Optimized Very Fast Decision Tree with Balanced Classification Accuracy and Compact Tree Size

Hang Yang, Simon Fong
Faculty of Science and Technology, University of Macau
Av. Padre Tomás Pereira Taipa, Macau, China
henry.yh@gmail.com, ccfong@umac.mo

***Abstract*—Very Fast Decision Tree (VFDT) in data stream mining has been widely studied for more than a decade. VFDT in essence can mine over a portion of an unbounded data stream at a time, and the structure of the decision tree gets updated whenever new data feed in; hence it can predict better upon the input of fresh data. Inherent from traditional decision trees that use information gains for tree induction, VFDT may suffer the same over-fitting problem where noise in the training data leads to excessive tree branches therefore decline in prediction accuracy. This problem is aggravated in stream mining because limited run-time memory (for storing the whole decision tree) and reasonable accuracy are often the criteria for implementing VFDT. Post-pruning that was a popular technique used in traditional decision tree to keep the tree size in check, however may not be applicable for VFDT in situ. In this paper a new model that extends from VFDT called Optimized VFDT is proposed, for controlling the tree size while sustaining good prediction accuracy. This is enabled by using an adaptive threshold tie and incremental pruning in tree induction. Experimental results show that an optimal ratio of tree size and accuracy can be achieved by OVFDT.**

Keywords- Stream Mining; Very Fast Decision Tree; OVFDT; Tree Pruning; Incremental Optimization

I. INTRODUCTION

Data stream mining (DSM) has been received a lot of research attentions because this state-of-art is able to meet the requirement of discovering knowledge from very large data in real-time. Traditional data mining methods, that require learning from the whole set of historical data, do not suit the fast changes in high speed data streams in real-time applications. Very Fast Decision Tree (VFDT) [1] is a well-known decision tree algorithm for DSM. Its underlying principle is a dynamic decision tree building process that uses a Hoeffding Bound (HB) to determine the conversion of a tree-leaf to a tree node by accumulating sufficient statistics of the new samples. But when the data streams are impaired by noise, VFDT faces a problem of tree size explosion and deterioration of prediction accuracy. This problem obstructs the development of VFDT applications in real-life. Imperfect data are inevitable because of unreliable communication hardware, or temporarily data loss due to network traffic fluctuation. It is an open problem for VFDT that the tree size grows tremendously when noise-infested data are streaming in, and the classifier's accuracy drops. A good decision tree should possess comprehensible tree size and acceptable accuracy. At times, tree size and accuracy are opposite of each other – despite too large the tree size impairs accuracy due to over-fitting, high prediction accuracy grounded from sufficiently large decision tree. Too coarse of a decision tree hardly can yield good prediction accuracy. In other words, optimal prediction accuracy would be resulted from a tree that is neither too large nor too small. The optimum is defined here as the balance between the highest possible prediction accuracy and the most compact tree size. In this paper, we devise a new version of VFDT called Optimized VFDT (OVFDT) that can provide sustainable prediction accuracy and regulate the growth of decision tree size to a reasonable extent, even in the presence of noise. The optimization mechanism covers both the tree construction and tree pruning processes. On one hand, the optimized splitting-estimation adapts to the distribution of the input data samples, and in turn influences the HB value which is a key factor in the decision tree construction. It is adaptive in a sense that no human intervention is required during the data stream mining; we let the incoming data itself decide how precisely (or how frequently) the tree-node splitting should be done. On the other hand, an incremental optimization is proposed here as a pre-pruning mechanism to constraint attribute-splitting from becoming an excessively large tree. By using synthetic and real-world datasets, the experimental results show that OVFDT finds an optimally compact decision tree that has good prediction accuracy as well.

This paper is structured as follow: Section 2 introduces a research background of VFDT, the effect of tie-threshold in tree-building and incremental optimization. Section 3 presents the details of our proposed model OVFDT, including the adaptive tie-threshold and incremental optimization mechanisms. Experiments are described in Section 4. Section 5 concludes.

II. RESEARCH BACKGROUND AND EXISTING PROBLEMS

A. VFDT Tree-growing Process by HB and Tie-threshold

VFDT is built by recursively replacing leaves with decision nodes as per new sample arrives. Sufficient statistics of attribute values are stored in each leaf. Heuristic evaluation function is used to determine split attributes converting from leaves to nodes. Nodes contain the split attributes and leaves contain only the class labels. When a sample enters, it traverses the tree from root to a leaf,

evaluating the relevant attribute at every single node. After the sample reaches a leaf, the sufficient statistics are updated. At this time, the system evaluates each possible condition based on attribute values: if the statistics are enough to support the one test over the others, a leaf is converted to a decision node. The tree-growing process that contains splitting check using heuristic evaluation function *G(.)* and HB. VFDT uses information gain as *G(.)*. $\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$. HB is used by the necessary number of samples to ensure control over error in attribute-splitting distribution selection. For n independent observations of a real-valued random variable r whose range is R, HB illustrates that with confidence level 1-δ, the true mean of r is at least $\bar{r} - \varepsilon$, where $\bar{r}$ is the observed mean of samples. For a probability the range R is 1, and for information gain the range R is $\log_2$Class#.VFDT makes use of HB to choose a split attribute as a decision node. Let x_a be the attribute with the highest *G(.)*, x_b be the attribute with the second-highest *G(.)*. $\Delta\bar{G} = \bar{G}(x_a) - \bar{G}(x_b)$ is the difference between the two top quality attributes. If $\Delta\bar{G} > \varepsilon$ with N samples observed in leaf, while the HB states with probability *1-*δ, that x_a is the attribute with highest value in *G(.)*, then the leaf is converted into a decision node which splits on x_a.

