Learn Smart with Less: Building Better Online Decision Trees with Fewer Training Examples

Ariyam Das, **Jin Wang**, **Sahil M. Gandhi**, **Jae Lee**, **Wei Wang** and **Carlo Zaniolo**
Department of Computer Science, University of California, Los Angeles
{ariyam, jinwang, sahilmgandhi, jlee734, weiwang, zaniolo}@cs.ucla.edu

Abstract

Online decision tree models are extensively used in many industrial machine learning applications for real-time classification tasks. These models are highly accurate, scalable and easy to use in practice. The Very Fast Decision Tree (VFDT) is the classic online decision tree induction model that has been widely adopted due to its theoretical guarantees as well as competitive performance. However, VFDT and its variants solely rely on conservative statistical measures like Hoeffding bound to incrementally grow the tree. This makes these models extremely circumspect and limits their ability to learn fast. In this paper, we efficiently employ statistical resampling techniques to build an online tree faster using fewer examples. We first theoretically show that a naive implementation of resampling techniques like non-parametric bootstrap does not scale due to large memory and computational overheads. We mitigate this by proposing a robust memory-efficient bootstrap simulation heuristic (Mem-ES) that successfully expedites the learning process. Experimental results on both synthetic data and large-scale real world datasets demonstrate the efficiency and effectiveness of our proposed technique.

1 Introduction

Decision trees have been widely adopted by machine learning practitioners across different domains for their efficiency [Bifet *et al.*, 2017], scalability [Abuzaid *et al.*, 2016] and comprehensibility [Freitas, 2014]. It has been used in myriad applications ranging from Higgs boson classification [Chen and He, 2014] to predicting protein-protein interactions [Kingsford and Salzberg, 2008] in computational biology. In the streaming scenario, incremental decision tree induction algorithms have emerged as the predominant choice for a broad array of industrial classification tasks like real-time telecommunications network management and planning [Bifet *et al.*, 2017], stock market prediction [Kargupta *et al.*, 2002], vehicle monitoring [Kargupta *et al.*, 2004], health indicator tracking [Haghi *et al.*, 2017] and biosensor measurements [Aggarwal, 2006].

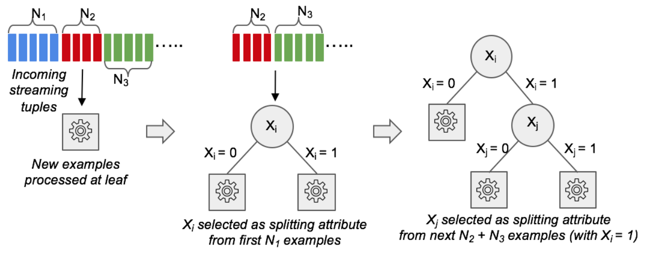
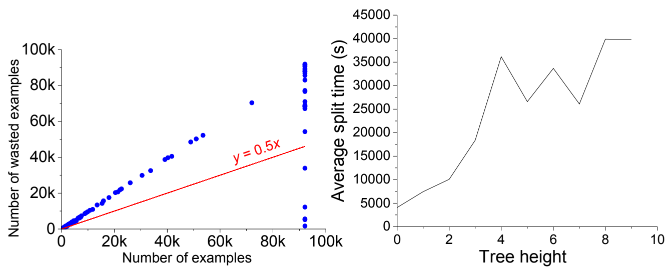

N_1
N_2
N_3
Incoming streaming tuples
New examples processed at leaf
N_2
N_3
X_i
$X_i = 0$
$X_i = 1$
X_i selected as splitting attribute from first N_1 examples
X_i
$X_i = 0$
$X_i = 1$
X_j
$X_j = 0$
$X_j = 1$
X_j selected as splitting attribute from next $N_2 + N_3$ examples (with $X_i = 1$)

Figure 1: Online decision tree construction

The Very Fast Decision Tree [Domingos and Hulten, 2000] (VFDT), a.k.a Hoeffding tree, is the most popular incremental decision tree induction algorithm. Figure 1 illustrates how a VFDT is built incrementally in a top-down manner as more examples are streamed in. It is typically constructed by observing "enough" training examples from an unbounded data stream and then deciding the splitting criterion from these observations with reasonably high confidence. The main idea is that for any small value δ, the splitting attribute chosen from a finite subsample would be the same as the one selected by traditional learners [Quinlan, 1993; Breiman *et al.*, 1984] with at least a probability of $1 - \delta$. The minimal sample size is ensured via the classic *Hoeffding bound inequality*, which determines the number of points that need to be observed before deciding the split. Theoretically, VFDT is guaranteed to be asymptotically identical to a decision tree built by a conventional learner [Breiman *et al.*, 1984; Quinlan, 1993]. However, the Hoeffding bound is a very conservative measure, since it is independent of the underlying data distribution. This means that a larger number of observations are needed at every node to make a split with the same confidence level $1 - \delta$, as compared to distribution-dependent bounds.

