Speeding up Very Fast Decision Tree with Low Computational Cost

Jian Sun[1], **Hongyu Jia**[1], **Bo Hu**[2], **Xiao Huang**[1], **Hao Zhang**[1], **Hai Wan**[1 *], **Xibin Zhao**[1]

[1] KLISS, BNRist, School of Software, Tsinghua University, China
[2] Beijing University of Posts and Telecommunications, China

{sunj17, jiahy19}@mails.tsinghua.edu.cn, huboniccolo@bupt.edu.cn,
{huangx19, hao-zhan17}@mails.tsinghua.edu.cn, {wanhai, zxb}@tsinghua.edu.cn

Abstract

Very Fast Decision Tree (VFDT) is one of the most widely used online decision tree induction algorithms, and it provides high classification accuracy with theoretical guarantees. In VFDT, the split-attempt operation is essential for leaf-split. It is computation-intensive since it computes the heuristic measure of all attributes of a leaf. To reduce split-attempts, VFDT tries to split at constant intervals (for example, every 200 examples). However, this mechanism introduces split-delay for split can only happen at fixed intervals, which slows down the growth of VFDT and finally lowers accuracy. To address this problem, we first devise an online incremental algorithm that computes the heuristic measure of an attribute with a much lower computational cost. Then a subset of attributes is carefully selected to find a potential split timing using this algorithm. A split-attempt will be carried out once the timing is verified. By the whole process, computational cost and split-delay are lowered significantly. Comprehensive experiments are conducted using multiple synthetic and real datasets. Compared with state-of-the-art algorithms, our method reduces split-attempts by about 5 to 10 times on average with much lower split-delay, which makes our algorithm run faster and more accurate.

1 Introduction

In recent years, the amount of data generated by industrial areas is growing significantly. [Dobre and Xhafa, 2014] pointed out that approximately 25 exabytes of data are generated every day. Traditional offline methods have to store all the data first, then apply specific algorithms to train models. However, if large amounts of data arrive continuously, due to the limited computing resources, the memory becomes a bottleneck. Training offline models after storing all data becomes increasingly impractical. Therefore, it is particularly important to design online algorithms to process data in real-time.

Over the past few decades, decision tree algorithms have been deeply studied and applied in various fields due to its high efficiency, accuracy, and interpretability. Traditional decision tree algorithms (ID3 [Quinlan, 1986], CART [Breiman *et al.*, 1984], C4.5 [Quinlan, 1993], etc.) are not suitable for streaming environments. When a tree grows, the best attribute is first selected based on all the data, and then apply it to guide the tree split for learning more information. However, it is impossible to store all data of a stream, thus decision trees need to determine how much data should be accumulated to select the right attribute. VFDT [Domingos and Hulten, 2000] and its variants (EFDT [Manapragada *et al.*, 2018], VFDTc [Gama *et al.*, 2003], etc.) are currently the most popular online decision tree methods, using Hoeffding bound as the theoretical guarantee to assume that a small number of samples are sufficient to select the best split attribute. By setting the parameter δ in Hoeffding bound, VFDT can ensure that the attribute selected during splitting is the best attribute with $1 - \delta$ confidence.

In specific implementation details, since the volume of data needed to be accumulated cannot be predicted in advance, a naive approach is to perform a split-attempt of a leaf for every newly arrived example to check whether the best attribute selected from the current data satisfies the Hoeffing bound. This approach results in high computational cost, as each split-attempt needs to calculate heuristic measure functions (i.e., information gain [Quinlan, 1993] or Gini index [Breiman *et al.*, 1984]) of all attributes. A large number of algorithms based on VFDT [Hulten *et al.*, 2001; Gama *et al.*, 2003; Bifet and Gavaldà, 2009; Ikonomovska *et al.*, 2011a; Ikonomovska *et al.*, 2011b] use the same approach to alleviate this problem by setting a hyperparameter n_{min} and performs split-attempt every n_{min} examples, therefore reducing split-attempts and shortening the running time. However, when n_{min} is too large, the period of split-attempt is prolonged and it is more likely to miss the optimal splitting time, thereby inducing long split-delay which makes the tree grow slower and reduce accuracy. Conversely, small n_{min} will increase runtime.

How to make the leaf split reasonably and fast without affecting the operational efficiency of the algorithm is the focus of this paper. Two main works of this paper are as follows:

1. A mechanism is proposed to incrementally update the heuristic measure of an attribute in constant time. By this mechanism, the heuristic measure can be updated immediately every time a single example arrives, instead

*Corresponding Author

of being calculated periodically like VFDT.

