Stream Mining Dynamic Data by Using iOVFDT

Yang Hang, Simon Fong
Department of Computer and Information Science
University of Macau
Taipa, Macau SAR
Email: henry.yh@gmail.com, ccfong@umac.mo

Abstract— **Dynamic data is referring to data that are being produced continuously and their volume can potentially amount to infinity. They can be found in many daily applications such as e-commerce, security surveillance and activities monitoring. Such data call for a new generation of mining algorithms, called stream mining that is able to mine dynamic data without the need of archiving them first. This paper[1] studies the efficacy of a prominent stream mining method, called iOVFDT that stands for Incrementally Optimized Very Fast Decision Tree, under the environments of dynamic data. Six scenarios of dynamic data which have different characteristics are tested in the experiment. Each type of dynamic data represents a decision-making problem which demands an efficient classification mechanism such as decision tee to quickly and accurately classify a new case into a defined group. iOVFDT is compared with other popular stream mining algorithms, and it shows its superior performance.**

Index Terms—**Classification; stream mining; dynamic data**

I. Introduction

Web mining [1] attracted much research attention nowadays because of the popularity of applications running on the Web as well as their practical importance in businesses that operate on the Web platform. Specifically the Web data pose certain challenging to data mining research community for they are relatively more unstructured, the data feeds are continuous and dynamic – the data may arrive asynchronously with possibly no end. Traditional data mining algorithms are designed to work with fixed training dataset; once a prediction model is trained based on the fixed training dataset, it is ready to test new data and make prediction. Should the underlying model need to be updated due to the change of data, the whole model would have to be reconstructed again.

In advent of the time critical applications, so-called real-time applications that work with dynamic data [2], a new breed of data mining models aka stream mining emerged. The model building algorithms of stream mining are designed in response to dynamic data which exhibit dynamic characteristics such as unpredictable missing values or noises, and endless data flow. For example, web logs that are generated from a busy e-commerce website are never-ending and they may come from a wide variety of activities on the website. Most of the commercial software tools, however, are based on traditional data mining algorithms, usually coupled with their database management software packages. In the data mining research community there are no shortage of stream mining algorithms nevertheless. A classical one is called Very Fast Decision Tree (VFDT) [3], which can build a decision tree simply by keeping track of the statistics of the attributes of the incoming data. When sufficient statistics have accumulated at each leaf, a node-splitting algorithm determines whether there is enough statistical evidence in favor of a node-split, which expands the tree by replacing the leaf with a new decision node. VFDT learns by incrementally updating the tree while scanning the data stream on the fly. This powerful concept is in contrast to a traditional decision tree that requires the reading up of a full dataset for tree induction. VFDT uses Hoeffding Bound (HB) for available memory usage in the node-splitting process. The obvious advantage is the anytime data mining capability, which frees it from the need to store up all of the data for training the decision tree.

Recently an innovative and effective incremental optimization model for VFDT is proposed in [4], which is called Incrementally Optimized Very Fast Decision Tree (I-OVFDT). The features of iOVFDT are summarized briefly as follow: (1) it combines incremental optimization to decision tree model for high-speed data streams; (2) it facilitates an optimization algorithm in which the parameters for tree-growing are automatically computed, instead of fixed values that have to be pre-defined by users; (3) the incremental model maintains a balanced tree model amongst accuracy, tree size and learning time. The features of iOVFDT as they can be seen above could potentially solve the shortcomings of stream mining of dynamic data.

The objective of this paper therefore is to study the efficacy of iOVFDT, under the environments of dynamic data. Six scenarios of dynamic data which have different characteristics are tested in the experiment. Each type of dynamic data represents a typical decision-making problem like those found in web applications, real-time security surveillance and activities monitoring. These applications often demand for an efficient and effective real-time classification to quickly and accurately classify a new test into a target group. In this paper iOVFDT is compared with other popular stream mining algorithms with respect to the six data `scenarios.

The rest of this paper is structured as follow: The theoretical formulation of iOVFDT is presented in Section 2. The dynamic data and their sources are

[1] This paper extends the work presented in Hang Yang, Simon Fong, Yain-Whar Si, "Multi-objective Optimization for Incremental Decision Tree Learning", in the Proceeding of the 14th International Conference on Data Warehousing and Knowledge Discovery, Springer LNCS, Vienna (Austria) September 3 - 6, 2012.

described in Section 3. The experiment and discussion on the results then follow in Section 4. A conclusion is given in Section 5.

II. iOVFDT Model

A. Very Fast Decision Tree (VFDT)

Assume S_i is a set of data stream with the form (*X*, *y*), where *X* is a vector of *d* attributes and *y* is the actual discrete class label. Attribute X_i is the i^{th} attribute in *X* and is assigned a value of $X_{i1}, X_{i2}\ldots X_{ij}$, where $1 \leq i, j \leq d$. Suppose *HT* is a tree induction using Hoeffding Bound (HB). The classification objective is to produce a decision tree model that predicts the classes in future examples from *N* past examples which represent a segment of data stream that arrived previously. The example size is unbounded that $N \rightarrow \infty$.

