \ldots, |b|\}$, $\ldots$, $q_i \in \{1, \ldots, K\}$.

Decision Trees for Mining Data Streams

Each element of Z belongs to one of the K different classes k_j. Entropy associated with the classification of Z is defined as

$$H(Z) = -\sum_{j=1}^{K} p_j \log_2 p_j,$$

where p_j is the probability that element from Z comes from class k_j. We estimate this probability by $\frac{n^j}{N}$, where n^j is the number of elements from class k_j.

Decision Trees for Mining Data Streams

Then

$$H(Z) = -\sum_{j=1}^{K} \frac{n^j}{N} \log_2 \frac{n^j}{N}.$$

Let us choose an attribute a, characterizing the elements of set Z. Then Z_{a_i} denotes a set of elements from Z, for which the value of a is a_i. The number of elements from set Z_{a_i} is labeled as n_{a_i}.

Decision Trees for Mining Data Streams

Then the weighted entropy for attribute a and set Z is given by

$$H_a(Z) = \sum_{i=1}^{|a|} \frac{n_{a_i}}{N} H(Z_{a_i}),$$

where

$$H(Z_{a_i}) = -\sum_{j=1}^{K} \frac{n_{a_i}^j}{n_{a_i}} \log_2 \frac{n_{a_i}^j}{n_{a_i}}$$

and $n_{a_i}^j$ denotes the number of elements in set Z_{a_i} from class k_j.

Decision Trees for Mining Data Streams

from class k_j. Information gain for attribute a is given by

$$Gain_a(Z) = H(Z) - H_a(Z).$$

Let us assume, that a is an attribute with the highest value of information gain, while b is the second-best attribute. Define

$$\begin{aligned} f(Z) &= Gain_a(Z) - Gain_b(Z) \\ &= H(Z) - H_a(Z) - (H(Z) - H_b(Z)) \\ &= H_b(Z) - H_a(Z). \end{aligned}$$

Decision Trees for Mining Data Streams

Heoffding inequality

Let Y_i for $i = 1, \ldots, N$ be independent random variables such that $Pr(Y_i \in [a_i, b_i]) = 1$. Then for $S = \sum_{i=1}^{N} Y_i$ for all $\epsilon > 0$ we have the inequalities

$$Pr(S - E[S] \geq \epsilon) \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{N}(b_i - a_i)^2}\right),$$

$$Pr(|S - E[S]| \geq \epsilon) \leq 2\exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{N}(b_i - a_i)^2}\right).$$

Obviously Hoeffding inequality holds only for real valued data

Unfortunately, by analyzing descriptions it can be easily seen that they do not fit the Hoeffding's inequality probabilistic model. Firstly, only numerical data are applicable to the Hoeffding's inequality. In the general case the data do not have to be numerical.

Secondly, split measures, like information gain and Gini index, can not be expressed as a sum S of elements Y_i. Moreover they are using only the frequency of elements.

McDIARMID'S INEQUALITY

$$Z = \{X_1, \ldots, X_i, \ldots, X_N\} \qquad (2)$$

Let Z, given by (2), be the set of independent random variables, with X_i taking values in a set A_i for each i. Let us define

$$Z' = \{X_1, \ldots, \hat{X}_i, \ldots, X_N\}, \qquad (17)$$

with $\hat{X}_i$ taking values in A_i. Observe that Z' differs from Z, given by (2), only in the ith element.

In this paper the basic tool to analyze data streams is the following McDiarmid's inequality [18].

McDiarmid's inequality

Suppose that the (measurable) function $\tilde{f}: \prod A_i \to \mathbb{R}$ satisfies

$$\sup_{x_1,\ldots,x_N,\widehat{x_i}} |\tilde{f}(Z) - \tilde{f}(Z')| \leq c_i,$$

for some constant c_i. Then

$$Pr(\tilde{f}(Z) - E[\tilde{f}(Z)] \geq \epsilon) \leq \exp\left(\frac{-2\epsilon^2}{\sum_{i=1}^{N} c_i^2}\right).$$

ID3 ALGORITHM ADOPTED

for STREAM DATA MINING

McDiarmid's bound for information gain

In this section we assume that the split measure is information gain. The following theorem guarantees that a decision tree learning system, applied to data streams, has the property that its output is nearly identical to that produced by a conventional learner.

Theorem

Let $Z = \{X_1, \cdots, X_N\}$ be the set of independent random variables, with each of them taking values in the set $A \times B \times \cdots \times \Lambda$. Then, for any fixed δ and any pair of attributes a and b, where $f(Z) = Gain_a(Z) - Gain_b(Z) > 0$, if

$$\epsilon = C_{Gain}(K, N)\sqrt{\frac{\ln(1/\delta)}{2N}},$$

then

$$Pr(f(Z) - E[f(Z)] > \epsilon) \leq \delta,$$

where

$$C_{Gain}(K, N) = 6(K \log_2 eN + \log_2 2N) + 2\log_2 K.$$

CART ALGORITHM ADOPTED

for STREAM DATA MINING

McDiarmid's bound for Gini index

Gini index for a set Z is defined as follows

$$Gini(Z) = 1 - \sum_{j=1}^{K} \left(\frac{n^j}{N} \right)^2 .$$

Let $A = \{a_1, ..., a_{|a|}\}$ be a set of values for attribute a. There are $2^{|a|-1} - 1$ different ways of partitioning set A into two disjoint subsets: $A_1 = \{a_{j_i} \in A : i \in \{1, ..., D\}\}$ and $A_2 = \{a_{j_i} \in A : i \in \{D+1, ..., |a|\}\} = A \backslash A_1$. Since $A = A_1 \cup A_2$, it is sufficient to consider only one of these sets, e.g. A_1. The set Z (Z') is partitioned into two disjoint subsets Z_1 and Z_2 (Z'_1 and Z'_2), according to sets A_1 and A_2.

