Multi-objective Optimization for Incremental Decision Tree Learning

Hang Yang, Simon Fong, and Yain-Whar Si

Department of Science and Technology, University of Macau,
Av. Padre Tomás Pereira Taipa, Macau, China
ya97404@gmail.com, {ccfong,fstasp}@umac.mo

Abstract. Decision tree learning can be roughly classified into two categories: static and incremental inductions. Static tree induction applies greedy search in splitting test for obtaining a global optimal model. Incremental tree induction constructs a decision model by analyzing data in short segments; during each segment a local optimal tree structure is formed. Very Fast Decision Tree [4] is a typical incremental tree induction based on the principle of Hoeffding bound for node-splitting test. But it does not work well under noisy data. In this paper, we propose a new incremental tree induction model called incrementally Optimized Very Fast Decision Tree (iOVFDT), which uses a multi-objective incremental optimization method. iOVFDT also integrates four classifiers at the leaf levels. The proposed incremental tree induction model is tested with a large volume of data streams contaminated with noise. Under such noisy data, we investigate how iOVFDT that represents incremental induction method working with local optimums compares to C4.5 which loads the whole dataset for building a globally optimal decision tree. Our experiment results show that iOVFDT is able to achieve similar though slightly lower accuracy, but the decision tree size and induction time are much smaller than that of C4.5.

Keywords: Decision Tree, Classification, Incremental Optimization, Stream Mining.

1 Introduction

How to extract knowledge efficiently from massive data has been a popular research topic. A decision tree, which presents the knowledge in a tree-like format, can be easily understood by both human and machine. Due to the high degree of comprehensibility, considered as one of the most important methods for classification.

In general, there are roughly two approaches for decision tree learning. The first approach loads full data, multi-scanning and analyzing them. This process builds a static tree model by greedy search, i.e. ID3 [1], C4.5 [2], CART [3]. When new data come, the whole data (including historical and fresh data) is re-loaded to update algorithm. The second approach only requires loading a small part of samples in terms of Hoeffding bound and comparing the best two values of heuristic function for node-splitting test, i.e. VFDT [4] (which will be introduced in Section 2) and its

A. Cuzzocrea and U. Dayal (Eds.): DaWaK 2012, LNCS 7448, pp. 217–228, 2012.

extensions [7,9,12,13]. Besides, static decision tree provides a global optimal model because it computes across the full samples by greedy search. Incremental tree maintains a local optimal model because it computes on a sufficient part of samples.

One challenge to decision tree learning is associated with noise, which generally renders a data stream "imperfect". The size of a decision tree model will grow excessively large under noisy data, so is an undesirable effect known as over-fitting. The imperfection significantly impairs the accuracy of a decision tree classifier through the confusion and misclassification prompted by the inappropriate data.

For static decision tree learning, pruning algorithms help keep the size of the decision tree in check, although the majority are post-pruning techniques that remove relevant tree paths after a whole model has been built from a stationary dataset [5, 6]. For incremental decision tree learning, post-pruning is not suitable because no extra time is available for stopping tree building and pruning the branches under high-speed data streams environment. It is said that the excessive invocation of tie breaking can cause significant decline in VFDT performance on complex and noise data [12], even with the additional condition by the parameter τ. MVFDT [7] uses an adaptive tie-breaking to reduce tree size for incremental tree.

In this paper, we propose a new incremental decision tree induction inheriting the usage of Hoeffding bound in splitting test, so called Incrementally Optimized Very Fast Decision Tree (iOVFDT). It contains a multi-objective incremental optimization mechanism so as to maintain a small tree size and comparable accuracy, even for imperfect data. For higher accuracy, four types of functional tree leaf are integrated with iOVFDT. In the experiment, we compare iOVFDT to a classical static tree induction C4.5 and pre-pruning incremental tree induction MVFDT. The objective of this paper is to shed light into the following research questions. What are the significant differences between static (global optimum) and incremental (local optimum) decision tree? The answer can be found in experiment and discussion sections, which also show the superior performance of our new algorithm.

The remainder of this paper is organized as follows. In the next section, we describe the classification problem for decision tree. In Section 3, we define the optimization problem for decision tree. Our new algorithm iOVFDT is presented in Section 4. Moreover, we provide the experimental comparison and discussion in Section 5. Finally, Section 6 concludes this paper.

