Classifying High-Speed Data Streams Using Statistical Decision Trees

Mirela Teixeira Cazzolato, Marcela Xavier Ribeiro

Universidade Federal de São Carlos
Departamento de Computação - São Carlos, Brazil
{mirela.cazzolato, marcela}@dc.ufscar.br

Abstract. Every day a large amount of data is collected by applications such as credit card transactions, monitoring networks and sensors. This type of data, called data streams, are generated in an automatic way, and its storage and knowledge extraction techniques differ from those used on traditional data. The classification task builds a model to describe and distinguish classes of data. In the context of data stream classification, many incremental techniques have been proposed. The existent methods tent to improve the classification accuracy as the number of processed examples increases. However, this characteristic makes the techniques conservative when the dataset is not too big, since they are dependent on the amount of data available. In this work we propose two algorithms that are not dependent on the number of examples read and that present a high accuracy and low execution time. We describe an incremental decision tree algorithm called StARMiner Tree (ST), which is based on Very Fast Decision Tree (VFDT) system, deals with numerical data and uses a method based on statistics as the heuristic to decide when to split a node, and also to choose the best attribute to be used in the test node. We also present a non-parametric version of ST called AST. We applied ST and AST in four datasets, one synthetic and three real-world, comparing their performance to the VFDT and VFDTcNB, which is an extension of VFDT and uses Naïve Bayes in the leaves. In all experiments ST and AST achieved better accuracy results, dealing well with noise data, describing the data from the earliest examples and maintaining a good execution time. The obtained results indicate that ST and AST are well-suited for data streams classification.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications—*Data Mining*

Keywords: automatic StARMiner tree, classification, data stream mining, incremental decision trees, StARMiner tree, VFDT

1. INTRODUCTION AND MOTIVATION

Data streams are obtained in a continuous way by applications such as sensor networks, credit card transactions and financial applications, generating large volumes of data. The classification task builds a model to describe classes of data. Traditional techniques for data mining require multiple scans on the data, which is not feasible for stream data [Rutkowski et al. 2013]. In the data streams context, incremental techniques are used to eliminate the need of rebuilding the model every time a new example arrives. Incremental decision trees are constructed based on sufficient statistics extracted from the examples, and they generally scan the data once.

According to Zia-Ur Rehman et al. [2012], classification using decision trees is a widely studied problem in data streams, and the main concern is when to split a decision node into multiple leaves. The VFDT (Very Fast Decision Tree) algorithm [Domingos and Hulten 2000] is one of the well-known decision tree algorithms for data streams classification. VFDT uses Hoeffding inequality to achieve a probabilistic bound on the accuracy of the constructed tree, ensuring that its output is asymptotically nearly identical to the output of a conventional learner. A disadvantage of VFDT is that the Hoeffding bound depends on the number of read examples, becoming conservative and requiring more examples

We would like to thank CAPES, CNPq and FAPESP for the financial support.

than necessary to describe the data. Taking this gap into account we propose a parametric incremental decision tree algorithm called StARMiner Tree (ST). ST is based on the general structure of VFDT, and proposes a decision tree model constructed from numeric data, using statistics as a heuristic to decide when to perform the division of a tree node and which attribute to use in the internal test. This heuristic uses the mean and deviation, and is appropriated to classify databases that follow the Normal Distribution. In general, events in a data stream are the result of many factors operating independently. A sum of independent random variables follow the Normal Distribution. Because of it, we can approximate most data streams distributions to the normal model. This is true specially in the context of this article where the number of data analyzed in each step tends to be large. Unlike VFDT, our algorithm is not dependent on the number of read examples, and can describe the data since the first examples. We also describe a non-parametric version of ST, called Automatic StARMiner Tree (AST). In this work we describe the behavior of the proposed algorithms using different datasets. In order to validate ST and AST we applied them in four experiments, one using a synthetic dataset and three using real-world datasets, comparing the obtained results with VFDT and VFDTcNB [Gama et al. 2003], which is an extension of VFDT and uses Naïve Bayes in the leaves.

This article is organized as follows. Section 2 presents the theoretical background of data streams classification using decision trees. In Section 3, we describe our proposed algorithms, ST and AST. Section 4 presents the experimental analysis and Section 5 summarizes the obtained results and the future work.