The tree is built by recursively replacing leaves with decision nodes. Statistics n_{ijk} of attribute X_{ij} values are stored in each leaf y_k. A heuristic evaluation function is used to determine split attributes for converting leaves to nodes. Nodes contain the split attributes and leaves contain only the class labels. When a sample enters, it traverses the tree from the root to a leaf, evaluating the relevant attributes at every node. At this time, the system evaluates each possible condition based on the attribute values – if the statistics are sufficient to support one test over the others, then a leaf is converted to a decision node. The splitting check uses a heuristic evaluation function *G(.)*. The necessary number of samples (sample#) uses *HB*, shown in (1), to ensure control over errors in the attribute-splitting distribution selection.

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}} \quad (1)$$

For *n* independent observations of a real-valued random variable *r* whose range is *R*, *HB* illustrates that with a confidence level of 1-δ, the true mean of *r* is at least $\bar{r} - \varepsilon$ where $\bar{r}$ is the observed mean of the samples. For a probability, the range *R* is 1 and for information gain, the range *R* is $log_2Class\#$. VFDT makes use of *HB* to choose a split attribute as a decision node. Let x_a be the attribute with the highest *G(.)*, and x_b the attribute with the second-highest *G(.)*. $\Delta\bar{G} = \bar{G}(x_a) - \bar{G}(x_b)$ is the difference between the two greatest-quality attributes. If $\Delta\bar{G} > HB$ with *n* samples is observed in a leaf, and *HB* states with probability *1-δ* that x_a is the attribute with the highest value in *G(.)*, then the leaf is converted into a decision node that splits on x_a.

B. Incremental Optimization Function for VFDT

Suppose the optimization problem Π is defined as a tuple $(X, \mathcal{F}, \Phi)$. The set *X* is a collection of objects, and the feasible solution $\mathcal{F}$ are subsets of *X* that collectively achieve a certain optimization goal. The set of all possible solutions is $\mathcal{F} \subseteq 2^X$, and $\Phi: \mathcal{F} \rightarrow \mathbb{R}$ is a cost function on these solutions. A weight ω_x with every object *x* of *X* is defined as $\Phi(S) = \sum_{x \in S} \omega_x$. The optimal solution $O(X, \mathcal{F}, \Phi)$ exists if *X* and Φ are awareness, and the subset $S \subseteq X, S \in \mathcal{F}$ is optimizing $\Phi(S)$.

The solution is to be integrated into the decision tree induction model, whose algorithm is based on Hoeffding tree (HT) using the HB in node-splitting control. Therefore, the incremental optimization function can be expressed as a sum of several sub-objective functions:

$$\Phi(HT_x) = \bigcup_{D=1}^{M} \Phi_D(HT_x) \quad (2)$$

where $\Phi_m : \mathcal{F} \rightarrow \mathbb{R}$ is a continuously differentiable function and M is the number of objects in the optimization problem. iOVFDT builds a desirable tree model combining with an incremental optimization mechanism, seeking a compact tree model that balances the tree size, prediction accuracy and learning time. Consequently, the fixed installed parameters are replaced by an adaptive mechanism when new data arrive. The optimization problem is considered as:

$$minimize\ \Phi(HT_x)\ subject\ to\ HT_x \in X \quad (3)$$

The proposed method shall find a general optimization function $\Phi(HT_x)$ in (2), considering prediction accuracy, tree size and learning speed at the same time, so *M* = 3.

When a new data segment arrives, it will be sorted from the root to a leaf in terms of the existing *HT* model. The data stream S_i contains information (*X*, *y*), where *X* is a vector of *d* attributes and *y* is the actual discrete class label in a supervised learning process. Attribute X_i is the i^{th} attribute in *X* and is assigned a value of $X_{i1}, X_{i2}\ldots X_{ij}$, where $1 \leq i,\ j \leq d$. Decision tree algorithm uses $\widehat{y_k} = HT(X)$ to predict the class when a new data sample (*X*, y_k) arrives. The prediction accuracy $accu_n$ is dynamically changing with the example size *n* growing in an incremental learning process, defined as:

$$accu_n = \frac{\sum_{i=1}^{n} Predict(S_i)}{n} \quad (4)$$

$$Predict(S_i) = \begin{cases} 1, if\ \widehat{y_k} = y_k \\ 0, if\ \widehat{y_k} \neq y_k \end{cases} \quad (5)$$

In order to measure the utility of the three dimensions using a minimizing function in (3), the measure of prediction accuracy is reflected by the prediction error in (6):

$$\Phi_1 = erro_n = 1 - accu_n \quad (6)$$

The classification goal is to produce a decision tree model *HT* from *N* examples that predicts the class $\widehat{y_k}$ in future examples with accuracy p. In data stream mining, the example size is very large, even unlimited that $n \rightarrow \infty$.