Variants of VFDT have been developed by researchers [11,12,13,14,15] by extending from the original version of VFDT which is based on the principles of Hoeffding Tree (HT). In HT, a tie breaking parameter τ is added as an additional splitting condition, so that whenever the HB becomes very small that means the difference between the best and the second best splitting attributions is not obvious, τ stands in as a quick deceive parameter to resolve the tie. Using an arbitrarily chosen value that is smaller than τ as a comparing reference, and the value is fixed throughout the operation, the candidate node is selected to be split on the current best attribute regardless of how close the second best candidate splitting attribute is. It is observed that the excessive invocation of tie breaking brings HT performance declining significantly on complex and noise data, even with the additional condition by the parameter τ. A proposed solution [2] to overcome this detrimental effect is an improved tie breaking mechanism, which not only considers the best and the second best splitting candidates in terms of heuristic function, but also uses the worst candidate. A correctly chosen τ can effectively control a minimum tree-growing speed that the tree size is supervised. However, there is no single default value that works always well in all tasks so far. The choice of τ hence depends on the data and their nature.

B. Tree Size Explosion Problem

Complexity of a decision tree structure in form of tree graph greatly influences its applicability. A significant advantage of decision tree classification is that the tree-like graph has a high degree of interpretability. Ideally we want a compact decision tree that possesses just sufficient rules for classification and prediction with certain accuracy and interpretability. One culprit that leads to tree size explosion is noise in the training data, a well know phenomenon is called over-fitting in decision trees. Noises in data samples are considered as a type of irrelevant or meaningless data that do not typically reflect the main trends but make the identification of these trends more difficult. However, prior to the start of the decision tree induction, we do not know which samples are noises; filtering noise is thus difficult.

Tree size explosion problem, not only exists in HT, but also in tradition decision tree (TDT) [8,9,10]. The tree explosion associates with many erroneous concepts caused by the non-informative noise data. In Table I, an experiment demonstrates the relationship between noise and tree size in VFDT (with a typical τ = 0.05). Experimental datasets are synthetic LED24 with different noise percentages (NP). At each time of node splitting, the corresponding HB is recorded. According to those recorded HB, the mean and the variance are computed respectively. Range is computed by taking the difference of the maximum and the minimum in the recorded values. With NP increasing, the accuracy of VFDT is declining, and the number of leaves is fluctuating as well.

TABLE I. MEAN AND VARIANCE CHANGE INDIFFERENT NPDATA

NP	VFDT		HB Mean		HB Variance	
	Acc	*Leaf#*	*Val*	*Range*	*Val*	*Range*
0	89.93	10	0.06	0.01	0.00	0.00
1	84.51	83	0.17	0.18	0.03	0.04
2	84.23	51	0.11	0.13	0.02	0.04
3	81.12	77	0.15	0.15	0.03	0.04
4	79.35	53	0.10	0.08	0.01	0.02
5	76.60	52	0.10	0.08	0.01	0.02
6	74.05	48	0.10	0.08	0.01	0.01
7	74.00	81	0.21	0.20	0.03	0.04
8	69.91	83	0.17	0.16	0.03	0.04
9	67.98	82	0.19	0.19	0.03	0.04
10	63.78	46	0.09	0.05	0.00	0.01

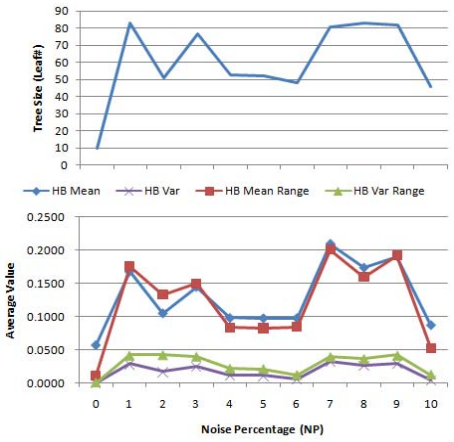

Figure 1. Tree size-growing corresponds to HB, mean and variance.

Obviously in Figure 1, we see the pattern of tree size is corresponding to HB mean, HB variance and their ranges. This phenomenon has a strong implication that a steady HB is desirable across different noise levels for constructing a decision tree. In other words, if we can hold tight of the HB

fluctuation the resultant decision tree could be relieved from the ill-effect of data noise, at least to certain extent. The mathematical property of HB is defined as a conservative function and has been used classically in VFDT for many years. (HB formulation is simple and works well in stream mining, it depends on the desired confidence and the number of observations.) We are inspired to modify the node splitting function based on the mean of HB instead of modifying the classical HB formulation. Intuitively speaking, holding on to the mean of HB is equivalent to evading the fluctuation of HB values, thereby reducing the noise effects.

C. Post-pruning and Pre-pruning Mechanisms

Over-fitting problem appears by inferring a larger tree-structure than it is justified by the training dataset. To alleviate this problem, decision tree pruning is proposed to remove nodes which, statistically, seem likely to arise from the noise data. Replacing the sub-tree by a single leaf with a lower estimated error is common in pruning algorithms.

Figure 2 shows the different procedures of post-pruning and pre-pruning in decision tree induction. In general, post-pruning is implemented after a full decision tree is grown. But pre-pruning is implemented during the tree-growth.

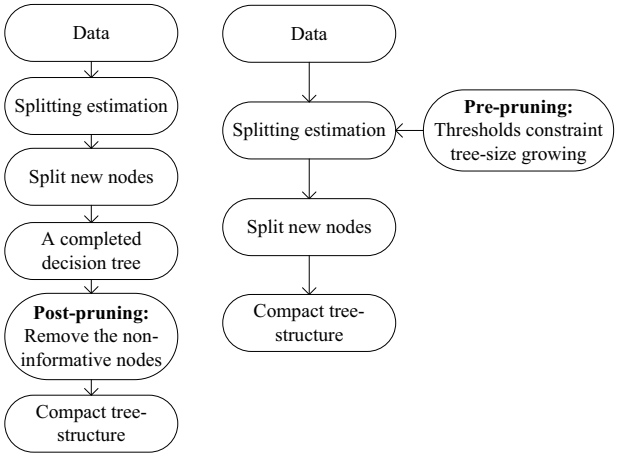

Figure 2. Comparison of post- and pre-pruning processes.

