# Incremental Decision Tree based on order statistics

Christophe Salperwyck, Vincent Lemaire

# Incremental Decision Tree based on order statistics

**Christophe Salperwyck**[1] **and Vincent Lemaire**[2]

**Abstract.** New application domains generate data which are not persistent anymore but volatile: network management, web profile modeling... These data arrive quickly, massively and are visible just once. Thus they necessarily have to be learnt according to their arrival orders. For classification problems online decision trees are known to perform well and are widely used on streaming data. In this paper, we propose a new decision tree method based on order statistics. The construction of an online tree usually needs summaries in the leaves. Our solution uses bounded error quantiles summaries. A robust and performing discretization or grouping method uses these summaries to provide, at the same time, a criterion to find the best split and better density estimations. This estimation is then used to build a naïve Bayes classifier in the leaves to improve the prediction in the early learning stage.

## 1 Introduction

Learning machines have shown their ability to deal with huge volumetry on real problems [15, 9]. Nevertheless most of the works were realized for data analysis on homogeneous and stationary data. Learning machines usually use data sets with fixed sizes and produce static models.

New application fields for data mining emerge in which data are not anymore persistent data table but rather "temporary" data. These data are called streaming data. Among these domains one finds: management of telecommunication networks, user modeling in a social network, web mining. One of the technical challenges is to design algorithms able to handle these new application constraints. As data arrive quickly and are visible only once, it is necessary to learn them as they arrive. Incremental learning appears as a natural solution to streaming problems.

Among the methods in incremental learning, models based on decision trees inspired by the algorithm "Very Fast Decision Tree" (VFDT) [8] are widely used. The tree construction is incremental and leaves are transformed into nodes as examples arrive. The new examples go down into the tree and are inserted variable by variable in a summary. A criterion (Gini or Entropy) uses this summary to find the cut points to transform a leaf into a node. The tree prediction can be improved by the addition of a local model in each leaf as in VFDTc [12].

The error rate of this kind of algorithm is more important in the early learning stage than a batch algorithm as C4.5. But having learnt several hundreds of thousand examples, this error rate becomes lower than C4.5 since C4.5 is not able to deal with millions of examples and thus has to use only a part of the available information.

[1] Orange Labs, Lannion, France and LIFL, Université de Lille 3, Villeneuve d'Ascq, France, email: christophe.salperwyck@orange.com
[2] Orange Labs, Lannion, France, email: vincent.lemaire@orange.com

In this article we focused on the construction of online decision trees. Section 2 presents existing approaches in terms of: split criterion, summaries in leaves, and local models. Our approach, based on order statistics, is detailed in the 3rd section. The experimental part which compares our approach with existing ones is in section 4. The last section concludes this article.

## 2 Previous works

Constructing an online decision tree is based on three main choices. In the first place, it is impossible in the data stream context, potentially of infinite size, to keep all the examples. The use of data summaries of limited size is necessary to be able to control the tree memory consumption. The fact that decisions are local to the leaf justifies storing summaries in each leaf. Secondly, cut points are chosen by the evaluation in every leaf of a criterion (generally the Gini or the entropy criterion). This choice being a definitive action has to be robust and made with a certain confidence. Finally before a split occurs, the available information in leaves is not used. Using a local model in each leaf allows exploiting this information to improve the global prediction of the tree. Figure 1 illustrates a decision tree and its three main components.

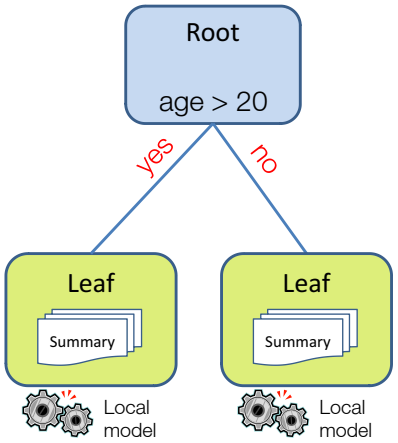


**Figure 1.** Main components of an online decision tree.

This section presents the different approaches used in the literature to answer the three main key points mentioned above. The quality of a decision tree depends on: (i) the summaries in the leaves, (ii) the split criterion, (iii) the local model.

