Solving Problems of Imperfect Data Streams by Incremetnal Decision Trees

Hang Yang
Department of Computer and Information Science
University of Macau
Macau SAR, China
henry.yh@gmail.com

***Abstract*—Big data is a popular topic that attracts highly attentions of researchers from all over the world. How to mine valuable information from such huge volumes of data remains an open problem. Although fast development of hardware is capable of handling much larger volume of data than ever before, in the author's opinion, a well-designed algorithm is crucial in solving the problems associated with big data. Data stream mining methodologies propose one-pass algorithms that discover knowledge hidden behind massive and continuously moving data. These provide a good solution for such big data problems, even for potentially infinite volumes of data. In this paper, we investigate these problems and propose an algorithm of incremental decision tree as the solution.**

***Index Terms*—Data stream Mining; Big data; Decision Trees; Classification Algorithms.**

I. Introduction

Big data has become a hot research topic, and how to mine valuable information from such huge volumes of data remains an open problem. Many research institutes worldwide have dedicated themselves to solving this problem. The solutions differ from traditional methods, where learning process must be efficient and incremental.

Processing big data presents a challenge to existing computation platforms and hardware. However, according to Moore's Law, CPU hardware may no longer present a bottleneck in mining big data due to the rapid development of the integrated circuit industry. Then, what is the key point of big data mining?

In author's opinion, a well-designed mining algorithm is crucial in solving the problems associated with massive data. The methodology shall efficiently discover the hidden information behind massive data and then present the real-time findings in a user-friendly way.

One on hand, amongst those methods of data mining, fortunately, the decision tree is a non-linear supervised-learning model, which classifies data into different categories and makes a good prediction for unseen data. The decision model is into a set of if-then-else rules within a tree-like graph. The high-degree comprehension of tree-like model makes it easy to understand the discovered knowledge from massive and big data, for both human and machine. Based on data stream mining, incremental decision tree has become a popular research topic.

On one hand, however, imperfect data problem is a barrier of the mining process. Missing data, either value- or case-based, will increase difficulties to data mining process. Noisy data are usually the culprits when contradicting samples appear. Bias data causes an irregular class distribution that will influence the reliability of evaluating model. On the other hand, decision tree model will face tree size explosion and detrimental accuracy problems when including imperfect data. In the past decade, incremental decision trees algorithms [1,2,3,4] apply the Hoeffding bound with a tie-breaking threshold, for dealing with the problem of tree-size explosion. This threshold is a fixed user-defined value. We do not know what the best configuration is unless all possibilities have been tried, but undesirable in practical. Although the pre-processing technique is to handle these imperfections, it may not be possible because of the nature of incremental access to the constantly incoming data streams. In addition, concept-drift problem is a characteristic of time-changing data, referring to that the most types of an attribute remain the same while only particular type changes with time. This problem will reduce the utility of a decision model that increases the difficulties of data mining.

II. Imperfect Data Streams

A. Nosiy Data

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

Noise data is considered a type of irrelevant or meaningless data that does not typically reflect the main trends but makes the identification of these trends more difficult. Non-informative variables may be potentially

doi:10.4304/jetwi.5.3.322-331

random noise in the data stream. It is an idealized but useful model, in which such noise variables present no information-bearing pattern of regular variation. Tree size explosion problem, not only exists in incremental trees [1,2], but also in traditional trees [5,6,7]. However, data stream mining cannot eliminate those non-informative candidates in preprocessing before starting classification mining, because the concept-drift problem may also bring non-informative variables into informative candidates.

On one hand, a previous study [8] reenacts this phenomenon that the inclusion of noise data reduces the accuracy and increasing model size. This consequence is undesirable in the decision tree classification. There has been an attempt to reduce the effect of noise by using supplementary classifiers for predicting missing values in real-time and minimizing noise in the important attributes [9]. Such methods still demand extra resources in computation.

B. Missing Data

It is known that a major cause of over-fitting in a decision tree is the inclusion of contradicting samples in the learning process. Noisy data and missing values are usually the culprits when contradicting samples appear. Unfortunately, such samples are inevitable in distributed communication environments such as wireless sensor network (WSN). Two measures are commonly employed to define the extent of values missing from a set of data [10]: the percentage of predictor values missing from the dataset (the value-wise missing rate) and the percentage of observation records that contain missing values (the case-wise missing rate). A single value missing from the data usually indicates transmission loss or malfunctioning of a single sensor. A missing data value record may result from a broken link between sensors. In WSN, can distinguish the missing data to two categories:

- Incomplete data with lost values: Because of an accidence of sensor itself, like a crash or a reboot, the instant data of the last event before the accidence will be lost. Hence, such kind of missing values is permanent, which is lost forever.
- Unstable data with late arrival: Because of temporal disconnection or network delay, data stream capture faces asynchronous issues. The missing value caused by asynchronous is not permanent, which is temporally lost and will arrive in a short while.

C. Bias Data

Bias data is also called imbalanced distribution data. The term "imbalanced" 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. Imbalanced classification is a common problem. This problem occurs when the classifier algorithm is trained with a dataset, in which one class has only a few samples, and there are a disproportionally large number of samples in the other classes. Imbalanced data causes classifiers to be over-fitted (i.e., produce redundant rules that describe duplicate or meaningless concepts), and, as a result, perform poorly, particularly in the identification of the minority class.

Most of the standard classification algorithms assume that training examples are evenly distributed among different classes. In practical applications where this was known to be untrue, researchers addressed the problem by either manipulating the training data or adjusting the misclassification costs. Resizing training data sets is a common strategy that attempts to downsize the majority class and over-samples the minority class. Many variants of this strategy have been proposed [10,11,12]. A second strategy is to adjust the costs of misclassification errors to be biased against or in favor of the majority and minority classes, respectively. Using the feedback from the altered error information, researchers then fine-tune their cost-sensitive classifiers and post-prune the decision trees in the hope of establishing a balanced treatment of each class in the new imbalanced data collected by the network [12,13]. However, they are not suitable for data stream mining because of the nature of incremental access to the constantly incoming streams.

D. Concept-drift Data

Data stream is also an infinite big data scenario that the underlying data distribution of newly arrival data may be appeared differently from the old one in the real world, so called concept-drift problem. For example, click-streams of user's navigating e-commerce website may reflect the preferences of purchase through the analysis systems. When people's preferences of product change, however, the old user's behavior model is not applicable any more that the drifting of concepts appears.

The hidden changes in the attributes of data streams will cause a drift of target concept. In terms of the occurring frequency, commonly it can be distinguished in two kinds: abrupt drift and gradual drift. For data streams, the data arrive continuously that the concept-drift is local, for instance, only particular types of attribute may change with time while the others remain the same.

III. INCREMENTAL DECISION TREE ALGORITHMS

A. Decision Tree Learning using Hoeffding Bound

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 I attributes and y is a discrete class label. I is the number of attributes in X. k is the index of class label. Suppose a class label with the k^{th} discrete value is y_k. Attribute X_i is the i^{th} attribute in X, and is assigned a value of x_{i1}, x_{i2}… x_{iJ}, where $1 \le i \le I$ and J is the number of different values of 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 stream mining, the example size is very large or unlimited that $N \rightarrow \infty$.