We demonstrate this by an example. We built a VFDT with 99% confidence (i.e. $\delta = 10^{-2}$) on 5 million streaming tuples produced from the standard MOA RandomTreeGenerator [Manapragada *et al.*, 2018] using the same parameters[1] mentioned in [Domingos and Hulten, 2000]. Figure 2(a) plots the total number of examples observed at a leaf before splitting vs. the number of *'redundant'* examples accumulated at

[1] 100 binary attributes and two classes



(a) Redundant examples (b) Time to split increases with depth

Figure 2: Observations over Decision Tree Node Split

the same leaf node. The latter represents the observations trailing a split, which have been collected at the leaf just to reach the eventual Hoeffding bound, but they do not change the selection of the best splitting attribute in any way. That is, the same best split could have been obtained without these 'redundant' examples, although the Hoeffding bound would not have been met. As shown in the figure, even for a moderate confidence level[2], majority of the leaves have 50% to 90% wasted examples (above the red line in Figure 2(a)).

In fact, this problem becomes more acute as the decision tree grows. Under the same experimental settings as before, Figure 2(b) shows how the average time taken by a leaf to split increases across the decision tree levels. This is primarily because the probability of an incoming tuple being assigned to a leaf reduces as the number of branches increases. This can be particularly concerning with today's modern shared memory and distributed memory architectures, where decision tree construction can be massively parallelized on multicore machines [Jin and Agrawal, 2003a] and on multi-node clusters [Abuzaid *et al.*, 2016]. However, the conservative nature of Hoeffding bound under these same parallel settings can force more resources (workers or threads) to idle-wait for longer durations. Naturally, these problems can be mitigated to a large extent, if an online induction tree model can learn *faster with fewer training examples* using tighter data distribution dependent bounds. Since the online induction tree model eventually grows and improves via leaf node splitting, this calls for a better and faster approach to split a leaf correctly. This is particularly needed when dealing with sensitive data streams (e.g. stock market or health sensor data), where inaccurate predictions can incur considerable damages.

In this paper, we propose a memory-efficient bootstrap simulation strategy (Mem-ES), which consumes fewer examples in deciding when to split a leaf. While simple adoption of resampling techniques like non-parametric bootstrap is not conducive to stream processing due to large memory and computational overheads (Section 3.2), Mem-ES only uses constant memory space per leaf to ensure accelerated learning and superior performance. Our proposed approach Mem-ES (Section 3.3) is able to estimate the distribution dependent bounds for node splitting and can operate robustly on any dataset. To the best of our knowledge, this is the first resource-efficient resampling strategy proposed in the context of incremental decision tree learners that empirically learns the distribution dependent bounds. We conduct extensive evaluations on Mem-ES using both synthetic data and large real-world datasets. Experimental results (Section 4) demonstrate the effectiveness of our proposed technique.

2 Related Work

2.1 Incremental Decision Tree Learning

Traditional decision tree [Quinlan, 1993] model is a cornerstone of classification tasks in machine learning. However, it is not well suited for real-time learning on data streams, where low latency bounded memory models are required. The most famous online decision tree induction algorithm, VFDT [Domingos and Hulten, 2000] mitigates this by learning from massive data streams incrementally in a *single* pass using *constant memory per leaf*. CVFDT [Hulten *et al.*, 2001] extends VFDT to incorporate gradual changes in the underlying data distribution for concept-drifting data streams. [Fan, 2004] and [Wang *et al.*, 2003] combined VFDT with other ensemble methods to improve the performance. Some previous studies also aimed at improving the node split latency of VFDT, which is the major bottleneck for the overall learning stage. [Jin and Agrawal, 2003b] deduced smaller theoretical bounds for the sample size required to make a split decision correctly. These bounds are independent of the input data distribution and have been derived from the mathematical properties of information gain [Quinlan, 1993] and Gini index [Breiman *et al.*, 1984]. However, unlike VFDT, these bounds are not agnostic to the split measures used. In addition, they also do not exploit the underlying input data distribution to come up with tighter bounds to hasten the split.

Recent studies like Extremely Fast Decision Tree [Manapragada *et al.*, 2018] (EFDT) improves the splitting process by allowing revision on the split decisions and achieves state-of-the-art performance on many datasets. Other recently proposed methods like One-Sided Minimum OSM [Losing *et al.*, 2018] utilizes local node statistics to optimize the frequency of evaluation of split decisions . These recent improvements still rely on the Hoeffding bound inequality for the actual split and are thus independent of our proposed method. Hence, our proposed algorithm Mem-ES can be plugged into these frameworks to further improve their performance, as shown in our experimental results. Many data stream mining systems also provide parallel and distributed implementations of online decision trees [Jin and Agrawal, 2003a], such as MOA [Bifet *et al.*, 2010] and STREAMDM-C++ [Bifet *et al.*, 2017], where our optimizations can also be integrated.