## 2.1 Summaries in the leaves

The purpose of data summaries in the leaves is to memorize stream characteristics. This summary is used to find the "best" cut point to transform a leaf into a node and also to build a local model in the leaf. In certain application domains, as for example the management of a network of telecommunication or energy, the stream is potentially of infinite size. The generated tree de facto possesses a large number of decision nodes but the available memory is often of limited size. Therefore these summaries need to have a low memory footprint with low errors and address the precision / memory tradeoff.

The paragraphs below present several summaries. Numerical and categorical attributes are generally handled by means of different summary techniques.

### *2.1.1 Numerical attributes*

Gama and Pinto propose a summary for numerical attributes called Partition Incremental Discretization - PiD [11] which is partially incremental. This solution is based on two levels. Level 1 realizes a first discretization where the counts are stored by interval. This first level implements an incremental algorithm which combines the methods "Equal Frequency" and "Equal Width" and has to contain more intervals than the level 2. Second level uses level 1 discretization to make the final discretization, which can be of several types: "Equal Frequency", "Equal Width", K-means, Recursive entropy discretization, Proportional discretization. The memory consumption of this method depends mainly on the number of intervals in the first level. The second level is not incremental.

Pfahringer et al. [21] carried out a study on summaries for numerical attributes; their study is dedicated to trees using the Hoeffding bound. They tested the following summaries methods:

- Very Fast Machine Learning (VFML): this very simple method takes the $k$ first values of the stream and uses them to build $k+1$ intervals. The next values are aggregated in the closest interval defined by the first values. The memory consumption depends on the parameter $k$. This method comes from the source code [18] provided by the authors of VFDT: Domingos and Hulten.
- Gaussian approximation (GA): the data distribution is supposed to be a normal law. The purpose is to estimate the three parameters of the law which define this Gaussian: the average, the standard deviation (or the variance) and the number of elements. These three values can be stored incrementally making this method perfectly adapted to an online use. This approximation is chosen by Kirkby [19] in all his experiments. The memory consumption is constant independently of the nature of the observed stream.
- Exhaustive binary trees (EBT): Gama et al. use this method for their VFDTc tree [12]. A binary search tree, for each numerical variable, is built incrementally. This tree also keeps in each node the counts of values smaller and bigger than the cut point. This structure allows an immediate access to the counts on both sides of a cut point. The tree memory consumption depends on the number of different values arriving in the leaf.
- GK: this method, proposed by Greenwald and Khanna [14], is a quantiles based summary. It maintains a sorted list of intervals and controls the error on the quantile position (detailed in section 3.1.1). Its memory consumption depends either on the maximal error tolerated on quantiles or on the number of intervals that can be kept.

### *2.1.2 Categorical attributes*

To our knowledge, in most of the publications related to online trees there is no dedicated summary for categorical attributes but just exhaustive counting. It means that for each categorical variable and for each value the number of occurrences is stored. It can be sufficient if the number of values is limited and thus the memory consumed is low. This method linearly depends on the number of different values in the data stream.

## 2.2 Split criterion

During the construction of the decision tree a split criterion is used to transform a leaf into a node. The objective is to produce the most homogeneous possible groups of individuals regarding the target variable. To transform a leaf into a node it is necessary to determine at the same time on which attribute to cut and on which value (cut point).

In the literature on offline and online decision trees, two main cut criteria are used: Gini in the CART algorithm and the "gain ratio" based on the entropy in C4.5. Those two criteria find the best cut point for an attribute and each of them provides a score. The node is split on the attribute having the best score. This process to transform a leaf into a node is repeated to produce the final decision tree.

The difference in building an online tree and an offline tree comes from the fact that data arrive continuously for the first one. The choice of the attribute to cut is made according to the summary and not on all data. The choice of transforming a leaf into a node, is a definitive action. To make sure that this choice is realized with a certain confidence, Domingos and Hulten suggest in VFDT the use of the Hoeffding bound [16]. This bound brings a guarantee on the choice of the good attribute. The Hoeffding bound was afterwards often used to build online decision tree: VFDTc [12], CVFDT [17], IADEM [22], "ensemble of Hoeffding trees" [19]... The Hoeffding bound is also used in this article to construct the proposed online tree. Detailed description of this bound is presented below.

