Incremental Optimization Mechanism for Constructing a Decision Tree in Data Stream Mining

Article *in* Mathematical Problems in Engineering · January 2013

DOI: 10.1155/2013/580397

2 authors, including:

Simon Fong
University of Macau
573 PUBLICATIONS **2,449** CITATIONS

SEE PROFILE

Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 580397, 14 pages
http://dx.doi.org/10.1155/2013/580397

Research Article

Incremental Optimization Mechanism for Constructing a Decision Tree in Data Stream Mining

Hang Yang and Simon Fong

Department of Computer and Information Science, Faculty of Science and Technology, University of Macau, Taipa, Macau

Correspondence should be addressed to Simon Fong; ccfong@umac.mo

Received 21 September 2012; Revised 21 January 2013; Accepted 29 January 2013

Academic Editor: Sabrina Senatore

Imperfect data stream leads to tree size explosion and detrimental accuracy problems. Overfitting problem and the imbalanced class distribution reduce the performance of the original decision-tree algorithm for stream mining. In this paper, we propose an incremental optimization mechanism to solve these problems. The mechanism is called Optimized Very Fast Decision Tree (OVFDT) that possesses an optimized node-splitting control mechanism. Accuracy, tree size, and the learning time are the significant factors influencing the algorithm's performance. Naturally a bigger tree size takes longer computation time. OVFDT is a pioneer model equipped with an incremental optimization mechanism that seeks for a balance between accuracy and tree size for data stream mining. It operates incrementally by a test-then-train approach. Three types of functional tree leaves improve the accuracy with which the tree model makes a prediction for a new data stream in the testing phase. The optimized node-splitting mechanism controls the tree model growth in the training phase. The experiment shows that OVFDT obtains an optimal tree structure in both numeric and nominal datasets.

1. Introduction

Decision-tree learning is one of the most significant classifying techniques in data mining and has been applied to many areas, including business intelligence, healthcare, and biomedicine. The traditional approach to build a decision-tree, designed by the Greedy Search, loads a full set of data into memory and partitions the data into a hierarchy of nodes and leaves. The tree cannot be changed when new data are acquired, unless the whole model is rebuilt by reloading the complete set of historical data together with the new data. This approach is unsuitable for unbounded input data such as data streams, in which new data continuously flow in at high speed. To this end, the incremental approach is proposed to build a decision-tree dynamically that the tree grows with new data input.

A new generation of algorithms has been developed for incremental decision-tree, a pioneer of which using a Hoeffding bound (HB) in node splitting is so called Very Fast Decision-tree (VFDT) [1]. It builds a decision-tree simply by keeping track of the statistics of the attributes of the incoming data. When sufficient statistics have accumulated at each leaf, a node-splitting algorithm determines whether there is enough statistical evidence in favor of a node-split, which expands the tree by replacing the leaf with a new decision node. This decision-tree learns by incrementally updating the model while scanning the data stream on the fly. This powerful concept is in contrast to a traditional decision-tree that requires the reading of a full dataset for tree induction. The obvious advantage is its real-time data mining capability, which frees it from the need to store up all of the data to retrain the decision-tree because the moving data streams are infinite. A research work [2] proofs the feasibility of classification algorithms for analyzing biosignals in the forms of infinite data streams, and it also provides a comparison of traditional decision-tree C4.5 and incremental decision-tree VFDT in practical.

On one hand, the challenge for data stream mining is associated with the imbalanced class distribution. The term "imbalanced data" refers to irregular class distributions in a dataset. For example, a large percentage of training samples may be biased toward class A, leaving few samples that describe class B. Both noise and imbalanced class distribution significantly impair the accuracy of a decision-tree classifier

through confusion and misclassification prompted by the inappropriate data. The size of the decision-tree will also grow excessively large under noisy data. To tackle these problems, some researchers applied data manipulation techniques to handle the imbalanced class distribution problems, including undersampling, resampling, a recognition-based induction scheme [3], and a feature subset selection approach [4]. On the other hand, despite the difference in their tree-building processes, both traditional and incremental decision-trees suffer from a phenomenon called overfitting when the input data are infected with noise. The noise confuses the tree-building process with conflicting instances. Consequently, the tree size becomes very large and eventually describes noise rather than the underlying relationship.

With traditional decision-trees, the underperforming branches created by noise and biases are commonly pruned by cross-validating them with separate sets of training and testing data. Pruning algorithms [5] help keep the size of the decision-tree in check; however the majority are post-pruning techniques that remove relevant tree paths after a whole model has been built from a stationary dataset. Post-pruning of a decision-tree in high-speed data stream mining, however, may not be possible (or desirable) because of the nature of incremental access to the constantly incoming data streams.

In this paper, we devise a new version of VFDT, so called Optimized VFDT (OVFDT), which can provide an incremental optimization on prediction accuracy and decision-tree model size. The motivations of OVFDT are.

(1) To handle the imbalanced class distribution problem, OVFDT proposes three types of functional tree leaf that improve the classification accuracy;

(2) To deal with the noisy data in data streams, OVFDT uses an adaptive tie-breaking threshold instead of a user predefined. We do not know what the best setting is unless all possibilities have been tried. However it is an obstacle for real-world application. By running simulation experiments, the optimized value of adaptive tie is proved to be ideal for constraining the optimal tree growth.

(3) To prevent the over-fitting problem, OVFDT contains an incremental optimization mechanism in the node-splitting test that obtains an optimal decision-tree amongst prediction accuracy and model size.

The rest of this paper is structured as follows: Section 2 introduces a research background of decision-tree learning for data streams, the effect of tiethreshold in tree building. Section 3 presents the details of OVFDT algorithm, in terms of a test-then-train approach. Experiments are described in Section 4. Section 5 concludes.

2. Background

2.1. Decision-Tree in Data Stream Mining. A decision-tree classification problem is defined as follows: N is the number of examples in a dataset with a form (X, y), where X is a vector of d attributes and y is a discrete class label. k is the index of class label. Suppose a class label with the kth discrete value is y_k. Attribute X_i is the ith attribute in X and is assigned a value of $x_{i1}, x_{i2}, \ldots, x_{iJ}$, where $1 \le i \le d$, and J is the number of different values X_i. The classification goal is to produce a decision-tree model from N examples, which predicts the classes of y in future examples with high accuracy. In data stream mining, the example size is very large or unlimited, $N \to \infty$.

VFDT [1] constructs an incremental decision-tree by using constant memory and constant time-per-sample. It is a pioneering predictive technique that utilizes the Hoeffding bound. The tree is built by recursively replacing leaves with decision nodes. Sufficient statistics n_{ijk} of attribute X_i with a value of x_{ij} are stored in each leaf with a class label assigning to a value y_k. A heuristic evaluation function $H(\cdot)$ is used to determine split attributes for converting leaves to nodes. Nodes contain the split attributes, and leaves contain only the class labels. The leaf represents a class according to the sample label. When a sample enters, it traverses the tree from the root to a leaf, evaluating the relevant attributes at every node. Once the sample reaches a leaf, the sufficient statistics are updated. At this time, the system evaluates each possible condition based on the attribute values; if the statistics are sufficient to support one test over the others, then a leaf is converted to a decision node. The decision node contains the number of possible values for the chosen attribute according to the installed split test. The main elements of VFDT include, first, a tree-initializing process that initially contains only a single leaf and, second, a tree-growing process that contains a splitting check using a heuristic function $H(\cdot)$ and a Hoeffding bound (HB). VFDT uses information gain as $H(\cdot)$.