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 (HB) that $HB = \sqrt{R^2 \ln\left(\frac{1}{\delta}\right)/2n}$, where R is the range of classes distribution and n is the number of instances which have fallen into a leaf. Sufficient

statistics is used to record the counts of each value x_{ij} of attribute X_i belonging to class y_k. The solution, which doesn't requiring the full historical data, is a node-splitting criterion using a HB. To evaluate a splitting-value for attribute X_i, it chooses the best two values. Suppose x_{ia} is the best value of $H(.)$ where $x_{ia} = \arg\max H(x_{ij})$; suppose x_{ib} is the second best value where $x_{ib} = \arg\max H(x_{ij}), \forall j \neq a$; suppose $\Delta H(X_i)$ is the difference of the best two values for attribute X_i, where $\Delta H(X_i) = \Delta H(x_{ia}) - \Delta H(x_{ib})$. Let n be the observed number of instances, HB is used to compute high confidence intervals for the true mean r_{true} of attribute x_{ij} to class y_k that $r - HB \leq r_{true} < r + HB$ where $r = (1/n)\sum_i^n r_i$. If after observing n_{min} examples, the inequality $r + HB < 1$ holds, then $r_{true} < 1$, meaning that the best attribute x_{ia} observed over a portion of the stream is truly the best attribute over entire stream. Hence, a splitting-value x_{ij} of attribute X_i can be found without full attribute-values, even when we don't know all values of X_i (from x_{i1} to x_{iJ}).

When data contains imperfect values, it may confuse the values of heuristic function. The difference of the best two heuristic evaluation for attribute X_i, where $\Delta\bar{H}(X_i) = H(x_{ia}) - H(x_{ib})$, may be negligible. To solve this problem, a fixed tie-breaking τ, which is a user pre-defined threshold for incremental learning decision tree, is proposed as a pre-pruning mechanism to control the tree growth speed [2]. This threshold constrains the node-splitting condition that $\Delta\bar{H}(X_i) \leq HB < \tau$. 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 no a unique τ suitable for all problems. In other words, there is not a single value that works well in all tasks. The choice of τ hence depends on the data and their nature.

B. Evoluation in the Past Decade

According to node-splitting process of a decision tree, we can distinguish it into two categories: singletree algorithm and multi-tree algorithm. Singletree is a decision model that only builds one tree in the tree-building approach while does not require any optional branches or alternative trees. Multi-tree builds a decision tree model dependent on many other trees at the same time. The advantage of singletree is lightweight favored for data streams environment and easy to implement, although in some cases, multi-tree may bring a higher accuracy.

VFDT is the pioneer singletree of using HB to construct incremental decision tree for high-speed data streams, but it can't handle concept drift. Functional tree leaf is originally proposed to integrate to incremental decision tree [3]. Consequently, 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. OcVFDT [14] provides a solution to deal with unlabeled samples based on VFDT and POSC4.5. The experiment shows four fifths of samples are unlabeled, while the performance still gets close to VFDT of fully labeled streams. OcVFDT is a one-class classification that classifiers are trained to distinguish only a class of objects from all other objects. FlexDT [15] proposes a Sigmoid function to handle noisy data and missing values. Sigmoid function is used to decide what true node-splitting value, but sacrificing algorithm speed.

For handling concept-drift problem, CVFDT [2] proposed a fixed size of sliding-window that integrated to VFDT. It constructs an alternative tree in the tree growing. When tree model is out-of-date within a window, the alternative branch will replace the old one so that it adapts to concept-drift data. HOT [16] 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. ASHT [4] 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.

IV. Hypothesis and Motivation

A. Hypothesis

The research is on the basic of the following assumptions:

One-pass Process The feature of proposed method implement as a one-pass approach, which requires loading and computing the data records only one time. Therefore, this is potentially applicable for big data, even unbounded data problem.

Data Volume The data is multi-dimensional, with bounded and constant values of attributes. The data is also labeled. A data record is called the instance. The data has a large scale of instances, even infinite. The algorithm builds an incremental decision tree, in which suppose there enough instances for the node splitting using the HB.

Imperfect Data The imperfect data include: the noisy data, the data with missing values, the data with imbalanced class distribution, as well as the data with concept-drift.

Performance Measures Accuracy is the number of correctly classified instances divided by the number of total instances. Tree size is the number of the rules in a decision tree. This also equals to the number of leaves in the tree model. Learning speed is the time to construct the decision tree. It is an immediate time in the incremental learning process. Memory cost is the memory size used to build the tree model.

Classifier Due to the one-pass process, the incremental decision tree implements a test-then-train process. When a new instance arrives, it will traverse from the root to a leaf according to the tree model. This is also a testing process. During the traversing, the node splitting is triggered so that tree model is trained incrementally. Besides, the post-pruning mechanism is infeasible since the nature of fast-moving data scenario. No extra time is allowed to stop tree building and prune tree structure.

Application The result of the proposed methodology is a decision tree model, which presents rules from the root to the leaves. The tree-like structure shows a collection of complex rules intuitively in terms of IF-THEN-ELSE rules. Both human and machine can understand this rules easily.

B. Motivation

In this paper, we propose an incremental decision tree learning method that is suitable for big data analysis. What is the difference between traditional decision tree learning and incremental decision tree learning?

In Figure 1, we provide an example of traditional decision tree learning. The criteria of splitting-node selection is based on heuristic function. For example, ID3 algorithm uses the information entropy while C4.5 applies the information gain as the heuristic function. In general, the traditional tree learning requires loading the full data and analyzes the whole training data to build a decision tree. The splitting criteria is according to the heuristic result, splitting from the attribute with the larger heuristic value, until all candidates become internal nodes.

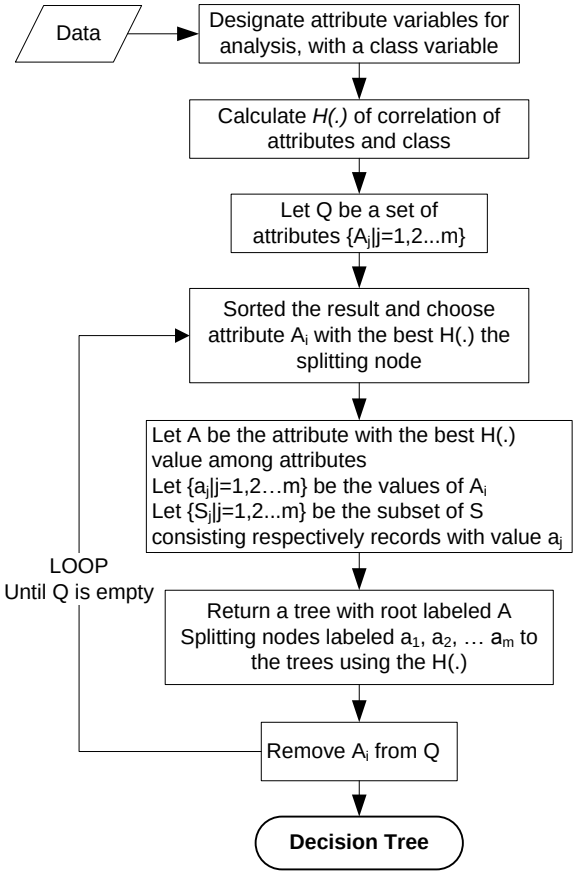

Figure 1. Workflow of Buiding A Traditional Decision Tree.