2 Optimization Problem for Decision Tree

Suppose D is the full set of data samples 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 j is the range of attribute X_i, $|X_i| = j$ and $1 \leq i \leq d$. Class y_k is the k class in y and is assigned a value of $y_1, y_2\ldots y_k$, where K is the total number of discrete classes. The classification problem for decision tree is defined as follows: construct a decision tree classifier $DT(X)$ so as to satisfy a classifying goal $DT(X) \leftarrow \hat{y}$, which uses the attribute vector X to provide a predicted class $\hat{y}$. The tree induction builds a tree model from a set of alternatives, minimizing the error between predicted class and actual class (1).

$$\text{minimize} \sum_{k=1}^{K} |Error_k\ |, where\ Error_k\ = \begin{cases} 1, & if\ \widehat{y_k} \neq\ y_k \\ 0, & otherwise \end{cases} \quad (1)$$

The static decision tree learning, i.e. ID3, C4.5, CART, etc., looks for an attribute with the best value of heuristic function *H(.)* as splitting-attribute by greedy search. Post-pruning mechanism removes noisy branches so as to minimize an error-based cost function after full tree built. Hence, it improves accuracy by reducing tree size. The constructed tree model *DT* searches a global optimal solution from the entire dataset *D* so far timestamp *t*, where $D = \sum_1^t D_i$. When new data D_{t+1} comes at timestamp *t+1*, it re-computes on full data D' to update *DT,* where $D' = \sum_1^{t+1} D_i$.

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

where *R* is the range of classes distribution and *n* is the number of instances which have fallen into a leaf, δ is the confidence to support this evaluation. Different from static tree learning, incremental decision tree learning operates continuously arrival data $D_1, D_2, \ldots, D_t$. In the t^{th} splitting test, it only scans the newly received data D_t and update the sufficient statistics by Hoeffding bound (*HB)* in (2). Hence incremental decision tree is also called Hoeffding tree (*HT)*. Let X_{ja} be the attribute X_j with the highest value of *H(.)*, X_{jb} be the attribute with the second-highest *H(.)*. $\Delta H = H(X_{ja}) - H(X_{jb})$ is the difference between the two top quality attributes. If $\Delta H > HB$ with *n* samples observed in leaf, while the *HB* states with probability *1-δ*, that X_{ja} is the attribute with highest value in *H(.)*, then the leaf is converted into a decision node which splits on X_{ja}.

The constructed tree is a local optimum that satisfies the data *D* at timestamp *t*. Said it a local optimum because we never know what are the new arrival data at timestamp *t+1*, even if they contains imperfect values. Let p_l be the probability that an example that reaches level *l* in a decision tree falls into a leaf at that level. If HT and DT use the same heuristic function for node-splitting evaluation, the possibility that the $HT_\delta \neq DT$ is not greater than δ/p, where $E(HT \neq DT) \leq \delta/p$ [4]. Therefore, we can know that: for the full data *D,* where $D = \sum_1^t D_i$ at timestamp *t*, an incremental tree HT_t uses the same *H*(.) with *DT* that tree-branches of HT_t should be a subset of *DT* that $HT_t \subset DT$ with probability δ/p at least.

3 Incrementally Optimized Very Fast Decision Tree (iOVFDT)

3.1 Metrics

Here section will provide iOVFDT in detailed. 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 $\Phi: HT \rightarrow \mathbb{R}$ 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:

$$\Phi(HT_x) = \bigcup_{D=1}^{M} \Phi_D(HT_x) \tag{3}$$

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 is given in (4):

$$minimize\ \Phi(HT_x)\ subject\ to\ HT_x \in X \tag{4}$$

iOVFDT uses $HT(X) \to \hat{y}$ to predict the class when a new data sample (X, y) arrives. So far timestamp t, the prediction accuracy $accu_t$ defined as:

$$accu_t = \frac{\sum_{i=1}^{t} Predict(D_i)}{|D_t|} \tag{5}$$

$$Predict(D_i) = \begin{cases} 1, if\ \widehat{y_k} = y_k \\ 0, if\ \widehat{y_k} \neq y_k \end{cases} \tag{6}$$