In TDT, the widely-used pruning methods for minimizing errors are error-based pruning [8] and cost-complexity pruning [9]. Laplace correction [10] is applied to a loss-based pruning mechanism working with C4.5, and it seems to have a better performance than error-based pruning. The traditional approach is to first grow a full decision tree and prune the non-informative nodes backward, which is also called post-pruning. Using the same heuristic function, post-pruning approach may obtain a globally fit decision tree because it covers a full training set of observed samples.

However, it is impossible to load the full dataset in data stream mining. Post-pruning after the full-grown tree generated is not suitable for stream mining over unbounded data. Pre-pruning uses some thresholds to constraint the splitting speed, e.g., tie-breaking threshold in VFDT [1,2]. Hence pre-pruning is applicable for HT because the tree is changing continuously with no end.

D. Incremental Optimization

HT handles streaming data that tree-structure keeps on updating when new data arrive. HT only requires reading some samples satisfying the statistical bound (HB) to construct a decision tree. Since HT 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. In this paper, for optimizing HT, we propose to use incremental optimization methods.

Incremental optimization requires solutions that evolve over time in response to environmental changes. In general, there are three performance metrics for incremental problems: *ratio*, *sum* and *demand* [3]. *Ratio metric* uses a worst-case measurement to determine the distance between the optimal solution and the solution made at each time step. *Sum metric* is the expected value metric over all time steps. A weight function while summing solution values can easily settle the problem of natural bias for late-stage solution. *Demand metric* is a decision metric measuring the degree of specific quantitative requirements satisfaction.

Π : an optimization problem, $\Pi \rightarrow (\mathrm{X}, \mathcal{F}, v)$
X : a set of collected objects
$\mathcal{F}$: a set of feasible solutions, $\mathcal{F} \subseteq 2^X$
v : a valuation function, $\mathcal{F} \rightarrow \mathbb{R}$
S : a feasible solution, $S \subseteq \mathcal{F}$
S^* : an optimal solution, $S^* \in \mathcal{F}$
The optimization goal is to find the S^*:
$$v(S^*) \geq v(S) \text{ or } v(S^*) \leq v(S)$$

Figure 3. Single-level optimization model.

k : total level of an optimization task
l : a specific level, $1 \leq l \leq k$
Π^k : an optimization problem with k levels
X : a set of collected objects
$\mathcal{F}_l$: a set of feasible solutions in level l
ω_l : a given weight for level l
d_l : a given demand metric for level l
v : a valuation function, $\sum_{l=1}^{k} \mathcal{F}_l \rightarrow \mathbb{R}$
S : an incremental solution for Π^k,
$S = (S_1, S_2, \ldots, S_k)$ where $S_l \in \mathcal{F}_l$
S_l^* : an optimal solution at level l
S_l : a feasible solution at level l
The optimization goal is to find the S:
- The sum objective:
$$max \sum_{l=1}^{k} \omega_l \times v(S_l) \;\; OR \; min \sum_{l=1}^{k} \omega_l \times v(S_l)$$
- The ratio objective:
$$max_l \frac{v(S_l)}{v(S_l^*)} \;\; OR \;\; min_l \frac{v(S_l)}{v(S_l^*)}$$
- The demand objective
$$v(S_l) \leq d_l \text{or} \;\; v(S_l) \geq d_l$$

Figure 4. Multi-level incremental optimization model.

According to the complexity of tasks, normal optimization method that analyzes the full dataset can be considered as a single-level optimization problem in Figure 3, which is described as [4,5]. In the full dataset, heuristic evaluation can find all feasible solutions in a closed range.

In HT, the full dataset is unknown that heuristic evaluation can only search for the feasible solutions within a certain known range. The uncertainties in the unknown range make it hard to find global optima. To solve this problem, [6,7] proposed a multi-level incremental optimization as shown in Figure 4. In each level, an optimal solution is found progressively, using objective function to evaluate the feasible solution in each level (splitting estimation).

III. OPTIMIZED VERY FAST DECISION TREE (OVFDT)

A. Assumptions

OVFDT is an extended version of the original VFDT, which is proposed for achieving reasonable prediction accuracy and compact tree size. OVFDT is designed to solve the over-fitting problem of tree size explosion in noise data, by an adaptive tie evaluation and extra pruning conditions. In this paper, OVFDT is based on the following assumptions:

- Tree-structure is updated when a new attribute is split as a new node in certain period. Such attribute-splitting improves the current tree's accuracy in such period, but the split enlarges the tree.
- A branch-path from the root to a leaf presents a concept, which can be structured as an IF-THEN-ELSE rule. The number of leaves is the number of concepts included in a decision tree. The classifying accuracy relates to the number of concepts. In other words, accuracy is related to the tree size.
- Suppose an efficient heuristic function *G(.)* is obtained, and attribute-splitting estimation is going to be evaluated by this value.
- The best tie-breaking τ is not known until all the possibilities have been tried on the whole training set.

B. Incremental Optimization Model

In VFDT, the estimation for splits and ties are only executed once for every n_{min} which is a value arbitrarily chosen by the user, number of samples that arrive at a leaf. Instead of a pre-configured tie threshold, OVFDT uses an adaptive tie threshold that is calculated by incremental computing. The key objective here is to know when to do a split and when not to, as the incoming data arrive and updating of the tree is considered in each round. This process takes cues from the data themselves, by considering the HB mean per n_{min} amount of arriving data. The process iterates. In addition, OVFDT uses some constraints in attribute-splitting estimation, which narrows the scope of splitting and reduces the tree size-growing speed – this is known as the pre-pruning process.

Incremental optimization model used in OVFDT is presented in Figure 5. The performance optimization target is represented by a smaller tree size and a higher accuracy. A *gain* is used to measure the change between two HT structures in two different attribute-splitting estimations. The optimization goal is to minimize the sum of ratio in (2).