Differently in Figure 2, incremental learning process using Hoeffding bound in the splitting criteria. It does not require loading the full data, instead, it only needs a part of data to train decision tree model. When new data arrives, the sufficient statistics are updated. If checking condition satisfied, it will compare splitting candidates with the best and the second best heuristic result. In this case, the tree model is updated incrementally, with newly arrival data.

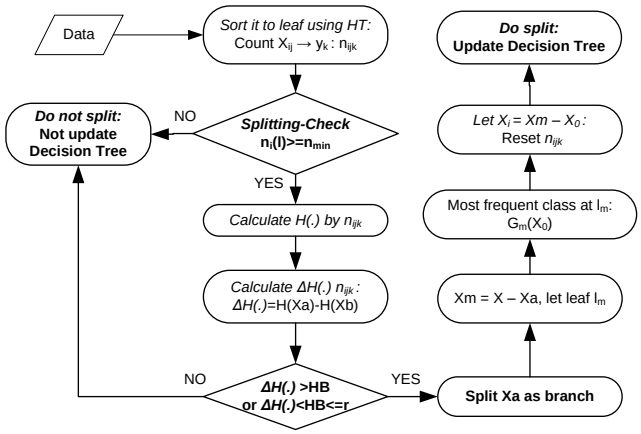

Figure 2. Workflow of Buiding An Incremental Decision Tree.

From the comparison above, obviously, the traditional method is not suitable for big data scenario, because loading full data is inapplicable in practical. That is why we propose an incremental method to deal with big data. The incremental process is applicable for continuously arrival data, even infinite data scenario.

V. METHODOLOGY DESIGN

A. Overall Workflow

The proposed methodology, which inherits the use of HB, implements on a test-then-train approach (Figure 3) 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 model updating) of the decision tree then occurs incrementally.

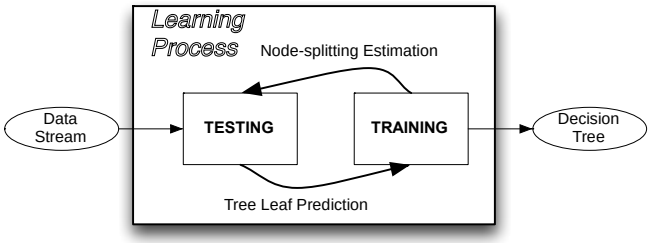

Figure 3. Test-then-train Workflow.

B. Auxiliary Reconciliation Control

The Auxiliary Reconciliation Control (ARC) is a set of data pre-processing functions used to solve the problem of missing data streams. The ARC can be programmed as a standalone program that may run in parallel and in synchronization with the test-and-train operation. Synchronization is facilitated by using a sliding window that allows one segment of data to arrive at a time at regular intervals. When no data arrive, the ARC simply

stands still without any action. The operational rate of the sliding window should be no greater than the speed at which the decision tree building is operated and faster than the speed at which the sensors transmit data.

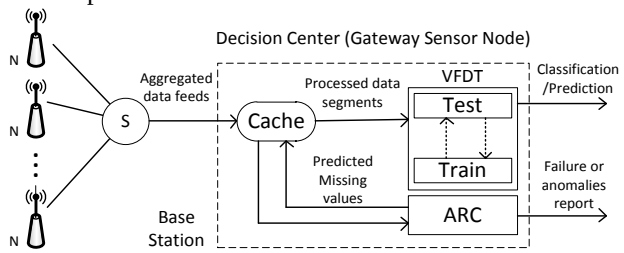

Figure 4. The workflow of ARC in a gateway sensor node.

To tackle the problem of missing values in a data stream, a number of prediction algorithms are commonly used to guess approximate values based on past data. Although many algorithms can be used in the ARC, that deployed should ideally achieve the highest level of accuracy while consuming the least computational resources and time. Some popular choices we use here for simulation experiments include, but are not limited to, mean, naïve Bayesian, and C4.5 decision tree algorithms for nominal data, and mean mode, linear regression, discretized naïve Bayesian and M5P algorithms for numeric data. Missing value estimation algorithms require a substantial amount of past data to function. For example, before using a C4.5 decision tree algorithm as a predictor for missing values, a classifier must be built using statistics from a sample of sufficient size.

C. Functional Tree Leaf

Functional tree leaf [3], can further enhance the prediction accuracy via the embedded Naïve Bayes classifier.. In this paper, we embed the functional tree leaf to improve the performance of prediction by *HT* model. When these two extensions – an optimized node-splitting condition ($\Delta\bar{H} > HB$ or $\mathrm{Opt}.\Phi(HT_x{}^*) > Max.\Phi(HT_x)$ or $\mathrm{Opt}.\Phi(HT_x{}^*) < Min.\Phi(HT_x)$) and a refined prediction using the functional tree leaf – are used together, the new decision tree model is able to achieve unprecedentedly good performance, although the data streams are perturbed by noise and imbalanced class distribution.

For the actual classification, iOVFDT uses a decision tree model HT_F to predict the class label $\widehat{y_k}$ with functional tree leaf F when a new sample (*X*, *y*) arrives, defined as $HT_F(X) \rightarrow \widehat{y_k}$. The predictions are made according to the observed class distribution (OCD) in the leaves called functional tree leaf *F*. Originally in VFDT, the prediction uses only the majority class F^{MC}. The majority class only considers the counts of the class distribution, but not the decisions based on attribute combinations. The naïve Bayes F^{NB} computes the conditional probabilities of the attribute-values given a class at the tree leaves by naïve Bayes network. As a result, the prediction at the leaf is refined by the consideration of each attribute's probabilities. To handle the imbalanced class distribution in a data stream, a weighted naïve Bayes F^{WNB} and an error-adaptive $F^{Adaptive}$ are proposed in this paper. These four types of functional tree leaves are discussed in following paragraphs.

Let Sufficient statistics n_{ijk} be an incremental count number stored in each node in the iOVFDT. Suppose that a node $Node_{ij}$ in *HT* is an internal node labeled with attribute x_{ij} and *k* is the number of classes distributed in the training data, where $k \geq 2$. A vector V_{ij} can be constructed from the sufficient statistics n_{ijk} in $Node_{ij}$, such that $V_{ij} = \{n_{ij1}, n_{ij\,2} \ldots n_{ij\,k}\}$. V_{ij} is the OCD vector of $Node_{ij}$. OCD is used to store the distributed class count at each tree node in iOVFDT to keep track of the occurrences of the instances of each attribute.

***Majority Class Functional Tree Leaf*:** In the OCD vector, the majority class F^{MC} chooses the class with the maximum distribution as the predictive class in a leaf, where F^{MC}: arg max $r = \{n_{i,j,1}, n_{i,j,2} \ldots n_{i,j,r} \ldots n_{i,j,k}\}$, and where $0<r<k$.