Let us define

$$Gini_{A_1}(Z) = \frac{n_1}{N}\left(1-\sum_{j=1}^{K}\left(\frac{n_1^j}{n_1}\right)^2\right) + \frac{n_2}{N}\left(1-\sum_{j=1}^{K}\left(\frac{n_2^j}{n_2}\right)^2\right) = Gini_{A_2}(Z),$$

where n_l, $l = 1, 2$, is a number of elements in set Z_l and n_l^j is a number of elements in set Z_l from class k_j

Among all the possible partitions of set A into two disjoint sets A_1 and A_2, we choose one with the lowest value of $Gini_{A_1}(Z)$. This value is a Gini index for attribute a

$$Gini_a(Z) = \min_{A_1}\{Gini_{A_1}(Z)\}.$$

We will call it an optimal partition.

A split measure for attribute a, based on Gini index (Gini gain), is given by:

$$\Delta Gini_a(Z) = Gini(Z) - Gini_a(Z).$$

Let us assume that a is an attribute with the highest value of $\Delta Gini_a(Z)$ and b is the second-best attribute. Define

$$\begin{aligned} f(Z) &= \Delta Gini_a(Z) - \Delta Gini_b(Z) \\ &= Gini_b(Z) - Gini_a(Z). \end{aligned}$$

Theorem

Let $Z = \{X_1, \cdots, X_N\}$ be the set of independent random variables, with each of them taking values in the set $A \times B \times \cdots \times \Lambda$. Then, for any fixed δ and any pair of attributes a and b, where $f(Z) = Gini_b(Z) - Gini_a(Z) > 0$, if

$$\epsilon = 8\sqrt{\frac{\ln(1/\delta)}{2N}}$$

then

$$Pr(f(Z) - E[f(Z)] > \epsilon) \leq \delta$$

Corollary

Suppose a and b are attributes for which the values of Gini gain, calculated from the data sample Z, satisfy $\Delta Gini_a(Z) > \Delta Gini_b(Z)$. For any fixed δ and ϵ given by (37), if $f(Z) > \epsilon$, then with probability $1 - \delta$ attribute a is better to split than attribute b, according to whole data stream. Moreover, if a and b are attributes with the highest and the second highest values of Gini gain, then with probability $1-\delta$, a is the best attribute to split according to the whole stream.

Example

Let us assume that we have set Z of 10000 data. We want to check if attribute a is better to split than attribute b, with probability $1 - \delta = 0,95$. To check this we use expressions (37) and (38).

First let us compute the value of ϵ given by formula (37). For our data $\epsilon = 0,0979099$. Note that in this case, contrary to Theorem 1, the value of ϵ does not depend on the number of classes K. According to Corollary 2 attribute a is better to split the node than attribute b, with probability $1 - \delta$, if $f(Z) > \epsilon$. In our case this condition is satisfied for $Gini_b(Z) - Gini_a(Z) > 0,0979099$.

If $Gini_b(Z) - Gini_a(Z) \leq \epsilon$, then we can not determine which attribute is better.

Example

Let us assume that the calculated value of $f(Z) = Gini_b(Z) - Gini_a(Z)$ is equal to 0,354. We want to know how many data elements N we should have, to say that attribute a is better to split than attribute b with probability $1-\delta = 0,975$. According to inequality (31), with ϵ given by (37), N is a sufficient number if

$$f(Z) > 8 \cdot \sqrt{\frac{\ln(1/\delta)}{2N}}.$$

Contrary to the case of information gain (see Example 2), N can be determined using simple algebra

$$N > \frac{64 \ln(1/\delta)}{2 \cdot (f(Z))^2}.$$

In our case the number of elements N should satisfy $N > 941,9718$. So if we have 942 elements or more, we can say that with probability $0,975$ attribute a is better to split than attribute b.

The McDIARMID CART TREE ALGORITHM

For convenience the following notations will be introduced:

- $\mathscr{A}$ - set of all attributes
- a - any attribute from set $\mathscr{A}$
- a_{MAX1} - attribute with the highest value of $G(\cdot)$
- a_{MAX2} - attribute with the second highest value of $G(\cdot)$

Table 1: The McDiarmid tree algorithm

Inputs:	Z	is a sequence of examples,
	$\mathscr{A}$	is a set of discrete attributes,
	$G(\cdot)$	is a split evaluation function,
	δ	is one minus the desired probability of choosing the correct attribute at any given node.
Output:	$McDT$	is a decision tree.

Procedure McDiarmidTree$(Z, \mathscr{A}, G, \delta)$
Let $McDT$ be a tree with a single leaf l_1 (the root).
Let $\mathscr{A}_{l_1} = \mathscr{A}$
For each class k_r
    For each attribute $a \in \mathscr{A}$
        For each value a_λ of attribute a
            Let $n^r_{a_\lambda}(l_1) = 0$.
For each example X in Z
    Sort X into a leaf l using $McDT$.
    For each attribute $a \in \mathscr{A}_l$
        For each value a_λ of attribute a
            If value of example X for attribute a equals a_λ and X comes from class r
                Increment $n^r_{a_\lambda}(l)$.
    Label l with the majority class among the examples seen so far at l.