A *tree-path*, starting from the root to a leaf, represents a regression pattern – the class $\hat{y}$ stated in the leaf. When an internal node splits to be a new leaf, the total number of leaves grows. A decision model is a tree-like structure that presents the patterns of non-linear relationship mapping between *X* and the class by the tree-paths. The number of leaves in the decision model represents the number of patterns/rules in this model. Therefore, *the definition of tree size is the number of leaves in decision model.* When a leaf is being generated, the tree size grows. The data flow continuously that the decision model incrementally refreshes when a new leaf is created. Therefore, the tree size function is:

$$\Phi_2 = size_n = \begin{cases} size_{n-1} + 1\ , if\ \Delta\bar{G} > HB \\ \\ size_{n-1} \quad , otherwise \end{cases} \quad (7)$$

The conditions of iOVFDT node splitting, which inherits the use of HB in (1), where $\Delta\bar{G} = \bar{G}(x_a) - \bar{G}(x_b)$ is the difference between the two best attributes.

iOVFDT is a one-pass algorithm that builds a decision model using a single scan over the training data. The sufficient statistics, which counts the number of examples passed an internal node, are the only updated elements in the one-pass algorithm. The calculation is a "plus one" incremental process that consumes little computation resource. Hence, the computation speed of such "plus one" operation for a new example passing is supposed as a constant value $Rate$ in learning process. The number of examples that have passed within an interval period of in node splitting control determines the learning time. Originally in VFDT, n_{min} is a fixed value to control the interval time checking node splitting.

$$\Phi_3 = Time_n = Rate \times (n_{y_k} - n_{min}) \qquad (8)$$

Suppose 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 in (8), during which how many have examples passed. The optimum tree model is the HT_x structure with the minimum $\phi(x)$. The area of this triangle $\Phi_x(HT_x)$ changes when node splitting happens and the HT updates. A min-max constraint of the optimization goal in (3) controls the node splitting, which ensures the new tree model keeps a $\Phi_{HT}(x)$ within a considerable range. Suppose $Max\text{: } \Phi(HT_x)$ is a HT model with the maximum utility so far, and $Min\text{: } \Phi(HT_x)$ is a HT model with the minimum utility. The optimum model should be within this min-max range, near the $Mean\text{: } \Phi(HT_x)$:

$$Mean\text{: } \Phi(HT_x) = \frac{\text{Max.}\Phi(HT_x) - Min.\Phi(HT_x)}{2} \qquad (9)$$

According to the Chernoff bound [5], we know:

$$|\text{Opt: } \Phi(HT_x{}^*) - Mean\text{: } \Phi(HT_x)| \leq \sqrt{\frac{\ln(1/\delta)}{2n}} \qquad (10)$$

where the range of $\Phi_x(HT_x)$ is within the min-max constraint that $\text{Min: } \Phi(HT_x) < \text{Opt: } \Phi(HT_x{}^*) < \text{Max: } \Phi(HT_x)$. Therefore, if $\Phi(HT_x)$ exceeds this constraint, the existing HT is not suitable to embrace the new data input and that the tree model should not be updated. The node-splitting condition is adaptively re-optimized in OVFDT so that: $\Delta\bar{G} > HB$, or $\text{Opt: } \Phi(HT_x{}^*) > Max\text{: } \Phi(HT_x)$, or $\text{Opt: } \Phi(HT_x{}^*) < Min\text{: } \Phi(HT_x)$, instead of a fixed tie-breaking threshold.

C. Enhanced Prediction with Functional Leaf

The prediction of HT can be further enhanced by an embedded Naïve Bayes classifier, *Functional Tree Leaf* (FTL) [6]. For classification with FTL, iOVFDT uses $\hat{y}_k = HT_{\mathcal{F}}(X)$ to predict the class label when a new sample (X, y) arrives. The predictions are made according to cache called Observed Class Distribution (OCD) in the FTL, $\mathcal{F}$. Originally in VFDT the prediction used only the majority class Functional Tree Leaf $\mathcal{F}^{MC}$. The majority class only considers the counts of the class distribution, but not the decisions based on combinations of attributes. The naïve Bayes Functional Tree Leaf $\mathcal{F}^{NB}$ was proposed to compute the conditional probabilities of the attribute-values given a class at the tree leaves by naïve Bayes. As a result, the prediction at the leaf is refined by the consideration of the probabilities of each attribute. To handle the imbalanced class distribution in a data stream, a weighted naïve Bayes Functional Tree Leaf $\mathcal{F}^{WNB}$ and an adaptive Functional Tree Leaf $\mathcal{F}^{Adaptive}$ are proposed in this paper. The sufficient statistics n_{ijk} is 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}. Suppose that k is the number of classes distributed in the training data, where $k \geq 2$. A vector V_{ij} is 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 observed class distribution (OCD) vector of $Node_{ij}$. OCD stores the count of distributed class at each tree node in OVFDT. It helps to keep track of the occurrences of the instances of each attribute.