2. RELATED WORK

In this section we describe some of the main decision tree methods for data streams classification, briefly highlighting the contributions of each one. Domingos and Hulten [2000] presented a basic decision tree algorithm for data stream classification called Hoeffding Tree (HT). In the same work, it was proposed a framework based on HT, called VFDT (Very Fast Decision Tree) system. VFDT allows the use of Information Gain and Gini Index as the attribute evaluation measure, and adds several refinements to the original algorithm, HT. In order to find the best attribute to test at a given node, it may be sufficient to consider only a small subset of training examples that pass through that node [Domingos and Hulten 2000]. After processing a given stream of examples, the first ones will be used to choose the root test node. Once the root attribute is chosen, the next examples will be passed down to the corresponding leaves of the tree and used to choose the appropriate attributes there, and so on recursively. When a new example is read at a leaf l, the algorithm extracts only the sufficient statistics needed to maintain the model. After n_{min} examples seen at l, VFDT verifies if l should be divided, becoming a test node. The parameter n_{min} is used to reduce the time spent to decide about the split of a leaf, since it is not computational efficient to try to split a node every time a new example arrives, and one example itself have a small influence on the decision of split or not a node.

VFDT uses a statistical result known as Hoeffding bound (see Equation 1) to decide how many examples are necessary to be observed at each leaf l before split it. According to Domingos and Hulten [2000] and Bifet [2010]: let r be a real-valued random variable whose range is R (e.g., for a probability the range is one, and for an information gain the range is $log(c)$, where c is the number of classes). Suppose we have made n independent observations of this variable, and computed their 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

$$\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}. \tag{1}$$

The VFDT system is a very popular decision tree algorithm which has been constantly adapted and modified. Gama et al. [2003] proposed an extension of VFDT called VFDTcNB. VFDTcNB uses Information Gain as the heuristic to split a node, handles numeric attributes and uses Naïve Bayes at the leaves, which according to the authors it is a more powerful technique to classify examples.

The OVFDT (Optimized Very Fast Decision Tree) was proposed by Yang and Fong [2011] to control the tree size while keeping a good accuracy. According to the authors, this is enabled by using an adaptive tie threshold and incremental pruning in tree induction. Li et al. [2009] proposed the OcVFDT (One-class Very Fast Decision Tree) algorithm, applied to one-class classification. It is based on VFDT and POSC4.5 [Letouzey et al. 2000], an induction tree algorithm built from positive and unlabelled examples only. The EBT (Empirical Bernstein Tree) was proposed by Zia-Ur Rehman et al. [2012] to replace the Hoeffding bound by the empirical Bernstein's bound, in order to achieve a better probabilistic bound on the accuracy of the decision tree.

One of the issues regarding the data stream classification is the concept drift problem, which occurs when the concept defining the target being learned begins to shift over time. Hulten et al. [2001] proposed the CVFDT system, an efficient algorithm based on VFDT to construct decision trees from continuous-changing data streams. The algorithm grows alternative subtrees, and whenever the current model becomes questionable it is replaced by a more accurate alternative subtree. Another decision tree algorithm proposed to the concept drift problem is the H-CVFDT [Ouyang et al. 2008]. According to the authors the H-CVFDT uses an extended hash table and classifies continuous attributes of data streams with concept drift.

VFDT has the ability to handle large amounts of data maintaining a good accuracy, with theoretical guarantees concerning the use of Hoeffding bound. A disadvantage of being so general is that the Hoeffding bound is conservative, requiring more examples than necessary to describe the data [Zia-Ur Rehman et al. 2012]. We are particularly interested in this problem, because in some domains it is important to have a faster convergence, with a model that describes the data since the first examples available. To perform this, we adapted the VFDT algorithm to fast converge when choosing an node attribute. We replaced the use of the $\bar{G}$ (Information Gain or Gini Index) and the Hoeffding bound to an statistical evaluation of the attribute behavior based in the mean, standard deviation, and a hypotheses test, employing some ideas presented by Ribeiro et al. [2005]. The method proposed by Ribeiro et al. [2005] identifies the most significant attributes in a dataset, i.e., the ones which describe more classes and have uniform behavior. The original algorithm uses three parameters set by the user. In the work of Watanabe et al. [2012], an automatic estimation of these parameters was proposed, and we also adapted the automatic parameter estimation to work in our proposed method.

In this article we describe the StARMiner Tree (ST) and the Automatic StARMiner Tree (AST) algorithms, which are based on the principles of VFDT, but utilize a method based on statistics as a heuristic to choose the best attribute to be used in the test at a node. The details of implementation are described in the next section.