2. We introduce the candidate attribute set which keeps track of the top K attributes that are most likely to split. Only the heuristic measures of attributes in the set are updated continuously, which further reduces the computational cost. To cover a wider range of attributes and make the set indeed save the true top K attributes, we propose an adaptive candidate set updating mechanism that dynamically changes candidate attributes when their performance deteriorates.

We evaluate and verify the algorithm on multiple synthetic and real datasets. Compared with the existing methods, our method can effectively reduce split-attempts by 5 to 10 times on average and even hundreds of times in some data streams, which greatly reduces the computational cost. What's more, we shorten split-delay more stably in both stationary and non-stationary environments. These make our algorithm run faster and more accurate.

The rest of this paper is organized as follows: Section 2 introduces the related work. We describe our algorithm and the corresponding theoretical derivation in Section 3. Section 4 shows the experimental results and analysis. Finally, we conclude this paper in Section 5.

2 Related Work

Traditional decision tree algorithms are based on batch data, but [Domingos and Hulten, 2000] proposes VFDT for establishing decision trees using stream data based on Hoeffding bound [Hoeffding, 1994]. Suppose we have n independent observations of real-valued random variable r with range R and mean $\bar{r}$. The Hoeffding bound states that the true mean of the variable r is at least $\bar{r} - \epsilon$ with probability $1 - \delta$, where,

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

Denote $G(X_i)$ as the heuristic measure of attribute X_i at the leaf node. Assume that after observing n pieces of data, X_a and X_b are the attributes with highest and second-highest G. Let $\Delta G = G(X_a) - G(X_b)$ and if $\Delta G > \epsilon$, then the attribute X_a has a probability of $1 - \delta$ as the best attribute of the current leaf node. Since the calculation of ΔG needs to be performed on all attributes, VFDT uses the hyperparameter n_{min} to periodically check if $\Delta G > \epsilon$ satisfied. The introduction of n_{min} effectively improves the runtime of VFDT. But fixed n_{min} will also delay the split and affect the performance of VFDT.

In recent studies, more and more people focus on accelerating the learning of VFDT, because the Hoeffding bound is still a conservative measure [Das *et al.*, 2019]. [Manapragada *et al.*, 2018] proposed EFDT (Extremely Fast Decision Tree) which uses a more loose judgment to speed up the splitting of leaves. If $\Delta G = G(X_a) - G(X_\phi) > \epsilon$, which means splitting on the best attribute works better than not splitting, then EFDT will split attribute X_a at the leaf node. However, this method requires a periodic check because the best attributes are prone to change. To accelerate the learning process, Mem-ES [Das *et al.*, 2019] abandons the theoretical guarantee of Hoeffding's inequality. Based on the principle of Bag of Little Bootstraps [Kleiner *et al.*, 2012], Mem-ES employs a sampling method to speed up splitting. But this method retains some examples at leaves, which lead to large memory consumption.

Scholars have begun to pay attention to the hyperparameter n_{min} in the past few years, [García-Martín, 2017] assumes that the ΔG value calculated for the first time does not change. As ϵ in Equation 1 decreases with the number of samples n increasing, it estimates that at least how many samples are required to be observed in order to make ϵ small enough to satisfy the inequality $\Delta G > \epsilon$. [Losing *et al.*, 2018] proposes the OSM algorithm, which assumes that $G(X_b)$, the heuristic measure of the second-best attribute, does not change, the value of $G(X_a)$ increases as much as possible as the data arrives. In this case, OSM estimates how many samples are needed to satisfy the Hoeffding bound, such that $G(X_a) - G(X_b) > \epsilon$. Obviously, the value of ΔG or $G(X_b)$ actually changes dynamically. Although these approaches may effectively reduce split-attempts, they will increase the split-delay and have a negative impact on the accuracy of the algorithm.

Compared with previous algorithms, the algorithm proposed in this paper can use both the historical and the latest data distribution information, generate fewer split-attempts and less split-delay, it can also improve the accuracy to a certain extent. Besides, our algorithm can be well embedded in the above algorithms such as VFDT, EFDT and so on.