Parameters and variables:

i : the index of splitting estimation
m : total number of estimations seen so far
HT_i : HT structure after i^{th} estimation
HT^* : optimized HT structure
C_T: number of samples fall to existent leaves
C_F: number of samples don't fall to existent leaves
ΔC_i: difference of C_T and C_F at i^{th} esimtation
μ_l: a binary variable taking value 1 if ε relates to leaf l; and 0, otherwise
ε_i: HB value at i^{th} estimation , computed by (1)
$\overline{\varepsilon_l}$: the mean of ε_l
$\overline{\varepsilon_l}^{MAX}$: the maximum mean of ε_l
$\overline{\varepsilon_l}^{MIN}$: the minimum mean of ε_l
p: probobility variable of attribute $-$ splitting
ΔG: difference between the best and the second best heuristic evalating values
$TreeSizeGain()$: tree size change function
$AccuGain()$: accuracy change function
P(): probability function of meeting the splitting requirements

The incremental optimization aims to find HT^*:

$$HT^* = \arg\min \sum_{i=1}^{m} \frac{TreeSizeGain(HT_i)}{AccuGain(HT_i)} \quad (2)$$

Subject to:

$$\Delta C_i = C_T - C_F \quad (3)$$

$$\overline{\varepsilon_l} = \frac{1}{m}\sum_{i=1}^{m} \mu_l \cdot \varepsilon_i \quad (4)$$

$$\overline{\varepsilon_l}^{MAX} = \arg max\, \overline{\varepsilon_l} \quad (5)$$

$$\overline{\varepsilon_l}^{MIN} = \arg min\, \overline{\varepsilon_l} \quad (6)$$

$$p = \begin{cases} P\left(\Delta G \le \overline{\varepsilon_l}^{MIN}\right) \cdot P(\Delta C_i < \Delta C_{i-1}) \\ P\left(\Delta G \le \overline{\varepsilon_l}^{MIN}\right) \cdot P(\Delta C_i < 0) \\ P(\overline{\varepsilon_l}^{MIN} < \Delta G < \overline{\varepsilon_l}^{MAX}) \cdot P(\Delta C_i < \Delta C_{i-1}) \end{cases} \quad (7)$$

$$L_i = L_{i-1} + n_{min} \cdot p \quad (8)$$

$$AccuGain(HT_i) = \Delta C_i - \Delta C_{i-1} \quad (9)$$

$$TreeSizeGain(HT_i) = L_i - L_{i-1}\ , (L_0 = 1) \quad (10)$$

Figure 5. OVFDT incremental optimization model.

HT algorithm uses sufficient statistics to count the number of samples falling into a leaf, traversing from the root to a leaf, over the existing tree-structure. Suppose C_T is the count of the new samples that fall into the existent leaves, C_F is the count of the new samples that do not fall into the existent leaves. At the *i*-th splitting estimation, the measurement of accuracy ΔC_i is calculated by the difference of C_T and C_F, in (3). If $\Delta C_i \ge 0$, it means the number of

correctly classified samples is no less than that of wrongly classified in the current tree-structure; otherwise, the current tree-graph needs to be updated. *The first pre-pruning condition is the tree-structure need not be updated until* $[\Delta \boldsymbol{C_i} < 0]$**,** *with a probability of* $P(\Delta C_i < 0)$ *at a splitting estimation.*

Suppose $AccuGain$ is the gain of accuracy at the *i*-th estimation, being calculated in (9). If $AccuGain(HT_i) \geq 0$, it means the measurement of precision at the HT-structure is no worse than that of the last time estimation; otherwise, the old tree-structure needs to be updated. Therefore, *the second pre-pruning condition is the tree-structure needs not be updated until* $[\Delta \boldsymbol{C_i} < \Delta \boldsymbol{C_{i-1}}]$**,** *with a probability of* $P(\Delta C_T < \Delta C_F)$ *at a splitting estimation.*

OVFDT uses the mean of HB evaluated by the *m* times of splitting-estimations in (3). $\bar{\varepsilon}_l^{MIN}$ is the minimum mean of recorded HB in *m* times splitting estimations, falling to the leaf *l*, in (6); $\bar{\varepsilon}_l^{MAX}$ is the maximum mean of recorded HB in *m* splitting estimations, falling to the leaf *l,* in (5). The adaptive tie substitutes a user-defined value with a scope of $[\bar{\varepsilon}_l^{MIN}, \bar{\varepsilon}_l^{MAX}]$. Instead of using a user-supplied tie value in splitting, the third pre-pruning condition is enforced: *the tree-structure needs not be updated until* $[\Delta G \leq \bar{\varepsilon}_l^{MIN}]$*, with a probability of* $P(\Delta G \leq \bar{\varepsilon}_l^{MIN})$ *in a splitting estimation; or the tree-structure needs not be updated until* $\left[\bar{\varepsilon}_l^{MIN} < \Delta G < \bar{\varepsilon}_l^{MAX}\right]$ *with* $P(\bar{\varepsilon}_l^{MIN} < \Delta G < \bar{\varepsilon}_l^{MAX})$ *at a splitting estimation.*

The estimation is implemented once for every n_{min} number of samples that arrives at a leaf. The tree size increases by *l* when a new node splits. The number of samples meets the first pruning condition is $(n_{min} \cdot p)$, where p is calculated by (7). Only one value of p can be chosen at one splitting estimation. The calculation of tree size at estimation i is given in (8).