To measure the utility of the three dimensions via the minimizing function in (4), the measure of prediction accuracy is reflected by the prediction error in (7):

$$\Phi_1 = erro_t = 1 - accu_t \tag{7}$$

iOVFDT is a new methodology for 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 $\Phi(HT_x)$ in (3), 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 can be defined as:

$$\Phi_2 = size_t = \begin{cases} size_{t-1} + 1 & , if\ \Delta\bar{H} > HB \\ size_{t-1} & , otherwise \end{cases} \tag{8}$$

iOVFDT 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. n_{min} is a fixed value for controlling interval time checking node splitting.

$$\Phi_3 = time_t = R \times (n_{y_k} - n_{min}) \tag{9}$$

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 in (9) – 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 shown in Figure 1. A utility function computes the area of this triangle, reflecting a relationship amongst the three objects in (10):

$$\Phi(HT_x) = \frac{\sqrt{3}}{4} \cdot (Erro_x \cdot Size_x + Erro_x \cdot Time_x + Size_x \cdot Time_x) \quad (10)$$

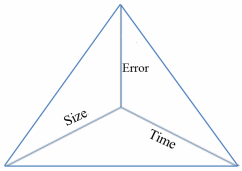

Fig. 1. A multiple objectives optimization model

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 $Mean.\Phi(HT_x)$:

$$Mean.\Phi(HT_x) = \frac{\text{Max.}\Phi(HT_x) - Min.\Phi(HT_x)}{2} \quad (11)$$

According to the Chernoff bound [8], we know:

$$|\text{Opt.}\,\Phi(HT_x{}^*) - Mean.\Phi(HT_x)| \leq \sqrt{\frac{\ln(1/\delta)}{2n}} \quad (12)$$

where the range of $\Phi_x(HT_x)$ is within the min-max model and $\text{Min.}\,\Phi(HT_x) < \text{Opt.}\,\Phi(HT_x{}^*) < \text{Max.}\,\Phi(HT_x)$. 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 adaptively optimized in iOVFDT such that: $\Delta\bar{H} > HB$ or $\text{Opt.}\,\Phi(HT_x{}^*) > Max.\Phi(HT_x)$ or $\text{Opt.}\,\Phi(HT_x{}^*) < Min.\Phi(HT_x)$,

3.2 Functional Tree Leaf Integration

Functional tree leaf [9], 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 $\text{Opt.}\,\Phi(HT_x{}^*) > Max.\Phi(HT_x)$ or $\text{Opt.}\,\Phi(HT_x{}^*) < Min.\Phi(HT_x)$) 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 to predict the class label $\widehat{y_k}$ with functional tree leaf when a new sample (X, y) arrives, defined as $HT(X) \rightarrow \widehat{y_k}$. 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 $V_{F^{\mathrm{MC}}}$ is the OCD with the majority class F^{MC}; suppose $V_{F^{\mathrm{NB}}}$ is the OCD with the naïve Bayes F^{NB} and suppose that $V_{F^{\mathrm{WNB}}}$ 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 $F = \{E_{F^{\mathrm{MC}}}, E_{F^{\mathrm{NB}}}, E_{F^{\mathrm{WNB}}}\}$.

3.3 Tree-Building Process

In this section, a pseudo code summaries the process of tree-growth presented in previous parts. When new data stream comes, it will be sorted by current HT and given a predictive class label. The OCD, which is stored on those pass-by internal nodes, is updated. Comparing the predicted class to the actual class, the prediction error is updated (Line 1 – 5). If the number of samples seen so far is greater than the pre-defined interval number, the node-splitting evaluation should be performed (Line 7 – 19). If node-splitting condition that the difference of best two values of heuristic function *H(.)* is greater than *HB*, or the value of optimization function is out of a min-max range, the attribute with the highest *H(.)* value should split to a new leaf (Line 12 – 17). Meanwhile, new model size and learning time are updated by (8) and (9).