If the examples seen so far at l are not of the same class, then

Compute $\overline{G}_l(a)$ for each attribute $a \in \mathscr{A}_l$ using the counts $n^r_{a\lambda}(l)$.

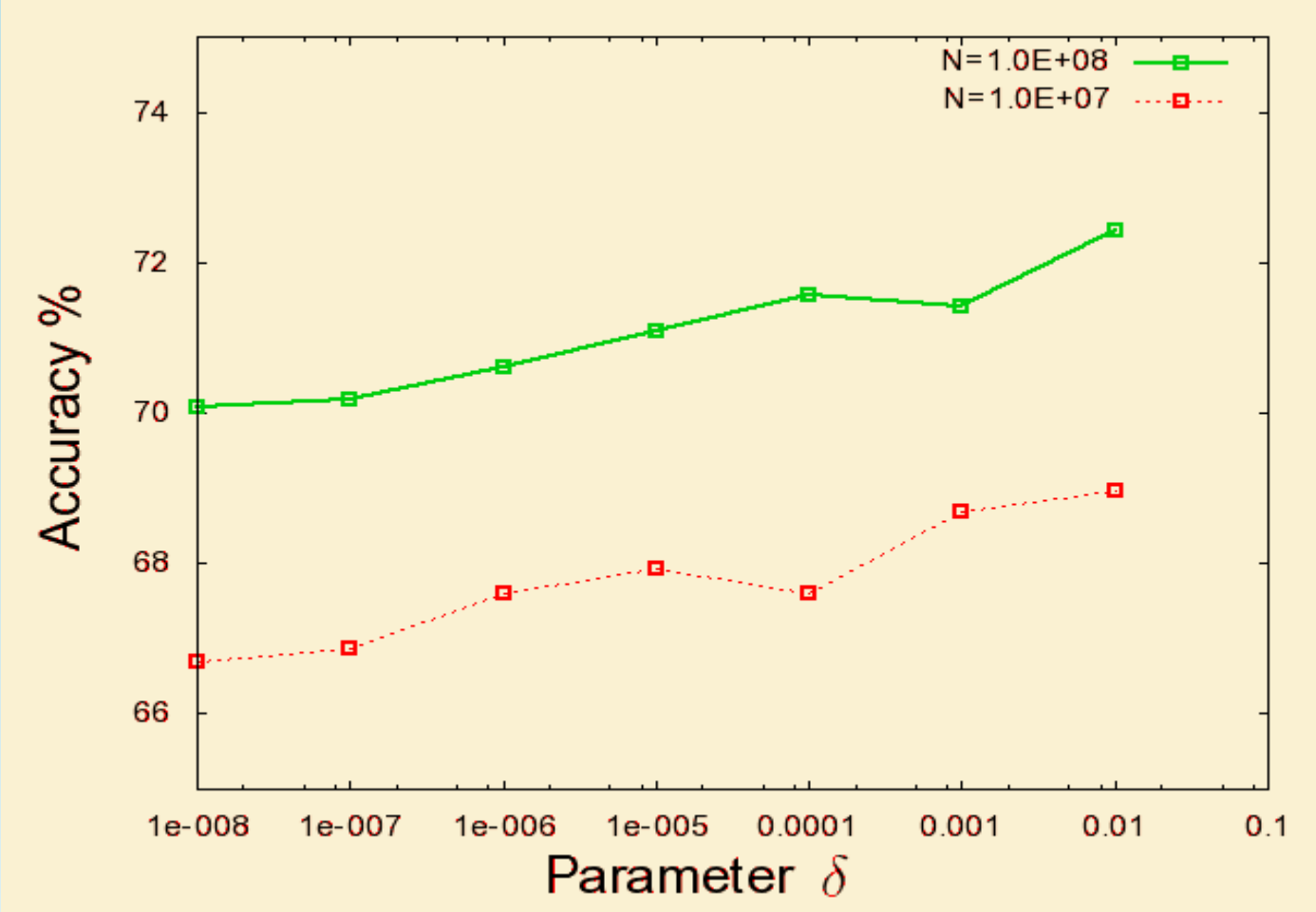
Let a_{MAX1} be the attribute with the highest $\overline{G}_l$.

Let a_{MAX2} be the attribute with the second-highest $\overline{G}_l$.