Comparing OVFDT to the original VFDT tie-breaking mechanism, here we want to prove that VFDT is not better than OVFDT at any given time. The attributes included in a data stream can be considered as random variables that the statistics of HB is Gaussian distribution. $P(\bar{\varepsilon}_l^{MIN} < \Delta G < \bar{\varepsilon}_l^{MAX})$is greater than that of $P(\Delta G < \bar{\varepsilon}_l^{MIN})$. In addition, $P(\Delta C_i < 0)$ has a narrower range than $P(\Delta C_i < \Delta C_{i-1})$. Therefore, the variable p should take a value from: $\left[P\left(\bar{\varepsilon}_l^{MIN} < \Delta G < \bar{\varepsilon}_l^{MAX}\right) \cdot P(\Delta C_i < \Delta C_{i-1})\right] > \left[P\left(\Delta G \leq \bar{\varepsilon}_l^{MIN}\right) \cdot P(\Delta C_i < \Delta C_{i-1})\right] > [P\left(\Delta G \leq \bar{\varepsilon}_l^{MIN}\right) \cdot P(\Delta C_i < 0)]$. In original VFDT, the tie-breaking is determined by a user-defined value τ, where the splitting condition is that $[\Delta G \leq \tau]$. Because $\bar{\varepsilon}_l$ is the mean calculated incrementally over the data samples, the range of $P(\Delta G \leq \tau)$ is wider than that of $P\left(\bar{\varepsilon}_l^{MIN} < \Delta G < \bar{\varepsilon}_l^{MAX}\right)$. For this reason, $P(\Delta G \leq \tau) > P_3 > P_1 \geq P_2$. When VFDT and OVFDT have the same *AccuGain*, VFDT's *TreeSizeGain* is bigger than that of OVFDT. In other words, OVFDT has a smaller ratio than original VFDT.

C. OVFDT Tree-building Process

The pseudo code of the OVFDT algorithm is presented in Figure 6. It contains three sub-procedures:

- Initialize a tree when the first data stream comes (Figure 7);
- Whenever new data arrives, traverse the samples from the root to a leaf, updating the sufficient statistics and ΔC_i (Figure 8);
- Implement the splitting estimation to determine whether the attribute shall split or not, and return an updated HT (Figure 9).

Input:
S: a stream of samples
X: a set of symbolic attributes
G(.): a split heuristics evaluation function
δ: one minus the desired probability of choosing the correct attribute at any given node
n_{min}: # sample between estimations for growth
Output:
HT: a decision tree
Procedure OptimizeVFDT(*S, X, G,* δ, n_{min}):
A data stream *S* arrives
IF *HT*= $\emptyset$, THEN *InitializeTree(S,X)*
An initialized *HT* with a single root is obtained
IF *HT*≠ $\emptyset$, THEN, *TraverseNewStream(S,X)*
$n_{ijk}(l)$, ΔC_i are computed
Recall ΔC_{i-1}
Perform *i*-th estimation of attribute-splitting
Split (ΔC_i, ΔC_{i-1}, *S, X, G,*δ, n_{min}*)*
Return *HT*

Figure 6. OVFDT Procedure: Optimize VFDT.

Procedure InitializeTree(*S,X*):
Let *HT* be a tree with a single leaf *l* (the root)
Let $X_l = X \cup \{X_\emptyset\}$
Let $\bar{G}_l(X_\emptyset)$ be the $\bar{G}$ obtained by predicting the most frequent class in *S*
FOR each class y_k
FOR each value x_{ij} of each attribute $X_i \in X$
Let $n_{ijk}(l) = 0$
END-FOR
END-FOR
Return HT

Figure 7. OVFDT Procedure: Initialize a Tree.

Procedure TraverseNewStream(*S,X*):
A new sample (*x,y*) in *S* arrives:
Sort (*x,y*) into a leaf *l* using *HT*
IF $y \in X_l$, THEN increment: $C_T = C_T + 1$
IF $y \notin X_l$, THEN increment: $C_F = C_F + 1$
Compute $\Delta C_i = C_T - C_F$
FOR each x_{ij} in x such that $X_i \in X_l$
Increment $n_{ijk}(l)$
END-FOR
Return ΔC_i

Figure 8. OVFDT Procedure: Traverse a a new stream.

Procedure Split (ΔC_i, ΔC_{i-1}, *S, X, G,δ,* n_{min}):
Label *l* with the majority class among the samples seen so far at *l*
Let n_l be the number of samples seen at *l*
IF the samples seen so far at *l* are not all of the same class and ($n_l \bmod n_{min}$) = 0, THEN
FOR each attribute $X_i \in X_l - \{X_\emptyset\}$
Compute $\bar{G}_l(X_i)$
Let X_a be the attribute with highest $\bar{G}_l$
Let X_b be the attribute with 2nd highest $\bar{G}_l$
Compute ε using (1)
Let $\Delta\bar{G}_l = \bar{G}_l(X_a) - \bar{G}_l(X_b)$
END-FOR
IF ($\Delta\bar{G}_l > \varepsilon$) or
[($\Delta\bar{G}_l \leq \bar{\varepsilon}_l^{MIN}$) and ($\Delta C_i < \Delta C_{i-1}$)] or
[($\Delta\bar{G}_l \leq \bar{\varepsilon}_l^{MIN}$) and ($\Delta C_i < 0$)] or
[($\bar{\varepsilon}_l^{MIN} < \Delta\bar{G}_l \leq \bar{\varepsilon}_l^{MAX}$) and ($\Delta C_i < \Delta C_{i-1}$)]
Replace *l* by an internal node splits on X_a
FOR each branch of split
Add a new leaf l_m and let $X_m = X - \{X_a\}$
Let $\bar{G}_m(X_\emptyset)$ be the $\bar{G}$ obtained by predicting the most frequent class at l_m
FOR each class y_k and each value x_{ij} of each attribute$X_i \in X_m - \{X_\emptyset\}$
Let $n_{ijk}(l_m) = 0$
END-FOR
Estimation time increment: *m=m+1*
Compute $\bar{\varepsilon}_l$using (3)
$\bar{\varepsilon}_l^{MAX} = \arg max\, \bar{\varepsilon}_l$
$\bar{\varepsilon}_l^{MIN} = \arg min\, \bar{\varepsilon}_l$
END-FOR
END-IF
END-IF
Return *HT*

Figure 9. OVFDT Procedure: Estimate Splitting.