***Naïve Bayes Functional Tree Leaf*:** In the OCD vector $V_{i,j} = \{n_{i,j,1}, n_{i,j,2} \ldots n_{i,j,r} \ldots n_{i,j,k}\}$, where *r* is the number of observed classes and $0<r<k$, the naïve Bayes F^{NB} chooses the class with the maximum possibility, as computed by the naïve Bayes, as the predictive class in a leaf. $n_{ij,r}$ is updated to $n'_{i,j,r}$ by the naïve Bayes function such that $n'_{i,j,r} = \mathrm{P}(X|C_f) \cdot \mathrm{P}(C_f) \,/\, \mathrm{P}(X)$, where *X* is the new arrival instance. Hence, the prediction class is F^{NB}: arg max $r = \{ n'_{i,j,1}, n'_{i,j,2} \ldots n'_{i,j,r} \ldots n'_{i,j,k} \}$.

***Weighted Naïve Bayes Functional Tree Leaf*:** In the OCD vector $V_{i,j} = \{n_{i,j,1}, n_{i,j,2} \ldots n_{i,j,r} \ldots n_{i,j,k}\}$, where *k* is the number of observed classes and $0<r<k$, the weighted naïve Bayes F^{WNB} chooses the class with the maximum possibility, as computed by the weighted naïve Bayes, as the predictive class in a leaf. $n_{i,j,r}$ is updated to $n'_{i,j,r}$ by the weighted naïve Bayes function such that $n'_{i,j,r} = \omega_r \cdot \mathrm{P}(X|C_f) \cdot \mathrm{P}(C_f) \,/\, \mathrm{P}(X)$, where *X* is the latest received instance and the weight is the probability of class *i* distribution among all the observed samples, such that $\omega_r = \prod_{r=1}^{k} (v_r / \sum_{r=1}^{k} v_r)$, where $n_{i,j,r}$ is the count of class *r*. Hence, the prediction class is F^{WNB}: arg max $r = \{ n'_{i,j,1}, n'_{i,j,2} \ldots n'_{i,j,r} \ldots n'_{i,j,k} \}$.

***Adaptive Functional Tree Leaf*:** In a leaf, suppose that $V_F{}^{MC}$ is the OCD with the majority class F^{MC}; suppose $V_F{}^{NB}$ is the OCD with the naïve Bayes F^{NB} and suppose that $V_F{}^{WNB}$ is the OCD with the weighted naïve Bayes F^{WNB}. Suppose that *y* is the true class of a new instance *X* and E_F is the prediction error rate using a *F*. E_F is calculated by the average $E=error_i/n$, where *n* is the number of examples and $error_i$ is the number of examples mis-predicted using *F*. The adaptive Functional Tree Leaf chooses the class with the minimum error rate predicted by the other three strategies, where $F^{Adaptive}$: arg min $F = \{E_F{}^{MC}, E_F{}^{NB}, E_F{}^{WNB}\}$.

D. Incremental Optimization

The model is growing incrementally so as to update an optimal decision tree under continuously arriving data. Suppose that a decision tree optimization problem Π is defined as a tuple (X, HT, Φ). The set *X* is a collection of objects to be optimized and the feasible Hoeffding tree

HT solutions are subsets of *X* that collectively achieve a certain optimization goal. The set of all feasible solutions is $HT \subseteq 2^X$ and $\Phi: HT \to \mathbb{R}$ is a cost function of these solutions. The optimal decision tree *HT** exists if *X* and Φ are known, and the subset S is the set of solutions meets the objective function where *HT** is the optimum in this set. Therefore, the incremental optimization functions can be expressed as a sum of several sub-objective cost functions: $\Phi(HT_x) = \bigcup_{D=1}^{M} \Phi_D(HT_x)$, where $\Phi_m : HT \to \mathbb{R}$ is a continuously differentiable function and *M* is the number of objects in the optimization problem. The optimization goal: $minimize\ \Phi(HT_x)\ subject\ to\ HT_x \in X$. $HT(X) \to \hat{y}$ is used to predict the class when a new data sample (*X*, *y*) arrives. So far timestamp *t*, the prediction accuracy defined as: $accu_t = \frac{\sum_{i=1}^{t} Predict(D_i)}{|D_t|}, Predict(D_i) = \begin{cases} 1, if\ \widehat{y_k} = y_k \\ 0, if\ \widehat{y_k} \neq y_k \end{cases}$.

To measure the utility of the three dimensions via the minimizing function, the measure of prediction accuracy is reflected by the prediction error in: $\Phi_1 = 1 - accu_t$.

The new methodology is building a desirable tree model by combining with an incremental optimization mechanism and seeking a compact tree model that balances the objects of tree size, prediction accuracy and learning time. The proposed method finds an optimization function $\Phi(HT_x)$, where $M = 3$. When a new data arrive, it will be sorted from the root to a leaf in terms of the existing *HT* model. When a leaf is being generated, the tree size grows. A new leaf is created when the tree model grows incrementally in terms of newly arrival data. Therefore, up to timestamp *t* the tree size is:

$$\Phi_2 = \begin{cases} size_{t-1} + 1 & , if\ \Delta\bar{H} > HB \\ size_{t-1} & , otherwise \end{cases}.$$

It is a one-pass algorithm that builds a decision model using a single scan over the training data. The *sufficient statistics* that count the number of examples passed to an internal node are the only updated elements in the one-pass algorithm. The calculation is an incremental process, which tree size is "plus-one" a new splitting-attribute appears. It consumes little computational resources. Hence, the computation speed of this "plus one" operation for a new example passing is supposed as a constant value *R* in the learning process. The number of examples that have passed within an interval period of in node splitting control determines the learning time that $\Phi_3 = R \times (n_{y_k} - n_{min})$. n_{min} is a fixed value for controlling interval of node splitting.

Suppose that n_{y_k} is the number of examples seen at a leaf y_k and the condition that checks node-splitting is $n_{y_k} mod\ n_{min} = 0$. The learning time of each node splitting is the interval period – the time defined as Φ_3 – during which a certain number of examples have passed up to timestamp *t*.

Returning to the incremental optimization problem, the optimum tree model is the HT_x structure with the minimum $\phi(x)$. A triangle model is provided to illustrate the relationship amongst the three dimensions – the prediction accuracy, the tree size and the learning time. The three dimensions construct a triangle utility function in Figure 5. A utility function computes the area of triangle, reflecting a relationship amongst the three objects in:

$$\Phi(HT_x) = \frac{\sqrt{3}}{4} \cdot (\Phi_1 \times \Phi_2 + \Phi_1 \times \Phi_3 + \Phi_2 \times \Phi_3)$$

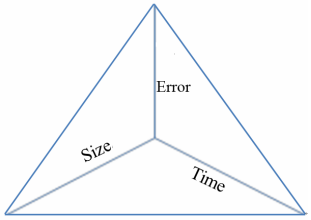

Figure 5. Three-obective Optimization.