**Hoeffding bound** provides an upper bound on the observed mean of a random variable and its true mean. Consider a real-valued random variable $r$ whose range is $R$. Suppose we have made $n$ independent observations of this variable, and computed its mean $\bar{r}$. The Hoeffding bound states that, with probability $1-\delta$, the true mean of the variable is at least $\bar{r}-\epsilon$, where $\epsilon = \sqrt{\frac{R^2}{2n} ln(\frac{1}{\delta})}$. The interest of this bound is that it depends only on: i) the range $R$, ii) the number of observations $n$, iii) the desired confidence $\delta$.

This bound guarantees the choice (with a probability $1-\delta$) of the attribute to cut. The bound is applied on the average of an evaluation split criterion $G$. The best attribute $a$ is definitively considered better than the best second attribute $b$ if $\bar{G}_a - \bar{G}_b > \epsilon$.

## 2.3 Local model

Gama et al. [12] observed empirically that 100 to 1000 examples are needed before transforming a leaf into a node. These examples in leaves are not used to improve the global model as long as the leaf is not transformed into a node. They suggest using these examples by adding in every leaf a local model (called "functional tree leaves" by Gama et al.). The reader can note that such kind of technique, which uses local models positioned in leaves, exists for a long time. For example the algorithm NBTree [20] uses a decision tree with a naïve Bayes classifier in leaves.

A good local model for online decision trees has to consume a small amount of memory, be fast to build and be fast to return a prediction. It has to be in adequacy with the summaries built in leaves. A good ability to predict with a small number of examples is required because summaries in leaves can be based on few examples. A study [24] on the speed (in number of examples) of different classifiers shows that the forests of tree and the naïve Bayes classifier are classifiers which require few examples to learn. Among those two classifiers only the naïve Bayes classifier respects at best the conditions required by the online construction. Indeed, it does not require additional memory if the summary returns a density estimation by interval (this is the case for all the summaries presented previously). Furthermore it is fast to elaborate and has a low algorithmic complexity to predict. This classifier was also used in VFDTc and improved the prediction on several benchmark datasets [12].

## 3 Our approach

This paper introduces current works on the construction of online decision trees based on order statistics. For summaries we choose methods based on order statistics and addressing the precision / memory tradeoff. For the criterion, the choice turns to the MODL [5] method which finds Bayes optimal cut points with order statistics. The MODL approach also provides robust density estimation that can be used by a local model. In our case the naïve Bayes classifier is chosen.

### 3.1 Summaries in the leaves

The summaries used in the proposed approach have at the same time a fixed memory consumption and strong guarantees on the error over the counts. These summaries are described below.

#### *3.1.1 Numerical attributes: quantiles summaries (GK)*

Quantiles provides order statistics on the data. The $\phi$-quantile, with $\phi \in [0, 1]$ is defined as the element in the position $\lceil \phi N \rceil$ on a sorted list of $N$ values. $\epsilon$ is the maximum error on the position of the element: an element is an $\epsilon$ approximation of a $\phi - quantile$ if its rank is between $\lceil (\phi - \epsilon)\, N \rceil$ and $\lceil (\phi + \epsilon)\, N \rceil$. It corresponds to an "Equal Frequency" discretization; the number of quantiles being in that case the number of intervals.

The GK quantiles summary [14] is an algorithm to compute quantiles using a memory of $O(\frac{1}{\epsilon} log(\epsilon N))$ in the worst case. This method does not need to know the size of the data in advance and is insensitive to the arrival order of the examples. The algorithm can be configured either with the number of quantiles or with a bound on the error. Its internal structure is based on a list of tuples $< v_i, g_i, \Delta_i >$ where :

- $v_i$ is a value of an explanatory variable of the data stream
- $g_i$ corresponds to the number of values between $v_{i-1}$ and $v_i$
- $\Delta_i$ is the maximal error on $g_i$

Some insights about the choice of this summary are given in [23]. This method is studied in [21] and implemented with a summary per class and per attribute. It is evaluated as being less successful than the GA or VFML methods. However we adapted the GK summary to store directly the class counts in tuples. Therefore just a summary per attribute is needed and not a summary per class and per attribute. Experimentally this modification improves the quality of prediction (due to the lack of place the experiments related to this point are not presented in this article).