3. PROPOSED ALGORITHMS

In this section we first describe the StARMiner Tree (ST), an statistical incremental decision tree algorithm. When a new example arrives, ST classifies it into its correspondent leaf, according to the class value. The ST decision tree is built incrementally over time, by deciding to split a node when a minimum number of examples n_{min}, from more than one class, is read in that leaf node. In the decision tree, each internal node corresponds to a test using an attribute, and each leaf l corresponds to a class value, that stores the sufficient statistics extracted from the examples seen at l. The sufficient statistics kept for each attribute at l are, for each class value k, the mean μ_{lk}, the standard deviation σ_{lk} and the number of examples observed at that leaf n_{lk}. To convert a leaf into a conditional node, the algorithm verifies the best attribute to be used in the test based on the mean, standard deviation, and a hypotheses test (see conditions 1, 2 and 3). In fact, a node split occurs when there is sufficient evidence that a new conditional node is needed. To check this, at least n_{min} examples should be read in that node. The decision of split a node is computed based in an incremental technique that compute the mean, standard deviation over the attribute values for each class, and execute an statistical hypothesis test over the mean of each class. The ST algorithm is

Algorithm 1: The StARMiner Tree

Input : $\Delta\mu_{min}$, the minimum difference allowed; σ_{max}, the maximum deviation allowed; γ_{min}, the minimum confidence
Output: A decision tree ST.

```
1  Let ST be a tree with a single leaf (the root)
2  foreach training example e do
3      Sort example e into leaf l using ST
4      Update sufficient statistics in l (μ_lk, σ_lk and n_lk for each class value k of the dataset)
5      Increment n_l, the number of examples seen at l
6      if n_l mod n_min = 0 and all examples seen at l are not all of same class then
7          L = SelectBestAttributes(Δμ_min, σ_max, γ_min)
8          if L ≠ ∅ then
9              Let X_a be the best attribute of L
10             Replace l with an internal node that splits on X_a
11             foreach branch generated by the split do
12                 Add a new leaf with initialized sufficient statistics at the leaf l (μ_lk, σ_lk and n_lk for each class
                   value k of the dataset)
```

Function SelectBestAttributes($\Delta\mu_{min}$, σ_{max}, γ_{min})

```
1  Let L be the list attributes L = {a_1, ..., a_i}
2  Let L_1 be an empty list of attributes
3  foreach attribute a_i ∈ L do
4      foreach class x_j ∈ X do
5          if (μ_{a_i}(T_{x_j}) − μ_{a_i}(T − T_{x_j})) ≥ Δμ_min then
6              if σ_{a_i}(T_{x_j}) ≤ σ_max then
7                  Compute Z_{i_j} = (μ_{a_i}(T_{x_j}) − μ_{a_i}(T − T_{x_j})) / (σ_{a_i}(T_{x_j}) / √(|T_x|))
8                  Get Z_1 and Z_2 values
9                  if Z_{i_j} < Z_1 or Z_{i_j} > Z_2 then
10                     Add a_i to L_1
11 return L_1
```

presented in Algorithm 1, followed by a more detailed explanation.

The algorithm starts with a tree ST containing a single empty leaf node (line 1). Each new example e is classified into its corresponding leaf l (according to its attribute values) using ST (lines 2 and 3). At line 4 the sufficient statistics at l (μ_{lk}, σ_{lk}, and the number of examples observed n_{lk} for each class k) necessary to build the tree are collected and updated. The number of examples seen at leaf l (n_l) is incremented at line 5. Although ST is not dependent on the number of read examples, we continue to use the n_{min} parameter because it is not computational efficient to try to split a node every time a new example arrives, and one example has a small influence on the decision of split or not a node. Thus, when a minimal number of examples n_{min} is read, it is verified if all data seen at l are not all of the same class (line 6). If they are of the same class, the algorithm continues to receive examples to process until $n_l \ mod \ n_{min} = 0$, i.e. until the algorithm receives n_{min} more examples. If the conditions in line 6 are satisfied, the Function SelectBestAttributes is called in line 7.

The Function SelectBestAttributes selects the attributes that satisfy both of the following conditions.

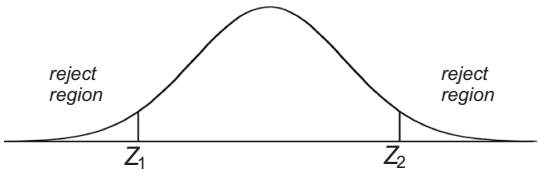
Condition 1. The a_i attribute should have a behavior at class x_j different to its behavior in other classes (line 5 of Function SelectBestAttributes);

Condition 2. The a_i attribute should present an uniform behavior in the data from class x_j (line 6 of Function SelectBestAttributes).