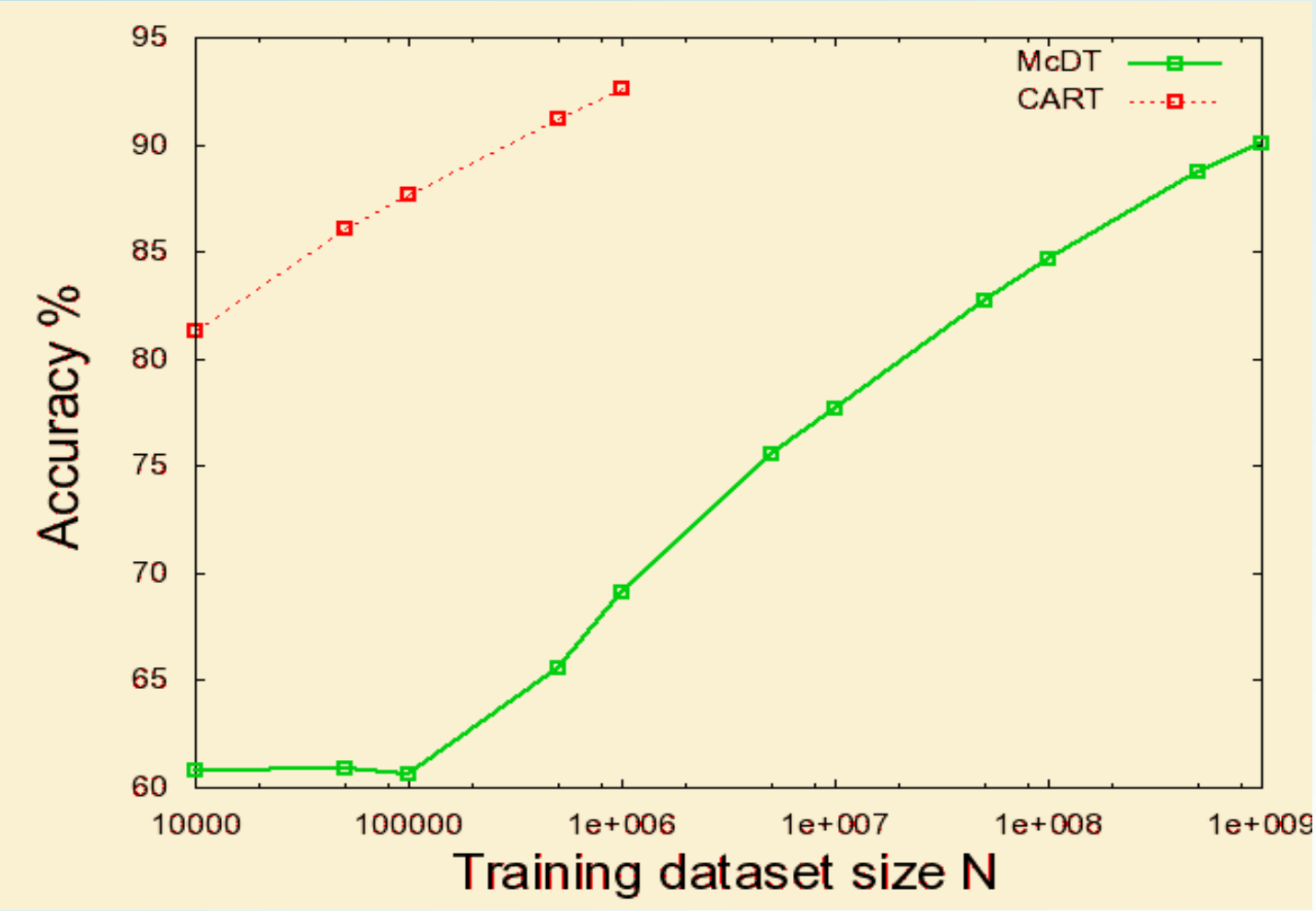
Compute ϵ using equation (29) for information gain or (37) for Gini gain.

If $\overline{G}_l(a_{MAX1}) - \overline{G}_l(a_{MAX2}) > \epsilon$, then
Replace l by an internal node that splits on a_{MAX1}.
For each branch of the split
Add a new leaf l_m, and let $\mathscr{A}_{l_m} = \mathscr{A} \backslash \{a_{MAX1}\}$ at l_m.
For each class k_r
For each attribiute $a \in \mathscr{A}$
For each value a_λ of a
Let $n^r_{a_\lambda}(l_m) = 0$.