The formula of HB is shown in (1). HB controls over errors in the attribute-splitting distribution selection, where R is the range of classes' distribution and n is the number of instances that have fallen into a leaf. δ is one minus the desired probability of choosing the correct attribute at any given node. To evaluate a splitting value for attribute X_i, it chooses the best two values. Suppose x_{ia} is the best value of $H(\cdot)$, where $x_{ia} = \arg\max H(x_{ij})$; suppose x_{ib} is the second best value, where $x_{ib} = \arg\max H(x_{ij}), \forall j \ne a$; suppose $\Delta H(X_i)$ is the difference of the best two values for attribute X_i, where $\Delta H(X_i) = \Delta H(x_{ia}) - \Delta H(x_{ib})$. HB is used to compute high confidence intervals for the true mean r_{true} of attribute x_{ij} to class y_k that $r - \text{HB} \le r_{\text{true}} < r + \text{HB}$, where $r = (1/n)\sum_i^n r_i$. If, after observing $n_{\min}$ examples, the inequality $r + \text{HB} < 1$ holds, then $r_{\text{true}} < 1$, meaning that the best attribute x_{ia} observed over a portion of the stream is truly the best attribute over the entire stream. Thus, a splitting value x_{ij} of attribute X_i can be found without full attribute values even when we do not know all values of X_i. In other words, it does not train a model from full data, and the tree is growing incrementally when more data come. Consider,

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

In the past decade, several research papers have proposed different methodologies to improve the accuracy of VFDT. HOT [6] proposes an algorithm producing some optional

tree branches at the same time, replacing those rules with lower accuracy by optional ones. The classification accuracy has been improved significantly, while learning speed is slowed because of the construction of optional tree branches. Some of options are inactive branches consuming computer resource. Functional tree leaf is originally proposed to integrate into incremental decision-tree in VFDTc [7]. Consequently, the *Naïve Bayes* classifier on the tree leaf has improved classification accuracy. The functional tree leaf is able to handle both continuous and discrete values in data streams, but no direct evidence shows it can handle such imperfections like noise and bias in data streams. FlexDT [8] proposes a Sigmoid function to handle noisy data and missing values. The Sigmoid function is used to decide what true node-splitting value, but sacrificing algorithm speed. For this reason, the lightweight algorithm with fast learning speed is favored for data streams environment. CBDT [9] is a forest of trees algorithm that maintains a number of trees, each of which is rooted on different attributes and grows independently. It is sensitive to the concept drift in data streams according to the sliding-window mechanism. VFDR [10] is a decision rule learner using HB. Likewise VFDT, VFDR proposes a rule expending mechanism that constructs the decision rules (ordered or unordered) from data stream on the fly. VFDT handles streaming data that tree structure keeps on updating when new data arrive. It only requires reading some samples satisfying the statistical bound (referring to the HB) to construct a decision-tree. Since it cannot analyze over the whole training dataset in one time, normal optimization methods using full dataset to search for an optima between the accuracy and tree size do not work well here. Our previous work has provided a solution for sustainable prediction accuracy and regulates the growth of the decision-tree to a reasonable extent, even in the presence of noise. Moderated Very Fast Decision-tree (MVFDT) [11] is a novel extension of the VFDT model that includes optimizing the tree-growing process via *adaptive tie-breaking threshold* instead of a user predefined value in VFDT.

There are two popular platforms for implementing stream-mining decision-tree algorithms. Very Fast Machine Learning (VFML) [12] is a C-based tool for mining time-changing high-speed data streams. Massive Online Analysis (MOA) [13] is Java-based software for massive data analysis, which is a well-known open source project extended from WEKA data mining. In both platforms, the parameters of VFDT must be preconfigured. For different tree induction tasks, the parameter setup is distinguished.

MOA is an open source project with a user friendly graphic interface. It also provides several ways to evaluate algorithm's performance. Hence, some VFDT-extended algorithms have been built in this platform. For example, the VFDT algorithms embedded in MOA (released on November 2011) are Ensemble Hoeffding Tree [14] is an online bagging method with some ensemble VFDT classifiers. Adaptive Size Hoeffding Tree (ASHT) [15] is derived from VFDT adding a maximum number of split nodes. ASHT has a maximum number of split nodes. After one node splits, if the number of split nodes is higher than the maximum value, then it deletes some nodes to reduce its size. Besides, it is designed for handling concept-drift data streams. AdaHOT [15] is also derived from HOT. Each leaf stored an estimation of current error. The weight of node in voting process was proportional to the square of inverse of error. AdaHOT combines HOT with a voting mechanism on each node. It also extends the advantages using optional trees to replace the tree branches of bad performance. Based on an assumption "there has been no change in the average value inside the window," ADWIN [16] proposes a solution to detect changes by a variable-length window of recently seen instances. In this paper, the OVFDT algorithm is developed on the fundamental of MOA platform.

2.2. Relationship amongst Accuracy, Tree Size, and Time. When data contains noisy values, it may confuse the result of heuristic function. The difference of the best two heuristic evaluations for attribute X_i, where $\Delta\overline{H}(X_i) = H(x_{ia}) - H(x_{ib})$, may be negligible. To solve this problem, a fixed tie-breaking τ, which is a user predefined threshold for incremental learning decision-tree, is proposed as prepruning mechanism to control the tree growth speed [17]. This threshold constrains the node-splitting condition that $\Delta\overline{H}(X_i) \leq \text{HB} < \tau$. An efficient τ guarantees a minimum tree growth in case of tree-size explosion problem. τ must be set before a new learning starts; however, so far there has not been a unique τ suitable for all problems. In other words, there is not a single default value that works well in all tasks so far. The choice of τ hence depends on the data and their nature. It is said that the excessive invocation of tie breaking brings the performance of decision-tree learning declining significantly on complex and noise data, even with the additional condition by the parameter τ.

In addition to the tie-breaking threshold τ, $n_{\min}$ is the number of instances a leaf should observe between split attempts. In other words, τ is a user-defined value to control the tree growing speed, and $n_{\min}$ is a user-defined value to control the interval time to check node splitting. The former is used to constrain tree size and the latter is used to constrain the learning speed. In order to optimize accuracy, tree size, and speed for decision-tree learning, first of all, an example is given to demonstrate the relationship amongst these three factors for data streams.

In this example, we use VFDT, which is a classical incremental decision-tree using HB in node splitting, to evaluate synthetic datasets added with bias classes. We use MOA to generate two typical datasets: LED24 is a nominal dataset, and Waveform21 is a numeric dataset. Both datasets share the origins with the sample generators donated by UCI machine learning repository. LED24 uses 24 nominal attributes to classify 10 different classes, and Waveform21 uses 21 numeric attributes to classify 3 different classes. The data stream problem is simulated by large numbers of instances as many as one million. The accuracy, tree size, and time are recorded with changing the predefined values of τ and $n_{\min}$. From Table 1, we can see the following.

(i) In general, the bigger tree size brings a higher accuracy, even caused by the over-fitting problem, but taking more learning time.

Table 1: (a) Comparison of VFDT using different τ and $n_{\min}$, (b) comparison of VFDT using different $n_{\min}$.

(a)

τ	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
LED24 ($n_{\min}$ = 200)										
Accuracy (%)	75.88	76.97	77.14	77.42	77.47	77.50	77.56	77.56	77.56	77.56
No. leaf	143.00	522.00	1124.00	1857.00	2618.00	3723.00	3743.00	3743.00	3743.00	3743.00
Time (sec)	8.70	9.91	10.92	11.51	11.92	12.78	12.32	12.32	12.43	12.45
Waveform21 ($n_{\min}$ = 200)										
Accuracy	85.00	86.53	86.61	86.72	86.72	86.72	86.72	86.72	86.72	86.72
No. leaf	506.00	1565.00	2492.00	2623.00	2623.00	2623.00	2623.00	2623.00	2623.00	2623.00
Time	17.80	18.50	18.89	18.69	18.77	18.72	18.53	18.72	18.74	18.72

(b)

$n_{\min}$	200	300	400	500	600	700	800	900	1000
LED24 (τ = 0.7)									
Accuracy	77.5611	77.5867	77.4565	77.3472	77.2557	77.1417	77.1412	77.0847	76.9887
No. leaf	3743	2405	1826	1383	1244	1057	935	804	689
Time	11.66887	10.93567	10.40527	10.07766	9.469261	9.42246	9.032458	9.172859	8.642455
Waveform21 (τ = 0.4)									
Accuracy	86.7218	86.5226	86.3028	85.9499	85.9119	85.6378	85.6707	85.7318	85.2165
No. leaf	2623	1800	1363	1103	940	806	703	644	572
Time	18.31452	17.70611	17.34731	17.03531	16.84811	16.58291	16.61411	16.61411	16.2709

(ii) τ is proposed to control the tree size growing. A bigger τ brings a faster tree size growth, but longer computation time. But because the memory is limited, the tree size does not increase, while τ reaches a threshold (τ = 0.7 for LED24; τ = 0.4 for Waveform21).

(iii) $n_{\min}$ is proposed to control the learning time. A bigger $n_{\min}$ brings a faster learning speed, but smaller tree size and lower accuracy.