2.2 Resampling Methods

Statistical resampling techniques provide conceptually simple and powerful tools to (1) measure an estimate and (2) assess the quality of the corresponding estimation from an empirical distribution. The classical Bootstrap method [Bickel and Freedman, 1981; Gine and Zinn, 1990] was the first to

[2]More stringent confidence levels are used in [Domingos and Hulten, 2000; Manapragada *et al.*, 2018].

quantify the uncertainty in an estimator using confidence intervals via repeated Monte Carlo trials, where in each trial the estimator was computed over a resample drawn from the entire observed data. These estimates were more accurate, robust and consistent than those obtained using asymptotic approximations [Hall, 2013] on a wide domain of problems. Variants of the bootstrap algorithm like subsampling [Politis *et al.*, 1999] and m out of n bootstrap [Bickel *et al.*, 2012] are computationally much less demanding than the classic bootstrap algorithm, since the estimators are repeatedly computed on significantly smaller resamples. Unlike the traditional bootstrap method, these variants are less generic in nature, and often require rescaling and analytic asymptotic approximation of the output. This is addressed by the most recent Bag of Little Bootstraps (BLB) [Kleiner *et al.*, 2012] method, which combines the weighted results of bootstrapping multiple small subsets of a larger dataset in a robust manner. Our proposed algorithm Mem-ES is a variant of BLB that performs the Monte Carlo approximation over incoming streaming tuples under bounded memory constraints using a computationally efficient strategy.

3 Methodology

3.1 Preliminary

First we provide a detailed overview of the general online decision tree induction algorithm. Suppose $\mathcal{S}$ denotes a sequence of examples s, where $s = \langle \mathbf{x}, y \rangle$. Let $\mathbf{X}$ denote the set of attributes an example s has. The goal is to predict the class label y given the attribute values $\mathbf{x}$ of an incoming example s. VFDT, in particular, adopts the *Hoeffding bound* to decide when a node in the decision tree can be split with a confidence level of $1-\delta$. Formally, the Hoeffding bound inequality states that for n independent observations of a real-valued random variable r with range R and observed mean $\bar{r}$, the true mean of $r \geq \bar{r} - \epsilon$ with probability $1-\delta$, where

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

Thus intuitively, if $G(X_i)$ denotes the information gain[3] [Quinlan, 1993] for an attribute $\mathbf{X}_i$ computed from n training examples, and $G(\mathbf{X}_a), G(\mathbf{X}_b)$ indicate the highest and second highest information gain among all the attributes, then $G(\mathbf{X}_a) - G(\mathbf{X}_b) > \epsilon$ implies $\mathbf{X}_a$ can be judged as the best splitting attribute with probability $1-\delta$ without considering additional points for this particular split. In other words, by Hoeffding bound, we have $G(\mathbf{X}_a) - G(\mathbf{X}_b) > 0$ with probability $1-\delta$ over entire data.

Algorithm 1 demonstrates the process of constructing an online decision tree. The decision tree is first initialized with a single node (line: 2). Then for each incoming example $s \in \mathcal{S}$, it assigns the example into a leaf node l using the existing decision tree (line: 5). The leaves do not store the actual examples. Instead they maintain statistics like the *number of examples* seen for each attribute $\mathbf{X}_i \in \mathbf{X}$, with value j and class label $y = k$, which are denoted as n_{ijk} and are used

[3]Other heuristic measures such as Gini index [Breiman *et al.*, 1984] can also be used.

Algorithm 1: Online Decision Tree Induction ($\mathcal{S}$, $\mathbf{X}$, δ)

```
Input: $\mathcal{S}$: A sequence of examples; $\mathbf{X}$: The set of
        attributes; $\delta$: One minus the desired probability
Output: $\mathcal{T}$: Online decision tree learned from $\mathcal{S}$
1  begin
2      Initialize $\mathcal{T}$ with a single root node;
3      Initialize the statistics for tree growth;
4      foreach $s \in \mathcal{S}$ do
5          Sort $s$ into leaf node $l$ using $\mathcal{T}$;
6          Update the statistics at $l$ for tree growth;
7          if Examples at $l$ are not from the same class then
8              async Attempt to Split($l, \mathbf{X}, \delta$);
9      return $\mathcal{T}$;
10 end
```

Function `Attempt to Split` (l, $\mathbf{X}$, δ) in VFDT

```
1  begin
2      $\mathbf{X}_a, \mathbf{X}_b \leftarrow$ The two attributes with highest $G(\mathbf{X}_i)$
       values, where $\mathbf{X}_i \in \mathbf{X}$;
3      Compute $\epsilon$ using Equation 1;
4      $G(\mathbf{X}_\emptyset) \leftarrow$ Gain corresponding to no split;
5      if $G(\mathbf{X}_a) - G(\mathbf{X}_b) > \epsilon$ and $G(\mathbf{X}_a) \neq G(\mathbf{X}_\emptyset)$ then
6          Replace $l$ with an internal node;
7      foreach branch of split on $\mathbf{X}_a$ do
8          Add a new child $l_m$ to $l$ and set $\mathbf{X}_m \leftarrow \mathbf{X} - \mathbf{X}_a$;
9  end
```

for computing $G(\mathbf{X}_i)$. These statistics are updated along with new incoming examples (line: 6). A leaf node in VFDT is split (line: 8) when its best attribute satisfies the Hoeffding inequality (as shown in the function `Attempt to Split`). The split attempts on different leaves can be executed in parallel and asynchronously [Jin and Agrawal, 2003a]. However, more importantly, as shown in equation 1, the Hoeffding bound does not take into account the underlying distribution of the examples seen and hence most leaf nodes consume considerably more examples than necessary in order to make a successful split with reasonable confidence.