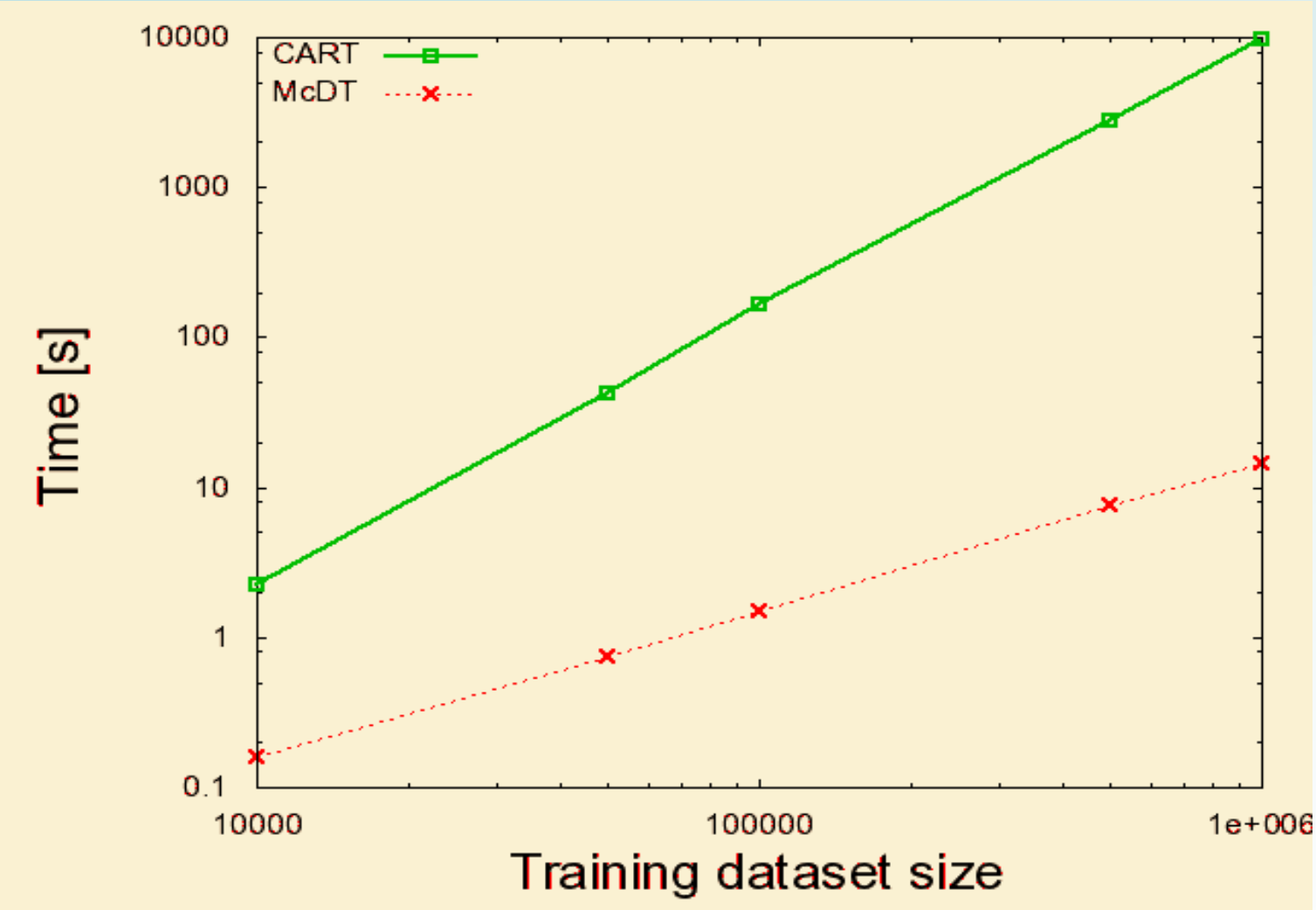
Return $McDT$.



Accuracy as a function of parameter δ



Training dataset size N



Training dataset size

ID3 ALGORITHM ADOPTED

for STREAM DATA MINING

Gaussian based approximation

for information gain

The estimated values of p_L, p_{1L} and p_{1R}, calculated from the sample, are denoted by $\overline{p_L}$, $\overline{p_{1L}}$ and $\overline{p_{1R}}$, respectively.

$$\overline{g_x} = g(\overline{p_L}, \overline{p_{1L}}, \overline{p_{1R}})$$

Let us define a threshold th as a fraction of elements of the first class (p_1), below which we assume that there is no need for splitting the node. Therefore, all the calculations are performed only if the following inequality is satisfied

$$\frac{1-p_1}{p_1} \leq \frac{1-th}{th} = C.$$

For example, for threshold value $th = 0,05$, C is equal to 19. To simplify the form of the following theorem, we introduce $Q(C)$ as

$$Q(C) = \frac{\log_2^2 e^2}{Ce^2} + \frac{\log_2^2 e^2}{e^2} + \frac{\log_2 e}{e} + \frac{\log_2^2(2C)}{4}.$$

The main result of this paper is the following theorem stating that if the difference between the information gain estimates obtained for two attributes is greater than a specific value, given by (46), then with a fixed probability there is, roughly speaking, a statistical difference between the true information gains. This allows either to determine, from a recent fragment of data, the best feature to split on or to say that the information to determine the split is statistically insufficient.

Theorem

Let us consider two attributes a^x and a^y, for which we have calculated the values of information gain $\overline{g_x} = g(\overline{p_L^x}, \overline{p_{1L}^x}, \overline{p_{1R}^x})$ and $\overline{g_y} = g(\overline{p_L^y}, \overline{p_{1L}^y}, \overline{p_{1R}^y})$. If the difference of these values satisfies the following condition

$$\overline{g_x} - \overline{g_y} > \epsilon,$$

where

$$\epsilon = z_{(1-\delta)} \frac{\sqrt{2Q(C)}}{\sqrt{n}}$$

and $z_{(1-\delta)}$ is the $(1-\delta)$-th quantile of the standard normal distribution $N(0,1)$, then g_x is greater than g_y with probability $1-\delta$. If a^x and a^y are attributes with the highest values of information gain, then a^x can be chosen to split the considered leaf node, with the level of confidence $1-\delta$.