A proposed solution [18] to overcome this detrimental effect is an improved tie-breaking mechanism, which not only considers the best (x_{ia}) and the second best (x_{ib}) splitting candidates in terms of heuristic function, but also uses the worst candidate (x_{ic}). At the same time, an extra parameter is imported, α, which determines how many times smaller the gap should be before it is considered as a tie. The attribute splitting condition becomes as the following: when $\alpha \times (H(x_{ia}) - H(x_{ib})) < (H(x_{ib}) - H(x_{ic}))$, the attribution x_{ia} will be split as a node. Obviously, this approach uses two extra elements, α and x_{ic}, which bring extra computation to the original algorithm. However, the only way to detect the best tie-breaking threshold for a certain task is trying all the possibilities in VFDT. It is impractical for real-world applications. In this paper, we propose the adaptive tie-breaking threshold using the incremental optimization methodology. The breakthrough of our work is the optimized node-splitting control, which will be specified in the following sections.

3. Proposed Methodology

3.1. Motivation and Overview. OVFDT, which inherits the use of HB, implements on a test-then-train approach

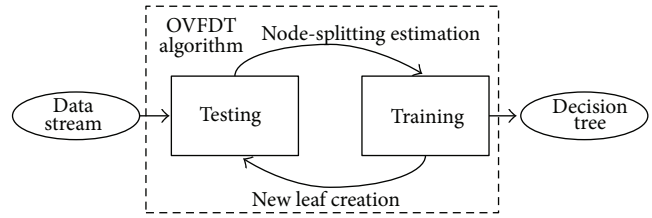

Figure 1: A test-then-train OVFDT workflow.

(Figure 1) for classifying continuously arriving data streams, even for infinite data streams. The whole test-then-train process is synchronized such, that when the data stream arrives, one segment at a time, the decision-tree is being tested first for prediction output, and training (which is also known as updating) of the decision-tree then occurs incrementally. The description of testing process will be explained in Section 3.3 in details, and the training process will be explained in Section 3.4. Ideally, the node-splitting test updates tree model in order to improve the accuracy, while a bigger tree model takes longer computation time. The situation to do the node-splitting check is when the number of instances in a leaf l is greater than the predefined value $n_{\min}$.

Imperfect data streams, including noisy data and bias class distribution, decline the performance of VFDT. Figure 2 shows the results of accuracy, tree size, and computation time using VFDT the same dataset structure added with imperfect values. The ideal stream is free from noise and has a uniform proportion of class samples, which is rare in real world. Comparing ideal data streams with imperfect data streams, we conclude Lemma 1.

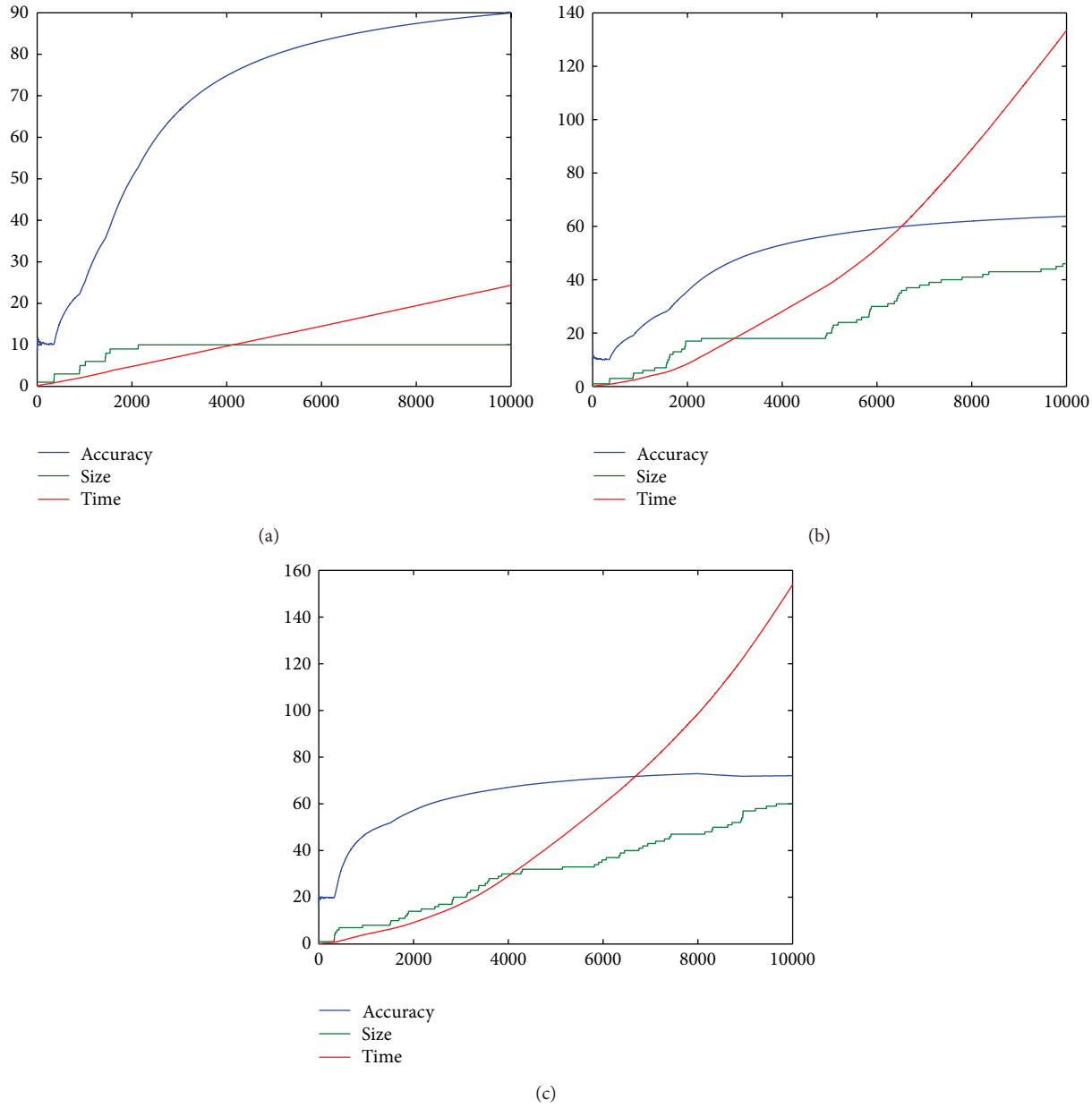

FIGURE 2: VFDT performance for: (a) ideal data, (b) data with noise, (c) data with noise and bias. X-axis presents the accuracy and y-axis the number of samples.

Lemma 1. *Imperfections in data streams worsen the performance of VFDT. The tree size and the computation time are increased, but the accuracy is declined. In other words, the optimization goal is to increase the accuracy but not enlarge the tree size, within an acceptable computation time. Naturally a bigger tree size takes longer computation time. For this reason, the computation time is dependent on the tree size.*

In the decision-tree model, each path from the root to a leaf is considered as a way to present a rule. To ensure a high accuracy, there must be sufficient number of rules, which is the number of leaves in the tree model. Suppose the Hoeffding Tree (HT) is the decision-tree algorithm using Hoeffding bound (HB) as the node-splitting test. Let $\text{Accu}(\text{HT}_{m\text{th}})$ be the accuracy function for the decision-tree structure HT at the mth node-splitting estimation, and let $\text{Size}(\text{HT}_{m\text{th}})$ be the tree size; then

$$\text{Accu}\left(\text{HT}_{m\text{th}}\right) = R\left(\text{Size}\left(\text{HT}_{m\text{th}}\right)\right), \tag{2}$$

where $R(\cdot)$ is a mapping function of tree size to accuracy. Most incremental optimization functions can be expressed as the sum of several subobjective functions:

$$\Phi\left(x\right) = \sum_{m=1}^{M} \Phi_m\left(x\right), \tag{3}$$

where $\Phi_m : \chi \subset \mathbb{R}^p \rightarrow \mathbb{R}$ is a continuously differentiable function whose domain χ is a nonempty, convex, and closed set. We consider the following optimization problems:

$$\text{maximize } \Phi(x) \text{ subject to } x \in \chi. \tag{4}$$

Based on Lemma 1, we propose a solution to optimize the decision-tree structure by improving the original VFDT that:

$$\Phi_m(x) = \frac{\text{Accu}(\text{HT}_{m\text{th}}) - \text{Accu}(\text{HT}_{m\text{th}})}{\text{Size}(\text{HT}_{m\text{th}}) - \text{Size}(\text{HT}_{m\text{th}})}. \tag{5}$$

The tree model is updated when a node splitting appears. Original VFDT considers the HB as the only index to split node. However, it is not enough. In terms of the above optimization goal, OVFDT proposes an optimized node-splitting control during the tree-building process.