Algorithm 1 Online Decision Tree Induction

Input: S: A sequence of examples; G: A heuristic measure function; $\mathbb{X}$: A set of attributes; δ: One minus the desired probability of choosing the correct attribute at any given node; τ: The threshold of tie-split; n_{min}: The period of split-attempt.
Output: HT: Online decision tree learned from S

1: Initialize HT with a single root node
2: Initialize the statistics for tree growth
3: **for** each example s in S **do**
4:     Sort s into a leaf l using HT
5:     Update the statistics at l for tree growth
6:     **if** the weight of examples at l mod n_{min} equals 0 **then**
7:         *AttemptToSplit(l, G, $\mathbb{X}$, δ, τ)*
8:     **end if**
9: **end for**
10: **return** HT

3 Methodology

3.1 Preliminary

Algorithm 1 describes an overview of VFDT. VFDT is first initialized by a single root node (line: 1). For each example s in the data stream S, VFDT finds the leaf node l that the example s falls into, and then updates the statistics of attributes in l (lines: 3-5). l will not attempt to split until the weight of examples observed in l is an integer multiple of n_{min}.

Function 2 describes the process of split-attempt operation. The first step is to calculate the heuristic measure of all at-

Function 2 AttemptToSplit(l, G, $\mathbb{X}$, δ, τ)

1: X_a, X_b are the two attributes with highest G_l
2: Compute ϵ using equation 1
3: **if** $G(X_a) - G(X_b) > \epsilon$ or $\epsilon < \tau$ **then**
4:     Replace l with an internal node
5:     **for** each branch of the split **do**
6:         Add a new leaf l_m and let $\mathbb{X}_m = \mathbb{X} - \{X_a\}$
7:     **end for**
8: **end if**

tributes, and then find the best attribute X_a and the second-best attribute X_b (line: 1). The Hoeffding bound is calculated according to Equation 1. If $G(X_a) - G(X_b) > \epsilon$ or $\epsilon < \tau$, the attribute X_a will be used to split the leaf node (lines: 3-8). When two attributes have very similar G values, if we only use the condition $G(X_a) - G(X_b) > \epsilon$, potentially many examples will be required to split a leaf. VFDT uses condition $\epsilon < \tau$ to determine whether the tie situation occurs. It uses the current best attribute to split when the tie situation does occur. The split in this situation is called tie-split. The parameter τ essentially specifies the maximum number of examples that can be accumulated at a leaf.

3.2 Incremental Heuristic Measure

[Sovdat, 2014] proposes the updating formulas of computing entropy and Gini index for time-changing data streams, and we further extend it to the heuristic measure calculation of attributes. Assuming that the current node can be split into q child nodes using attribute X_i. The data set in the current node is D, and the number of classes is c, then the following symbols can be defined:

- n: the total weight of examples in D;
- $n_{i,j}$: the total weight of examples that would be passed into the jth child node if the split was made by the attribute X_i;
- $n_{i,j,k}$: the total weight of examples from the kth class that would be passed into the jth child node if the split was made by the attribute X_i.

For the clarity of the explanation, the index i will be neglected, i.e., $n_{i,j} \equiv n_j, n_{i,j,k} \equiv n_{j,k}$.

If VFDT uses information gain as the heuristic measure, the conditional entropy on the attribute X_i is:

$$G(D|X_i) = -\frac{1}{n}\sum_{j=1}^{q}\sum_{k=1}^{c} n_{j,k}\log\frac{n_{j,k}}{n_j} \tag{2}$$

Theorem 1. *For any given example s with weight w and class m ($m \in [1, c]$), suppose s will be passed into the rth child node if the split was made by the attribute X_i, then the new conditional entropy of the attribute X_i is $\hat{G}(D|X_i) = \frac{1}{n+w}(nG(D|X_i) - \Delta)$, where $\Delta = \log[(\frac{n_r}{n_r+w})^{n_r}(\frac{n_{r,m}+w}{n_{r,m}})^{n_{r,m}}(\frac{n_{r,m}+w}{n_r+w})^{w}]$.*

Proof. According to the assumptions of Theorem 1, both n_r and $n_{r,m}$ will increase w, and the values of other parameters remain unchanged. Then

$$\begin{aligned}
&\hat{G}(D|X_i) - \frac{n}{n+w}G(D|X_i)\\
&= -\frac{1}{n+w}(n_{r,m}\log\frac{n_{r,m}+w}{n_{r,m}} + w\log\frac{n_{r,m}+w}{n_r+w})\\
&- \frac{1}{n+w}\sum_{k=1}^{c} n_{r,k}\log\frac{n_r}{n_r+w}\\
&= -\frac{1}{n+w}\log[(\frac{n_r}{n_r+w})^{n_r}(\frac{n_{r,m}+w}{n_{r,m}})^{n_{r,m}}(\frac{n_{r,m}+w}{n_r+w})^{w}]
\end{aligned} \tag{3}$$

This completes the demonstration of Theorem 1. □

When VFDT uses Gini index as the heuristic measure, the term in which does not depend on the attribute, will be neglected since the subject of interest is the difference of the Gini index for two attributes. Thus the approximate Gini index of the attribute X_i is:

$$G(D|X_i) = \frac{1}{n}\sum_{j=1}^{q}\sum_{k=1}^{c}\frac{n_{j,k}^2}{n_j}. \tag{4}$$

We can prove Theorem 2 in the same way as theorem 1.