Example

Let us consider a two class case with threshold $th = 0,01$. Suppose that after 10000 observations the difference $\overline{g_x} - \overline{g_y}$ was equal to 0,09371. We want to know if we should make a split with 0,95 level of confidence. From normal distribution tables we get $z_{(0,95)} = 1,64854$. First we need to calculate C

$$C = \frac{0,99}{0,01} = 99.$$

After calculating C we can compute the value of function $Q(C)$

$$\begin{aligned} Q(C) &= \frac{\log_2^2 e^2}{99e^2} + \frac{\log_2^2 e^2}{e^2} + \frac{\log_2 e}{e} + \frac{\log_2^2(2 \cdot 99)}{4} \\ &\approx 16,22062. \end{aligned}$$

Example

Then, according to (45), we obtain

$$z_{(1-\delta)}\frac{\sqrt{2Q(C)}}{\sqrt{n}} \approx 1,64854\frac{\sqrt{2\cdot 16,22062}}{\sqrt{10000}}$$
$$\approx 0,093894 > \overline{g_x} - \overline{g_y} = 0,09371.$$

Therefore we should not yet make a split. Assume that after 10100 observations the difference $\overline{g_x} - \overline{g_y}$ was equal to 0,09356. Now we get

$$z_{(1-\delta)}\frac{\sqrt{2Q(C)}}{\sqrt{n}} \approx 16,22062\frac{\sqrt{2\cdot 1,22062}}{\sqrt{10100}}$$
$$\approx 0,093429 < \overline{g_x} - \overline{g_y}.$$

This time our result shows, that with 0,95 level of confidence, g_x is greater than g_y, so we can make a split.

Example

In this example we compare our method with the McDiarmid bound for information gain [29]. We will calculate the number of data elements needed to make a split. Let us consider the following situation. We calculated the difference $Gain_{a^x}(Z) - Gain_{a^y}(Z) = \overline{g_x} - \overline{g_y} = 0,2479$ and we want to know, with probability 95%, if the attribute a^x is the best to make a split. The number of classes is $K = 2$ and the threshold th is 0,02.

a) First we use the McDiarmid bound for information gain. According to inequality (20), n is sufficient number if $f(Z) = Gain_{a^x}(Z) - Gain_{a^y}(Z) > \epsilon$, where ϵ is given by (19). From equation (21)

Example

we can not compute analytically the number of data elements. However we can solve this problem numerically. In this case, for $n = 4'345'299$ obtained ϵ was equal to 0,2479000136. This value is greater than $f(Z)$. Therefore the number of data is too small. However for $n = 4'345'300$ the value of ϵ was 0,2478999886. This value is smaller than $f(Z)$ and therefore n is large enough to say that attribute x is better to split than y, with probability 95%.

b) Now we will compute the sufficient number of data elements using method presented in this paper. First we have to calculate C and $Q(C)$ according to (43) and (44), i.e.

$$C = \frac{1 - 0,2}{0,02} = 49,$$

$$Q(C) \approx 12,61905957.$$

Example

Using inequality (45) we calculate n as follows

$$\overline{g_x} - \overline{g_y} > z_{(1-\delta)} \frac{\sqrt{2Q(C)}}{\sqrt{n}},$$

$$\sqrt{n} > z_{(1-\delta)} \frac{\sqrt{2Q(C)}}{\overline{g_x} - \overline{g_y}},$$

$$n > z^2_{(1-\delta)} \frac{2Q(C)}{(\overline{g_x} - \overline{g_y})^2}.$$

$$z^2_{(1-\delta)} \frac{2Q(C)}{(\overline{g_x} - \overline{g_y})^2} \approx (1,64854)^2 \frac{2 \cdot 12,61905957}{(0,2479)^2}$$
$$\approx 1116,099494$$

Therefore if the number of data elements is grater than 1116 then, with probability 95%, attribute a^x is the best and the split should be made.

Example

As we can see the obtained number of data elements in the case of the McDiarmid bound is incomparably higher than obtained based on our new method. The observed relationship can be confirmed by analytical way. Suppose that we want to get the same value of parameter ϵ_M given by formula (19) (McDiarmid) and ϵ given by (46) (Gaussian Approximation). It requires different values of n. Let us denote by n_M and n_G values of parameter n for the McDiarmid and Gaussian Approximation, respectively. Comparing equations (19) and (46) we can determine, by simple algebra, the relationship

$$n_G = \frac{z^2_{1-\delta} Q(C) n_M}{9 \ln(1/\delta) \log_2^2(2^{\frac{4}{3}} e^2 n_M^3)}.$$

Example

This relationship is shown in Figure 1 for different values of parameter δ. The axis OX shows the number of data elements for the McDiarmid method. The axis OY shows corresponding percentage of data ($\frac{n_G}{n_M} \cdot 100\%$) that would have to be used in Gaussian Approximation method in order to $\epsilon = \epsilon_M$. One can see that the method presented in this paper allows to reduce the number of data required to make a split.