3.2. OVFDT Test-Then-Train Process. Data streams are open-ended problems that traditional sampling strategies are not viable in the nonstopping streams scenario. OVFDT is an improved version of the original VFDT and its extensions using HB to decide the node splitting. The most significant contribution is OVFDT that can obtain an optimal tree structure by balancing the accuracy and tree size. It is useful for data mining especially in the events of the tree size explosion, when the decision-tree is subject to imperfect streams including noisy data and imbalanced class distribution.

HT algorithms run a test-then-train approach to build a decision-tree mode. When new stream arrives, it will be sorted from the root to a predicted leaf. Comparing the predictive class with the true class of this data stream, we can maintain an error matrix for every tree leaf in the testing process. In terms of the stored statistics matrix, the decision-tree model is being updated in the training process. Table 2 presents the differences between OVFDT and HT algorithms (including the original VFDT and its extensions). Pseudocode 1 shows the input parameters, and the output of OVFDT, and the approach presented as pseudocode.

3.3. OVFDT Testing Approach. Suppose X is a vector of d attributes, and y is the class with k different values included in the data streams. For decision-tree prediction learning tasks, the learning goal is to induce a function of $\widehat{y}_k = \text{HT}_{\mathcal{F}}(X)$, where $\widehat{y}_k$ is the predicted class by a Hoeffding tree (HT) according to a functional tree leaf strategy $\mathcal{F}$. When a new data stream (X, y_k) arrives, it traverses from the root of the decision-tree to an existing leaf by the current decision-tree structure, provided that the root has existed initially. Otherwise, the heuristic function is used to construct a tree model with a single root node. When new instance comes, it will be sorted from the root to a leaf by the current tree model. The classifier on the leaf can further enhance the prediction accuracy via the embedded classifiers. OVFDT contains three classifiers f to improve the performance of prediction. They are the *Majority Class* (f^{MC}), *Naïve Bayes* (f^{NB}), and *Weighted Naïve Bayes* (f^{WNB}).

Suppose $\widehat{y_k}$, the predicted class value, and y_k is actual class in data streams with a vector of attribute X. A sufficient statistics matrix stores the number of passed-by samples, which contain attribute X_i with a value x_{ij} belonging to a certain y_k so far. We call this statistics table Observed Class Distribution (OCD) matrix. The size of OCD is $J \times K$, where J is the total number of distinct values for attribute X_i and K is the number of distinct class values. Suppose n_{ijk} is the sufficient statistic that reflects the number of attribute X_i with a value x_{ij} belonging to class y_k. Therefore, OCD on node X_i is defined as

$$\text{OCD}_{X_i} = \begin{bmatrix} n_{i11} & \cdots & n_{iJ1} \\ \vdots & \ddots & \vdots \\ n_{i1K} & \cdots & n_{iJK} \end{bmatrix}. \tag{6}$$

For a certain leaf that attributes X_i with a value of x_{ij},

$$\text{OCD}_{x_{ij}} = \{n_{ij1} \ \cdots \ n_{ijK}\}. \tag{7}$$

Majority Class classifier chooses the class with the maximum value as the predicted class in a leaf. Thus, f^{MC} predicts the class with a value that

$$\arg\max k = \{n_{ij1} \ \ldots \ n_{ijk} \ \ldots \ n_{ijK}\}. \tag{8}$$

Naïve Bayes classifier chooses the class with the maximum possibility computed by *Naïve Bayes* as the predictive class in a leaf. The formula of *Naïve Bayes* is

$$p_{ijk} = \frac{P(x_{ij} \mid y_k) \cdot P(y_k)}{P(x_{ij})}. \tag{9}$$

OCD of leaf with value x_{ij} is updated incrementally. Thus, f^{NB} predicts the class with a value that

$$\arg\max k = \{p_{ij1} \ \cdots \ p_{ijk} \ \cdots \ p_{ijK}\}. \tag{10}$$

Table 2: The comparison between VFDT and OVFDT.

Approach	VFDT	OVFDT
Testing	Sort new data by current HT Update the sufficient statistics FTL of MC, NB classifiers Assign a predicted class by FTL	Sort new data by current HT Update the sufficient statistics FTL of MC, NB, and WNB classifiers Assign a predicted class by FTL *Update incremental* *Sequential error*
Training	Check node splitting by HB Check node splitting by fixed τ Tree model HT update	Check node splitting by HB Check node splitting by *adaptive* τ Check node splitting by *statistical error* Tree model HT update

*FTL is functional tree leaf; MC is *Majority Class*, NB is *Naïve Bayes*, and WNB is *Weighted Naïve Bayes*; HT is the decision tree using a Hoeffding bound.

INPUT:
S: A stream of sample
X: A set of symbolic attributes
$H(\cdot)$: Heuristic function using for node-splitting estimation
δ: One minus the desired probability of choosing a correct attribute at any given node
$n_{\min}$: The minimum number of samples between check node-splitting estimation
$\mathcal{F}$: A functional tree leaf strategy
OUTPUT:
HT: A decision tree
PROCEDURE: OVFDT (S, X, $H(\cdot)$, δ, $n_{\min}$, $\mathcal{F}$)
(1) A data stream S arrives
(2) IF HT is null, THEN *initialize*HT(S, X, $H(\cdot)$, δ, $n_{\min}$, $\mathcal{F}$)
ELSE *traverse*HT(S, HT, $\mathcal{F}$) and update ΔC
(3) Label l as the predicted class among the samples seen so far
(4) Let n_l be the number of samples seen at the leaf l
(5) IF the samples seen so far at leaf l do not all belong to the sameclass
and (n_l mod $n_{\min}$) is zero, THEN *doNodeSplitting*(ΔC, S, X, $H(\cdot)$, δ, $n_{\min}$)
(6) Return HT

Pseudocode 1: Pseudocode of input and the test-then-train approach.

PROCEDURE: *traverse*HT(S, HT, $\mathcal{F}$)
(1) Sort S from the root to a leaf by HT. Update OCD in each node: $n_{ijk}(l)$++
(2) Switch ($\mathcal{F}$)
(3) Case $\mathcal{F}^{\mathrm{MC}}$: predict the class y_k' with max $n_{ijk}(l)$
(4) Case $\mathcal{F}^{\mathrm{NB}}$: predict the class y_k' with max NB prob.
(5) Case $\mathcal{F}^{\mathrm{WNB}}$: predict the class y_k' with max WNB prob.
(6) IF y_k' equals to the actual class label in S, THEN C_T++
(7) ELSE C_F++
(8) $\Delta C = C_T - C_F$
(9) Return ΔC

Pseudocode 2: Pseudocode of testing approach.

Weighted Naïve Bayes classifier proposes to reduce the effect of imbalanced class distribution. It chooses the class with the maximum possibility computed by *Weighted Naïve Bayes* as the predictive class in a leaf:

$$p_{ijk} = \omega_{ijk} \frac{P\left(x_{ij} \mid y_k\right) \cdot P\left(y_k\right)}{P\left(x_{ij}\right)}, \quad \text{where } \omega_{ijk} = \frac{n_{ijk}}{\sum_{k=1}^{K} n_{ijk}}. \tag{11}$$

OCD of leaf with value x_{ij} is updated. Thus, $\mathcal{F}^{\mathrm{WNB}}$ predicts the class with a value that

$$\arg\max k = \left\{p_{ij1} \ \cdots \ p_{ijk} \ \cdots \ p_{ijK}\right\}. \tag{12}$$

After the stream traverses the whole HT, it is assigned to a predicted class $\hat{y}_k$, which $\hat{y}_k \leftarrow$ Classifier (HT, $\mathcal{F}$, X) according to the functional tree leaf $\mathcal{F}$. Comparing the predicted class $\hat{y}_k$ to the actual class y_k, the statistics of correctly C_T and incorrectly C_F prediction are updated immediately. Meanwhile, the sufficient statistics, n_{ijk}, which is a count of attribute x_i with value j belonging to class y_k, are updated in each node. This series of actions is so called a testing approach in this paper. Pseudocode 2 gives the pseudocode of this approach. According to the functional tree leaf strategy, the current HT sorts a newly arrived sample (X, y_k) from the root to a predicted leaf $\hat{y}_k$. Comparing the predicted class $\hat{y}_k$ to the actual class y_k, the sequential-error statistics of C_T and C_F prediction are updated immediately.