The area of this triangle $\Phi(HT_x)$ changes when node splitting happens and the *HT* updates. A min-max constraint of the optimization goal in (4) controls the node splitting, which ensures that the new tree model keeps a $\Phi(HT_x)$ within a considerable range. Suppose that $Max.\Phi(HT_x)$ is a *HT* model with the maximum utility so far and $Min.\Phi(HT_x)$ is a *HT* model with the minimum utility. The optimum model should be within this min-max range, near $Mean.\Phi(HT_x)$:

$$Mean.\Phi(HT_x) = \frac{\text{Max.}\Phi(HT_x) - Min.\Phi(HT_x)}{2}$$

According to the Chernoff bound, we know:

$$|\text{Opt.}\Phi(HT_x{}^*) - Mean.\Phi(HT_x)| \leq \sqrt{\frac{\ln(1/\delta)}{2n}}$$

where the range of $\Phi_x(HT_x)$ is within the min-max model $\text{Min.}\Phi(HT_x) < \text{Opt.}\Phi(HT_x{}^*) < \text{Max.}\Phi(HT_x)$. Therefore, if $\Phi(HT_x)$ goes beyond this constraint, the existing *HT* is not suitable to embrace the new data input and the tree model should not be updated. Node-splitting condition is:

$$\Delta\bar{H} > HB,$$
$$\text{or Opt.}\Phi(HT_x{}^*) > Max.\Phi(HT_x),$$
$$\text{or Opt.}\Phi(HT_x{}^*) < Min.\Phi(HT_x).$$

,

VI. Evaluation

A. Synthetic Data Streams

Hyper-plane data is another typical data streams for concept-drift study [4,17]. We use MOA hyper-plane data generator to simulate the data streams without noise-included (10 attributes and 2 classes, 2 of 10 attributes are randomly drifting). The performance measurement is Interval Test-then-train Evaluation in MOA. The aforementioned contents have verified that Error-adaptive is the best strategy of functional tree leaf, hence, it is applied in this test.

The synthetic streams are marked when attributes drifting. A piece of streams is visualized (50 instances included) in Figure 6. Similar result appears that iOVFDT

outperforms the other two algorithms. In addition, it is obvious that: when a drift occurs, the accuracy is declining consequently. This test shows iOVFDT has a good performance dealing with attributes drifting.

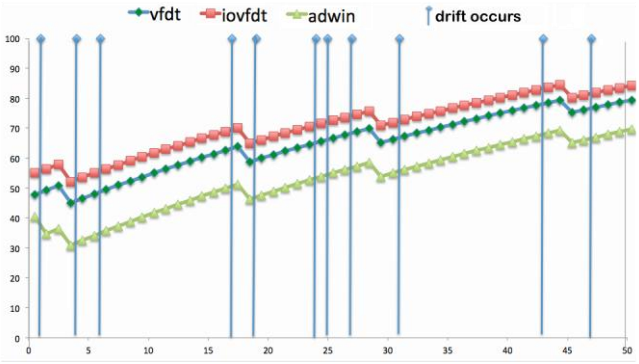

Figure 6. Concept-drift evaluation for hyper-plane data streams.

B. Sensor Data with Missing Values

The complex nature of incomplete and infinite streaming data in WSNs has escalated the challenges faced in data mining applications concerning knowledge induction and time-critical decision-making. Traditional data mining models employed in WSNs work, which are mainly on the basis of relatively structured and stationary historical data, and may have to be updated periodically in batch mode. The retraining process consumes time as it requires repeated archiving and scanning of the whole database. Data stream mining is a process that can be undertaken at the front line in a manner that embraces incoming data streams.

To the best of the author's knowledge, no prior study has investigated the impact of imperfect data streams or solutions related to data stream mining in WSNs, although the pre-processing of missing values is a well-known step in the traditional knowledge discovery process. We propose a holistic model for handling imperfect data streams based on four features that riddle data transmitted among WSNs: missing values, noise, delayed data arrival and data fluctuations. The model has a missing value predicting mechanism called the auxiliary reconciliation control (ARC). A bucket concept is also proposed to smooth traffic fluctuations and minimize the impact caused by late arriving data. Together with the VFDT, the ARC-cache facilitates data stream mining in the presence of noise and missing values. To prove the efficacy of the new model, a simulation prototype is implemented based on ARC-cache and VFDT theories by using a JAVA platform. Experimental results unanimously indicate that the ARC-cache and VFDT method yield better accuracy in mining data streams in the presence of missing values than VFDT only. One reason for this improved performance is ascribed to the improved predictive power of the ARC in comparison with other statistical counting methods for handling missing values, as the ARC computes the information gains of almost all other attributes with non-missing data. In future research, we will continue to investigate the impact of noisy or corrupted data and irregular data stream patterns on data stream mining.

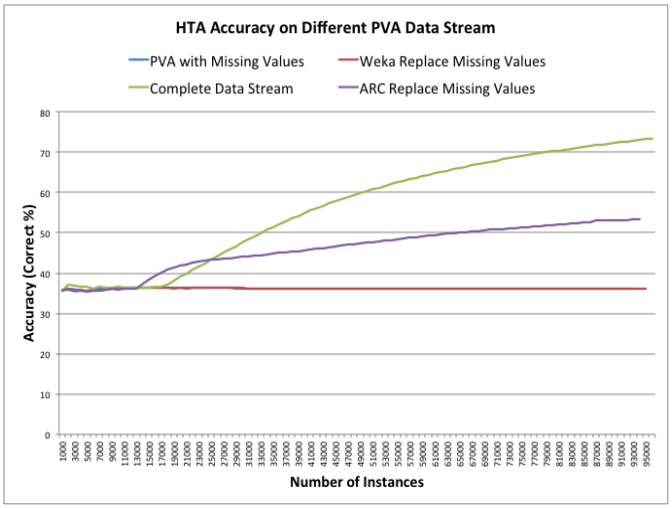

Figure 7. Performance of ARC-cache missing values replacement

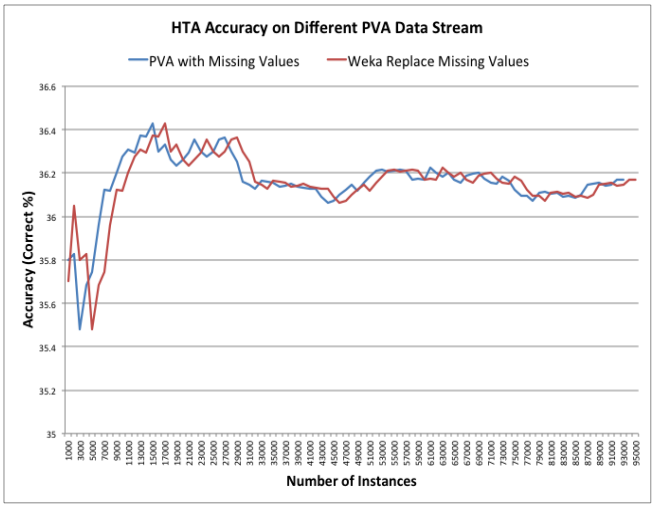

Figure 8. Magnified version of the diagram

In this part, we use a set of real-world data streams downloaded from the 1998 KDD Cup competition provided by Paralyzed Veterans of America (KDD Cup, 1998). The data comprise information concerning human localities and activities measured by monitoring sensors attached to patients. We use the learning dataset (127MB in size) with 481 attributes originally in both numeric and nominal form. Of the total number of 95,412 instances, more than 70% contain missing values.