3.2 Non-parametric Bootstrap Driven Split

According to the bootstrap principle, given any unknown distribution F and a sample S drawn i.i.d from F, the quality of an estimation θ of some unknown population value, associated with F, can be assessed by drawing *with replacement* sufficient resamples from S of size $|S|$. The confidence interval of θ computed via a form of Monte Carlo approximation from the resampling (or empirical) distribution F^* holds well in practice, since by the law of large numbers the relative variation among F and F^* are similar [Gine and Zinn, 1990]. Interestingly, bootstrap does not necessarily improve upon the actual value of the estimation θ. Instead it provides a good assessment of the quality of the estimate via standard error or confidence intervals.

We next summarize a bootstrap driven split attempt method: For a given leaf node l, let $\mathcal{A}$ denote all examples observed in l. $\mathbf{X}_a, \mathbf{X}_b$ are the best and second best splitting attributes based on $G(\mathbf{X}_a)$ and $G(\mathbf{X}_b)$ respectively as computed from $\mathcal{A}$. Let $\theta = G(\mathbf{X}_a) - G(\mathbf{X}_b)$. Now, we can perform the classical bootstrap for T Monte Carlo iterations, where in each iteration i we draw a sample S_i with replacement from $\mathcal{A}$ and compute from S_i, $\theta_i^* = G(\mathbf{X}_a) - G(\mathbf{X}_b)$ and $\Delta_i^* = \theta_i^* - \theta$. Given a series of sorted Δ_i^*, for $1 \leq i \leq T$, we can select Δ_L^* and Δ_U^* as the lower and upper percentiles for calculating the bootstrap confidence interval corresponding to $1-\delta$ confidence. For example, for a confidence level of 95%, Δ_L^* and Δ_U^* would be the 2.5th and 97.5th percentile respectively. Thus, the bootstrap confidence interval for the estimate θ is given by $[\theta - \Delta_U^*, \theta - \Delta_L^*]$. Now, considering only the lower bound of the confidence interval, if equation 2 is satisfied, then $\theta = G(\mathbf{X}_a) - G(\mathbf{X}_b) > 0$ holds with probability $1-\delta$. Therefore, we can split the node on $\mathbf{X}_a$.

$$\theta > \Delta_U^* \tag{2}$$

It is important to note that for calculating Δ_U^*, we only need to store a few top Δ_i^* values. For example, for calculating the 97.5th percentile of 100 Δ_i^* values, we only need to store and maintain the four top Δ_i^* values. Note this bootstrap driven method decides to split based on the empirical distribution. However, this accelerated split comes at the cost of larger memory and computational overheads, which are discussed next.

Space and Time Complexity Analysis

Unlike the VFDT, a bootstrap driven split attempt would need to store the examples in the leaf in order to be able to resample from it. Considering nominal data with c classes and d attributes, where each attribute can have at most v values, VFDT can maintain the n_{ijk} counts at each leaf in $O(dvc)$ memory. Thus for l leaves, total space complexity for VFDT is given by $O(ldvc)$. On the contrary, in order to store examples at the leaves, non-parametric bootstrap would require a space of $O(ldn)$, where n is the highest number of points accumulated at any leaf. Since $n >> vc$, the bootstrap driven split method has a space complexity of $O(ldn)$.

Information gain computation takes $O(c)$ time and each of the d attributes needs to calculate at most v information gains. Thus the time to find the best and second best split attribute at any node in VFDT is $O(dvc)$. However, in bootstrap, T iterations are conducted, where in each iteration resamples are drawn in $O(n)$ time and Δ_U^* is computed in $O(vc)$ time. Again, since $n >> vc$, the overall time complexity for bootstrap based split attempt at any leaf is $O(Tn + dvc)$, or simply $O(Tn)$, assuming the split attempts at each node are conducted in parallel and asynchronously (see Algorithm 1). Thus, in terms of both space and time complexity, the dependency on n is the major bottleneck, which makes the bootstrap based approach practically hard to scale.

3.3 Memory-Efficient Bootstrap Simulation (Mem-ES)

To mitigate the above problems, we propose the Mem-ES method based on the principle of Bag of Little Bootstraps (BLB) [Kleiner *et al.*, 2012]. BLB selects few small samples (possibly disjoint) and then *artificially* generates large bootstrap samples from them, which are consequently used to compute the quality of the estimators. Under BLB, we simulate selecting large bootstrap samples of size n from a considerably smaller sample of size w ($w << n$) by drawing n trials from a multinomial distribution with parameters $n, \mathbf{1}_w/w$, where $\mathbf{1}_w/w$ denotes the 1-by-w vector of multinomial probabilities, each initialized with value $\frac{1}{w}$. The final estimator quality is assessed by averaging across all the results. However, this BLB template was proposed in the context of bounded static data. Mem-ES extends BLB and adapts it for unbounded data streams. For example, Algorithm 3 uses Mem-ES to specifically check for a potential split with high statistical confidence in a memory-efficient manner by only relying on the most recent batch of w points at any leaf to simulate the bootstrap process. Also note that in Algorithm 3, unlike the non-parametric bootstrap, the size of $\mathcal{A}$ is bounded to an user-defined value w (lines: 3 and 18). We can easily integrate Mem-ES into VFDT or other variants by simply replacing line 8 in Algorithm 1. We next discuss some key aspects of Mem-ES.