To store OCD for OVFDT, $\mathcal{F}^{\mathrm{MC}}$, $\mathcal{F}^{\mathrm{NB}}$, and $\mathcal{F}^{\mathrm{WNB}}$ require memory proportional to $O(N \cdot I \cdot J \cdot K)$, where N is the number of nodes in tree model and I the number of attributes; J is the maximum number of values per attribute; K is the number of classes. OCD of $\mathcal{F}^{\mathrm{NB}}$ and $\mathcal{F}^{\mathrm{WNB}}$ are converted from that of $\mathcal{F}^{\mathrm{MC}}$. Therefore, we do not require extra memory. When required, it can be converted from $\mathcal{F}^{\mathrm{MC}}$.

3.4. OVFDT Training Approach. Right after the testing approach, the training follows. Node-splitting estimation is used to initially decide if HT should be updated or not that depends on the amount of samples received so far that can potentially be represented by additional underlying rules in the decision-tree. In principle, the optimized node-splitting estimation should be applied on every single new sample that arrives. Of course this will be too exhaustive, and it will slow

down the tree building process. Instead, a parameter $n_{\min}$ is proposed in VFDT that only do the node-splitting estimation when $n_{\min}$ examples have been observed on a leaf. In the node-splitting estimation, the tree model should be updated when a heuristic function $H(\cdot)$ chooses the most appropriate attribute with the highest heuristic function value $H(x_a)$ as a splitting node according to HB and tie-breaking threshold. The heuristic function is implemented as an information gain here. This situ of node-splitting estimation constitutes the so-called training phase.

The node-splitting test is modified to use a dynamic tie-breaking threshold τ, which restricts the attribute splitting as a decision node. The τ parameter traditionally is pre-configured with a default value defined by the user. The optimal value is usually not known until all of the possibilities in an experiment have been tried. An example has been presented in Section 2.2. Longitudinal testing of different values in advance is certainly not favorable in real-time applications. Instead, we assign a dynamic tie threshold, equal to the dynamic mean of HB at each pass of stream data, as the splitting threshold, which controls the node splitting during the tree-building process. Tie breaking that occurs close to the HB mean can effectively narrow the variance distribution. HB mean is calculated dynamically whenever new data arrives.

The estimation of splits and ties is only executed once for every $n_{\min}$ (a user-supplied value) samples that arrive at a leaf. Instead of a preconfigured tie, OVFDT uses an adaptive tie that is calculated by incremental computing. At the ith node-splitting estimation, the HB estimates the sufficient statistics for a large enough sample size to split a new node, which corresponds to the leaf l. Let T_l be an adaptive tie corresponding to leaf l, within k estimations seen so far. Suppose μ_l is a binary variable that takes the value of 1 if HB relates to leaf l and 0 otherwise. T_l is computed by (13). To constrain HB fluctuation, an upper bound T_l^{UPPER} and a lower bound T_l^{LOWER} are proposed in the adaptive tie mechanism. The formulas are presented in

$$T_l = \frac{1}{k} \sum_{i=1}^{k} \mu_l \times \text{HB}_i, \tag{13}$$

$$T_l^{\text{UPPER}} = \arg\max T_l, \tag{14}$$

$$T_l^{\text{LOWER}} = \arg\min T_l. \tag{15}$$

For lightweight operations, we propose an error-based prepruning mechanism for OVFDT, which stops noninformative split node before it splits into a new node. The prepruning takes into account the node-splitting error both globally and locally.

According to the optimization goal mentioned in Section 3.1, besides the HB, we also consider the global and local accuracy in terms of the sequential error statistics of C_T and C_F prediction computed by functional tree leaf. Let ΔC_m be the difference between C_T and C_F, and m is the index of testing approach. Then ΔC_m reflects the global accuracy of the current HT prediction on the newly arrived data streams. If $\Delta C_m \geq 0$, the number of correct predictions is no less than the number of incorrect predictions in the current tree structure; otherwise, the current tree graph needs to be updated by node splitting. In this approach, the statistics of correctly C_T and incorrectly C_F prediction are updated. Suppose $\Delta C_m = C_T - C_F$, which reflects the accuracy of HT. If ΔC declines, it means the global accuracy of current HT model worsens. Likewise, comparing ΔC_m and ΔC_{m+1}, the local accuracy is monitored during the node splitting. If ΔC_m is greater than ΔC_{m+1}, it means the current accuracy is declining locally. In this case, the HT should be updated to suit the newly arrival data streams.

Lemma 2. *Monitor global accuracy. The model's accuracy varies whenever a node splits, and the tree structure is updated. Overall accuracy of current tree model is monitored during node splitting by comparing the number of correctly and incorrectly predicted samples. The number of correctly predicted instances and otherwise is recorded as global performance indicators so far. This monitoring allows the global accuracy to be determined.*

Lemma 3. *Monitor local accuracy. The global accuracy can be tracked by comparing the number of correctly predicted samples with the number of wrongly ones. Likewise, comparing the global accuracy measured at current node-splitting estimation with the previous splitting, the increment in accuracy is tracked dynamically. This monitoring allows us to check whether the current node-splitting is advantageous at each step by comparing with the previous step.*

Figure 3 gives an example why our proposed prepruning takes into account both the local and the global accuracy in the incremental pruning. At the ith node-splitting estimation the difference between correctly and incorrectly predicted classes was ΔC_i, and ΔC_{i+1} was at (i+1)th estimation. ($\Delta C_i - \Delta C_{i+1}$) was negative that the local accuracy of (i+1)th estimation was worse than its previous one, while both were on a global increasing trend. Hence, if accuracy is getting worse, it is necessary to update the HT structure.

Combining the prediction statistics gathered in the testing phase, Pseudocode 3 presents the pseudo code of the training phase in OVFDT in building an upright tree. The optimized node-splitting control is presented in Figure 3 Line 7. In each node-splitting estimation process, HB value that relates to a leaf l is recorded. The recorded HB values are used to compute the adaptive tie, which uses the mean of HB to each leaf l, instead of a fixed user-defined value in VFDT.

4. Evaluation

4.1. Evaluation Platform and Datasets. A Java package with OVFDT has been implemented with MOA toolkit as a simulation platform for experiments. The running environment is on a Windows 7 PC with Intel Quad 2.8 GHz CPU and 8 G RAM. In all of the experiments, the parameters of the algorithms are $\delta = 10^{-6}$ and $n_{\min} = 200$, which are default values suggested by MOA. *δ is the allowable error in split decision, and values closer to zero will take longer to decide; $n_{\min}$ is the number of instances a leaf should observe between*

PROCEDURE doNodeSplitting (ΔC, S, X, $H(\cdot)$, δ):
(1) FOR each attribute $X_i \in X_l - \{X_\emptyset\}$ at the leaf l
(2) Compute $H_l(X_i)$
(3) Let X_a be the attribute with highest $H_l(\cdot)$ and X_a the second
(4) Compute HB with δ
(5) Let $\Delta H_l = H_l(X_a) - H_l(X_b)$
(6) END-FOR
(7) IF ($\Delta H_l > HB$) or ($\Delta H_l \leq T_l^{\text{LOWER}}$ and $\Delta C_i < \Delta C_{i-1}$)
or ($\Delta H_l \leq T_l^{\text{LOWER}}$ and $\Delta C_i < 0$)
or ($T_l^{\text{LOWER}} < \Delta H_l \leq T_l^{\text{UPPER}}$ and $\Delta C_i < \Delta C_{i-1}$)
(8) Replace l by an internal node splits on X_a
(9) Update adaptive tie T_l^{LOWER} and T_l^{UPPER}
(10) FOR each branch of splitting
(11) Add a new leaf l_m and let $X_m = X - \{X_a\}$
(12) Let $H_l(X_\emptyset)$ be the $H_l(\cdot)$ obtained by predicting the class in S according to $\mathscr{F}$ at l_m
(13) FOR each class y_k and each value x_{ij} of each attribute
(14) $X_i \in X_m - \{X_\emptyset\}$ and reset OCD: $n_{ijk}(l_m) = 0$
(15) END-FOR
(16) END-FOR
(17) END-IF
(18) Return updated HT

Pseudocode 3: Pseudocode of training approach.