ID3 ALGORITHM ADOPTED

for STREAM DATA MINING

The Gaussian decision tree algorithm - GDT

Table 1: The GDT

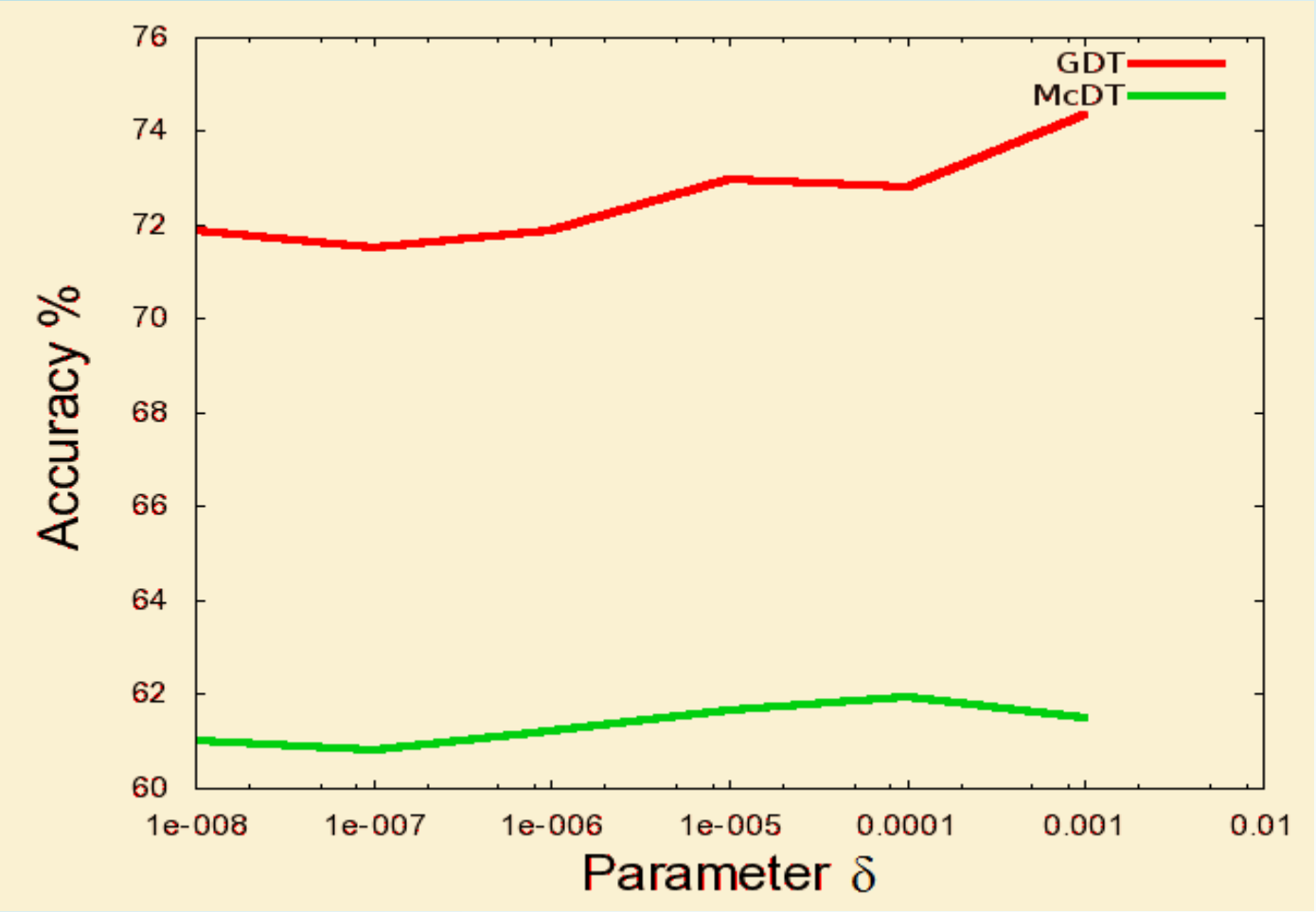
Inputs:	$\mathbf{S}$	is a sequence of examples,
	$\mathscr{A}$	is a set of discrete attributes,
	th	is a minimal fraction of elements belonging to the one class,
	δ	is one minus the desired probability of choosing the correct attribute at any given node.
Output:	GDT	is a decision tree.

Procedure GDT($\mathbf{S}, \mathscr{A}, th, \delta$)

01. Let GDT be a tree with a single leaf L_0 (the root).
02. Let $\mathscr{A}_0 = \mathscr{A}$
03. Let $C = \frac{1-th}{th}$, $last = 0$
04. For each attribute $a^i \in \mathscr{A}$
05. For each value a^i_λ of attribute a^i
06. $n^1_{i,\lambda,0} = 0$, $n^2_{i,\lambda,0} = 0$.
07. For each example s in $\mathbf{S}$
08. Sort s into a leaf L_q using GDT.
09. For each attribute $a^i \in \mathscr{A}_q$
10. For each value a^i_λ of attribute a^i
11. For each class $k = 1, 2$
12. If value of example s for attribute a^i is equal to a^i_λ and s is from the k-th class then
13. Increment $n^k_{i,\lambda,q}$.

14. Label L_q with the majority class among the examples seen so far at L_q.
15. If $\frac{\max\{n_q^1, n_q^2\}}{\min\{n_q^1, n_q^2\}} < C$ then
16. For each attribute $a^i \in \mathscr{A}_l$
17. For each partition of the set A^i into A_L^i, A_R^i
18. Compute $\overline{g_q}(A_L^i)$ using the counts $n_{i,\lambda,q}^k$, $k = 1, 2$.
19. $\overline{g_{i,q}} = \max_{A_L^i \in \mathbf{V}_i} \{\overline{g_q}(A_L^i)\}$
20. $a^x = \arg \max_{a^i \in \mathscr{A}_q} \{\overline{g_{i,q}}\}$
21. $a^y = \arg \max_{a^i \in \mathscr{A}_q \setminus \{a^y\}} \{\overline{g_{i,q}}\}$

22. Compute ϵ using formula (46)
23. If $\overline{g_{x,q}} - \overline{g_{y,q}} > \epsilon$, then
24. Replace L_q by an internal node that splits on a^x.
25. For both branches of the split
26. Add a new leaf L_{last+1} and let $\mathscr{A}_{last+1} = \mathscr{A}_q \backslash \{a^x\}$ at L_{last+1}.
27. For each attribute $a^i \in \mathscr{A}_{last+1}$
28. For each value a^i_λ of a^i
29. $n^1_{i,\lambda,last+1} = 0$, $n^2_{i,\lambda,last+1} = 0$.
30. $last = last + 1$
31. Return GDT.