Theorem 2. *For any given example s with weight w and class m ($m \in [1, c]$), suppose s will be passed into the rth child node if the split was made by the attribute X_i, then the new Gini index of the attribute X_i is $\hat{G}(D|X_i) = \frac{1}{n+w}(nG(D|X_i) + \Delta)$, where $\Delta = \frac{2wn_{r,m}+w^2}{n_r+w} - \sum_{k=1}^{c}\frac{w(n_{r,k})^2}{n_r(n_r+w)}$.*

3.3 Incremental Measure Algorithm Based on Candidate Attributes (IMAC)

According to Theorem 1 and Theorem 2, when the heuristic measure is information gain or Gini index, it can be incrementally calculated respectively. Based on this, an incremental algorithm is proposed, which we call IMAC. The algorithm is described in Algorithm 3 and Function 4.

IMAC removes the parameter n_{min} and uses a new parameter η instead, which represents the minimum total weight of examples that a leaf node needs to accumulate. The parameter η is used only once throughout the life cycle of the leaf node, it has the effect of "cold start".

IMAC maintains a candidate attribute set $\mathbb{P}$ in each leaf node, which stores the top K attributes that are most likely to split. In our algorithm, K is equal to 10% of the number of attributes. To avoid too many or too few candidate attributes are selected, the value of K is limited between 5 to 10. As long as $\mathbb{P}$ is not empty, the heuristic measure of the current candidate attributes will be incrementally updated according to Theorem 1 or 2 (lines: 6-8). When the difference between the heuristic measure of the best attribute and the second-best attribute in $\mathbb{P}$ is greater than the Hoeffding bound, the split-attempt operation will be executed at the leaf (lines: 10-13). In function 4, if the split-attempt operation fails, IMAC will replace $\mathbb{P}$ with new top K attributes according to the rank of current values of the heuristic measure (line: 9).

Algorithm 3 Online Decision Tree Induction with IMAC

Input: S: A sequence of examples; G: A heuristic measure function; $\mathbb{X}$: A set of attributes; δ: One minus the desired probability of choosing the correct attribute at any given code; τ: The threshold of tie-split; η: The minimum total weight of examples must be accumulated at a leaf; K: The maximum size of the candidate attribute set.
Output: HT: Online decision tree learned from S

```
Initialize HT with a single root node
Initialize the statistics for tree growth and an empty candidate attribute set ℙ
for each example s in S do
    Sort s into a leaf l using HT
    Update the statistics at l for tree growth
    if candidate attribute set ℙ_l at l is not empty then
        for each attribute X_i ∈ ℙ_l do
            Update the heuristic measure of X_i using Theorem 1 or 2
        end for
        X_a, X_b are the two attributes of ℙ_l with highest G_l
        Compute ϵ using Equation 1
        if G(X_a) − G(X_b) > ϵ or ϵ < τ then
            AttemptToSplitWithIMAC(l, G, 𝕏, δ, τ, K)
        end if
    else if the weight of examples at l is greater than η then
        AttemptToSplitWithIMAC(l, G, 𝕏, δ, τ, K)
    end if
end for
return HT
```

Function 4 AttemptToSplitWithIMAC(l, G, $\mathbb{X}$, δ, τ, K)

```
X_a, X_b are the two attributes with highest G_l
Compute ϵ using Equation 1
if G(X_a) − G(X_b) > ϵ or ϵ < τ then
    Replace l with an internal node
    for each branch of the split do
        Add a new leaf l_m with empty ℙ_m and let 𝕏_m = 𝕏 − {X_a}
    end for
else
    Replace ℙ_l with new top K attributes according to the rank of G_l
end if
```

3.4 Adaptive Switching Mechanism for Candidate Attributes

Theoretically, the result based on theorem 1 or theorem 2 should be accurate. However, the following situations may occur, resulting in an inaccurate result:

- In some improved algorithms based on VFDT, to speed up the processing of numerical attributes, it is assumed that the distribution of the values of a numerical attribute follows a normal distribution, and only the mean and variance of numerical attributes are stored in a leaf node [Gama *et al.*, 2004]. When calculating the heuristic measure of a numerical attribute, the optimal split-point of the attribute needs to be considered additionally. At a given split-point, the approximate weight greater than or less than the split-point is estimated by the normal distribution. While in the incremental method of IMAC, the weight is calculated accurately at a given split-point. The difference between the incremental method and the approximation method on a numerical attribute will lead to a large deviation finally.
- The candidate attribute set $\mathbb{P}$ may not have captured the true split attribute. On the one hand, due to the small amount of data at the beginning, the difference of the heuristic measure between the true best attribute and other attributes is not obvious, and even the true best attribute may be smaller. If $\mathbb{P}$ is initialized at this time point, the true best attribute may not be included. On the other hand, if concept drift occurs, the data distribution can change over time [Schlimmer and Granger, 1986; Widmer and Kubat, 1996]. Thus the best attribute may change at different stages, which is difficult to capture. For numerical attributes, even if the best attribute does not change, the optimal split-point may be different at different stages by calculating the data weight approximately.

To solve these problems, we give the following rules:

Illegal Attribute. Assume X_a is the attribute with highest G. For any other attribute X_i, X_i is said to be illegal if $G(X_a) - G(X_i) > \epsilon$. We believe that there is sufficient evidence to show that X_a is better than X_i, X_i will not be considered anymore and should be added into the illegal attribute set $\mathbb{I}$.

Potential Attribute. Assume X_a is the attribute with highest G. For any other attribute X_i, X_i is said to be potential if $G(X_a) - G(X_i) \leq \epsilon$ and X_i is not in the candidate set $\mathbb{P}$, then X_i should be added into the potential attribute set $\mathbb{M}$.

A hyperparameter μ is used to check the candidate set $\mathbb{P}$ every μ examples. Assume X_p is the worst attribute with lowest G in $\mathbb{P}$, and X_s is the best attribute with highest G in $\mathbb{M}$. Since the attribute information in $\mathbb{M}$ is not updated incrementally in time, the latest value of the heuristic measure of X_s needs to be recalculated. The one with the larger G of the two attributes will be added to $\mathbb{P}$ and the other will be added to $\mathbb{I}$ (according to the rule of illegal attribute) or $\mathbb{M}$ (according to the rule of potential attribute).

Space and Time Complexity Analysis

If c is the number of classes, d is the number of attributes and each nominal attribute can have at most v values, then VFDT maintains the statistics of all attributes at each leaf in $O(dvc)$ memory. Since IMAC introduces $\mathbb{P}$ to save the incremental information of top K candidate attributes, it requires extra $O(Kvc)(K < d)$ memory. $\mathbb{M}$ and $\mathbb{I}$ only store the names and lastest G values of attributes, the memory required is $O(d)$. In conclusion, the total required memory of each leaf in IMAC is still $O(dvc)$.

The time to calculate the heuristic measure of all attributes in VFDT is $O(dvc)$. Assume that the total weight of the data observed at a leaf is n, n_v ($n_v = n/n_{min}$) split-attempts are required on average in VFDT, and n_i split-attempts are re-

Dataset	#Samples	#Feat	#Class	Type	Stationarity
SEA_g	5M	3	2	artificial	No
LED_g	1M	24	10	artificial	No
AGR_g	10M	9	2	artificial	No
RBF	2M	200	15	artificial	Yes
RTG	10M	400	25	artificial	Yes
Covertype	581K	54	7	real	No
KDD99	4.9M	41	23	real	No
MNIST8M	8.1M	784	10	real	Yes

Table 1: Datasets information.

quired in IMAC. Thus the time complexity of split-attempts is $O(n_v dvc)$ in VFDT. Except for split-attempt operation in IMAC, $\mathbb{P}$ needs to be checked periodically in $O(n_c Kvc)$ ($n_c = n/\mu$) time, the information of candidate attributes are updated incrementally in $O(nK)$ time. Thus the time of split-attempt and the time of additional operations introduced in IMAC is $O(n_i dvc + n_c Kvc + nK)$. If $\frac{n_i}{n_v} < 1 - \frac{Kn_{min}}{\mu d} - \frac{Kn_{min}}{dvc}$, IMAC will faster than VFDT. We define a new function $\varphi = 1 - \frac{Kn_{min}}{\mu d} - \frac{Kn_{min}}{dvc}$. φ is an increasing function with d, v and c as arguments. From this, we know that, in large-scale and complex data flow scenarios, IMAC is very likely to be faster than VFDT. In Section 4, IMAC usually reduces split-attempts by about 10 times compared with VFDT, and sometimes even hundreds of times. In industrial scenarios, data is usually very complex, with hundreds of attributes, and some attributes may have hundreds of different state values. IMAC is more suitable for these complex scenarios.