In common with the previous experiment, we compare the ARC-Cache and VFDT method with the standard missing values replacement method found in WEKA using means. The results of the comparison are shown in Figure 7 and 8. Considering the number of attributes is very large, we apply a moderate window size (W = 100) for the ARC to operate. A complete dataset given by PVA is used to test the ARC-Cache (115MB). The experiment results demonstrate that using WEKA mean values to replace missing data yields the worst level of VFDT classification accuracy. Although using the ARC-Cache to deal with missing values in the dataset

does not yield results as accurate as the complete dataset without any missing values, ARC-Cache performance is much better than that achieved using WEKA means to replace missing values. The enlarged chart shows the WEKA replacement approach has very little effect in maintaining the level of performance because of the very high percentage of missing data (70%) in this extreme example.

C. UCI Data Streams

Dynamic data dominate many modern computer applications nowadays. They are characterized to be vast in size, fast moving in speed and consist of many attributes, which do not make sense individually, but they describe some behavioral patterns when analyzed together over some time. Traditional data mining algorithms are designed to load a full archive of data, and then build a decision model. New data that arrive would have to be accumulated with the historical dataset, and together they would be scanned again for rebuilding an up-to-date decision tree.

TABLE I.

DESCRIPTION OF DYNAMIC DATA

Name	Abbr.	#Ins	#Attr	#Cls	#Nom	#Num
iPod Sales	IP	7882	29	3	16	12
Internet Usage	IU	10104	72	5	71	0
Network Attack	NA	494021	42	23	3	38
Cover Type	CT	581012	55	7	42	12
Robot Sensor	RS	5456	25	4	0	24
Person Activity	PA	164860	6	11	2	3

Six scenarios of dynamic data are tested in the experiment, shown in Table 1. Each type of dynamic data represents typical decision-making problems on the topics of web applications, real-time security surveillance and activities monitoring. The data of web applications are Internet Usage (IU) data and iPod Sales on eBay (IP) data, which are generated from the recording of user's click-streams on the websites. The data of real-time security surveillance are Network Attack (NA) and Cover Type (CT) data. The data of activities monitoring are Robot Sensor (RS) and Person Activity (PA) data. The datasets are extracted from real-world applications that are available for download from UCI Machine Learning Repository.

TABLE II.

ACCURACY ANAYLSIS OF DYNAMIC DATA

Method\Data	RS	IP	IU	CT	PA	NA
C4.5 Pruned	99.45	99.82	82.34	91.01	75.20	99.94
C4.5 Unpruned	99.65	99.70	81.50	92.77	74.10	99.95
Incre.NB	~~55.35~~	~~89.90~~	~~75.29~~	~~60.52~~	~~49.28~~	~~96.55~~
VFDT	~~40.21~~	~~90.7~~	~~79.0~~	~~67.45~~	~~43.75~~	~~98.2~~
		~~1~~	~~6~~			~~7~~
VFDT_NB	~~55.35~~	99.07	82.03	~~77.16~~	~~61.03~~	99.68
VFDT_ADP	~~55.35~~	99.21	82.31	~~77.77~~	~~61.01~~	99.79
iOVFDT_MC	~~71.92~~	~~81.79~~	~~78.24~~	~~70.52~~	~~59.15~~	~~99.23~~
iOVFDT_NB	81.60	98.78	~~78.65~~	90.66	73.45	99.69
iOVFDT_WNB	81.91	98.13	~~78.95~~	90.51	72.35	99.69
iOVFDT_ADP	83.32	98.92	79.84	90.59	73.52	99.85
Standard Deviation.	20.21	6.09	2.31	11.80	11.28	1.08
Variance	408.54	37.14	5.31	139.17	127.13	1.16
Average	72.41	95.61	79.71	80.90	64.28	99.26

From Table 2, in general, it is observed that C4.5 had better accuracy than the other methods in all tested datasets because it built its decision model from the full dataset. Therefore it can attain a globally best solution by going through all the training data at one time. The other methods are incremental learning process that obtained a locally optimum solution in each pass of data stream. The strikethroughs indicate those accuracies that are below the average. Obviously, one can see that only C4.5 and iOVFDT_ADP (iOVFDT with Error-adaptive functional tree leaf) are able to achieve a 'full win' of satisfactory accuracies over the average across all the datasets. Fig. 8.1 shows a graphical representation of the accuracies in the form of a stacked bar chart – despite C4.5, the iOVFDT family of algorithms (except MC) obtains pretty good accuracies. Therefore, when batch learning such as C4.5 is not feasible or available in scenarios of dynamic data stream mining, iOVFDT_ADP would be a good candidate.

Table 3 shows the model size (the number of nodes / the number of leaves) which is calculated as the number of leaves over the number of nodes for different datasets. For all dataset, C4.5 built the decision model requiring largest tree size. Naïve Bayes does its prediction by using distribution probabilities, so that the decision model does not exhibit a tree-like structure. Although smaller tie-breaking threshold might bring respectively smaller tree size for VFDT, the accuracy is obviously worse than iOVFDT. It is interesting to see that the size of a globally best model (C4.5) is not much bigger than a locally optimum model (iOVFDT) because the latter algorithm allows tree to grow incrementally over time.

TABLE III.

MODEL SIZE ANAYLSIS OF DYNAMIC DATA

	RS	IP	IU	CT	PA	NA
C4.5 Pruned	18/35	20/39	847 /911	10149 /20297	13265 /24120	724 /838
C4.5 Unpruned	22/43	24/46	1028 /1267	14903 /29805	6467 /10357	679 /801

IncreNB	N/A	N/A	N/A	N/A	N/A	N/A
VFDT	1/1	5/9	46/47	127/253	167/187	87/94
VFDT_NB	1/1	5/9	46/47	127/253	167/187	87/94
VFDT_ADP	1/1	5/9	46/47	127/253	167/187	87/94
iOVFDT_MC	22/43	6/11	325/329	1280/2559	2211/2500	185/249
iOVFDT_NB	22/43	8/15	325/329	1864/3727	2440/2821	188/255
iOVFDT_WNB	22/43	8/15	325/329	1864/3727	2233/2551	188/255
iOVFDT_ADP	22/43	8/15	325/329	1864/3727	2440/2821	188/255

The speed of learning decision model was reflected by the time in seconds as shown in Table 4. In general, C4.5 has the slowest learning speed for all datasets. Comparing the average learning times of VFDT to iOVFDT, our experiment result shows both algorithms have a very similar learning speed. iOVFDT has a learning speed almost as fast as the original VFDT. This implies that the improved version, iOVFDT can achieve smaller tree size, good accuracy without incurring cost of slowing down the learning speed. Fast learning speed is important and applicable to time-critical applications.

TABLE IV.