#### *3.1.2 Categorical attributes: Count-min Sketch (CMS)*

The purpose of the Count-min Sketch [7] is to find the top-k most frequent values in a data stream with a maximal error $\epsilon$ on their counts. A counting matrix of size $t \times b$ is used for the storage. This method uses $t$ hash functions $h_i$ in $\{1, \ldots, b\}$ which select the cell in the matrix to increment: $\forall i = 1, \ldots, t \quad h_i(x) \leftarrow h_i(x) + 1$.

The values for $t$ and $b$ is computed by means of two parameters $\delta$ and $\epsilon$. To estimate the frequency $\hat{f}$ of a value with an error inferior to $\epsilon n$ and a probability of at least $1 - \delta$ then it is necessary to take $t = log\frac{1}{\delta}$ and $b = O(\frac{1}{\epsilon})$. The frequency of a value $v$ is estimated by the minimum of $h_i(x)$: $\hat{f} = \underset{i}{argmin}(h_i(x))$.

The Count-min Sketch is adapted to store class counts. Figure 2 presents this adaptation.

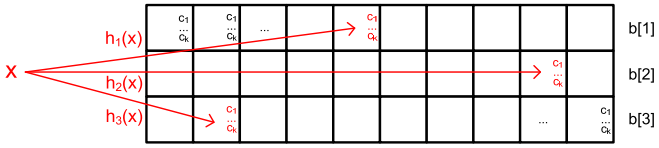


**Figure 2.** Count-min Sketch adapted for classification.

Using the Count-min Sketch is relevant when the number of different values is large. In the opposite case a simple counting is better as it neither introduces errors nor consumes a large amount of memory. Some other methods with guarantee could be used as well [6].

### 3.2 Criterion

To be coherent with our summaries (GK and CMS), the MODL criterion based on order statistics is chosen to find cuts and groups respectively for a numerical and a categorical variable. The MODL approach, designed for discretization and value groupings, also returns the quality of a cut or a grouping. This indication $l$, named 'level' ($l \in [0, 1]$), corresponds to a compression ratio. It indicates the information contained in a numerical or categorical variable when considering a target variable.

The MODL discretization [5] and grouping [4] are supervised and do not need any parameters. They are based on class counts and evaluate all possible intervals for the numerical variables and groups for the categorical variables. The quality evaluation of the model is based on a Bayesian approach. For numerical variables, its purpose is to find the best discretization parameters: number of intervals, frontiers of the intervals and class distribution in the intervals in a Bayesian sense. For categorical variables the method performs grouping in a similar way.

The tree, proposed in this article, is built online in the same manner as VFDT. It uses the Hoeffding bound but the Entropy Gain is replaced by the MODL criterion. The MODL criterion has an additional advantage because it returns a value of criterion $l > 0$ if and only if the discretization model / grouping model is better than the model which returns the majority class. This property allows an automatic "pre-pruning" of the tree while in VFDT this pruning must be separately implemented. What is more this criterion estimates not only binary cut points but can also estimate many cuts for each attribute, which allows to build trees having nodes with more than two sons.

### 3.3 Local model: density estimation with a two levels approach based on order statistics

We choose the naïve Bayes classifier as the local model in the leaves for our tree. Naïve Bayes has good performances when it is built with few data and its prediction has low algorithmic complexity. This classifier requires an estimation of the class conditional density. This density is estimated for all intervals or groups of values calculated by the MODL method applied respectively to GK or CMS summaries contained in the leaves. This discretization, applied on the summary, corresponds to the two levels discretization method proposed by Gama with its PiD method (see 2.1.1). However our two levels approach is completely based on order statistics and can treat indifferently numerical or categorical attributes. For numerical variables, we choose as the first level the GK quantiles summary and as the second level the MODL discretization method. For categorical variables, we choose as the first level the CMS summary (or a simple counting) and as the second level the MODL grouping method.