Majority Class Functional Tree Leaf: In the ODC vector, the majority class Functional Tree Leaf $\mathcal{F}^{MC}$ chooses the class with the maximum distribution as the predictive class in a leaf, where $\mathcal{F}^{MC}$: arg max $f = \{n_{i,j,1}, n_{i,j,2} \ldots n_{i,j,r} \ldots n_{i,j,k}\}$, 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 Functional Tree Leaf $\mathcal{F}^{NB}$ 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 $n'_{i,j,r} = \text{P}(X|C_f) \cdot \text{P}(C_f) \,/\, \text{P}(X)$, where X is the new arrival instance. Hence, the prediction class is $\mathcal{F}^{NB}$: arg max $i = \{n'_{i,j,1}, n'_{i,j,2} \ldots n'_{i,j,r} \ldots n'_{i,j,k}\}$.

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 Functional Tree Leaf $\mathcal{F}^{WNB}$ 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 $n'_{i,j,r} = \omega_r \cdot \text{P}(X|C_f) \cdot \text{P}(C_f) \,/\, \text{P}(X)$, where X is the new arrival instance, and the weight is the probability of class i distribution amongst all the observed samples such that $\omega_r = v_r / \sum_{r=1}^{k} v_r$, where $n_{i,j,r}$ is the count of class r. Hence, the prediction class is $\mathcal{F}^{WNB}$: arg max $f = \{n'_{i,j,1}, n'_{i,j,2} \ldots n'_{i,j,r} \ldots n'_{i,j,k}\}$.

Adaptive Functional Tree Leaf: In a leaf, suppose that $V_{\mathcal{F}}{}^{MC}$ is the observed class distribution vector with the majority class Functional Tree Leaf $\mathcal{F}^{MC}$; suppose $V_{\mathcal{F}}{}^{NB}$ is the observed class distribution vector with the naïve Bayes Functional Tree Leaf $\mathcal{F}^{NB}$; and suppose that $V_{\mathcal{F}}{}^{WNB}$ is the observed class distribution vector with the weighted naïve Bayes Functional Tree Leaf $\mathcal{F}^{WNB}$. Suppose that y is the true class of a new instance X. Suppose that $E_{\mathcal{F}}$ is the prediction error rate using a Functional Tree Leaf $\mathcal{F}$. $E_{\mathcal{F}}$ 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 $\mathcal{F}$. The adaptive Functional Tree Leaf chooses the class with the minimum error rate predicted by the other three strategies, where $\mathcal{F}^{Adaptive}$: arg min $\mathcal{F} = \{E_{\mathcal{F}}{}^{MC}, E_{\mathcal{F}}{}^{NB}, E_{\mathcal{F}}{}^{WNB}\}$.

D. *iOVFDT Algorithm*

iOVFDT tree-building algorithm is presented in the pseudo codes here. On the base of the metrics, the input parameters are given in Figure 1. The overall of iOVFDT is described in Figure 2. If it is a new tree model, the tree should be initialized with a single root (Figure 3). When new data stream arrives, it traverses from the root to a predicted FTL according to the existing tree model. If the node-splitting check met, the node-splitting estimation is implemented in Figure 5.

```
INPUT:
S :       A stream of sample
X :       A set of symbolic attributes
G(.) :    Heuristic function using for node-splitting
          estimation
δ :       One minus the desired probability of choosing a
          correct attribute at any given node
n_min :   The minimum number of samples between check
          node-splitting estimation
F :       A functional tree leaf strategy
OUTPUT:
HT :      A decision tree
```

Figure 1. Global variables of iOVFDT.

```
PROCEDURE: OVFDT(S, X, G(.), δ, n_min, F)
1. A data stream S arrives
2. IF HT is null, THEN initializeHT(S, X, G(.), δ, n_min, F)
   ELSE traverseHT(S, HT, F) and update Error
3. Label l as the predicted class among the samples seen so far at
   the leaf l.
4. Let $n_{yk}$ be the number of samples seen at the leaf with class $y_k$.
5. IF the samples seen so far at leaf l do not all belong to the
   same class, and ( $n_{yk}$ mod $n_{min}$ ) is zero, THEN
6. Calculate Size by (4)
7. Calculate Time by (5)
8. doNodeSplittingEstimation(S, X, G(.), δ, n_min)
9. Return HT
```

Figure 2. Overall approach of iOVFDT algorithm.

```
PROCEDURE: initializeHT(S, X, G(.), δ, n_min)
1. Let HT be a tree with a single leaf l (the root). Let $X_l = X \cup \{X_\emptyset\}$
2. Let $G_l(X_\emptyset)$ be the G(.) obtained by predicting the class in S,
   according to F.
3. FOR each class $y_k$
4. // $y_k$ is the class lable y with the k th label
5.     FOR each value $x_{ij}$ of attribute $X_i \in X$
6.         Reset OCD: $n_{ijk}(l)=0$
7.         // $n_{ijk}(l)$ is the count of attribute with $x_{ij}$ and $y_k$ at leaf l
8.     END-FOR
9. END-FOR
10. Return HT with a single root
```