To satisfy these conditions the algorithm uses the two following constraints of interest, received as parameter: (i) $\Delta\mu_{min}$ is the minimum difference between the means of the attribute a_i at examples

	γ_{min}		
	0.9	**0.95**	**0.99**
Z_1	-1.64	-1.96	-2.58
Z_2	1.64	1.96	2.58

(a) Z values according to γ_{min}



(b) Reject regions

Fig. 1: The rejected regions obtained according to the γ_{min} parameter

from class x_j, and the examples from the remaining classes; and (ii) σ_{max} is maximum deviation allowed for the attribute a_i at examples from class x_j.

Finally, the ST uses the constraint of interest γ_{min} to verify the followed condition.

Condition 3. Reject with minimum confidence γ_{min} the hypothesis H that the means $\mu_{a_i}(T_{x_j})$ and $\mu_{a_i}(T - T_{x_j})$ are statistically equal at the sets of examples from class x_j and from the other classes, i.e. T_{x_j} and $T - T_{x_j}$ respectively.

This condition is verified according to equation presented at the line 7 of Function SelectBestAttributes. At line 8 (Function SelectBestAttributes) the Z values are obtained according to the γ_{min} parameter, as it is shown in Figure 1(a). At line 9 (Function SelectBestAttributes) it is checked if the Z_{i_j} value is between the rejected regions, as it is illustrated in Figure 1 (b). If this condition is satisfied, the attribute is added to the list L_1. As the result, the Function SelectBestAttributes returns a list L_1 of attributes that satisfy the three conditions described above. At line 8 of the Algorithm 1 it is checked if the list L of attributes returned is not empty, i.e., if at least one attribute has been selected. If more than one attribute has been selected, at line 9 (Algorithm 1) it is chosen the attribute X_a which, respectively, identifies more classes, have higher $\mu_{a_i}(T - T_{x_j})$ and lower $\sigma_{a_i}(T_{x_j})$. Then X_a is used to split the leaf l turning it into an internal node at line 10. For each branch of the split the algorithm adds a new leaf l and initialize the sufficient statistics μ_{lk}, σ_{lk} and n_{lk} (for each class value k) (lines 12 and 13, Algorithm 1). ST can generate and adapt the built model without the restriction of reading a large number of examples, being more flexible than VFDT in this context.

The estimation of parameters is not always easy to accomplish, sometimes requiring prior knowledge of the used data. In the work of Watanabe et al. [2012] an automatic estimation of parameters was proposed to work in an algorithm based on statistical association rules, used to automatically select the most significant features to produce rules. The algorithm uses the principles presented by Ribeiro et al. [2005], also used to construct our proposed algorithm, ST. In this work we propose and extension of ST, the Automatic StARMiner Tree (AST), which uses the automatic estimation proposed by Watanabe et al. [2012] to calculate the constraints $\Delta\mu_{min}$ and σ_{max}, given as parameters in the original algorithm ST. These values are computed as follows:

$$\Delta\mu_{min} = min(m_{a_i}) + (max(m_{a_i}) - min(m_{a_i}))^{2.5}, \tag{2}$$

where $m_{a_i} = min(||\mu_{a_i}(T_{x_j}) - \mu_{a_i}(T - T_{x_j})| - |\sigma_{a_i}(T_{x_j}) - \mu_{a_i}(T_{x_j})| - |\mu_{a_i}(T - T_{x_j}) - \sigma_{a_i}(T - T_{x_j})||)$.

$$\sigma_{max} = \gamma_{min} * max(\sigma a_i(T_{x_j})) \tag{3}$$

In the Equation 2 the exponent has been modified, because it showed better results in empirical experiments. Equation 3 shows the maximum deviation allowed σ_{max}, which is the minimum confidence multiplied by the maximum deviation of an attribute in a class value. AST is presented in Algorithm 2. The algorithm is very similar to ST. The difference is that the AST receives as input just the minimum confidence, and the two other constraints μ_{min} and Δ_{max} are computed (line 7).

In the next section we present an experimental comparison of VFDT, VFDTcNB (an extension of VFDT that uses Naïve Bayes in the leaves), and our proposed algorithms, ST and AST.