The main advantages of the MODL method are its robustness and the absence of parameters. The MODL approach discretizes a numerical variable in several intervals if and only if this discretization is better (in a Bayesian sense) than the model with a single interval (the approach is similar for grouping). If the most probable model is with an interval it means that the attribute is not informative given the observed data. In that case the majority class is predicted. Kirkby in [19] studies exactly this behavior by comparing Hoeffding trees predicting the majority class in leaves and those having a naïve Bayes predictor in leaves. He observes that sometimes the majority class prediction is better than naïve Bayes. Out of this fact he proposes a method choosing automatically either one model or the other by estimating their error rate on the examples which pass in leaves. As the MODL discretization returns just one interval if a variable is not informative, our two levels approach based on MODL method should predict the majority class as well in that case.

## 4 Experimentations

This section aims to compare our approach to the existing methods described in section 2. We first present the data streams used, then the compared methods and finally the obtained results.

### 4.1 Data streams used

In order to have enough data for testing online algorithms, artificial generators have been developed to generate stream containing millions of examples. An overview of these generators is presented in [19] and [10]. For the experiments in this paper the following generators were used:

- Random RBF data: is generated by first creating a random set of centers for each class. Each center is randomly assigned a weight, a central point per attribute, and a standard deviation. To generate new instances, a center is chosen at random taking the weights of each center into consideration. Attribute values are randomly generated and offset from the center, where the overall vector has been scaled so that its length equals a value sampled randomly from the Gaussian distribution of the center. The particular center chosen determines the class of the instance. Random RBF data contains only numeric attributes as it is non-trivial to include nominal values. We used 1000 centers and 50 attributes in our experiments.
- Random Tree: data generated by a randomly constructed decision tree consisting of $rt1$ nominal attributes with $rt2$ values each, $rt3$ numeric attributes, $rt4$ classes, a tree depth of $rt5$, with leaves starting at level $rt6$ and a $rt7$ chance of leaves. The final tree has a certain number of nodes and leaves. Different settings for parameters $rt1, ..., rt6$ generate different data streams. This generator gives preferential treatment to decision tree classifiers. We used the following parameters: $rt1 = 10, rt2 = 5, rt3 = 10, rt4 = 3, rt5 = 5, rt6 = 0.15$.
- Waveform: it produces 21 attributes, all of which include noise. It differentiates between 3 different classes of waves, each of which is generated from a combination of two or three base waves. This generator is based on a normal law and gives a preferential treatment to classifiers which assume that data follow a normal law.
- Function (F6): data generation based on the Function F6 described in the appendix of [1]. The stream contains 6 numerical attributes and 3 categorical attributes and a level noise of 5%.

### 4.2 Algorithms compared

For our experiments we used the MOA toolbox: Massive Online Analysis [2] developed by the university of Waikato which takes up the VFML library supplied by Hulten and Domingos [18]. This toolbox contains stream generators and many online algorithms with which we wish to compare. We made an extension package[3] for the MOA toolbox with our new summaries, new split criterion and new local model.

All the tested trees come from the same algorithm based on the Hoeffding trees. This algorithm contains three parameters: i) the number of examples to be considered before trying to find a cut point; ii) the confidence in the split regarding the Hoeffding bound and iii) the parameter $\tau$ which is used to force the choice between two attributes when the criterion difference is too small so that the Hoeffding bound can not decide between them. The configuration of all the trees versions tested here is the same, namely: $n = 200$, $\delta = 10^{-6}$, $\tau = 0.05$. The same settings were used and described by Kirkby in [19].

The summaries in leaves for categorical attributes do not use, in these experiments, the count-min sketch and the MODL grouping because the datasets used contain few different values. The summary, for the categorical attributes, is thus based on a simple counting. The numerical attributes are summarized either by a Gaussian approximation or by quantiles using the GK method.

The attribute and the cut point selected to transform a leaf into a node are the ones maximizing the MODL criterion and which are non zero ($l > 0$). Only binary splits for numerical attributes are considered. For categorical attributes the criterion is evaluated on the model with a leaf per value.

The tested approaches were either without local model or with a naïve Bayes classifier in every leaf. Table 1 presents a synthetic view of the algorithms presented below:

- **HT: Hoeffding Tree -** This algorithm corresponds to the Hoeffding Tree named VFDT in [8]. The summaries are based on density estimation using a Gaussian by class (estimated as being the most successful in [21]). For numerical attributes 10 binary cuts by class, taken between the minimum and the maximum observed, are estimated. This tree does not possess a classifier in leaves.