Input:
  D_t: a data stream (X,y) arriving at timestamp t;
  H(.): the heuristic function for splitting test;
  $\mathcal{F}$: a strategy of functional tree leaf;
  n_{min}: the minimum interval between node-splitting tests;
  δ: a desired probability for Hoeffding bound;
Output: Incremental decision tree HT
1. A data stream $D_t = (X, y)$ arrives;
2. If HT isn't initialized, let HT be a tree with a single leaf
   l (the root);
3. Sort D_t from the root to a leaf by HT, using • to give a
   predicted class $\widehat{y_k} \leftarrow HT(X)$;
4. Update OCD on each pass-by node;
5. Compare predicted class $\widehat{y_k}$ to actual class y_k, and update
   error in (7);
6. Let n_k be the number of instances seen at the leaf with class
   Y_k.
7. If all instances seen so far at leaf k don't belong to the
   same class, and $(n_k \bmod n_{min} = 0)$ {
8. Update the learning time in (9);
9. Let X_a and X_b the attributes with highest and the 2nd
    highest heuristic function H(.).
10. Let $\Delta H = H(X_a) - H(X_b)$;
11. Compute HB and update $\Phi(HT_t)$ in (3);
12. If $(\Delta H > HB, or\ \Phi(HT_t) > Max.\Phi(HT), or\ \Phi(HT_t) < Min.\Phi(HT)$ {
13. Replace leaf k by a node splits on X_a;
14. Update the tree size in (8);
15. If $\Phi(HT_t) > Max.\Phi(HT)$ then $Max.\Phi(HT) = \Phi(HT_t)$
16. If $\Phi(HT_t) < Min.\Phi(HT)$ then $Min.\Phi(HT) = \Phi(HT_t)$
17. } End-if;
18. Reset OCD on the new leaf;
19. } End-if;
20. Return HT_t

4 Experiment

4.1 Setup

In this section, we compared C4.5, MVFDT to iOVFDT tree inductions in order to show the difference between global (C4.5) and local (MVFDT, iOVFDT) search optimal tree models. The heuristic function for splitting attribute is information gain. The experimental platform was built on Java. For C4.5, WEKA [10] tree classification J48 was used, both un-pruning and pruning mechanisms; for MVFDT, there are loose and strict pruning mechanisms, both of which have been verified to outperform VFDT described in [7]; for iOVFDT, it was programmed and integrated with MOA [11]. Majority Class (MC), Naïve Bayes (NB), Weighted Naïve Bayes (WNB) and Hybrid Adaptive (ADP) functional tree leaves are integrated in iOVFDT.

The running environment was a Windows 7 PC with an Intel 2.8GHz CPU and 8G RAM. Besides, un-pruned and pruned C4.5 algorithms were applied in this experiment so as to analyze the tree size. The heuristic evaluation of node-splitting was information gain in both methods.

4.2 Datasets

The datasets, including discrete, continuous and mixed attributes, were either synthetically generated by MOA generator, or downloaded from UCI repository.

Table 1. The description of experimental datasets

Data Name	#Nominal	#Numeric	#Class	Source	Size
LED24	24	0	10	Synthetic	10^6
Waveform	0	21	3	Synthetic	10^6
Cover Type	42	12	7	UCI	581,012

Synthetic Data *LED data* was generated by MOA. We added 10% noisy data to simulate imperfect data streams. The LED24 problem used 24 binary attributes to classify 10 different classes. *Waveform* was generated by the MOA generator. The goal of this task was to differentiate between three different classes of Waveform. It had 21 numeric attributes contained noise. **UCI Data** *Cover Type* was used to predict forest cover types from cartographic variables. It is a typical imbalanced class distribution data that all are real life samples.

4.3 Result Analysis

The performance measurements were evaluated in three aspects: accuracy, tree size and learning speed. The measurement of accuracy was 10 folds cross-validation. The number of leaves in the tree mode computed tree size. Learning speed was reflected by the time taken to build decision tree.

Discrete Data. Obviously, un-pruned C4.5 resulted lowest accuracy, biggest tree size and slowest speed (Figure 2). Hence un-pruned C4.5 had the worst performance in this test. Compared with iOVFDT, pruned C4.5 had better accuracy for small data

size. But when the data size grew, iOVFDT outperformed pruned C4.5, because of the smaller tree size and the faster speed. iOVFDT with Hybrid Adaptive function tree leaf obtained the best accuracy but similar speed with the others. Both strict and loose pruning MVFDT had lower accurate than iOVFDT.