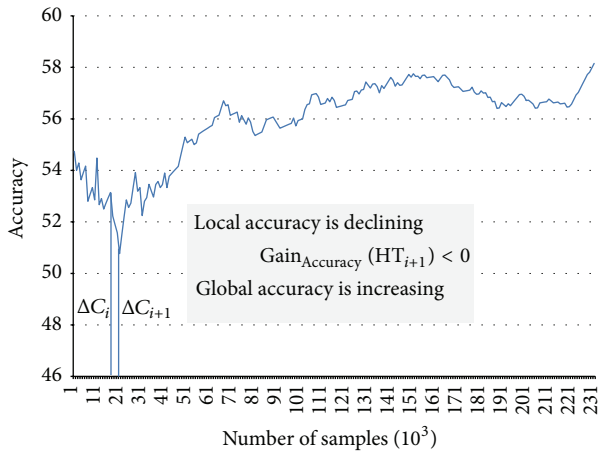

Figure 3: Incremental pruning.

split attempts. The main goal of this section is to provide evidence of the improvement of OVFDT compared with the original VFDT.

The experimental datasets, including pure nominal datasets, pure numeric datasets, and mixed datasets, are either synthetics generated by the MOA Generator or extracted from real-world applications that are publicly available for download from the UCI repository. The descriptions of each experimental dataset are listed in Table 3. LED24 dataset is generated by MOA Generator. In the experiment, we add 10% noisy data to simulate imperfect data streams. The LED24 problem uses 24 binary attributes to classify 10 different classes. Waveform21 dataset is also generated by MOA. The dataset is donated by David Aha to the UCI repository. The goal of the task is to differentiate between three different classes of Waveform. It has 21 numeric attributes which contained noise. Cover Type is used to predict forest cover type from cartographic variables, and this data is collected from real world [19].

The benchmarking algorithms are VFDT [1], HOT [6] and ADWIN [16]. VFDT and HOT are the representative learning methods without sliding-window criteria for handling concept drift; AWDIN uses an adaptive window technique. The paper [16] claimed that ADWIN performed as well or only slightly worse than the best window for each rate of change in CVFDT. This justifies why CVFDT was not being compared in this test.

4.2. Held-Out Evaluation. The first evaluation simulates a holdout testing approach. The datasets are divided to two parts: 70% for training model and 30% for testing model. In Figures 4 and 5 for LED24 and Waveform21, OVFDT algorithms have the best accuracy and compact tree size. For Cover Type data, HOT obtains a higher accuracy but much bigger model size than the others. OVFDT has the mechanism of optimizing node splitting so as to balance the prediction accuracy and the model size consequently.

Besides, we use the receiver operating characteristic (ROC) as a standard method for analysing and comparing classification result. It provides a convenient graphical display of the trade off between true and false positive classification rates for two-class problems. In the decision-tree classification, the ROC is used for more than two classes. In this

Table 3: Description of experimental datasets.

Name	Type	No. nom. attr.	No. num. attr.	No. class	Source	No. ins.
LED24	Nominal	24	0	10	Synthetic	10^6
Waveform21	Numeric	0	21	3	Synthetic	10^6
Cover type	Mix	42	12	7	UCI	581012

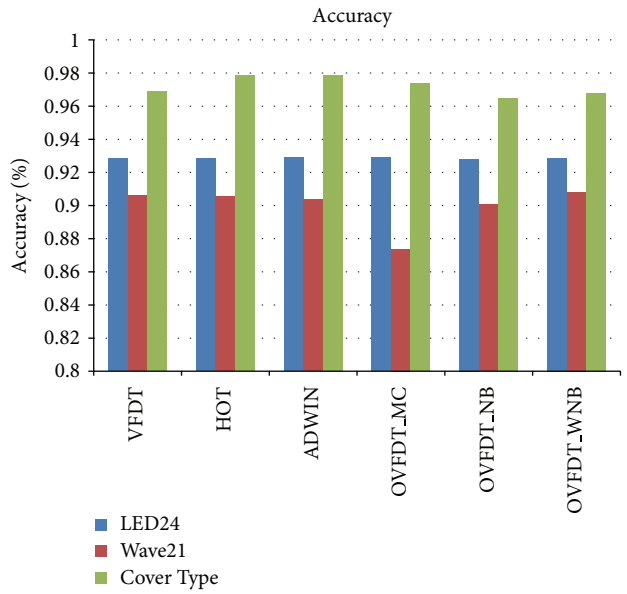

Figure 4: Accuracy of Holdout Test.

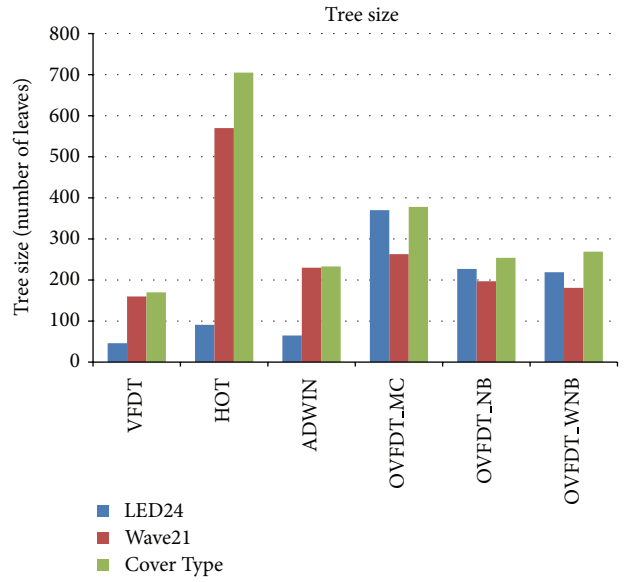

Figure 5: Tree Size of Holdout Test.

case, we apply the multi class ROC analysis to evaluate the performance of the tree-learning algorithm:

$$C = \begin{bmatrix} C_{11} & \cdots & C_{1d} \\ \vdots & \ddots & \vdots \\ C_{d1} & \cdots & C_{dd} \end{bmatrix}. \tag{16}$$

Likewise, two-class ROC statistics, for each class from i to d in a multi-class ROC analysis, can be assigned to the samples with class i as positive; otherwise they are assigned as negative in Figure 6. True positives (TP) are examples correctly labeled as positives, calculated by (17). False positives (FPs) refer to negative examples incorrectly labeled as positive, calculated by (18). True negatives (TNs) correspond to negatives correctly labeled as negative, calculated by (20). Finally, false negatives (FNs) refer to positive examples incorrectly labeled as negative, calculated by (19). Each class i can be converted into a two-class problem, with corresponding values for TP, TN, FP, and FN.

Precision-Recall is a well-known analysis method for ROC evaluation. In pattern recognition, precision refers to the fraction of retrieved instances that are relevant, whereas recall is the fraction of relevant instances that are retrieved. The values of precision and recall range from 0 to 1. A precision score of 1 for a class i means that every item labeled as belonging to class i does, indeed, belong to class i. A recall

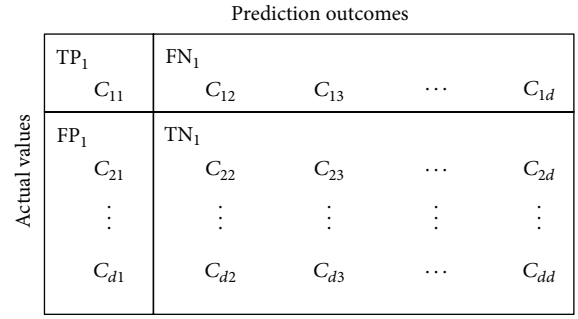

Figure 6: Sample of ROC Matrix.

score of 1 means that every item from class i was correctly labeled as belonging to class i. Precision-Recall scores are not analyzed in isolation: the F_β-measure [20] is a weighted harmonic mean of the Precision-Recall measure, and the F_1-measure evenly weights precision and recall scores, with a best value of 1 and a worst value of 0. In addition, the true positive rate (TPR) and false positive rate (FPR) are commonly used as benchmarks in ROC analysis. According to the ROC matrix shown in Figure 5, the calculation of TPR is given in (21). The calculation of FPR is given in (22). Precision is calculated by (23). F-measure calculations are in

Table 4: Precision-Recall results of holdout test.