[3] Extension available here: `http://chercheurs.lille.inria.fr/salperwy/index.php?id=moa-extension`

- **HTNB: Hoeffding Tree using a naïve Bayes -** This version is identical to the previous one but possesses a naïve Bayes classifier in the leaves. The conditional densities are estimated: i) for numerical variables by means of the Gaussian estimation parametrized by 3 values stored in the summary ($\mu$, $\sigma$, $n$); and ii) for categorical variables by means of the counts per class and value.
- **HTmGKc: Hoeffding Tree using MODL split criterion -** This version corresponds to the version which we described in the section 3. The summary for the numerical variables is based on the GK quantiles summary with tuples containing directly the counts per class. Each variable is summarized by 10 tuples. 10 possible cut points are taken from the summary and are evaluated with the MODL criterion described in section 3.2. This version does not possess local model in leaves.
- **HTmGKcNBm: Hoeffding Tree using MODL split criterion and naïve Bayes classifier -** This classifier is identical to the previous version "HTmGKc" but the leaves contain a naïve Bayes classifier. Conditional estimation densities needed by the classifier come from counts per class per value for categorical variable. To have more robust estimation on numerical variables density estimations are computed on the intervals found by the MODL discretization [5].

| Method | Summary | Criterion | Discretization | Local model |
|---|---|---|---|---|
| HT | Gaussian | Entropy Gain | - | - |
| HTNB | Gaussian | Entropy Gain | - | NB |
| HTmGKc | GK 10 tuples | MODL Level | - | - |
| HTmGKcNBm | GK 10 tuples | MODL Level | MODL on GK | NB |

**Table 1.** Specification of the tested algorithm (NB: naïve Bayes)

### 4.3 Results

Our experimental results are presented in Figure 3 where a curve is plotted for each data set. The evaluation method is the "hold-out test set" and has been chosen among methods described in [13]. For each generator a stream containing 11 million examples has been created. The 1st million of examples is used as a test set. The remaining 10 millions examples represent the training dataset. Trees accuracies are evaluated every 300,000 examples. In a general way our method behaves well on the tested streams and is competitive compared to the other methods. The MODL split criterion applied on the GK summaries, for numerical variables, and on the counts per class, for categorical variables, is globally better than the Entropy Gain criterion calculated on Gaussian summaries, for numerical variables, and the counts per class, for the categorical variables.

The contribution of the naïve Bayes classifier in leaves is debatable with Gaussian summaries because sometimes the accuracy of the global classifier (the entire tree) is either significantly improved (WaveForm) or significantly degraded (F6). With our two levels summaries, the naïve Bayes classifier improves the accuracy of the entire tree especially in the beginning of training. There is no degradation thanks to the robustness of the MODL approach. It creates intervals only if they contain information. If the variable is not informative no discretization model is proposed. The estimation based on these intervals is then provided to the naïve Bayes classifier.

The density estimation using a Gaussian approximation gives better results on "WaveForm" data streams. This result is not surprising because it seems normal that the naïve Bayes classifier having for density estimation a Gaussian approximation works well on data coming from "WaveForm" generator which uses a Gaussian to generate data.

The behavior of the algorithms in low memory environments was not studied in this article. However there are several techniques that we could apply. One of the simplest methods consists in deleting summaries for the less interesting variables. The MODL level is calculated for all variables and gives a precise indication of the information contained in a variable. Therefore it could be used to rank variables and choose which ones to remove from summaries. A second technique consists in deactivating the less promising leaves and thus limits the development of a part of the tree.

The concept drift (see `http://www.cs.waikato.ac.nz/~abifet/PAKDD2011/`) was not studied in this paper but several techniques for concept drift detection will be developed in future works using the summaries and the MODL approach. One of them will use two summaries: one for the past distribution and one for the current. These two summaries will be used to detect drift using the MODL approach (in a similar way to [3]).