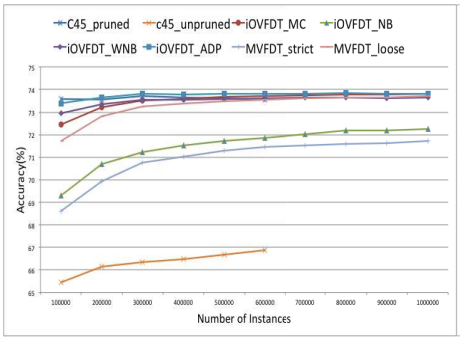

Fig. 2a. Prediction Accuracy for LED24 data

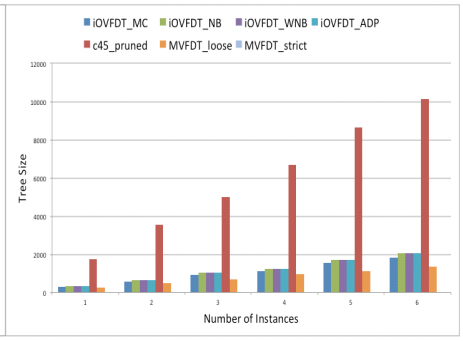

Fig. 2b. Tree size for LED 24

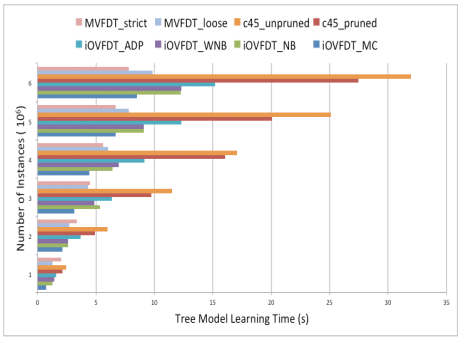

Fig. 2c. Learning time for LED 24

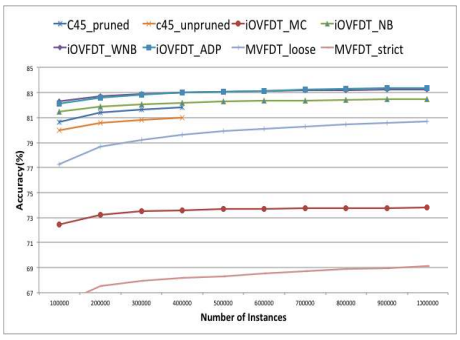

Fig. 3a. Prediction Accuracy for Waveform data

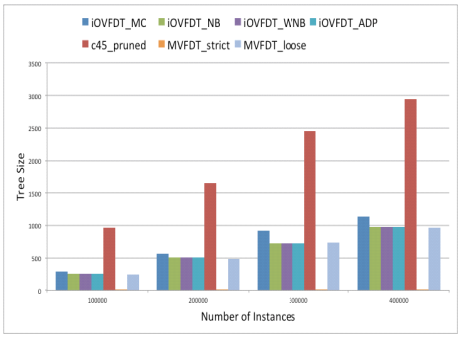

Fig. 3b. Tree size for Waveform

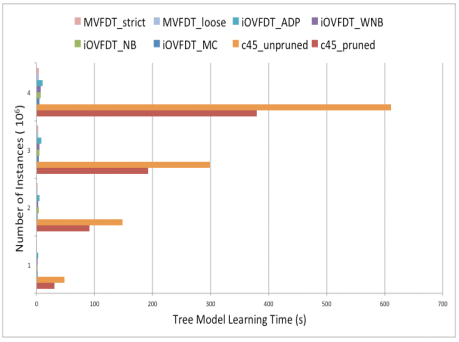

Fig. 3c. Learning time for Waveform

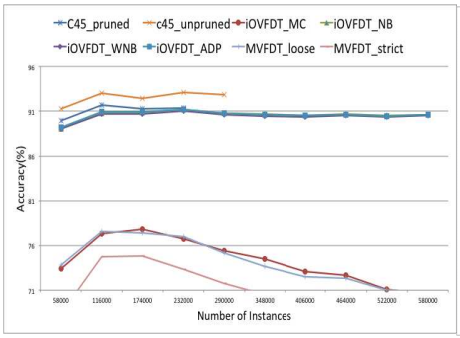

Fig. 4a. Prediction Accuracy for Cover Type data

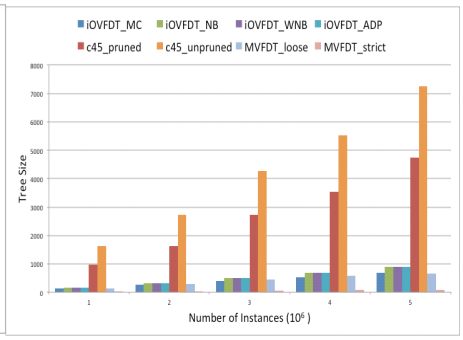

Fig. 4b. Tree size for Cover Type

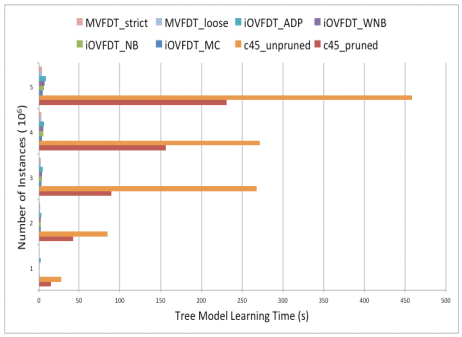

Fig. 4c. Learning time for Cover Type

Continuous Data. iOVFDT$_{MC}$ and MVFDT used majority class in leaf that didn't have the mechanism dealing with numeric data, so the accuracy of them was the worst in this test. For large data size, iOVFDT$_{ADP}$ had the best accuracy and smaller tree than the others. The speed of C4.5 was dramatically longer than iOVFDT (Figure 3).

Mixed Data. Although C4.5 had a little higher accuracy than iOVFDT, it took much longer learning time. iOVFDT$_{ADP}$ had similar accurate with pruned C4.5 but smaller model size. iOVFDT$_{MC}$ had worse performance than other iOVFDTs, and its accurate declined with more data arrived. In addition, the result shows iOVFDT had an obviously better accurate than MVFDT (Figure 4).

4.4 Discussion

For noise-included data, we designed an experiment to show the effects for the different decision trees. 10% noisy data were inserted into LED24 dataset, alternatively, at every other ten thousand instances. Full data had one million instances. C4.5 loaded the full data for training the decision tree and obtained a global accuracy around 91%. iOVFDT loaded the data incrementally, and trained the decision tree progressively. As it can be seen from Figure 5, the local accuracy of iOVFDT was affected by the noisy data. The accuracy dropped whenever noise appeared at the section of data, and it bounced back as soon as the next section of noise-free data emerged. The fluctuation continued but gradually diminished over the