	LED24			Waveform21			Cover Type		
	Recall	Precision	F_1	Recall	Precision	F_1	Recall	Precision	F_1
VFDT	0.620	0.650	**0.635**	0.878	**0.846**	0.862	0.508	0.563	0.534
HOT	0.620	0.650	**0.635**	0.875	**0.846**	0.860	**0.627**	**0.725**	**0.672**
ADWIN	0.592	**0.665**	0.627	0.873	0.843	0.858	**0.627**	**0.725**	**0.672**
OVFDT_mc	0.609	0.659	0.633	0.830	0.799	0.814	0.566	0.643	0.602
OVFDT_nb	**0.621**	0.648	0.634	0.866	0.840	0.853	0.624	0.502	0.556
OVFDT_wnb	0.619	0.652	**0.635**	**0.886**	**0.846**	**0.865**	**0.627**	0.539	0.580

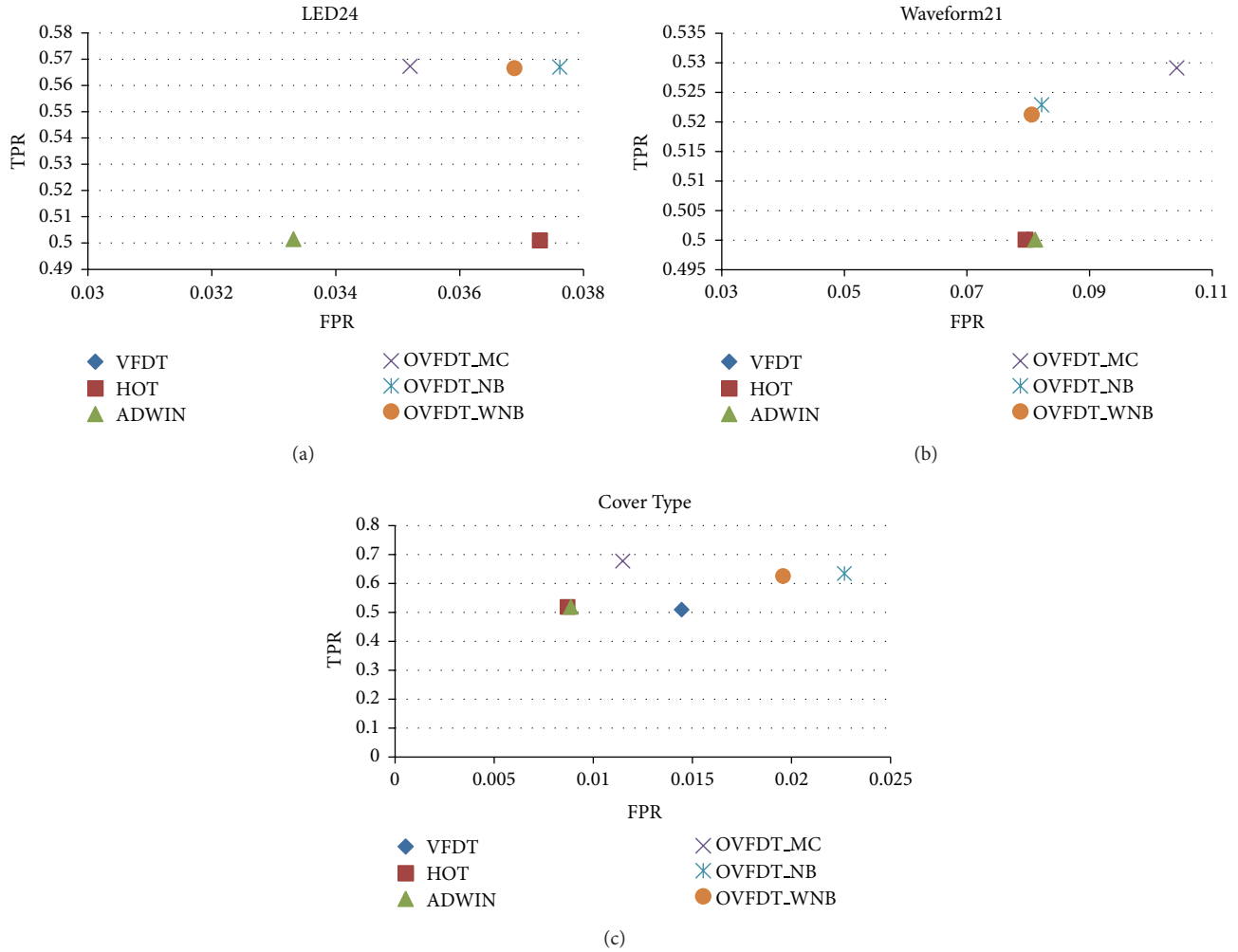

Figure 7: (a) ROC space of LED24. (b) ROC space of Waveform21. (c) ROC space of Cover Type.

(24). The experimental result of Precision-Recall test is shown in Table 4. Consider

$$\mathrm{TP}_i = C_{ii}, \tag{17}$$

$$\mathrm{FP}_i = \left(\sum_{j=1}^{d} C_{ji}\right) - C_{ii}, \tag{18}$$

$$\mathrm{FN}_i = \left(\sum_{j=1}^{d} C_{ij}\right) - C_{ii}, \tag{19}$$

$$\mathrm{TN}_i = \left(\sum_{i=1}^{d}\sum_{j=1}^{d} C_{ij}\right) - \mathrm{TP}_i - \mathrm{FP}_i - \mathrm{FN}_i, \tag{20}$$

$$\mathrm{TPR}_i = \mathrm{Recall}_i = \frac{\mathrm{TP}_i}{(\mathrm{TP}_i + \mathrm{FN}_i)} = \frac{C_{ii}}{\sum_{j=1}^{d} C_{ij}}, \tag{21}$$

$$\mathrm{FPR}_i = \frac{\mathrm{FP}_i}{(\mathrm{TN}_i + \mathrm{FP}_i)} = \frac{\left(\sum_{j=1}^{d} C_{ji}\right) - C_{ii}}{\left(\sum_{i=1}^{d}\sum_{j=1}^{d} C_{ij}\right) - \left(\sum_{j=1}^{d} C_{ij}\right)}, \tag{22}$$