Algorithm 3: Attempt to split with Mem-ES

```
Input: $l$: Leaf to be split; $\mathbf{X}$: The set of attributes; $\delta$: One
       minus the desired probability; $s$: current example;
       $\mathcal{A}$: Queue of fixed size $w$; $n$: User-specified
       parameter; $T$: Number of Monte Carlo iterations;
       $r, sum$: Variables initialized with 0
1  begin
2      Enqueue $s$ to $\mathcal{A}$;
3      if $|\mathcal{A}|$ is not full then
4          return;
5      $Q \leftarrow$ Min-Heap of fixed size for $\Delta_U^*$ computation;
6      $r \leftarrow r + 1$; // count of disjoint sets
7      $\mathbf{X}_a, \mathbf{X}_b \leftarrow$ The two attributes with highest $G(\mathbf{X}_i)$
       values computed from $n_{ijk}$ counts at $l$;
8      $\theta \leftarrow G(\mathbf{X}_a) - G(\mathbf{X}_b)$;
9      for $t \in [1, T]$ do
10         Draw sample $S_t$ from Multinomial$(n, \mathbf{1}_w/w)$;
11         $\theta_t^* \leftarrow G(\mathbf{X}_a) - G(\mathbf{X}_b)$ from $S_t$;
12         $\Delta_t^* \leftarrow \theta_t^* - \theta$;
13         if $\Delta_t^* > min(Q)$ or $Q$ is not full then
14             Add $\Delta_t^*$ to $Q$;
15     $\Delta_U^* \leftarrow min(Q)$;
16     $sum \leftarrow sum + \Delta_U^*$;
17     $\overline{\Delta_U^*} \leftarrow \frac{sum}{r}$;
18     Empty $\mathcal{A}$ and $Q$;
19     if $\theta > \overline{\Delta_U^*}$ then
20         Replace $l$ with an internal node;
21         foreach branch of split on $X_a$ do
22             Add a new child $l_m$ to $l$ and set $\mathbf{X}_m \leftarrow \mathbf{X} - \mathbf{X}_a$;
23 end
```

Discussion

Each leaf node in Mem-ES maintains at most w last seen examples. Thus, the overall memory complexity for Mem-ES is $O(ldvc + ldw)$. Typically, w values (as used in our experiments) are comparable to vc.

Similarly, we can draw a sample of size n from the multinomial distribution of w distinct objects in $O(w)$ time, thereby the overall time complexity for a split attempt at any leaf is reduced to $O(Tw + Tvc + dvc)$. Typically Tw is comparable to dvc. However, online decision tree construction is mostly I/O or network bound [Domingos and Hulten, 2000], since the streaming rate is primarily throttled by the I/O rate or network bandwidth. As such, the in-memory computation time of a split attempt, specially under parallel settings, is largely overshadowed by the time required to read the corresponding tuples. Lastly, it is worth mentioning that in practice, instead of two attributes (line: 7 in Algorithm 3), we maintain top 4 or 5 promising attributes since X_a, X_b may change as more examples arrive.

4 Evaluation

4.1 Experimental Setup

Baseline Methods

We evaluate the effectiveness of Mem-ES against two category of baselines:

• First, we integrate Mem-ES into VFDT (denoted as *VFDT+ME*) and compare its performance against standard VFDT and [Jin and Agrawal, 2003b] (denoted by '*VFDT+IG*'). Simple VFDT uses Hoeffding bound, whereas *VFDT+IG* uses sample size estimates deduced theoretically from the property of information gain.

• Second, we incorporate Mem-ES into a VFDT variant like EFDT and benchmark its performance. *EFDT+ME* denotes integration of Mem-ES into EFDT.

Datasets

We evaluate the performance of Mem-ES against the above baselines on two large real world classification datasets used in [Manapragada *et al.*, 2018] and one synthetic dataset.

• Gas Sensors dataset(Gas) from UCI repository consists of 900K+ records. The data have 15 continuous attributes and 3 classes in total.

• Human Activity Recognition (WISDM) dataset [Kwapisz *et al.*, 2011] consists of 1M+ records. The data have 5 continuous attributes and 6 classes in total.

• In addition, we also created a synthetic dataset(SYN) spanning across 10M+ records with 100 binary attributes and 2 classes (generated from the standard MOA RandomTreeGenerator) to test Mem-ES for scalability.

VFDT and its variants like EFDT theoretically converge towards the decision tree built by a traditional learner when the incoming streaming examples are i.i.d. Hence, the data sets are shuffled for the experiments as prescribed in [Manapragada *et al.*, 2018]. Here we report the metrics averaged over 5 such shuffles.