staggering input data with a damping effect. As it was shown in Figure 6, the performance curve of iOVFDT was converging when the total data volume increased. In order to verify the steady-state of the iOVFDT model without the need of using an exceedingly large amount training data, a fitting curve was derived from the existing iOVFDT curve by using Single Moving Average algorithm. The forecasted performance was estimated to be 88.05% from the fitted curve, with the reliability of mean absolute percentage error being 0.98%, Thiel's U value at 0.5118 (which is far better than random guess) and Durbin-Watson value at 1.21 that supported the fact of autocorrelation exhibited by the fitted curve.

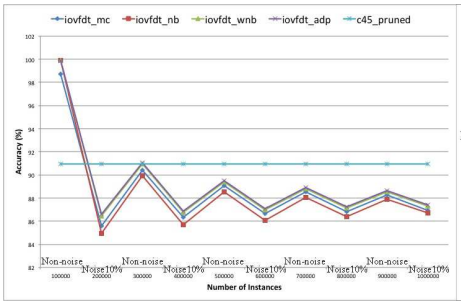

Fig. 5. Performance of global (C4.5) and local (iOVFDT) optimal trees for noise-included data

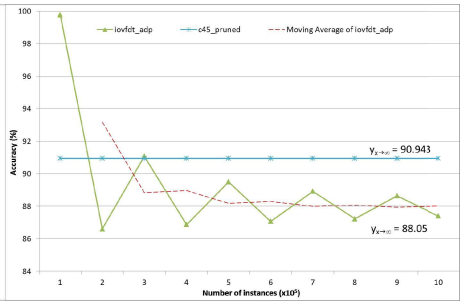

Fig. 6. Longitudinal accuracy of global (C4.5) and local (iOVFDT) optimal trees

In other words, when the training data size approaches infinity the expected accuracy of the iOVFDT model should be 88.05% as implied by the fitted curve. This observation essentially meant that iOVFDT could achieve an accuracy of 88.05% in a long run; and it would do so by incrementally updating its decision tree while new input data continuously streamed in. In contrast, C4.5 required first accepting all the data for learning a decision tree for obtaining a globally optimal accuracy value. The performance difference of 2.893% gained by C4.5 was done at the cost of loading in the full dataset that sometimes may be impractical. Our experiment in this case has shown that the accuracy of iOVFDT that was affected by a series of local optimums was still reasonably substantial in long run. And iOVFDT outperformed C4.5 by the merits of compact tree size, fast learning process and the incremental ability to handle infinite data streams.