Figure 3. Tree initialization of iOVFDT algorithm.

```
PROCEDURE: traverseHT(S, HT, F)
1. Sort S from the root to a leaf by HT. Update OCD in each node:
   $n_{ijk}(l)$ ++
2. Switch (F)
3.     Case $F^{MC}$: predict the class $y'_k$ with max $n_{ijk}(l)$
4.     Case $F^{NB}$: predict the class $y'_k$ with max NB prob.
5.     Case $F^{WNB}$: predict the class $y'_k$ with max WNB prob.
6.     Case $F^{Adaptive}$: predict the class $y'_k$ using F with $Error_{min}$
7. IF $y'_k$ equals to the actual class label in S, THEN $C_T$ ++
8. ELSE $C_F$ ++
9. $Error = C_F/(C_T+C_F)$
10. Return Error
```

Figure 4. New data traversing of iOVFDT algorithm.

```
PROCEDURE doNodeSplittingEstimation(S, X, G(.), δ)
1. FOR each attribute $X_i \in X_l - \{X_\emptyset\}$ at the leaf l
2.     Compute $G_l(X_i)$
3.     Let $X_a$ be the attribute with highest $G_l(.)$ and $X_b$ with the
   2nd highest $G_l(.)$
4.     Compute HB with δ
5.     Let $\Delta G_l = G_l(X_a) - G_l(X_b)$
6. END-FOR
7. Calculate $\Phi_x(HT_x)$, according to Error, Time and Size in (9)
8. IF ($\Delta G_l > HB$) or ($\Phi_x(HT_x) < \Phi_{min}(HT_x)$)
    or ($\Phi_x(HT_x) > \Phi_{max}(HT_x)$)
9.     Replace l by an internal node splits on $X_a$
10.    IF ($\Phi_x(HT_x) < \Phi_{min}(HT_x)$) $\Phi_{min}(HT_x) = \Phi_x(HT_x)$
11.    IF ($\Phi_x(HT_x) > \Phi_{max}(HT_x)$) $\Phi_{max}(HT_x) = \Phi_x(HT_x)$
12.    FOR each branch of splitting
13.        Add a new leaf $l_m$ and let $X_m = X - \{X_a\}$
14.        Let $G(X_\emptyset)$ be G(.) obtained by predicting the class in S,
           according to F at $l_m$
15.        FOR each class $y_k$ and each value $x_{ij}$ of each attribute
16.            $X_i \in X_m - \{X_\emptyset\}$ and reset OCD: $n_{ijk}(l) = 0$
17.        END-FOR
18.    END-FOR
19. END-IF
20. Return updated HT
```

Figure 5. Node-splitting of OVFDT algorithm.

III. Experimental Platform and Datasets

As a simulation platform for experiments, an iOVFDT Java package integrated with MOA toolkit [7] is programmed. The running environment is a Windows 7 PC with Intel Quad 2.8GHz CPU and 8G RAM. In all of the experiments, the parameters of the algorithms were $\delta=10^{-6}$ and $n_{min}=200$, which are default values recommended by MOA. δ is the allowable error in split decision and values closer to zero will require a longer time to decide (used in HB calculation); n_{min} is the default number of instances a leaf should observe between split attempts.

Six scenarios of dynamic data which possess different characteristics are put under test in the experiment, with iOVFDT and other variants of VFDT. The data mining and data stream mining algorithms that are chosen for tests are as follow: C4.5 [8], both pruned and un-pruned versions – that represent traditional data mining algorithm; and the rest that represent data stream mining algorithms – Incremental Naïve Bayes, the original version of VFDT, VFDT with FTL of Naviie Bayes, VFDT with FTL of Adaptive, and the family of iOVFDT algorithms – iOVFDT with FTL of Majority Class, iOVFDT with FTL of Naïve Bayes, iOVFDT with FTL of Weighted Naïve Bayes, and iOVFDT with FTL of Adaptive.

The representative dynamic data include the datasets of Robot Sensor (RS), iPod Sales on eBay (IP), Internet Usage (IU), Cover Type (CT), Person Activity (PA) and Network Attack (NA).

The datasets are extracted from real-world applications which are available for download from UCI Machine Learning Repository (archive.ics.uci.edu/ml/datasets.html) and KDD Cup Data Center (www.sigkdd.org/kddcup). UCI Dataset Archive is a popular place for researchers downloading publicly available data for testing machine learning algorithms. KDD Cup is the annual Data Mining and Knowledge Discovery competition organized by ACM Special Interest Group on Knowledge Discovery and Data Mining. The datasets are described below.