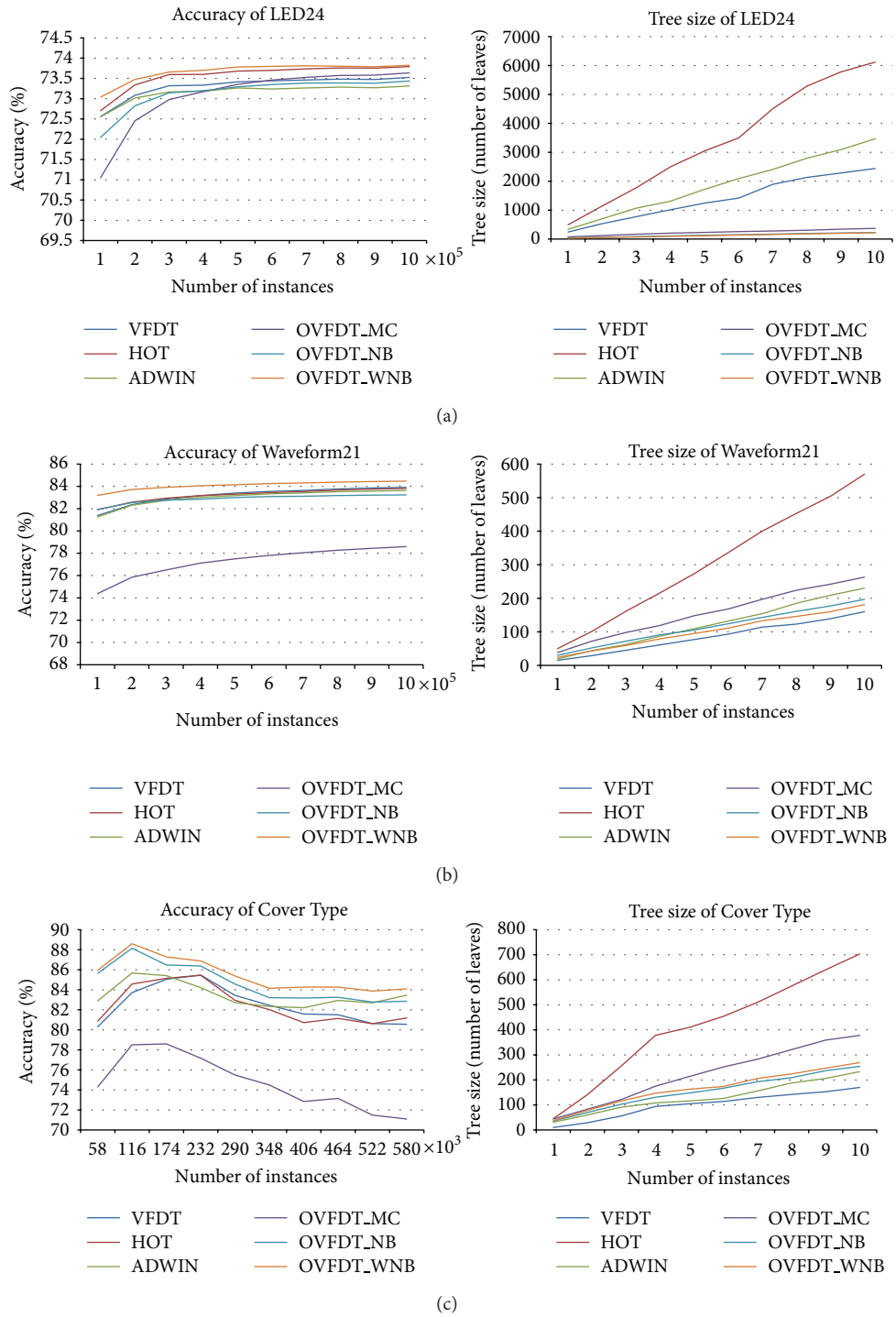

Figure 8: (a) Test-then-train evaluation of LED24. (b) Test-then-train evaluation of Waveform21. (c) Test-then-train evaluation of Cover Type.

$$\text{Precision}_i = \frac{\text{TP}_i}{(\text{TP}_i + \text{FP}_i)} = \frac{C_{ii}}{\left(\sum_{j=1}^{d} C_{ji}\right)}, \tag{23}$$

$$F_1\text{Measure}_i = \frac{2\text{TP}_i}{\text{TP}_i + \text{FN}_i + \text{TP}_i + \text{FP}_i} = \frac{2C_{ii}}{\sum_{j=1}^{d} C_{ij} + \sum_{j=1}^{d} C_{ji}}. \tag{24}$$

The pairwise of Precision-Recall results are shown in Table 4. OVFDT with WNB functional tree leaf has the best statistical results in Waveform21 data, which contains numeric attributes only. For the other two data, OVFDT also has the better TPR that represents that it has higher sensitivity in this experiment. For the ROC curve, Figure 7 displays the TPR-FPR analysis in an ROC space. Obviously, the plots of

OVFDT are higher than the other algorithms in ROC space that we say the proposed methods outperform others in this holdout test.

4.3. Test-Then-Train Evaluation. The second evaluation implements a test-then-train approach that is an incremental evaluation. When new instances come, they are used to test the current model tree that we write down the statistical results of accuracy and model size. After that, those instances are used to train and update the model tree. From Figure 8, we can see that OVFDT with WNB functional tree leaf obtains the best accuracy amongst the tested algorithms while smaller tree size relatively. Since the tree induction that uses optional tree leaves, HOT results much bigger model size than the other methods. This incremental evaluation shows OVFDT's outperformance on the aspects of accuracy and compact tree size.

5. Conclusion

Imperfect data stream leads to tree size explosion and detrimental accuracy problems. In original VFDT, a tie-breaking threshold that takes a user-defined value is proposed to alleviate this problem by controlling the node-splitting process that is a way of tree growth. But there is no single default value that always works well and that user-defined value is static throughout the stream mining operation. In this paper, we propose an Optimized-VFDT (OVFDT) algorithm that uses an adaptive tie mechanism to automatically search for an optimized amount of tree node splitting, balancing the accuracy and the tree size, during the tree-building process. The optimized node-splitting mechanism controls the attribute-splitting estimation incrementally. Balancing between the accuracy and tree size is important, as stream mining is supposed to operate in limited memory computing environment, and a reasonable accuracy is needed. It is a known contradiction that high accuracy requires a large tree with many decision paths, and too sparse the decision-tree results in poor accuracy. In the experiment, we use holdout test and incremental test to evaluate the proposed model. The results show that OVFDT achieves a better performance in terms of high prediction accuracy and compact tree size than the other VFDTs. This advantage can be technically accomplished by means of simple incremental optimization mechanisms as described in this paper. They are light weighted and suitable for incremental learning. The contribution is significant because OVFDT can potentially be further modified into other variants of VFDT models in various applications, while the best possible (optimal) accuracy and minimum tree size can always be guaranteed.

References

[1] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*, pp. 71–80, ACM, August 2000.

[2] S. Fong, H. Yang, S. Mohammed, and J. Fiaidhi, "Stream-based biomedical classification algorithms for analyzing biosignals," *Journal of Information Processing Systems*, vol. 7, no. 4, pp. 717–732, 2011.

[3] N. Chawla, N. Japkowicz, and A. Kolcz, "Special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations*, vol. 6, no. 1, 2004.

[4] D. Mladenic and M. Grobelnik, "Feature selection for unbalanced class distribution and naive bayes," in *Proceedings of the 16th International Conference on Machine Learning*, pp. 258–267, Morgan Kaufmann, Boston, Mass, USA, 1999.

[5] T. Elomaa, "The biases of decision tree pruning strategies," in *Advances in Intelligent Data Analysis*, vol. 1642 of *Lecture Notes in Computer Science*, pp. 63–74, 1999.

[6] B. Pfahringer, G. Holmes, and R. Kirkby, "New options for hoeffding trees," in *Proceedings of the Australian Conference on Artificial Intelligence*, pp. 90–99, 2007.

[7] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, pp. 523–528, ACM, Washington, DC, USA, August 2003.

[8] S. Hashemi and Y. Yang, "Flexible decision tree for data stream classification in the presence of concept change, noise and missing values," *Data Mining and Knowledge Discovery*, vol. 19, no. 1, pp. 95–131, 2009.

[9] H. Stefan, P. Russel, and S. K. Yun, "CBDT: a concept based approach to data stream mining," in *Advances in Knowledge Discovery and Data Mining*, vol. 5476 of *Lecture Notes in Computer Science*, pp. 1006–1012, 2009.

[10] J. Gama and P. Kosina, "Learning decision rules from data streams," in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*, T. Walsh, Ed., vol. 2, pp. 1255–1260, AAAI Press, 2011.

[11] H. Yang and S. Fong, "Moderated VFDT in stream mining using adaptive tie threshold and incremental pruning," in *Proceedings of the 13th International Conference on Data Warehousing and Knowledge Discovery (DaWaK '11)*, pp. 471–483, Springer, Toulouse, France, 2011.

[12] G. Hulten P Domingos, "VFML—a toolkit for mining high-speed time-changing data streams," 2003, http://www.cs.washington.edu/dm/vfml/.

[13] B. Albert, H. Geoff, P. Bernhard et al., "MOA: a real-time analytics open source framework," vol. 6913 of *Lecture Notes in Computer Science*, pp. 617–620.

[14] N. Oza and S. Russell, "Online bagging and boosting," in *Artificial Intelligence and Statistics*, pp. 105–112, Morgan Kaufmann, Boston, Mass, USA, 2001.

[15] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*, pp. 139–147, ACM, Paris, France, July 2009.

[16] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing," in *Proceedings of the of SIAM International Conference on Data Mining*, pp. 443–448, 2007.

[17] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*, pp. 97–106, ACM, August 2001.

[18] H. Geoffrey, K. Richard, and P. Bernhard, "Tie breaking in hoeffding trees," in *Proceedings of the 2nd International Workshop on Knowledge Discovery from Data Streams*, pp. 107–116, Porto, Portugal, 2005.

[19] A. Frank and A. Asuncion, *UCI Machine Learning Repository*, University of California, School of Information and Computer Science, Irvine, Calif, USA, 2010, http://archive.ics.uci.edu/ml/.

[20] J. P. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. E. Brodley, "Pruning decision trees with misclassification costs," in *Proceedings of the 10th European Conference on Machine Learning (ECML '98)*, pp. 131–136, Springer, 1998.