IV. EXPERIMENTS

OVFDT algorithm has been implemented as a package class in Massive Online Analysis (MOA) which is an open-source Java platform for stream mining. The experiment is run on a workstation with 2.83GHz CPU and 8G RAM. In the experiment, a variety of large data streams, with nominal and numeric attributes, are used to stress test the OVFDT. The experiment aims to verify if OVFDT outperforms HT-based algorithms with different user-defined values of tie. OVFDT is compared with VFDT [1], Genuine Tie Detection Tree (GTDT) [2] and Hoeffding Option Tree (HOT) [16].

The parameters in MOA are set as follow: n_{min} is 200, δ is 10^{-6}, numeric estimator is *Gauss100*. The tie-threshold value $\tau \in (0,1)$. The default value suggested by MOA is 0.05. The synthetic datasets are generated by MOA standard data generator classes, and real-world datasets are downloaded from UCI machine learning archive. The datasets consist of three types: nominal, numeric and mixed.

Since the datasets have different characteristics, we use their overall averaged values to evaluate the performance which is represented by three key indicators: the rate of accuracy, the number of concepts (tree size) that also represents the complexity of a decision tree, and time taken. The simulation shows the pruning mechanism has played a significant role for balancing the accuracy and the tree size growth. The experiment of VFDT with different ties is used to validate that OVFDT is better than VFDT in all tie values.

Accuracy: In general, the accuracy of a VFDT increases with more and more data stream samples arrive, as long as there is no concept-drift. Classification accuracy for VFDT is commonly evaluated as the percentage of correctly classified instances over the total number of samples.

From the experiment result in Table II, HOT has the highest accuracy, for all types of datasets. It uses additional option nodes that allow several tests to be applied, leading to multiple HTs as separate paths. The rank of average accuracy (Ac) results as: $\mathrm{Ac_{HOT}} > \mathrm{Ac_{T.5}} > \mathrm{Ac_{GT}} > \mathrm{Ac_{OVFDT}} > \mathrm{Ac_{T.05}}$.

Tree Size: Although HOT obtains the best accuracy in our experiment, it results in the biggest tree size. As seen from Table IV, when a small tie (0.05) is to control attribute-splitting, consequently the strict splitting-condition leads to a least amount of leaves (leaf#). The experiment shows the leaf# (Ln): $\mathrm{Ln_{T.05}} < \mathrm{Ln_{OVFDT}} < \mathrm{Ln_{T.5}} < \mathrm{Ln_{GT}} < \mathrm{Ln_{HOT}}$.

The more leaves are, the more concepts that the decision tree embraces. However it may be a result of noise causing an excessively large tree. On the other hand, if the tree size is too small, that means insufficient decision paths are there, hence low in classification accuracy for the decision tree.

Time: The computation time is the total time taken in seconds for processing a given full set of data stream. The results are in Table VI. For the nominal datasets, VFDT with a small tie (0.05) has the shortest running time obviously. In the same numeric estimating mechanism (*Gauss100*), VFDT with tie (0.5) runs a little faster than the others. However, HOT took the longest time amongst all the experiments. Because the adaptive tie computing mechanism and pruning are embedded, OVFDT consumed slightly longer time than the others. The time taken (T) comparison: $\mathrm{T_{T.05}} < \mathrm{Ln_{T.5}} < \mathrm{Ln_{GT}} < \mathrm{Ln_{OVFDT}} < \mathrm{Ln_{HOT}}$.

Optimization: So far we find the tree size of OVFDT is smaller than all other methods, although its accuracy is not the highest. The aim of optimizing the HT algorithm is to achieve an optimal balance between accuracy and tree size. The HT algorithm performance is represented by the ratio *TreeSizeGain*/*AccuracyGain* which is an important indicator for evaluating the incremental optimization. From Table V, VFDT using tie (0.05) has the smallest tree size. We use VFDT (Tie0.05) as a benchmark. The measurement M_x is:

$$M_x = \frac{\mathrm{Leaf\#}(HT_{\mathrm{x}}) - \mathrm{Leaf\#}(HT_{\mathrm{benchmark}}) \Big/ \mathrm{Leaf\#}(HT_{\mathrm{benchmark}})}{\mathrm{Accu}(HT_{\mathrm{x}}) - \mathrm{Accu}(HT_{\mathrm{benchmark}}) \Big/ \mathrm{Accu}(HT_{\mathrm{benchmark}})}$$

TABLE II. ACCURACY COMPARISON

	Data Name	VFDT (Tie0.05)	VFDT (Tie0.5)	VFDT (GT)	HOT	O-VFDT
Nominal	LED_NP0	89.93	99.84	99.89	99.84	99.87
	LED_NP10	63.78	73.68	73.80	73.72	73.71
	LED_NP20	44.47	51.16	51.15	51.16	51.15
	LED_NP30	27.97	32.85	32.85	32.87	32.81
	CONNECT	67.30	69.00	69.00	71.79	67.50
	NURSERY	82.09	83.78	83.78	85.46	83.06
	Average	*62.59*	*68.39*	*68.41*	*69.14*	*68.01*
Numeric	SEA_NP0	98.52	99.31	99.31	99.56	98.40
	SEA_NP10	88.52	89.24	89.24	89.29	88.70
	SEA_NP20	78.76	79.08	79.08	79.58	78.99
	WAVE21	76.48	80.63	80.63	82.57	79.53
	WAVE40	76.46	80.54	80.54	82.56	79.07
	RBFS	82.37	86.86	86.86	89.46	84.01
	RBFC	66.42	85.72	85.72	90.86	70.31
	Average	*81.07*	*85.91*	*85.91*	*87.70*	*82.72*
Mix	RTS	92.89	94.09	94.09	94.83	93.59
	RTC	95.01	95.73	95.73	96.00	95.39
	Average	*93.95*	*94.91*	*94.91*	*95.42*	*94.49*
Overall Average		***75.40***	***80.09***	***79.76***	***80.96***	***78.05***