Environment

Our experiments are conducted on a standard commodity machine with 4 cores and 32GB memory running Ubuntu 14.04 LTS. We used Java implementation of Mem-ES and other baselines with 8 threads. The main program was tasked with reading the incoming data tuple and assigning it to a leaf in the decision tree. Other workers concurrently and asynchronously attempt splits at different leaves. All the experiments were performed using a confidence level of 98%, $T = 150$, $w = 40$. We used the standard value of $n_{min} = 200$.

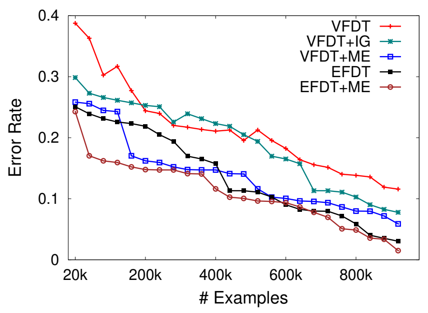
4.2 Results and Discussion

In this section, we will try to answer the following questions: (1) Does Mem-ES help to learn a decision tree *correctly* with *fewer* examples? (2) Is memory a major bottleneck for Mem-ES or can it scale at an affordable cost?

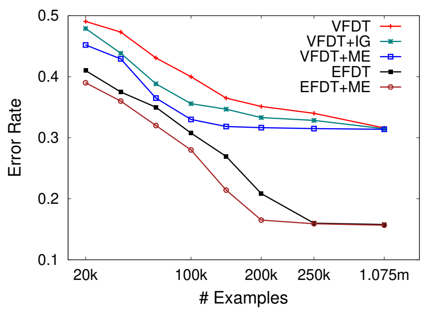
We examine the first question by plotting the number of instances seen vs. the error rate for all three data sets. Figure 3 shows the corresponding three plots. Figure 3(a) shows that *VFDT+ME* learns better from considerably fewer examples, as compared to VFDT and *VFDT+IG*, yielding around 8 percentage points lower error rate than VFDT and *VFDT+IG* in the first 200K examples. In addition, Figure 3(a) further shows that even *EFDT+ME* learns faster with fewer number of examples than EFDT, although the gap reduces after 150K examples. Interestingly, simple *VFDT+ME* has outperformed the optimized EFDT for the first 400K examples on the Gas data. Figure 3(b) on WISDM data further reiterates that Mem-ES learns and converges faster with fewer examples than other baseline methods. In fact, Figure 3(b) exemplifies the utility of Mem-ES as a 'plug-in', since it can be incorporated into VFDT to outperform VFDT and *VFDT+IG* and can similarly be integrated into EFDT to improve its performance. For some datasets, VFDT variants can outperform EFDT, as shown in Figure 3(c), where *VFDT+IG* produces lower error rate than simple EFDT. Nevertheless, Mem-ES still yields a marginal improvement (2 percentage points lower error in first 400K examples) over *VFDT+IG*.

It is worth highlighting here that all online models eventually converge as examples stream in [Manapragada *et al.*, 2018]. But online models that learn and converge faster with fewer examples are naturally more preferable. This is more important for sensitive data streams, where inaccurate classifications can incur significant penalties. Thus, the nomenclature of 'fast' in systems like VFDT or EFDT refers to how many fewer examples are required in the learning stage. In other words, at any time instant during the learning stage, the accuracy of the online learner is determined mainly from its ability to learn from fewer examples. And in this context of learning from fewer examples, Figure 3 presents Mem-ES as a very effective and robust strategy that works well across different data sets, since *EFDT+ME* (Figures 3(a), 3(b)) or *VFDT+ME* (Figure 3(c)) produces the best results.

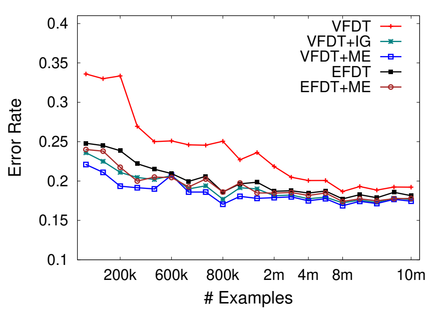
Next, we investigate the memory cost of Mem-ES by examining how the number of nodes and the memory consumption varies as the examples stream in. Figures 4 and 5 show this interplay with double Y-axis graphs, where the line graph indicates the total memory consumed and the vertical bar represents the total number of nodes created in the corresponding decision tree. Figure 4 presents the comparison between VFDT, *VFDT+IG* and *VFDT+ME*. As shown in Figure 4, the three methods incrementally builds the tree, but *VFDT+ME* constructs at the fastest rate i.e. it builds a more deeper decision tree model as compared to other baselines after processing the same number of examples. Also recall that Mem-ES needs to store at most w examples at each leaf. As a result, *VFDT+ME* ends up requiring more memory than VFDT or *VFDT+IG*, since it grows better trees with more nodes at an expedited rate as well as retains some data points at the leaves. But nevertheless, even for the syn-