LEARNING SPEED ANAYLSIS OF DYNAMIC DATA

Methods\Data	RS	IP	IU	CT	PA	NA
C4.5 Pruned	0.30	0.22	0.40	931.34	180.88	120.68
C4.5 Unpruned	0.80	0.40	0.39	1717.35	121.62	187.44
IncreNB	0.26	0.10	0.24	11.98	0.95	17.77
VFDT	0.18	0.07	0.19	6.65	0.63	4.50
VFDT_NB	0.13	0.09	0.24	9.88	0.96	6.64
VFDT_ADP	0.14	0.09	0.30	10.18	1.28	7.95
iOVFDT_MC	0.12	0.08	0.20	6.86	0.64	4.36
iOVFDT_NB	0.17	0.09	0.30	8.80	0.97	6.62
iOVFDT_WNB	0.16	0.10	0.29	8.61	0.98	6.41
iOVFDT_ADP	0.13	0.11	0.31	13.09	1.26	6.78
Avg. C4.5	0.55	0.31	0.40	1324.35	151.25	154.06
Avg. Increm.NB	0.26	0.10	0.24	11.98	0.95	17.77
Avg. VFDT	0.15	0.08	0.24	9.57	0.96	6.36
Avg. iOVFDT	0.14	0.10	0.27	8.34	0.96	6.04

D. Real-time Recommendation Data

Recommendation system is an important application of data mining that tries to refer the right products to the right customers in the right time. We use some real-life online recommendation data from the GroupLens Research:

MovieLens www.grouplens.org/node/73

Book-cross www.informatik.uni-freiburg.de/~cziegler/BX/

They are the typical dataset for the recommending system. This data is consisted of three files: movie/book information, user information, and rating. The three files are joined together by the user ID and movie/book ID.

After combining the data, MoiveLens includes 1,000,209 instances, 1 numeric attributes, 24 nominal attribute. The target class is the type of movie. There are 18 distinct types. Book-crossing includes 1,316,100 instances, 2 numeric and 5 nominal attributes. The target class is the country where the users are. There are 61 investigated countries. For a recommendation system, the classification model is used to predict what type of the movie does the user like, or which region does the user live in, from the previous rating data. The benchmark algorithms are VFDT, ADWIN and iOVFDT, with Error-adaptive functional tree leaf.

For MovieLens data, after normalized the result, we can see the comparison of these three algorithms in Figure 9. In general, iOVFDT and ADWIN have better accuracy than VFDT, but ADWIN results bigger model size than iOVFDT, as well as the learning time. For Book-crossing data, the accuracy and tree size analysis are shown in Figure 10 and 11 respectively. It reflects that ADWIN still obtains a bigger tree size. iOVFDT outperforms the others in terms of the accuracy and the tree size.

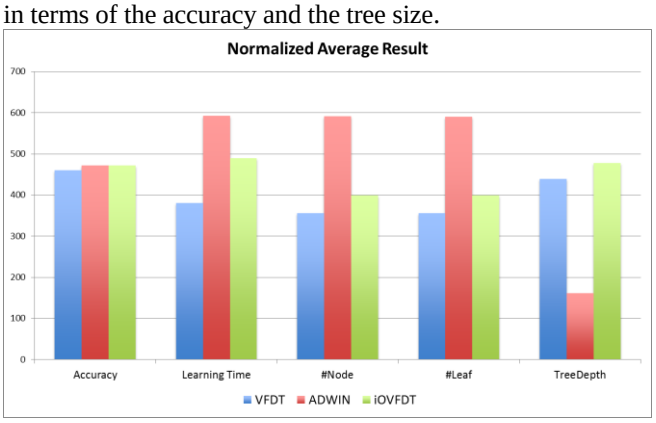

Figure 9. Normalized comparison result of MovieLens data

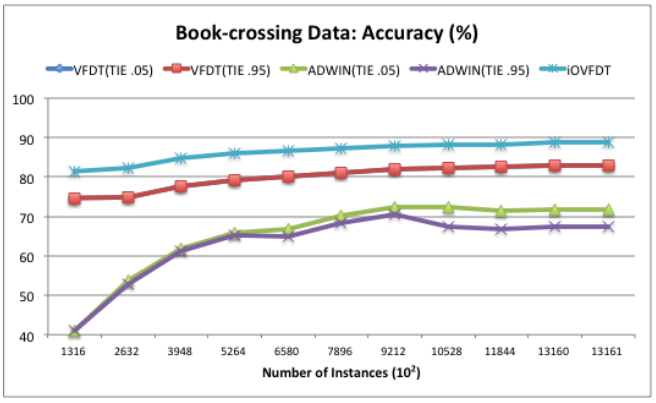

Figure 10. Accuracy of Book-crossing data

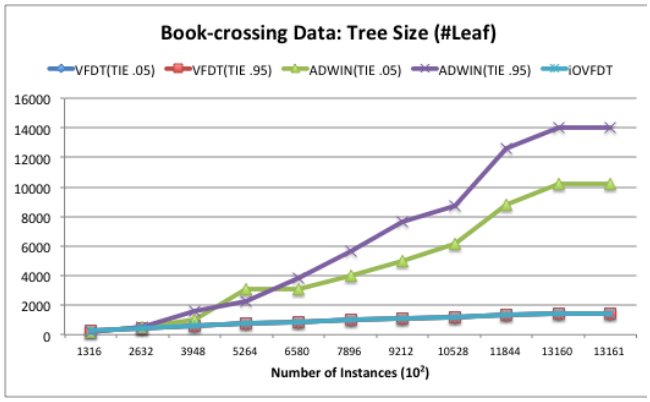

Figure 11. Tree size of Book-crossing data

VII. Coclusions

How to uncover the knowledge hidden within massive and big data efficiently, remains an open question. In the opinion of author, a well-designed algorithm is crucial in solving the problems associated with big data.

A data stream model is usually defined as a model in which data move continuously at high-speed. Most big data can be considered as data streams, in which many new data are generated in a short time, and moving continuously. Data streams contain very large volumes of data, which cannot be stored in either internal or external memory. A one-pass algorithm, therefore, forms the basis of data stream mining, which briefly stores a sufficient statistical matrix when new data passes, but does not require the full dataset being scanned repeatedly. However, imperfect data streams, like missing values, noise, imbalanced distribution and concept-drift, are common in the real world applications. To the best knowledge of the author, no suitable methods have solved all above problems well so far.

The main contributions of this research propose:

- An incremental decision tree algorithm handling imperfect data streams.
- A mechanism so called Auxiliary Reconciliation Control (ARC) is used to handle the missing data.
- An adaptive-tie breaking threshold is robust to the noisy data.
- A new functional tree leaf of weighted Na ïve Bayes is brought forward to deal with imbalanced distributions in data streams.
- A test-then-train learning approach monitors the performance of decision model in real-time so that the model is sensitive to concept-drift occurrence.

Experiment shows the proposed methodology can solve the aforementioned problems as a result.

References

[1] Domingos P., and Hulten G.. 2000. ‘Mining high-speed data streams’, in Proc. of 6th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD’00), ACM, New York, NY, USA, pp. 71-80.

[2] Hulten G., Spencer L., and Domingos P., 2001. ‘Mining time-changing data streams’, in Proc. of 7th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD’01), ACM, New York, NY, USA, pp. 97-106.

[3] Gama.J. Rocha R. and Medas P., 2003.‘Accurate decision trees for mining high-speed data streams’, in Proc. of 9th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD’03), ACM, New York, NY, USA, pp. 523-528.

[4] Bifet A. and Gavalda R. 2007. “Learning from time-changing data with adaptive windowing”. In Proc. of SIAM International Conference on Data Mining, pp. 443–448.

[5] Quinlan R, 1986. Induction of Decision Trees, Machine Learning, 1(1), pp.81-106.