Algorithm 2: The Automatic StARMiner Tree

```
Input  : γ_min, the minimum confidence
Output: A decision tree AST.
Let AST be a tree with a single leaf (the root)
foreach training example e do
    Sort example e into leaf l using AST
    Update sufficient statistics in l (μ_lk, σ_lk and n_lk for each class value k of the dataset)
    Increment n_l, the number of examples seen at l
    if n_l mod n_min = 0 and all examples seen at l are not all of same class then
        Compute m_{a_i}, Δμ_min and σ_max
        L = SelectBestAttributes(Δμ_min, σ_max, γ_min)
        if L ≠ ∅ then
            Let X_a be the best attribute of L
            Replace l with an internal node that splits on X_a
            foreach branch generated by the split do
                Add a new leaf with initialized sufficient statistics at the leaf l (μ_lk, σ_lk and n_lk for each class
                value k of the dataset)
```

4. EXPERIMENTAL ANALYSIS

In this section we present four experiments performed using datasets of different sizes, with and without noise. We compared the results of ST and AST with VFDT and VFDTcNB, in terms of final and mean accuracy (percentage of correct classifications), accuracy variance, kappa statistic, F-measure, tree size (number of nodes) and execution time (which is the average execution time of ten experiments performed for each dataset using each algorithm) using the prequential validation. The prequential validation, also called interleaved test-then-train, is a scheme used to interleave the test and train phases of the classification process. According to Patil and Attar [2011] each example can be used to test the model before it is used for training, and from this the accuracy can be incrementally updated.

4.1 Datasets and Configurations

We applied our algorithms using four datasets, one synthetic and three real-world. In Table I the datasets configurations are presented. The synthetic dataset *SEA* was generated using MOA[1], and according to Hulten et al. [2001] it contains abrupt concept drift (see more details in [Street and Kim 2001]). The *Electricity* dataset was obtained in the MOA[1] website. It contains data collected from the Australian New South Wales Electricity Market, where prices are not fixed and are affected by demand and supply of the market. We have excluded two attributes from the original dataset (date and day) and normalized all data. We inserted levels of noise in the class attribute using the WEKA tool [Witten et al. 2011]. The *Sick Euthyroid* dataset was obtained at the UCI Repository[2]. We considered only the numerical attributes in the experiments. The *Skin Segmentation* was also obtained in the UCI Repository[2]. We compare the results using the StARMiner Tree (ST) and the Automatic StARMiner Tree (AST) to VFDT and VFDTcNB. The version of VFDT used in the experiments uses majority class in the leaves, while VFDTcNB uses Naïve Bayes, as described in the Related Work section. All the experiments were performed on a i7 / 2.8GHz CPU, 8GB memory computer, considering the parameter $\gamma = 0.9$ for all experiments. The configurations not cited were considered as the default employed on MOA[1]. In the experiment using the *SEA* generator the model was tested at each 50 thousand examples. Using the Electricity dataset we tested the examples at each 3 hundreds examples. With the Sick Euthyroid dataset the algorithms read the examples 5 times, since the dataset was smaller, and the model was tested at each 150 examples, considering the grace period $n_{min} = 200$. The configurations of ST are shown in the two last columns of Table I. These values were obtained after performing a set of experiments, with different values of $\Delta\mu_{min}$ and σ_{max}.

[1] *Massive Online Analysis*: http://moa.cms.waikato.ac.nz/
[2] *UCI - Machine Learning Repository:* http://archieve.ics.uci.edu/ml

Table I: Dataset configurations and ST parameters used in the experiments

	Dataset configurations			*ST Parameters*	
Datasets	n. instances	n. attributes	instances per class	$\Delta\mu_{min}$	σ_{max}
Electricity	45,312	6	"up" = 19,237 - "down" = 26,075	0.014	0.36
SEA	1 million	3	"class1" = 356,752 - "class2" = 643,248	0.014	0.36
Skin Segmentation	245,057	4	"skin" = 50,859 - "nonskin" = 194,198	0.01	0.04
Sick Euthyroid	3,163	5	"positive" = 293 - "negative" = 2,870	0.014	0.036

Table II: Experiment results using the Skin Segmentation and Sick Euthyroid datasets

		Experiment results				
Datasets		Mean Accuracy	Final Accuracy	Accuracy Variance	Kappa	F-Measure
Skin Segmentation	ST	99.68	99.9	0.235	99.69	0.990
	AST	99.07	99.4	0.193	98.13	0.977
	VFDT	98.08	99.3	4.03	97.83	0.951
	VFDTcNB	98.8	99.2	1.067	97.49	0.972
Sick Euthyroid	ST	98.22	98.1	0.103	89.67	0.903
	AST	97.85	97.9	0.379	88.97	0.879
	VFDT	95.57	95.8	0.52	79.38	0.787
	VFDTcNB	93.57	93.1	0.48	68.45	0.706