(a) Gas

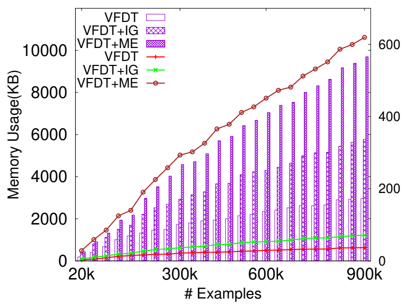


(b) WISDM

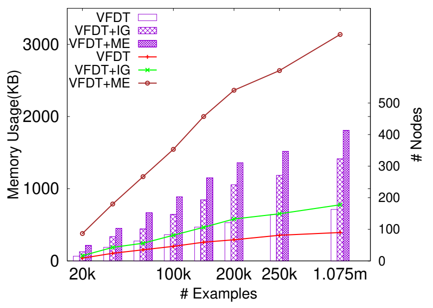


(c) SYN

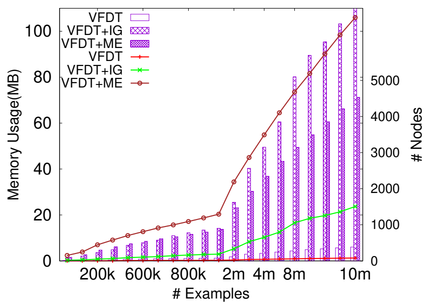
Figure 3: Comparison with state-of-the-art methods: Error Rate



(a) Gas

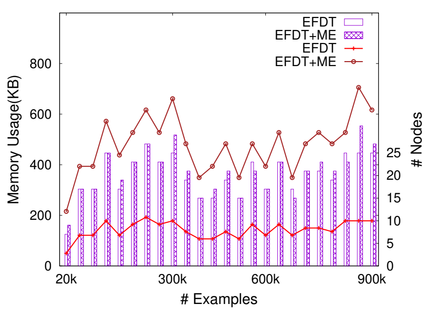


(b) WISDM

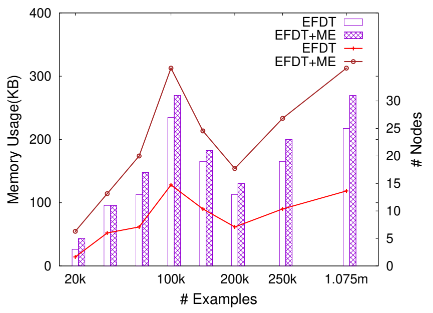


(c) SYN

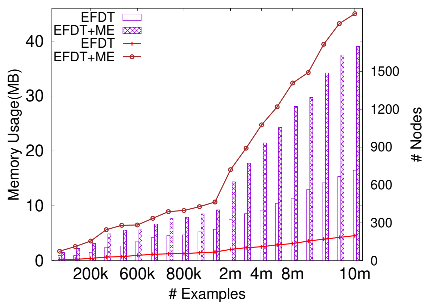
Figure 4: Tree growth and memory consumption: Mem-ES vs. VFDT and *VFDT+IG*



(a) Gas



(b) WISDM



(c) SYN

Figure 5: Tree growth and memory consumption: Mem-ES vs. EFDT

thetic data of over 10M records (Figure 4(c)), *VFDT+ME* requires only around 100MB memory, which is an order of magnitude less than the default allocated heap size in standard Java virtual machines as used in our implementation. Furthermore, *VFDT+ME* requires less than 10MB and 3MB memory for Gas and WISDM datasets respectively. Figure 5 shows the corresponding plots for EFDT and *EFDT+ME*. EFDT re-examines the splits of internal nodes in the decision tree and aggressively prunes the sub-trees if a split better than the original one is found. Consequently, as shown in Figure 5, the number of nodes and memory consumption of EFDT and *EFDT+ME* can decrease as well. This makes Mem-ES more suitable for integration into strategies like EFDT where memory can be aggressively freed. *EFDT+ME* consumes around 700KB, 300KB and 45MB memory for Gas, WISDM and SYN datasets.

5 Conclusion

In this paper, we propose Mem-ES that efficiently performs resampling techniques to accelerate the node splits for online decision tree learning. The success of Mem-ES is particularly exciting, since the idea of applying resampling techniques like non-parametric bootstrap on data streams had always been considered very difficult in the past due to its cumbersome nature and high time and space complexity. But, this first of its kind realization and experimental validation of approximate bootstrapping can invite further research investigations in other stream mining algorithms used for frequent pattern mining [Das and Zaniolo, 2016], episode mining [Ao *et al.*, 2019; 2018], complex pattern detection and ranking [Gu *et al.*, 2016], where bootstrapping can be useful.

References

[Abuzaid *et al.*, 2016] F. Abuzaid, J. K. Bradley, F. T. Liang, A. Feng, L. Yang, M. Zaharia, and A. Talwalkar. Yggdrasil: An optimized system for training deep decision trees at scale. In *NIPS*, pages 3817–3825. 2016.

[Aggarwal, 2006] Charu C. Aggarwal. *Data Streams: Models and Algorithms (Advances in Database Systems)*. Springer-Verlag, 2006.