## 5 Conclusion

This paper constitutes a current work on an incremental method used to build an online decision tree with order statistics. Our approach was presented and compared to the previous works on various points needed to elaborate an online decision tree. The GK quantiles summaries and the CMS summary are used in leaves. The MODL method uses these summaries to provide at the same time a cut criterion / value grouping and a robust density estimation per interval / group. This estimation allows afterwards to build a naïve Bayes classifier in the leaves. The experiments showed that our tree performs well compared to existing methods thanks to the robust density estimation provided by the MODL discretization/grouping. Particularly the local classifier always improves the prediction in the early learning stage. The next steps to confirm the interest of our approach are: (i) experiment more data streams and real dataset, (ii) control memory consumption and (iii) extend the algorithm to handle drifts.

## REFERENCES


[1] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami, 'Database mining: A performance perspective', *IEEE Transactions on Knowledge and Data Engineering*, **5**, 914–925, (1993).

[2] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà, 'New ensemble methods for evolving data streams', *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, 139, (2009).

[3] Alexis Bondu and Marc Boullé, 'A supervised approach for change detection in data streams', in *IJCNN*, pp. 519–526, (2011).

[4] M. Boullé, 'A grouping method for categorical attributes having very large number of values', *Machine Learning and Data Mining in Pattern Recognition*, 228–242, (2005).

[5] Marc Boullé, 'MODL: A Bayes optimal discretization method for continuous attributes', *Machine Learning*, **65**(1), 131–165, (May 2006).

[6] G. Cormode and M. Hadjieleftheriou, 'Finding frequent items in data streams', *Proceedings of the VLDB Endowment*, **1**(2), 1530—-1541, (January 2008).

[7] Graham Cormode and S. Muthukrishnan, 'An improved data stream summary: the count-min sketch and its applications', *Journal of Algorithms*, **55**(1), 58–75, (April 2005).