Robot Sensor (RS)

Robot Sensor (RS) dataset [9] was released in 2010. It was a non-linear classification task that collected from a robot following a wall in clockwise direction. 24 ultrasound sensors collected data for robot navigating. It contained 24 continuous attributes and 4 distinct classes, which navigated the robot's movement. We used the numeric attributes to build a decision model, which classified to one of the four classes "Move-Forward", "Slight-Right-Turn", "Sharp-Right-Turn" and "Slight-Left-Turn". Together with Person Activity Analysis dataset, RS dataset represent body movement activities which are dynamic in nature. The values of the attributes of such data change collaboratively that represent certain posture or movement; it may not be making any sense when the attribute values are viewed individual; but when they are combined together they implicitly mean a movement. The data instances change rapidly at times; and the sensor data are being generated continuously as a data stream even when the robot is idle. Depending on the sampling rate of the sensors, the data stream could flow continuously in at a high speed. The challenge is up to the decision tree to classify what behavior the robot is exhibiting given the data feed.

iPod Sales on eBay (IP)

Auction platforms such as e-Bay serve as an important fusion example of internet community and e-commerce. Bidders trade among one another with a consideration of both the reputation of the other party as well as the quality of the product on bid. Often they want to attain an adequate price as a compromise between sellers and buyers. "How to obtain an optimal (reasonable) price?" is the question. In the light of this, some auction experts recommend shorter or longer bidding durations, lower or higher start prices, terminations of auctions on weekends or week days, and many other variants that will likely guarantee the success. In data mining a scientific answer to this question is attempted [10].

The dataset represent an e-commerce scenario where an electronics retailer has noticed that its internet auctions somehow gain quite different sales revenues. Thus, typical patters hopefully can be obtained via a decision tree for the explanation of this effect. In order to maximize the profit it is hoped that an optimal auction listing with the correctly set attribute values (start and end times, duration, additional attributes like optional product display enhancements, etc.). For this, a decision tree model is built from some historical set of data for which each new auction predicts whether the actual sales revenue is higher than the average sales revenue of the product category.

The training dataset contains the data of the previous auctions of the past months utilizing the e-Bay auction market data program in 2006. The data, provided in full conformance with protection of data privacy, contains all required information for the solution of this task. It has a sample of 8,000 online auctions from the category. "Audio&Hi-Fi:MP3-Player:Apple iPod" has to be used for the description (classifier model) of the protection of the sales revenue.

Additionally, the average sales revenue of the product category (item_leaf_category_name) has already been calculated as attribute category_avg_gms, and the attribute gms_greater_avg indicates, whether the realized sales revenue was higher than the average of the product category.

In our experiment, a collection of stream mining decision tree models are built for predicting whether the sales revenue of an unseen transaction will be higher than the average sales revenue of the product category. The transactions are dynamic in nature as there is no fixed amount or interval of the transaction may come.

Internet Usage (IU)

Internet Usage (IU) dataset was released in 1999. Data were collected from a survey [11] provided by the Graphics and Visualization Unit at Georgia Tech in 1997. It contained 72 discrete attributes of user's personal information and interests of using Internet. For a data mining utility, we use these data to build a decision model of user's occupations so that relevant advertising information will be delivered to approximate users. This is a typical scenario of Web recommendation where the information collected would be used to build a prediction model for automatically selecting a suitable product to recommend, or appropriate Web advertisement to display, to a Web visitor. The data exchanged are dynamic and interactive in nature. The interest or shopping trend may change over time, for instance due to seasonal changes and economy change.

Cover Type (CP)

Cover Type (CP) dataset was released in 1999. Inventory data of forestland were provided for natural eco-system management. A total of 42 categorical attributes and 12 continuous attributes were used for two predictive models, which predicted seven different cover lands in this study. Originally, two established predictive models (neural network and linear discriminant) were provided with this dataset [12]. It was said that the neural network model had higher absolute accuracy (70.58%) than the linear discriminant analysis model (58.38%). In this experiment, decision trees of stream mining are tested. CP is a typical kind of GIS land surveillance information where data are continuously sent from sensors for processing.

Person Activity Analysis (PA)

Person Activity Analysis (PA) dataset was released in 2010. It recorded the instances collected from four sensors on human body. Five people were examined in this study, each of who wore sensors on ankle left, ankle right, belt and chest. The instance includes two categorical attributes (identified the sensors and sequences), two time attributes (timestamp and date) and three numeric attributes collected from sensors. A decision model was established to classify human activities, e.g. walking, falling, etc. [13].

Network Attack Detection (NA)

Network Attack Detection (NA) was originally released for KDD 1999. It was used to establish a predictive model [14] to distinguish network intrusions and attacks from normal connections. A network connection was a packet of TCP transaction, in which data flows from and to an IP address to a targeted IP address under some network transferring protocols. 42 attributes were recorded in a connection instance. We used 10% of the full data (494021 instances) to build a predictive decision tree model to classify 23 distinct types of "bad" and "good" connections. A classification model can be adopted or embedded in network security software which recognizes the pattern of network attack – that of course change and evolve dynamically in operation.