TABLE III. ACCURACY COMPARISON OF DIFFERENT TIES IN VFDT

VFDT TIE	Nominal	Numeric	Mixed	Overall
0.05	62.59	81.07	93.95	75.40
0.10	58.37	84.29	94.47	75.28
0.15	67.43	85.32	94.66	79.41
0.20	67.59	85.84	94.29	79.67
0.25	68.21	85.88	94.91	80.01
0.30	68.56	85.91	94.91	80.17
0.35	67.96	85.91	94.91	79.93
0.40	68.01	85.91	94.91	79.95
0.45	68.37	85.91	94.91	80.09
0.50	68.39	85.91	94.91	80.10
0.55	68.40	85.91	94.91	80.11
0.60	68.40	85.91	94.91	80.11
0.65	68.40	85.91	94.91	80.11
0.70	68.41	85.91	94.91	80.11
0.75	68.41	85.91	94.91	80.11
0.80	68.41	85.91	94.91	80.11
0.85	68.41	85.91	94.91	80.11
0.90	68.41	85.91	94.91	80.11
0.95	68.41	85.91	94.91	80.11
Average	***67.43***	***85.53***	***94.79***	***79.53***

TABLE IV. TREE SIZE (LEAF#) COMPARISON

	Data Name	VFDT (Tie0.05)	VFDT (Tie0.5)	VFDT (GT)	HOT	O-VFDT
Nominal	LED_NP0	10	10	10	19	10
	LED_NP10	46	2440	3842	6117	398
	LED_NP20	46	2397	3680	2805	1454
	LED_NP30	38	2404	3636	5133	3636
	CONNECT	23	437	437	1493	41
	NURSERY	13	72	72	102	13
	Average	*29*	*1293*	*1946*	*2612*	*925*
Numeric	SEA_NP0	115	308	308	755	55
	SEA_NP10	418	3799	3799	6721	375
	SEA_NP20	371	3967	3967	16219	608
	WAVE21	156	3660	3660	9663	477
	WAVE40	160	3667	3667	10382	465
	RBFS	425	2819	2819	8506	384
	RBFC	407	3458	3458	11075	763
	Average	*293*	*3097*	*3097*	*9046*	*447*
Mix	RTS	1644	2473	2473	4243	1756
	RTC	629	1478	1478	1903	587
	Average	*1137*	*1850*	*1976*	*3073*	*1172*
Overall Average		***300***	***2052***	***2226***	***5694***	***729***

TABLE V. TREE SIZE (LEAF#) OF DIFFERENT TIES IN VFDT

VFDT TIE	Nominal	Numeric	Mixed	Overall
0.05	29	293	1137	300
0.10	848	1018	1269	984
0.15	168	2063	1571	1239
0.20	282	2732	2207	1682
0.25	404	2915	1976	1785
0.30	590	3062	1976	1928
0.35	701	3097	1976	1989
0.40	991	3097	1976	2105
0.45	1057	3097	1976	2132
0.50	1293	3097	1976	2226
0.55	1704	3097	1976	2390
0.60	1913	3097	1976	2474
0.65	1933	3097	1976	2482
0.70	1946	3097	1976	2487
0.75	1946	3097	1976	2487
0.80	1946	3097	1976	2487
0.85	1946	3097	1976	2487
0.90	1946	3097	1976	2487
0.95	1946	3097	1976	2487
Average	***1242***	***2755***	***1885***	***2034***

TABLE VI. COMPUTATION TIME COMPARISON

	Data Name	VFDT (Tie0.05)	VFDT (Tie0.5)	VFDT (GT)	HOT	O-VFDT
Nominal	LED_NP0	4.90	4.87	4.91	9.48	5.58
	LED_NP10	5.91	7.44	9.91	20.75	7.30
	LED_NP20	5.82	9.83	11.14	15.07	10.06
	LED_NP30	5.69	10.19	11.33	19.91	12.53
	CONNECT	0.62	0.64	0.67	1.20	0.64
	NURSERY	0.05	0.05	0.05	0.06	0.06
	Average	***3.83***	*5.50*	*6.33*	*11.08*	*6.03*
Numeric	SEA_NP0	2.43	2.37	2.45	4.26	3.42
	SEA_NP10	3.92	4.20	4.43	8.14	4.35
	SEA_NP20	3.92	4.80	4.76	19.41	4.40
	WAVE21	19.70	18.30	18.86	40.76	20.50
	WAVE40	37.74	34.68	34.90	77.05	38.44
	RBFS	9.36	8.61	8.88	22.14	10.12
	RBFC	46.16	40.48	40.69	103.13	49.08
	Average	*17.60*	***16.21***	*16.42*	*39.27*	*18.62*
Mix	RTS	9.94	11.45	9.69	12.81	11.03
	RTC	41.61	44.23	40.08	47.28	47.88
	Average	*25.77*	*27.84*	***24.88***	*30.05*	*29.45*
Overall Average		***12.89***	***13.17***	***13.26***	***26.58***	***14.71***