4 Experiment

4.1 Experimental Setup

Environment

All algorithms and experiments are implemented on the Massive Online Analysis (MOA) platform [Bifet *et al.*, 2010], which is one of the most popular open-source frameworks for data stream mining. Our optimizations can also be integrated on other platforms (i.e., STREAMDM C++ [Bifet *et al.*, 2017], Scikit-Multiflow [Montiel *et al.*, 2018]). Our code is available at GitHub[1]. All experiments are conducted on a standard server with 36 cores and 125GB memory. We evaluate the performance of three methods in this paper:

- The standard VFDT with periodic split-attempts.
- The current best algorithm OSM, which predicts the interval of split-attempts in VFDT and has the best performance optimization effect [Losing *et al.*, 2018].
- The algorithm IMAC in this paper, which determines the potential split timing in VFDT with incremental information.

VFDT with default parameters ($n_{min} = 200$, $\tau = 0.05$, $\delta = 1e-7$), uses the majority class in leaves for classification and information gain as the heuristic measure. Since n_{min} is 200, to compare with VFDT and OSM at the same level, parameter μ and η in IMAC are both set to 200.

Datasets

We use large streams consisting of well known real-world and synthetic datasets. Table 1 shows detailed information.

Synthetic data. Synthetic data (SEA [Street and Kim, 2001], LED [Breiman *et al.*, 1984], AGR [Agrawal *et al.*, 1993], RTG [Domingos and Hulten, 2000], RBF) are all generated using the API proposed by MOA. The name's subscript of synthetic data with concept drift is marked with g (i.e., LED_g).

Covertype. The forest covertype data set represents forest cover type for 30 x 30 meter cells obtained from the US Forest Service Region 2 Resource Information System (RIS) data.

KDD99. KDD99 dataset corresponds to a cyber-attack detection problem (i.e., attack or common access), an inherent streaming scenario since instances are sequentially presented as a time series.

MNIST8M. MNIST8M is the augmentation of original MNIST [LeCun *et al.*, 1998] database by using pseudo-random deformations and translations [Loosli *et al.*, 2007].

4.2 Results and Discussion

Split-delay equals to the difference between the actual split-time t_l and the true minimum split-time $\hat{t_l}$ at a leaf l:

$$\Delta_l = t_l - \hat{t_l} \tag{5}$$

The value of $\hat{t_l}$ can be obtained by a reference tree that sets parameter n_{min} to 1 [Losing *et al.*, 2018]. The average split-delay will be evaluated in our experiments. Reducing split-delay usually requires more split-attempts and reducing split-attempts usually introduces a larger split-delay. It is difficult to reduce split-attempts and split-delay at the same time.

A comprehensive evaluation result of the three algorithms lists in table 2. Since n_{min} is 200, the split-delay generated by VFDT is usually around 100 (half of n_{min}), whether it is in stationary or non-stationary streams.

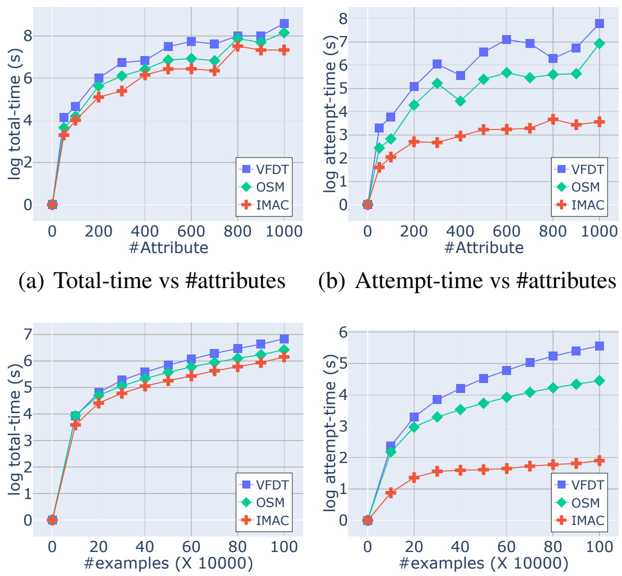
By comparison to VFDT, although OSM usually reduces split-attempts by 2 to 8 times, the performance of split-delay is particularly unstable. OSM can maintain roughly the same split-delay in some streams, but introduces a huge split-delay in others, sometimes even more than 10 times (for example, RBF, Covertype and KDD99). The reason is that OSM predicts the interval of the split-attempt base on the assumption that the current best attribute will not change. This may not be realistic because the best attribute may change drastically especially in non-stationary streams. What's more, a larger split-delay makes the tree grow slower, which usually decreases the accuracy of the tree (for example, the split-delay of AGR_g is beyond 600, results in 1% reduction in accuracy compared with VFDT).