4.2 Experiment Results

The mean and final accuracy obtained with the datasets Skin Segmentation and Sick Euthyroid are summarized in Table II. In both cases the best results were achieved by ST and AST, followed by VFDT and VFDTcNB. In Figure 2, it is shown that the ST achieved the best accuracy variation in almost all the classification process. AST also described well the data since the first examples, finishing the process with the second best accuracy. Although VFDT and VFDTcNB did not start well, after the first 100 thousand examples they achieved results very close to AST, finishing the classification almost equal. On the other hand, using the Sick Euthyroid dataset it is visible the difference between the use of ST and AST results (which are not dependent of the number of examples) and the VFDT and VFDTcNB. The accuracy variance of ST and AST was significantly smaller than VFDT and VFDTcNB, showing that the results of the proposed algorithms had a uniform behavior. The kappa statistic is a measure used to verify the difference between how much agreement is observed, in comparison with how much agreement is expected to be present. We used the reference of Landis e Koch [Emam 1999] to evaluate the agreement of the kappa values achieved. A complete agreement corresponds to 100%, and a lack of agreement corresponds to 0% (indicating random coincidences of rates). The kappa results achieved by using the Skin Segmentation dataset was "almost perfect" for all the algorithms. Using the Sick Euthyroid dataset the results of ST and AST were "almost perfect", while the results of VFDT and VFDTcNB were "substantial". Similarly, the F-measure values show that ST and AST were better than VFDT and VFDTcNB.

A significance test was performed to reject the hypothesis H_0 that the mean accuracies of the algorithms are statistically equal, with 95% of confidence. Table III shows the results of the two-sided T-Test performed between mean accuracies of two classifiers at a time, using the Skin Segmentation and the Sick Euthyroid datasets. A value "1" indicates that the row algorithm has an accuracy statistically superior to the column algorithm, rejecting H_0; "-1" indicates that the row algorithm has an accuracy value statistically inferior in comparison with the column, also rejecting H_0; a value of "0" indicates that both algorithms are statistically equal, rejecting H_0. The results show that ST had the best accuracy results in comparison with the others. Using the Skin Segmentation dataset AST obtained a better result than VFDT, and a statistically equal result than VFDTcNB. Finally, with the Sick Euthyroid dataset AST had better results than VFDT and VFDTcNB.

Figure 3 shows the tree sizes using the Skin Segmentation and Sick Euthyroid datasets. ST and AST created bigger trees in both experiments. However, the graphic presented in Figure 6 shows that the execution time of ST and AST using these two datasets was very close (when not smaller) than VFDT and VFDTcNB. This is plausible because the computation of the Information Gain used by VFDT and VFDTcNB is more computationally costly than the performance of the StARMiner split

Table III: Significance results using the Skin Segmentation and Sick Euthyroid datasets

Significance test of the accuracies using T-Test								
	Skin Segmentation				*Sick Euthyroid*			
	ST	AST	VFDT	VFDTcNB	ST	AST	VFDT	VFDTcNB
ST	0	1	1	1	0	1	1	1
AST	-1	0	1	0	-1	0	1	1
VFDT	-1	-1	0	0	-1	-1	0	1
VFDTcNB	-1	0	0	0	-1	-1	-1	0

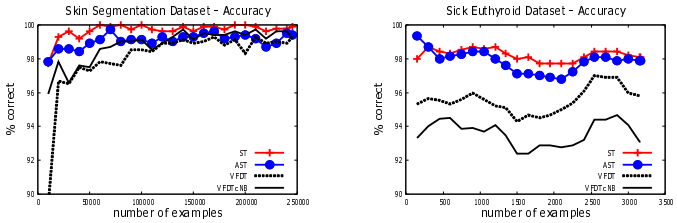


Fig. 2: Experiment using the Skin Segmentation and Sick Euthyroid datasets

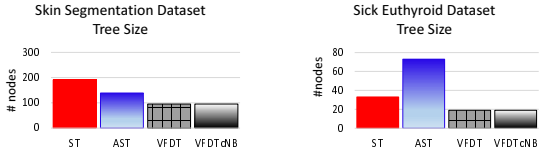


Fig. 3: Tree sizes using the Skin Segmentation and Sick Euthyroid datasets

criterion. ST and AST created bigger trees because they start to split the nodes before VFDT and VCFDTcNB (since the first available examples). This characteristic can increase the execution time, since a bigger model takes longer to classify the examples.

The accuracy results obtained using the SEA generator and the Electricity datasets are summarized in Table IV, according to the noise level, in terms of accuracy. The results obtained using the SEA generator show that ST and AST achieved the best accuracy on the data containing up to 15% of noise, and the VFDT achieved values very close to ST and AST with more than 15% of noise. AST obtained the best mean and final accuracy in almost all the experiments, independently on the noise level. The mean execution time obtained by each algorithm in the experiments are presented on Figure 6. Figure 4 shows the accuracy variation of the algorithms using the datasets containing noise. With the Electricity dataset the algorithms ST and AST acquired the best results with the first levels of noise. VFDT obtained better results with 25% and 30% of noise.