[6] Quinlan R, 1993. C4.5: Programs for Machine Learning, Morgan Kaufmann, San Francisco.

[7] Breiman L., Friedman J.H., Olshen R.A. and Stone C.J., 1984. 'Classification and Regression Trees', in Wadsworth & Brooks/Cole Advanced Books & Software, Monterey, CA.

[8] Yang H., and Fong S., 2011. ‘Moderated VFDT in Stream Mining Using Adaptive Tie Threshold and Incremental Pruning’, in Proc. of 13th international conference on Data Warehousing and Knowledge Discovery (DaWak2011), LNCS, Springer Berlin / Heidelberg, pp. 471-483.

[9] Farhangfar, A., Kurgan, L., & Dy, J. (2008). Impact of imputation of missing values on classification error for discrete data. *Pattern Recognition*, *41*(12), 3692-3705.

[10] Ding, Y., & Simonoff, J. S. (2010). An investigation of missing data methods for classification trees applied to binary response data. *The Journal of Machine Learning Research*, *11*, 131-170.

[11] Little, R. J., & Rubin, D. B. (1987). *Statistical analysis with missing data*(Vol. 539). New York: Wiley.

[12] Lakshminarayan, K., Harp, S. A., & Samad, T. (1999). Imputation of missing data in industrial databases. *Applied Intelligence*, *11*(3), 259-275.

[13] Street, W. N., & Kim, Y. (2001, August). A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*(pp. 377-382). ACM.

[14] Li, C., Zhang, Y., & Li, X. (2009, June). OcVFDT: one-class very fast decision tree for one-class classification of data streams. In *Proceedings of the Third International Workshop on Knowledge Discovery from Sensor Data*(pp. 79-86). ACM.

[15] Hashemi, S., & Yang, Y. (2009). Flexible decision tree for data stream classification in the presence of concept change, noise and missing values.*Data Mining and Knowledge Discovery*, *19*(1), 95-131.

[16] Pfahringer, B., Holmes, G., & Kirkby, R. (2007). New options for hoeffding trees. In *AI 2007: Advances in Artificial Intelligence* (pp. 90-99). Springer Berlin Heidelberg.

[17] Hoeglinger, S., Pears, R., & Koh, Y. S. (2009). CBDT: A Concept Based Approach to Data Stream Mining. In *Advances in Knowledge Discovery and Data Mining* (pp. 1006-1012). Springer Berlin Heidelberg.

Call for Papers and Special Issues

Aims and Scope

Journal of Emerging Technologies in Web Intelligence (JETWI, ISSN 1798-0461) is a peer reviewed and indexed international journal, aims at gathering the latest advances of various topics in web intelligence and reporting how organizations can gain competitive advantages by applying the different emergent techniques in the real-world scenarios. Papers and studies which couple the intelligence techniques and theories with specific web technology problems are mainly targeted. Survey and tutorial articles that emphasize the research and application of web intelligence in a particular domain are also welcomed. These areas include, but are not limited to, the following:

- Web 3.0
- Enterprise Mashup
- Ambient Intelligence (AmI)
- Situational Applications
- Emerging Web-based Systems
- Ambient Awareness
- Ambient and Ubiquitous Learning
- Ambient Assisted Living
- Telepresence
- Lifelong Integrated Learning
- Smart Environments
- Web 2.0 and Social intelligence
- Context Aware Ubiquitous Computing
- Intelligent Brokers and Mediators
- Web Mining and Farming
- Wisdom Web
- Web Security
- Web Information Filtering and Access Control Models
- Web Services and Semantic Web
- Human-Web Interaction
- Web Technologies and Protocols
- Web Agents and Agent-based Systems
- Agent Self-organization, Learning, and Adaptation
- Agent-based Knowledge Discovery
- Agent-mediated Markets
- Knowledge Grid and Grid intelligence
- Knowledge Management, Networks, and Communities
- Agent Infrastructure and Architecture
- Agent-mediated Markets
- Cooperative Problem Solving
- Distributed Intelligence and Emergent Behavior
- Information Ecology
- Mediators and Middlewares
- Granular Computing for the Web
- Ontology Engineering
- Personalization Techniques
- Semantic Web
- Web based Support Systems
- Web based Information Retrieval Support Systems
- Web Services, Services Discovery & Composition
- Ubiquitous Imaging and Multimedia
- Wearable, Wireless and Mobile e-interfacing
- E-Applications
- Cloud Computing
- Web-Oriented Architectrues

Special Issue Guidelines

Special issues feature specifically aimed and targeted topics of interest contributed by authors responding to a particular Call for Papers or by invitation, edited by guest editor(s). We encourage you to submit proposals for creating special issues in areas that are of interest to the Journal. Preference will be given to proposals that cover some unique aspect of the technology and ones that include subjects that are timely and useful to the readers of the Journal. A Special Issue is typically made of 10 to 15 papers, with each paper 8 to 12 pages of length.

The following information should be included as part of the proposal:

- Proposed title for the Special Issue
- Description of the topic area to be focused upon and justification
- Review process for the selection and rejection of papers.
- Name, contact, position, affiliation, and biography of the Guest Editor(s)
- List of potential reviewers
- Potential authors to the issue
- Tentative time-table for the call for papers and reviews

If a proposal is accepted, the guest editor will be responsible for:

- Preparing the “Call for Papers” to be included on the Journal’s Web site.
- Distribution of the Call for Papers broadly to various mailing lists and sites.
- Getting submissions, arranging review process, making decisions, and carrying out all correspondence with the authors. Authors should be informed the Instructions for Authors.
- Providing us the completed and approved final versions of the papers formatted in the Journal’s style, together with all authors’ contact information.
- Writing a one- or two-page introductory editorial to be published in the Special Issue.

Special Issue for a Conference/Workshop

A special issue for a Conference/Workshop is usually released in association with the committee members of the Conference/Workshop like general chairs and/or program chairs who are appointed as the Guest Editors of the Special Issue. Special Issue for a Conference/Workshop is typically made of 10 to 15 papers, with each paper 8 to 12 pages of length.

Guest Editors are involved in the following steps in guest-editing a Special Issue based on a Conference/Workshop:

- Selecting a Title for the Special Issue, e.g. “Special Issue: Selected Best Papers of XYZ Conference”.
- Sending us a formal “Letter of Intent” for the Special Issue.
- Creating a “Call for Papers” for the Special Issue, posting it on the conference web site, and publicizing it to the conference attendees. Information about the Journal and Academy Publisher can be included in the Call for Papers.
- Establishing criteria for paper selection/rejections. The papers can be nominated based on multiple criteria, e.g. rank in review process plus the evaluation from the Session Chairs and the feedback from the Conference attendees.
- Selecting and inviting submissions, arranging review process, making decisions, and carrying out all correspondence with the authors. Authors should be informed the Author Instructions. Usually, the Proceedings manuscripts should be expanded and enhanced.
- Providing us the completed and approved final versions of the papers formatted in the Journal’s style, together with all authors’ contact information.
- Writing a one- or two-page introductory editorial to be published in the Special Issue.

More information is available on the web site at http://www.academypublisher.com/jetwi/.

(Contents Continued from Back Cover)