Compared with VFDT, IMAC has a great effect on reducing split-attempts and substantially reduces it by more than 10 times in most cases and even 100 times sometimes. Even compared with OSM, IMAC can reduce split-attempts by more than 4 times normally. Although fewer split-attempts are used, less split-delay is introduced, which makes IMAC

[1]https://github.com/yearsj/IMAC

Datasets	#Attempts			Average delay			Total Time (s)			Accuracy (%)		
	VFDT	OSM	IMAC	VFDT	OSM	IMAC	VFDT	OSM	IMAC	VFDT	OSM	IMAC
SEA$_g$	25436	18172	**3414**	96.95	96.95	**21.74**	6.64	**6.60**	10.07	89.20	89.23	**89.26**
LED$_g$	4999	2652	**186**	113.80	102.87	**6.70**	2.20	**1.86**	2.79	31.05	31.03	**32.09**
AGR$_g$	33229	20496	**9688**	98.62	686.59	**79.53**	**21.01**	21.07	26.19	85.74	84.61	**86.20**
RBF	10008	1649	**105**	121.00	9149.00	**55.00**	76.11	33.97	**31.70**	56.24	55.58	**56.32**
RTG	35949	9565	**2499**	100.34	272.26	**45.84**	285.56	212.47	**151.99**	83.57	83.69	**83.84**
Covertype	2853	1237	**492**	**99.43**	1051.36	392.27	2.83	**2.10**	3.23	69.76	68.66	**70.23**
KDD99	4231	**105**	113	102.50	50553.50	**0.50**	9.80	**8.44**	9.75	99.25	99.03	**99.27**
MNIST8M	40436	11885	**527**	68.40	107.80	**54.20**	638.69	442.83	**316.15**	**58.03**	57.98	57.98

Table 2: A comprehensive evaluation of VFDT, OSM and IMAC, including split-attempts, the average of split-delay, total time and accuracy.



(a) Total-time vs #attributes (b) Attempt-time vs #attributes

(c) Total-time vs #examples (d) Attempt-time vs #examples

Figure 1: As the increase of #attribute and #examples, the variation of total-time and attempt-time on different algorithms in RTG. All time values are processed by logarithmic base 2.

grow faster and more accurate than VFDT and OSM under the same conditions in most circumstances. By calculating of candidate attributes incrementally, IMAC combines the historical distribution and the latest distribution to judge potential split-time, so that the split-delay is more stable. However, Covertype is excluded, whose delay reached 392.27 for $\mathbb{P}$ fails to capture the true best attribute so that the attribute is not considered when judging the potential split-time.

We only evaluate the running time of the online decision tree, excluding the time of reading and parsing data stream. When the size of stream data is very small (in VFDT, the total runtime is less than 30s in this experiment result), OSM is optimal for speeding up running time, while IMAC is slightly slower than VFDT because of additional operations (incremental calculation and candidate attributes switching). Although IMAC can reduce the split-attempts to a small extent, the effect of optimizing this operation is limited since the total time consumption of split-attempts is very small in these streams. Even if IMAC runs a little longer, the extra time is trivial (for example, in KDD99, IMAC is only 1s slower than OSM). While for large data streams, the speedup of the running time of IMAC is best. Compared with the time of additional operations, the time of split-attempts that IMAC reduces has a greater impact. For example, IMAC is 320s faster than VFDT and 120s faster than OSM in MNIST8M. Figure 1 shows the impact of the number of attributes and the number of examples on the optimization of runtime. Attempt-time is the time of split-attempts, and the additional operations used to reduce split-attempts in IMAC are also included. As the increase of attributes or examples, the attempt-time of VFDT and OSM shows a steeper rise, while IMAC rises to a certain level and almost stabilizes. Ideally, to ensure that leaves split in time, the split-delay should be equal to 0. Since IMAC generates less split-delay in most cases, it will make the tree grow a little deeper. The deeper trees will lead to a little longer training time and memory estimate time in IMAC so that the total runtime optimization is not obvious as attempt-time. It is also a reasonable phenomenon caused by accelerated splitting.