Figure 5 shows the kappa statistic of the experiments with the SEA and Electricity datasets. Using the SEA dataset, with 0% of noise the ST result is "substantial" and the other algorithms are "moderate". The kappa values tends to decrease as the noise increases. With 30% of noise the VFDT kappa result was "fair", while the others were "slight" results. Using the Electricity dataset the results were similar. Without noise the algorithms obtained "moderate" results. The agreement declined as the noise increased. AST finished with a "fair" result, close to the others, which achieved "slight" results. Summarizing, the significance of the results tends to decrease as the quantity of noise in the data increases. In general the results were close in both experiments.

Using the datasets containing noise, ST and AST constructed bigger trees in comparison with VFDT and VFDTcNB. Considering the Electricity dataset, ST had an average tree size of 317 nodes, AST 220 nodes, VFDT 37 nodes and VFDTcNB 38 nodes. With the SEA dataset, ST created trees with the average of 169 nodes, AST with 134, VFDT and VFDTcNB with 698 nodes. In Figure 6 it is possible to observe that using the Electricity dataset the algorithms VFDT and VFDTcNB achieved better execution times, and with the SEA generator the VFDT obtained a lower time, followed respectively

Table IV: Experiment results using the SEA generator and the Electricity dataset

Experiment results									
Datasets		Accuracy	*Noise / %*						
			0	5	10	15	20	25	30
SEA	ST	Mean	81.26	78.24	75.19	71.99	68.99	65.86	62.68
		Final	81.9	79.0	76.6	73.1	69.9	66.3	63.3
	AST	Mean	81.16	78.14	75.18	71.85	68.86	65.54	62.52
		Final	81.4	79.4	77.0	74.0	71.0	67.5	63.8
	VFDT	Mean	80.99	78.1	75.16	71.94	68.8	65.71	62.7
		Final	81.5	79.1	76.5	74.0	70.2	66.9	63.2
	VFDTcNB	Mean	80.68	77.76	74.62	71.26	68.41	64.93	61.73
		Final	81.7	79.0	76	72.2	69.3	65.9	61.3
Electricity	ST	Mean	78.93	75.21	71.36	68.68	65.61	62.06	59.05
		Final	81.3	76	71.2	71.6	64.5	62.4	59.3
	AST	Mean	79.15	75.71	72.2	69.54	66.06	63.23	60
		Final	80.1	76.2	75.8	68.7	68.0	62.3	59.9
	VFDT	Mean	74.64	72.02	70.45	67.42	64.19	61.75	58.7
		Final	75.8	72.2	71.06	68.7	67.6	63.2	62.9
	VFDTcNB	Mean	77.16	72.07	69.92	67.08	64.49	61.29	58.78
		Final	77.3	68.4	64.1	63.1	61.1	58.8	55.2

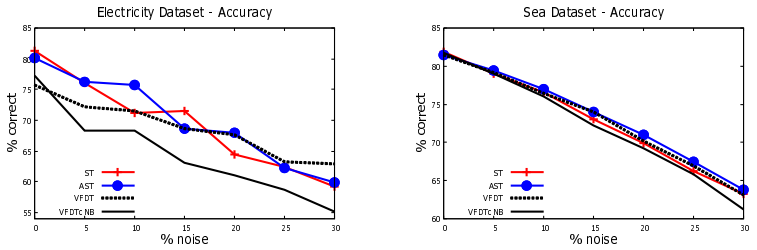


Fig. 4: Experiment with noise using the SEA and Electricity datasets

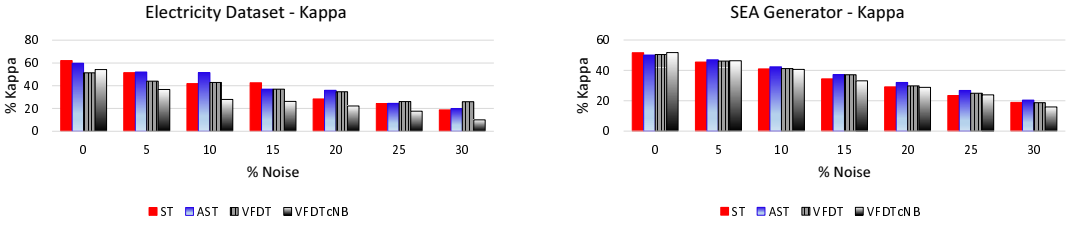


Fig. 5: Kappa values obtained using the Electricity and SEA datasets

by ST, AST and VFDTcNB.