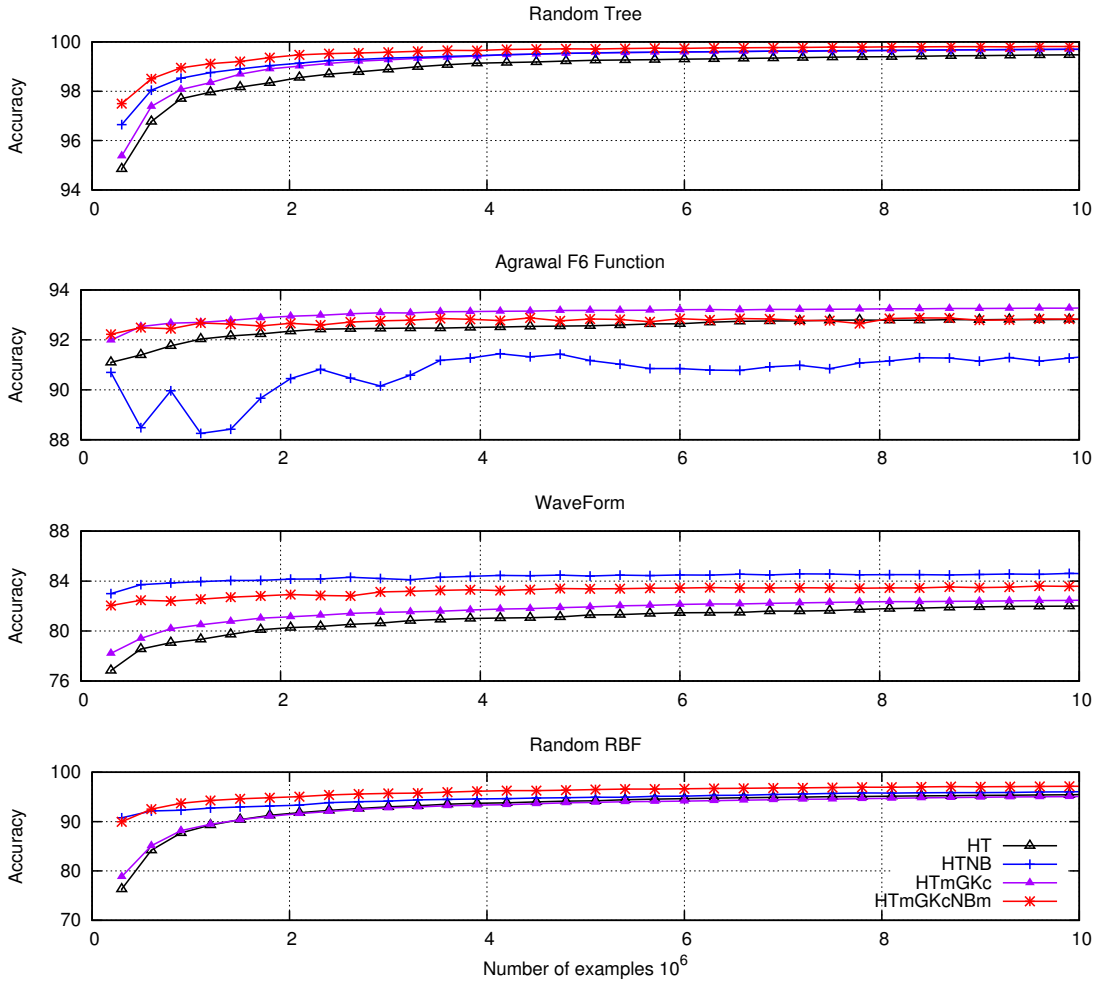


**Figure 3.** Accuracy versus the number of examples used to train the entire tree.


[8] P. Domingos and G. Hulten, 'Mining high-speed data streams', in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71–80. ACM New York, NY, USA, (2000).

[9] R. Féraud, M. Boullé, F. Clérot, F. Fessant, and V. Lemaire, 'The orange customer analysis platform', in *Industrial Conference on Data Mining (ICDM)*, pp. 584–594, (2010).

[10] J. Gama, *Knowledge Discovery from Data Streams*, Chapman and Hall/CRC Press, 2010.

[11] J. Gama and Carlos Pinto, 'Discretization from data streams: applications to histograms and data mining', in *Proceedings of the 2006 ACM symposium on Applied computing*, pp. 662–667, (2006).

[12] J. Gama, R. Rocha, and P. Medas, 'Accurate decision trees for mining high-speed data streams', in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 523–528. ACM New York, NY, USA, (2003).

[13] J. Gama, P.P. Rodrigues, and R. Sebastião, 'Evaluating algorithms that learn from data streams', in *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 1496–1500. ACM New York, (2009).

[14] M. Greenwald and S. Khanna, 'Space-efficient online computation of quantile summaries', *ACM SIGMOD Record*, **30**(2), 58–66, (2001).

[15] I Guyon, V. Lemaire, G. Dror, and D. Vogel, 'Analysis of the kdd cup 2009: Fast scoring on a large orange customer database', *JMLR: Workshop and Conference Proceedings*, **7**, 1–22, (2009).

[16] W Hoeffding, 'Probability inequalities for sums of bounded random variables', *Journal of the American Statistical Association*, (1963).

[17] G. Hulten, L. Spencer, and P. Domingos, 'Mining time-changing data streams', in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 97–106. ACM New York, NY, USA, (2001).

[18] Geoff Hulten and Pedro Domingos, 'VFML – a toolkit for mining high-speed time-changing data streams'. 2003.

[19] Richard Kirkby, *Improving Hoeffding Trees*, Ph.D. dissertation, University of Waikato, 2008.

[20] Ron Kohavi, 'Scaling up the accuracy of naive-Bayes classifiers: A decision-tree hybrid', in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, volume 7. Menlo Park, USA: AAAI Press, (1996).

[21] Bernhard Pfahringer, Geoffrey Holmes, and Richard Kirkby, 'Handling numeric attributes in hoeffding trees', *Advances in Knowledge Discovery and Data Mining*, 296–307, (2008).

[22] G. Ramos-Jimenez, J. del Campo-Avila, and R. Morales-Bueno, 'Incremental algorithm driven by error margins', *Lecture Notes in Computer Science*, **4265**, 358, (2006).

[23] C. Salperwyck and V. Lemaire, 'Incremental discretization for supervised learning', *CLADAG: CLAssification and Data Analysis Group - 8th International Meeting of the Italian Statistical Society*, (2011).

[24] C. Salperwyck and V. Lemaire, 'Learning with few examples: an empirical study on leading classifiers', in *The 2011 International Joint Conference on Neural Networks (IJCNN)*, pp. 1010–1019. IEEE, (2011).