5 Conclusion

In this paper, we further improved the performance of VFDT by replacing its periodic split-attempt mechanism. We calculated the heuristic measure incrementally and applied it to determine the potential split-time on candidate attributes. To make the candidate set to cover a wider range, we also proposed a dynamic candidate set switching mechanism. We conducted a comprehensive experiment on multiple synthetic datasets and real datasets. Compared with state-of-the-art algorithms, IMAC can not only reduce split-delay but also significantly reduce split-attempts, which makes IMAC run faster and more accurate. Moreover, we verified that IMAC is more efficient in large-scale data streams.

Acknowledgments

This research is sponsored in part by the NSFC Program (No.U1801263) and the Key-Area Research and Development Program of Guangdong Province (No.2020B010164001).

References

[Agrawal *et al.*, 1993] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Database mining: A performance perspective. *IEEE transactions on knowledge and data engineering*, 5(6):914–925, 1993.

[Bifet and Gavaldà, 2009] Albert Bifet and Ricard Gavaldà. Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*, pages 249–260. Springer, 2009.

[Bifet *et al.*, 2010] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604, 2010.

[Bifet *et al.*, 2017] Albert Bifet, Jiajin Zhang, Wei Fan, Cheng He, Jianfeng Zhang, Jianfeng Qian, Geoff Holmes, and Bernhard Pfahringer. Extremely fast decision tree mining for evolving data streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1733–1742. ACM, 2017.

[Breiman *et al.*, 1984] L Breiman, JH Friedman, R Olshen, and CJ Stone. Classification and regression trees. 1984.

[Das *et al.*, 2019] Ariyam Das, Jin Wang, Sahil M Gandhi, Jae Lee, Wei Wang, and Carlo Zaniolo. Learn smart with less: building better online decision trees with fewer training examples. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2209–2215. AAAI Press, 2019.

[Dobre and Xhafa, 2014] Ciprian Dobre and Fatos Xhafa. Intelligent services for big data science. *Future Generation Computer Systems*, 37:267–281, 2014.

[Domingos and Hulten, 2000] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Kdd*, volume 2, page 4, 2000.

[Gama *et al.*, 2003] João Gama, Ricardo Rocha, and Pedro Medas. Accurate decision trees for mining high-speed data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 523–528. ACM, 2003.

[Gama *et al.*, 2004] Joao Gama, Pedro Medas, and Ricardo Rocha. Forest trees for on-line data. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 632–636. ACM, 2004.

[García-Martín, 2017] Eva García-Martín. *Extraction and energy efficient processing of streaming data*. PhD thesis, Blekinge Tekniska Högskola, 2017.

[Hoeffding, 1994] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The Collected Works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

[Hulten *et al.*, 2001] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM, 2001.

[Ikonomovska *et al.*, 2011a] Elena Ikonomovska, João Gama, and Sašo Džeroski. Learning model trees from evolving data streams. *Data mining and knowledge discovery*, 23(1):128–168, 2011.

[Ikonomovska *et al.*, 2011b] Elena Ikonomovska, Joao Gama, Bernard Zenko, and Saso Dzeroski. Speeding-up hoeffding-based regression trees with options. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 537–544. Citeseer, 2011.

[Kleiner *et al.*, 2012] Ariel Kleiner, Ameet Talwalkar, Purnamrita Sarkar, and Michael I Jordan. The big data bootstrap. 2012.

[LeCun *et al.*, 1998] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Loosli *et al.*, 2007] Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. *Large scale kernel machines*, 2, 2007.

[Losing *et al.*, 2018] Viktor Losing, Heiko Wersing, and Barbara Hammer. Enhancing very fast decision trees with local split-time predictions. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 287–296. IEEE, 2018.

[Manapragada *et al.*, 2018] Chaitanya Manapragada, Geoffrey I Webb, and Mahsa Salehi. Extremely fast decision tree. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1953–1962. ACM, 2018.

[Montiel *et al.*, 2018] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. Scikit-multiflow: a multi-output streaming framework. *The Journal of Machine Learning Research*, 19(1):2915–2914, 2018.

[Quinlan, 1986] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[Quinlan, 1993] J Ross Quinlan. C 4.5: Programs for machine learning. *The Morgan Kaufmann Series in Machine Learning, San Mateo, CA: Morgan Kaufmann,— c1993*, 1993.

[Schlimmer and Granger, 1986] Jeffrey C Schlimmer and Richard H Granger. Incremental learning from noisy data. *Machine learning*, 1(3):317–354, 1986.

[Sovdat, 2014] Blaz Sovdat. Updating formulas and algorithms for computing entropy and gini index from time-changing data streams. *arXiv preprint arXiv:1403.6348*, 2014.

[Street and Kim, 2001] W Nick Street and YongSeog Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382. ACM, 2001.

[Widmer and Kubat, 1996] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23(1):69–101, 1996.