[Ao *et al.*, 2018] X. Ao, P. Luo, J. Wang, F. Zhuang, and Q. He. Mining precise-positioning episode rules from event sequences. *IEEE TKDE*, 2018.

[Ao *et al.*, 2019] X. Ao, H. Shi, J. Wang, L. Zuo, H. Li, and Q. He. Large-scale frequent episode mining from complex event sequences with hierarchies. *ACM TIST*, 2019.

[Bickel and Freedman, 1981] Peter J. Bickel and David A. Freedman. Some asymptotic theory for the bootstrap. *The Annals of Statistics*, 9(6):1196–1217, 11 1981.

[Bickel *et al.*, 2012] Peter J Bickel, Friedrich Götze, and Willem R van Zwet. Resampling fewer than n observations: gains, losses, and remedies for losses. In *Selected works of Willem van Zwet*, pages 267–297. Springer, 2012.

[Bifet *et al.*, 2010] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: massive online analysis. *Journal of Machine Learning Research*, 11:1601–1604, 2010.

[Bifet *et al.*, 2017] A. Bifet, J. Zhang, W. Fan, C. He, J. Zhang, J. Qian, G. Holmes, and B. Pfahringer. Extremely fast decision tree mining for evolving data streams. In *SIGKDD*, pages 1733–1742, 2017.

[Breiman *et al.*, 1984] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

[Chen and He, 2014] Tianqi Chen and Tong He. Higgs boson discovery with boosted trees. In *HEPML@NIPS*, pages 69–80, 2014.

[Das and Zaniolo, 2016] Ariyam Das and Carlo Zaniolo. Fast lossless frequent itemset mining in data streams using crucial patterns. In *SDM*, pages 576–584, 2016.

[Domingos and Hulten, 2000] Pedro M. Domingos and Geoff Hulten. Mining high-speed data streams. In *SIGKDD*, pages 71–80, 2000.

[Fan, 2004] Wei Fan. Systematic data selection to mine concept-drifting data streams. In *SIGKDD*, 2004.

[Freitas, 2014] Alex A. Freitas. Comprehensible classification models: A position paper. *SIGKDD Explor. Newsl.*, 15(1):1–10, 2014.

[Gine and Zinn, 1990] Evarist Gine and Joel Zinn. Bootstrapping general empirical measures. *The Annals of Probability*, 18(2):851–869, 04 1990.

[Gu *et al.*, 2016] Jiaqi Gu, Jin Wang, and Carlo Zaniolo. Ranking support for matched patterns over complex event streams: The CEPR system. In *ICDE*, 2016.

[Haghi *et al.*, 2017] Mostafa Haghi, Kerstin Thurow, and Regina Stoll. Wearable devices in medical internet of things: scientific research and commercially available devices. *Healthcare informatics research*, 23(1):4–15, 2017.

[Hall, 2013] Peter Hall. *The bootstrap and Edgeworth expansion*. Springer Science & Business Media, 2013.

[Hulten *et al.*, 2001] Geoff Hulten, Laurie Spencer, and Pedro M. Domingos. Mining time-changing data streams. In *SIGKDD*, pages 97–106, 2001.

[Jin and Agrawal, 2003a] Ruoming Jin and Gagan Agrawal. Communication and memory efficient parallel decision tree construction. In *SDM*, pages 119–129, 2003.

[Jin and Agrawal, 2003b] Ruoming Jin and Gagan Agrawal. Efficient decision tree construction on streaming data. In *SIGKDD*, pages 571–576, 2003.

[Kargupta *et al.*, 2002] H. Kargupta, B.-H. Park, S. Pittie, L. Liu, D. Kushraj, and K. Sarkar. Mobimine: Monitoring the stock market from a pda. *SIGKDD Explor. Newsl.*, 3(2):37–46, 2002.

[Kargupta *et al.*, 2004] H. Kargupta, R. Bhargava, K. Liu, M. Powers, P. Blair, S. Bushra, J. Dull, K. Sarkar, M. Klein, M. Vasa, and D. Handy. Vedas: A mobile and distributed data stream mining system for real-time vehicle monitoring. In *SDM*, 2004.

[Kingsford and Salzberg, 2008] Carl Kingsford and Steven L. Salzberg. What are decision trees? *Nature Biotechnology*, 26(1011), 2008.

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

[Kwapisz *et al.*, 2011] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *SIGKDD Explor. Newsl.*, 12(2):74–82, 2011.

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

[Manapragada *et al.*, 2018] Chaitanya Manapragada, Geoffrey I. Webb, and Mahsa Salehi. Extremely fast decision tree. In *SIGKDD*, pages 1953–1962, 2018.

[Politis *et al.*, 1999] Dimitris Politis, Joseph P Romano, and Michael Wolf. Weak convergence of dependent empirical measures with application to subsampling in function spaces. *Journal of statistical planning and inference*, 79(2):179–190, 1999.

[Quinlan, 1993] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[Wang *et al.*, 2003] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *SIGKDD*, pages 226–235, 2003.