The data structures of each experimental dataset are listed in Table I.

TABLE I. ATTRIBUTES OF THE DYNAMIC DATA USED IN EXPERIMENT

Name	Abbreviation	Sample size	No. of attributes	No. of classes	No. of nominal attributes	No. of numeric attributes
Robot Sensor	RS	5456	25	4	0	24
iPod Sales on eBay	IP	7882	29	3	16	12
Internet Usage	IU	10104	72	5	71	0
Cover Type	CT	581012	55	7	42	12
Person Activity	PA	164860	6	11	2	3
Network Attack	NA	494021	42	23	3	38

TABLE II. COMPARISON OF ACCURACY BY ALGORITHMS AND DATASETS

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~~	~~90.71~~	~~79.06~~	~~67.45~~	~~43.75~~	~~98.27~~
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

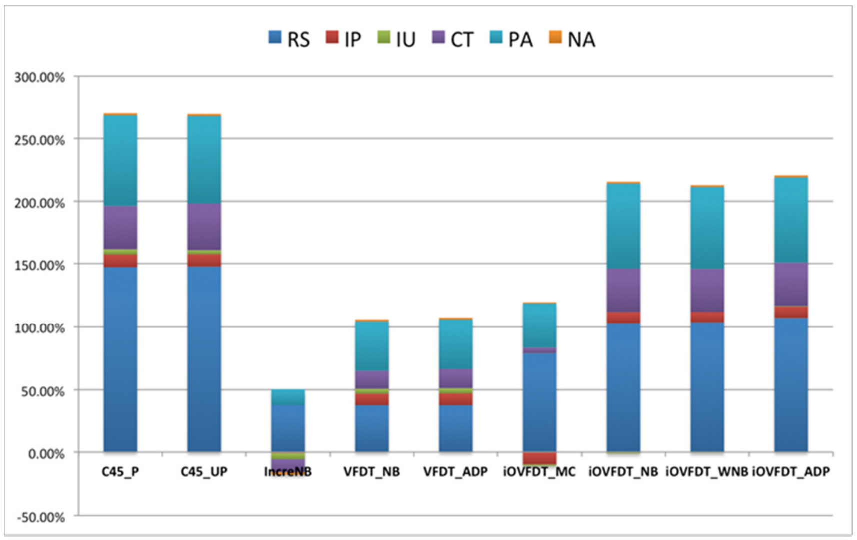

Figure 6. Accuracy improvement percentage.

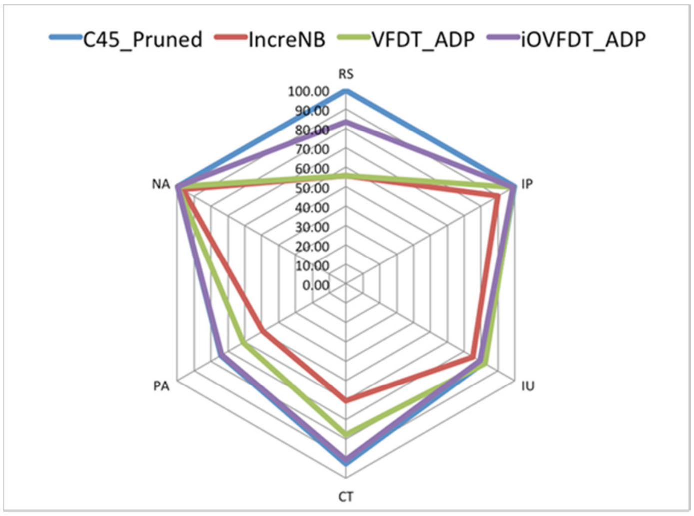

Figure 7. Accuracy radar chart of the selected methods.

The accuracy was the percentage of correctly classified number of total instances. Tree size is a sum of the number of leaves and the number of nodes. The number of leaves also indicates how many patterns in the form of decision paths or conditional rules that a decision tree model contains. The learning speed is measured by the average time of building a decision tree model per *N* samples.

IV. Experiment Results

We test our data stream mining algorithms with respect to three major performance indicators, namely, Classification Accuracy, Model Size and Learning Speed.

A. Classificatin Accuracy

From Table II, 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 are able to achieve a 'full win' of satisfactory accuracies over the average across all the datasets. Figure 6 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.

VFDT is chosen as a benchmark for the accuracy improvement analysis. We used the percentage of accuracy improvement to compare different methods accordingly. It is known however that incremental Naïve Bayes had the worst performance while C4.5 and iOVFDT (with functional tree leaf) had significantly better improvement than the others. Meanwhile, Figure 6 also shows that Adaptive (ADP) functional leaf obtained better accuracy than the other functional leaves, for either VFDT or iOVFDT. Then the four methods are selected as a comparison in Figure 7. Obviously pruned C4.5 has a best accuracy and iOVFDT_ADP is the second best, which had apparently higher accuracy than VFDT.