5. CONCLUSION

In this work we described two algorithms, the StARMiner Tree and the Automatic StARMiner Tree, which are statistical decision tree algorithms for data streams classification. We applied them in four datasets, one synthetic and three real-world. The obtained results show that ST and AST are well-suited to classify data with or without noise, in a small or large size. In almost all performed experiments ST and AST obtained the best mean and final accuracy, maintaining good execution time, in comparison with VFDT and VFDTcNB.

The ST and AST algorithms, when applied in datasets containing concept drift, should be used along with a drift detection method. Normally, a drift detection method is used independently on the classifier. When we use ST or AST in a context with concept drift, the algorithm should be used with a drift detection method that, for example, monitors the error rate of the model and triggers an alarm informing that the model should be replaced (or adapted), when it becomes outdated. Due

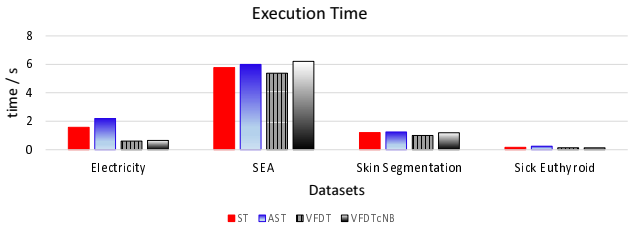


Fig. 6: Execution time of the algorithms.

to the fast convergence of the methods ST and AST, the algorithms construct the model since the first examples available, and may get more sensitive to changes in the distribution. In this case, it is important to chose a drift detection method that considers this sensibility, detecting the concept drift when the method does not describe the newest data properly.

REFERENCES

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

Domingos, P. and Hulten, G. Mining high-speed data streams. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Boston, Massachusetts, USA, pp. 71–80, 2000.

Emam, K. E. Benchmarking Kappa: Interrater Agreement in Software ProcessAssessments. *Empirical Software Engineering* 4 (2): 113–133, 1999.

Gama, J. a., Rocha, R., and Medas, P. Accurate Decision Trees for Mining High-Speed Data Streams. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington, D.C., pp. 523–528, 2003.

Hulten, G., Spencer, L., and Domingos, P. Mining Time-Changing Data Streams. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA, pp. 97–106, 2001.

Letouzey, F., Denis, F., and Gilleron, R. Learning from Positive and Unlabeled Examples. In *International Conference on Algorithmic Learning Theory*. pp. 71–85, 2000.

Li, C., Zhang, Y., and Li, X. OcVFDT: one-class very fast decision tree for one-class classification of data streams. In *International Workshop on Knowledge Discovery from Sensor Data*. Paris, France, pp. 79–86, 2009.

Ouyang, Z., Wu, Q., and Wang, T. An Efficient Decision Tree Classification Method Based on Extended Hash Table for Data Streams Mining. In *International Conference on Fuzzy Systems and Knowledge Discovery*. Vol. 5. pp. 313–317, 2008.

Patil, A. and Attar, V. Framework for performance comparison of classifiers. In *International Conference on Soft Computing for Problem Solving*, K. Deep, A. Nagar, M. Pant, and J. C. Bansal (Eds.). Advances in Intelligent and Soft Computing, vol. 131. Springer India, pp. 681–689, 2011.

Ribeiro, M. X., Balan, A. G., Felipe, J. C., Traina, A. J. M., and Traina-Jr, C. Mining Statistical Association Rules to Select the Most Relevant Medical Image Features. First International Workshop on Mining Complex Data, Houston, USA, pp. 91–98, 2005.

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

Street, W. N. and Kim, Y. A streaming ensemble algorithm (sea) for large-scale classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, CA, pp. 377–382, 2001.

Watanabe, C., Ribeiro, M., Traina, A., and Traina-Jr, C. A statistical associative classifier with automatic estimation of parameters on computer aided diagnosis. In *International Conference on Machine Learning and Applications*. Vol. 1. pp. 564–567, 2012.

Witten, I., Frank, E., and Hall, M. *Data Mining: Practical Machine Learning Tools and Techniques: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science, 2011.

Yang, H. and Fong, S. Optimized Very Fast Decision Tree with Balanced Classification Accuracy and Compact Tree Size. In *International Conference on Data Mining and Intelligent Information Technology Applications*. pp. 57 –64, 2011.

Zia-Ur Rehman, M., Li, T.-r., and Li, T. Exploiting empirical variance for data stream classification. *Journal of Shanghai Jiaotong University (Science)* vol. 17, pp. 245–250, 2012.