TABLE VII. OPTIMIZATION PERFORMANCE COMPARISON

Data Name	Avg.Ac	Avg.Ln	Acc.Gain	SizeGain	M_x
VFDT(T0.05)	75.4	300	0.00%	0.00%	-
VFDT(T0.10)	75.28	984	-0.16%	228.00%	-
VFDT(T0.15)	79.41	1239	5.32%	313.00%	58.85
VFDT(T0.20)	79.67	1682	5.66%	460.67%	81.34
VFDT(T0.25)	80.01	1785	6.11%	495.00%	80.96
VFDT(T0.30)	80.17	1928	6.33%	542.67%	85.78
VFDT(T0.35)	79.93	1989	6.01%	563.00%	93.71
VFDT(T0.40)	79.95	2105	6.03%	601.67%	99.70
VFDT(T0.45)	80.09	2132	6.22%	610.67%	98.18
VFDT(T0.50)	80.1	2226	6.23%	642.00%	102.99
VFDT(T0.55)	80.11	2390	6.25%	696.67%	111.53
VFDT(T0.60)	80.11	2474	6.25%	724.67%	116.01
VFDT(T0.65)	80.11	2482	6.25%	727.33%	116.44
VFDT(T0.70 ~ T0.95)	80.11	2487	6.25%	729.00%	116.70
VFDT (GT)	80.11	2487.07	5.34%	720.27%	134.88
HOT (Tie0.05)	81.3	5675.73	***6.91%***	1771.94%	256.43
HOT (Tie0.5)	76.22	533.13	0.23%	***75.84%***	329.74
OVFDT	78.4	727.8	3.10%	140.04%	***45.17***

Figure 10 visualizes the results of performance gain from Table VII. Amongst all the HT building methods in the experiment, the gradient of trend line of OVFDT is the smallest one. This means, the ratio (2) is the minimum (it is 45.17 in OVFDT). In other words, OVFDT can achieve the greatest gain in accuracy given the smallest required tree size.

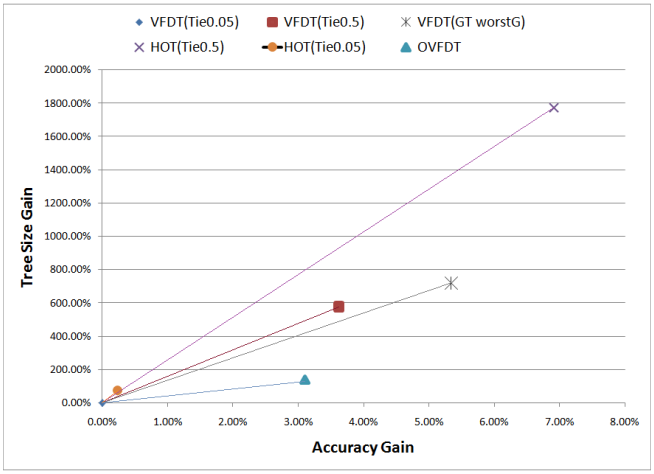

Figure 10. Incremental Optimization Performance.

V. 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 which 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 pre-pruning mechanism controls the attribute-splitting estimation incrementally. It does not require growing a full decision tree in advance and then removes the unnecessary nodes afterward. 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. The experiment results show that OVFDT meet the optimization goal and achieve a better performance gain in terms of high prediction accuracy and compact tree size than the other VFDTs. That is, with the minimum tree size, OVFDT can achieve the highest possible accuracy. This advantage can be technically accomplished by means of simple 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] G.Hulten, L.Spencer, and P.Domingos,"Mining time-changing data streams". Proc.7thACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD01). ACM Press, 2001, pp. 97-106

[2] H.Geoffrey, K.Richard and P.Bernhard, "Tie Breaking in Hoeffding trees". Proc. Workshop W6:Second International Workshop on Knowledge Discovery in Data Streams, 2005, pp.107-116

[3] J.R.K.Hartline, "Incremental Optimization", PhD Thesis, Faculty of the Graduate School, Cornell University, January 2008

[4] A. Gupta, M. P'al, R. Ravi, and A. Sinha. "Boosted sampling: approximation algorithms for stochastic optimization". Proc. 36th Annual ACM Symposium on Theory of Computing, ACM Press, 2004, pp. 417–426

[5] O. H. Ibarra and C. E. Kim. "Fast approximation algorithms for the knapsack and sum of subset problems".J. ACM, 22(4), 1975, pp.463–468

[6] C. G. Plaxton. "Approximation algorithms for hierarchical location problems". Proc. 35th Annual ACM Symposium on Theory of Computing,ACM Press, 2003, pp. 40–49

[7] R. R. Mettu and C. G. Plaxton. "The online median problem". Proc. 41st Annual Symposium on Foundations of Computer Science,IEEE Computer Society, 2000, pp. 339

[8] J. Ross Quinlan, C4.5: programs for machine learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1993

[9] L. Breiman, J. Friedman, R. Olshen, and C Stone. Classification and Regression Trees, Wadsworth,1984

[10] J.P. Bradford, C. Kunz, R. Kohavi, C. Brunk, and C. E. Brodley. Pruning Decision Trees with Misclassification Costs. Proc. 10th European Conference on Machine Learning (ECML '98), Springer-Verlag, London, UK, pp. 131-136, 1998.

[11] G.Hulten,L.Spencer, and PDomingos, Mining time-changing data streams. Proc. 7thACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD01). ACM, pp.97-106 2001

[12] S.Nishimura, M.Terabe,K.Hashimoto, and K.Mihara. Learning Higher Accuracy Decision Trees from Concept Drifting Data Streams. Proc. 21st international Conference on industrial, Engineering and Other Applications of Applied intelligent Systems: vol. 5027. Springer-Verlag, pp.179-188, 2008

[13] Gama, J., Rocha, R., and Medas, P. Accurate decision trees for mining high-speed data streams. In Proceedings of the Ninth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining. KDD '03. ACM, New York, NY, pp.523-528, 2003.

[14] Tao Wang, Zhoujun Li, Xiaohua Hu, Yuejin Yan, and Huowang Chen. A New Decision Tree Classification Method for Mining High-Speed Data Streams Based on Threaded Binary Search Trees. Emerging Technologies in Knowledge Discovery and Data Mining. Springer,pp.256-267, 2009.

[15] Li, C, Yang Zhang, and X Li. "OcVFDT: one-class very fast decision tree for one-class classification of data streams." SensorKDD 09 Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data. ACM, pp.79-86 2009.

B. Pfahringer, G. Holmes, and R. Kirkby."New Options for Hoeffding Trees", Proc. in Australian Conference on Artificial Intelligence, pp.90-99, 2007.