TABLE III. Comparison of Tree Size by algorithms and datasets

Methods\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

TABLE IV. COMPARISON OF LEARNING SPEED BY ALGORITHMS AND DATASETS

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	12.18	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	10.80	0.97	6.62
iOVFDT_WNB	0.16	0.10	0.29	10.61	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	10.34	0.96	6.04

B. *Model Size*

Table III shows the model size 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 the tree to grow incrementally over time.

D. *Learning Speed*

The speed of learning decision model was reflected by the time in seconds as shown in Table IV. 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.

V. CONCLUSION

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 behavourial patterns when analysed 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 over again for rebuilding an up-to-date decision tree. A new generation of stream mining is therefore invented to tackle this problem. Very Large Decision Tree (VFDT) is a classical stream mining algorithm which can incrementally starts a decision tree from scratch and updates it whenever new sample comes, without the need of reprocessing the whole dataset. An extended version of VFDT called incrementally optimized VFDT or iOVFDT inherits the advantages of VFDT plus an extra optimization that balances the performance criteria – accuracy, tree size and learning time.

In this paper, for the first time, six selected dynamic data each of which represents a typical application scenario were tested under iOVFDT and other variants. The results are encouraging as they show that iOVFDT is able to produce good classification accuracies almost on par with C4.5 without the drawbacks of batch learning. Superior performance results like compact tree size and reasonable learning time can be achieved too. These advantages make iOVFDT a suitable candidate algorithm for real-world applications especially those that deal with dynamic data and real-time constraints. In the future it is planned to test again with the same dynamic datasets but by using other kinds of stream mining algorithms than decision tree types. The experiments will be extended also to dynamic data that come in variable rates in order to observe how data latency, irregular dynamic data loads and information overwhelm may affect Web applications that are built by different stream mining algorithms.

REFERENCES

[1] Maged N. Kamel Boulos, "BlogBrain Ops: Proposal for a Semi-automatic Social Web Mining and Cyberinfluence Decision-support Tool for Info Ops Teams", Journal of Emerging Technologies in Web Intelligence, Vol 3, No 4 (2011), 317-322, Nov 2011, pp.317-322.

[2] Fujun Zhu et al., "Dynamic Data Integration Using Web Services", Proceedings of the IEEE International Conference on Web Services (ICWS'04), pp. 262-272.

[3] Pedro D., and Geoff H., "Mining high-speed data streams", in Proceeding of the sixth ACMSIGKDD international conference on Knowledge discovery and data mining, ACM, 2000, pp.71-80.

[4] Hang Yang, Simon Fong, Yain-Whar Si, "Multi-objective Optimization for Incremental Decision Tree Learning", in the Proceeding of the 14th International Conference on Data Warehousing and Knowledge Discovery, Springer LNCS, Vienna (Austria) September 3 - 6, 2012

[5] Chernoff H., "A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations", Annals of Mathematical Statistics, Vol. 23, 1952, pp.493-507.

[6] Gama J., Ricardo R., "Accurate decision trees for mining high-speed data streams", in Proceeding of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2003, pp.523-528.

[7] A. Bifet, G. Holmes, R. Kirkby, and Bernhard Pfahringer, "MOA: Massive Online Analysis", Journal of Machine Learning Research, Vol. 11, 2010, pp.1601-1604.

[8] S.B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques", Informatica Vol. 31, 2007, pp.249-268.

[9] Ananda L. et al, "Short-Term Memory Mechanisms in Neural Network Learning of Robot Navigation Tasks: A Case Study", Proceedings of the 6th Latin American Robotics Symposium (LARS'2009), Valparaíso-Chile, 2009, pp.1-6.

[10] Wei Chen, Simon Fong, "Social Network Collaborative Filtering Framework and Online Trust Factors: a Case Study on Facebook", The 5th International Conference on Digital Information Management (ICDIM 2010), July 2010, Thunder Bay, Canada, pp.266-273.

[11] Colleen M. Kehoe and James E. Pitkow, "Surveying the Territory: GVU's Five WWW User Surveys", The World Wide Web Journal, Vol. 1, no. 3, 1996, p.77-84.

[12] Zoran Obradovic and Slobodan Vucetic, "Challenges in Scientific Data Mining: Heterogeneous, Biased, and Large Samples", Technical Report, Center for Information Science and Technology Temple University, Chapter 1, pp.1-24

[13] B. Kaluza, V. Mirchevska, E. Dovgan, M. Lustrek, M. Gams, "An Agent-based Approach to Care in Independent Living", Ambient Intelligence, Lecture Notes in Computer Science, Springer, 2010, Volume 6439, pp.177-186.

[14] Wei Fan, Wenke Lee, Prodromidis, A., Chan, P.K., "Cost-based modeling for fraud and intrusion detection: results from the JAM project", DARPA Information Survivability Conference and Exposition, 2000, Vol. 2, pp.130-144.

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)