The dependence between the value of parameter δ and the accuracy of the Gaussian Decision Tree and the McDiarmid Tree algorithms.

The dependence between the number of training data and the accuracy of the Gaussian Decision Tree and the ID3 algorithm.

The dependence between the number of training data and the processing time of Gaussian Decision Tree and the ID3 algorithms.

Example for the GDT algorithm

- Data elements described by 3 nominal attributes:
  - attribute **a**, 3 possible values: a_1, a_2, a_3
  - attribute **b,** 2 possible values: b_1, b_2
  - attribute **c,** 2 possible values: c_1, c_2
- Two class problem (True (**T**), False (**F**))
- Split measure function: Information gain
- Parameters: $th = 0.01$, $\alpha = 0.05$, which gives: $C = 99$, $Q(C) \approx 16.22062$, $z_{1-\alpha} \approx 1.64854$
- ε bound from the GDT algorithm

$$\varepsilon = z_{1-\alpha}\frac{\sqrt{2Q(C)}}{\sqrt{N}} \approx \frac{9.38962}{\sqrt{N}}$$

Example for the GDT algorithm

Let us assume that the statistics in the considered tree node are given as follows:

attribute **a**

	a_1	a_2	a_3
T	2000	3000	5000
F	4000	3000	3000

attribute **b**

	b_1	b_2
T	5000	5000
F	4999	5001

attribute **c**

	c_1	c_2
T	7160	2840
F	3326	6674

The numbers of elements from class **T** and **F** are equal: the entropy is maximal:

E = 1

The weighted entropy for attributes **a**, **b**, and **c**:

E(a) = 0.95726235
E(b) = 0.99999999
E(c) = 0.89090685

This gives the following values of information gain:

G(a) = E – E(a) = 0.04273765
G(b) = E – E(b) = 0.00000001
G(c) = E – E(c) = 0.10909315

Example for the GDT algorithm

Let us assume that the statistics in the considered tree node are given as follows:

attribute **a**

	a_1	a_2	a_3
T	2000	3000	5000
F	4000	3000	3000

attribute **b**

	b_1	b_2
T	5000	5000
F	4999	5001

attribute **c**

	c_1	c_2
T	7160	2840
F	3326	6674

Information gain values:

G(a) = 0.04273765
G(b) = 0.00000001
G(c) = 0.10909315

Attributes **a** and **c** provide the highest value of information gain. The difference between them is:

G(c) – G(a) = 0.06635551

The bound ε for $N = 20000$ elements:

ε = 0.06639467

Since the difference is lower than ε:, i.e.

G(c) – G(a) < ε,

The considered node can not be split at the moment

Example for the GDT algorithm

Now the following data element reaches the considered node: $[a_3, b_2, c_2, F]$.
The statistics are changed as follows:

attribute **a**

	a_1	a_2	a_3
T	2000	3000	5000
F	4000	3000	3001

attribute **b**

	b_1	b_2
T	5000	5000
F	4999	5002

attribute **c**

	c_1	c_2
T	7160	2840
F	3326	6675

Current value of entropy:

E = 0.999999998

Current values of weighted entropy:

E(a) = 0.95728523
E(b) = 0.99999998
E(c) = 0.89088787

Current values of inormation gain:

G(a) = 0.04271477
G(b) = 0.00000002
G(c) = 0.10911212

Example for the GDT algorithm

Now the following data element reaches the considered node: $[a_3, b_2, c_2, F]$.
The statistics are changed as follows:

attribute **a**

	a_1	a_2	a_3
T	2000	3000	5000
F	4000	3000	3001

attribute **b**

	b_1	b_2
T	5000	5000
F	4999	5002

attribute **c**

	c_1	c_2
T	7160	2840
F	3326	6675

Previous values	**Current values**
G(a) = 0.04273765 G(b) = 0.00000001 G(c) = 0.10909315	G(a) = 0.04271477 G(b) = 0.00000002 G(c) = 0.10911212
G(c) – G(a) = 0.06635551	G(c) – G(a) = 0.06639736
ε = 0.06639467	ε = 0.06639301

Now the difference is higher than ε, i.e.

$G(c) - G(a) > \varepsilon$

The considered node is split according to the attribute **c**

Conclusions

a) We developed a theoretical tool to deal with data streams, in particular we replaced the Hoeffding bound (wrong technique) by another approaches.
b) We designed a decision tree learning system for stream data such that its output is nearly identical to that of a conventional learner.
c) We found analytically a number of samples of the infinite stream data such that a split in decision tree learning system can be made.

ICAISC 2014

http://icaisc.eu

13th International Conference
Artificial Intelligence and Soft Computing
Zakopane, Poland, June 1 - 5, 2014

organized by

the Polish Neural Network Society and the Academy of Management in Łódź (SWSPiZ)

in cooperation with

the Department of Computer Engineering at Częstochowa University of Technology and

the IEEE Computational Intelligence Society Poland Chapter

Dziękuję bardzo za uwagę