5 Conclusion

In this paper, we propose a new incremental decision tree learning so called incrementally Optimized Very Fast Decision Tree (iOVFDT). Different from static tree induction, iOVFDT updates the decision model without scanning the full data in infinite data streams scenario. An incremental optimization, which considers prediction accuracy, model size and learning time, is also used for node-splitting constraint in tree growing. In addition, four types of functional tree leaf are also embedded to improve the prediction accuracy.

In our experiment, the comparison of static decision tree and incremental decision tree is presented, using the same heuristic function for splitting test. The experimental

result compares iOVFDT to C4.5 and MVFDT, and we found that: iOVFDT has significantly smaller tree size and faster learning speed than C4.5. It also has higher accuracy than MVFDT in terms of the three-object incremental optimization mechanism. In the second experiment, the dataset is partitioned by interleaving segments of perfect and noisy data, so as to simulate existence of local optimums and to test the difference between global and local optimal decision trees. In general, global optimal decision tree, which was constructed by loading full data to find out a global optimal model in C4.5 tree induction, was able to obtain a high accuracy for small scale data. Because of multi-scanning over the full dataset, the size of decision tree model was very large even pruning was applied and the entire process was relatively slow. Local optimal decision tree, which was built by iOVFDT incrementally, was demonstrated to be applicable for large or, even infinite datasets. The proposed functional tree leaf of Hybrid Adaptive had the best performance in synthetic data. Compact tree size and short learning time make incremental model practical in real-time applications. The accuracy though may not be the highest, is on par with C4.5. By sacrificing slight accuracy, iOVFDT can be effectively used to handle infinite streams as well as to build an optimized tree within a short time.

Reference

1. Quinlan, R.: Induction of Decision Trees. Machine Learning 1(1), 81–106 (1986)
2. Quinlan, R.: C4.5: Programs for Machine Learning. MorganKaufmann, San Francisco (1993)
3. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth & Brooks/Cole Advanced Books & Software, Monterey (1984)
4. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proc of 6th ACM SIGKDD, pp. 71–80 (2000)
5. Elomaa, T.: The Biases of Decision Tree Pruning Strategies. In: Hand, D.J., Kok, J.N., Berthold, M. (eds.) IDA 1999. LNCS, vol. 1642, pp. 63–74. Springer, Heidelberg (1999)
6. Bradford, J., Kunz, C., Kohavi, R., Brunk, C., Brodley, C.: Pruning Decision Trees with Misclassification Costs. In: Nédellec, C., Rouveirol, C. (eds.) ECML 1998. LNCS, vol. 1398, pp. 131–136. Springer, Heidelberg (1998)
7. Yang, H., Fong, S.: Moderated VFDT in Stream Mining Using Adaptive Tie Threshold and Incremental Pruning. In: Cuzzocrea, A., Dayal, U. (eds.) DaWaK 2011. LNCS, vol. 6862, pp. 471–483. Springer, Heidelberg (2011)
8. Chernoff, H.: A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations. Annals of Mathematical Statistics 23, 493–507 (1952)
9. Gama, J., Ricardo, R.: Accurate decision trees for mining high-speed data streams. In: Proc. of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 523–528. ACM (2003)
10. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation. Morgan Kaufmann, San Francisco (2000)
11. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive Online Analysis. Journal of Machine Learning Research 11, 1601–1604 (2000)
12. Geoffrey, H., Richard, K., Bernhard, P.: Tie Breaking in Hoeffding trees. In: Gama, J., Aguilar-Ruiz, J.S. (eds.) Proceeding Workshop W6: Second International Workshop on Knowledge Discovery in Data Streams, pp. 107–116 (2005)
13. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proc. of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, pp. 97–106 (2001)