2018

Reducing labeling complexity in streaming data mining

Yesdaulet Izenov
Iowa State University

Reducing labeling complexity in streaming data mining

by

Yesdaulet Izenov

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Engineering (Software Systems)

Program of Study Committee:
Srikanta Tirthapura, Major Professor
Chinmay Hegde
Neil Zhenqiang Gong

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2018

TABLE OF CONTENTS

ABSTRACT

Supervised machine learning is an approach where an algorithm estimates a mapping function by using labeled data i.e. utilizing data attributes and target values. One of the major obstacles in supervised learning is the labeling step. Obtaining labeled data is an expensive procedure since it typically requires human effort. Training a model with too little data tends to overfit therefore in order to achieve a reasonable accuracy of prediction we need a minimum number of labeled examples. This is also true for streaming machine learning models. Maintaining a model without rebuilding and performing a prediction task without ever storing input samples are the key concepts of streaming machine learning models. A successful and widely used streaming model is the Hoeffding tree which has large labeling complexity. In this work, we present Frugal Hoeffding tree, a variation of the Hoeffding tree that uses less labeled data, and provides similar performance as the original Hoeffding tree. We conduct experiments on large real-world datasets where we compare the performances of traditional batch decision trees, the Hoeffding tree and the Frugal Hoeffding tree. We show that the Frugal Hoeffding tree consumes less labeled data yet can achieve classification performance similar to the Hoeffding tree.

CHAPTER 1. INTRODUCTION

We have many organizations that have terabytes of data that is produced everyday. Millions of transactions are being recorded in financial organizations, millions of calls are made everyday and web logs are continuously being produced every minute and these are only a few examples of application domains that have large-scale data. These examples constitute that the amount data has tremendously increased in many application domains and it has become almost impossible to store the data in memory. Therefore many of the machine learning models are being reconsidered and modified in a way that large scale data could be processed and analyzed using clusters of computational nodes.

Another important aspect to mention is the many of the machine learning models have been designed with an assumption that input data can fit in memory. However today this assumption is often violated thus in designing machine learning models, memory and computational resources are necessary to be taken into an account. In order to address these requirements streaming machine learning models are being designed, and therefore streaming machine learning models have been an active research area last couple of decades. Streaming algorithms are aimed to design models that utilize significantly less memory and computational resources because these models don't store input data. Streaming algorithms are useful only if they provide similar performance results as their batch analogs.

An early and effective machine learning method are the decision tree models. Decision trees have been successful and applied in many application domains providing a high quality prediction. Decision trees are attractive because of interpretability, simplicity, and robustness to outliers. A decision tree is constructed building a tree structure and it is grown to its maximum height. In a greedy manner decision tree algorithm finds the feature that will yield the largest information gain for a given node and various performance pruning techniques are applied in order to avoid overfitting. Decision tree algorithms have been evolved and a number of batch variations have been developed such as ID3(Iterative Dichotomiser 3) [1], C4.5 [2] and CART [3].

One of the widely used streaming variation of decision trees is Hoeffding trees. Hoeffding trees were originally proposed by Domingos and Hulten [4] and have been found further developments in many research works. Hoeffding tree is an incremental online decision tree algorithm that is able processes massive data streams. The main idea behind the Hoeffding trees is that the model builds an effective decision tree structure that has a mathematical guarantee of choosing best split nodes based on a subset of data and it produces asymptotically similar tree structure to the ones produced by a batch learner.

Like batch supervised machine learning models, streaming supervised machine learning models require a large amount of labeled data in order to achieve a high accuracy prediction. In large-scale data applications, this number of required labeled data can reach millions or even more units. Preparing and labeling such a large data is costly to label, and therefore in designing machine learning models one needs reduction of labeling complexity requirement additional to memory, computation and high prediction criteria.

In this work we propose Frugal Hoeffding tree model, a modification of the original Hoeffding tree that utilizes less labeled data and we show that it can achieve similar performance results as the original Hoeffding tree. We found that reduction of labeling effort in streaming algorithms hasn't been analyzed yet in research works. Our work shows that reducing the number of labeled data necessary for streaming Hoeffding tree can be achieved by applying several modifications and can provide similar performance results.

The contributions of this study are two-fold. First, this study proposes a simple concept which allows to achieve reduction of labeling complexity and it can be used in many streaming machine learning models too. In order to demonstrate it we apply the concept in the Hoeffding tree model. Second, the study evaluates the effectiveness of Frugal Hoeffding tree and compares to the original Hoeffding tree model. We show experimental results on three real-world datasets and compare performances of the batch and streaming decision tree algorithms.

CHAPTER 2. REVIEW OF LITERATURE

Domingos and Hulten [4] initially proposed VFDT(Very Fast Decision Tree learner) for categorical data that used the Hoeffding bound. The original algorithm assumes that streaming data has a stationary distribution. The designed algorithm was evaluated on a web-scaled dataset and the experimental results showed that the Hoeffding tree outperformed C4.5 model and also they showed that less amount of memory and computational resources were utilized by the streaming Hoeffding tree algorithm.

Hulten and Domingos [5] proposed CVFDT(Concept adapting Very Fast Decision Tree learner) algorithm that stays updated and efficient for continuously changing data streams also known as concept drift. Many machine learning algorithms and VFDT have the assumption that the streaming input data drawn from a stationary distribution. CVFDT utilizes a window-based concept and grows an alternate subtree that is replaced whenever the old tree becomes less efficient. The replacement happens when a concept drift occurs and makes the current model less accurate. The performance of CVFDT gives a nearly equivalent accuracy results as if VFDT is repeatedly grown on a window of samples.

Jin and Agrawal [6] revisit the Hoeffding tree model and propose the following improvements. They provide an efficient variation of the Hoeffding tree that can processes categorical and numerical attributes and achieve reduction in execution time. Also they propose a modification that utilizes the properties of heuristic evaluation function i.e. information gain and gini, and reduce the number of sample sizes required to obtain the bound condition for a split node.

CHAPTER 3. HOEFFDING TREES

Decision Trees

One of the most effective and widely-used classification methods is decision tree learning. These type of approaches induce models in the form of tree structures, where each node contains a split rule on an attribute, each branch from a node corresponds to a possible outcome of the split rule, and each leaf contains a class prediction. The label for an input example is obtained by passing the example down from the root to a leaf, testing the appropriate attribute at each node and following the branch corresponding to the attribute's value in the example. A decision tree is learned by recursively replacing leaves by split nodes, starting at the root. The attribute to split at a node is chosen by comparing all the available attributes and choosing the best one according to some heuristic measure. Classic decision tree learners like ID3 [1], C4.5 [2] and CART [3] assume that all training examples can be stored simultaneously in main memory, and thus severely limited in the number of examples they can learn from. Disk-based decision tree learners like SLIQ [7] and SPRINT [8] have been proposed. Despite the fact that these models largely increase the size of usable data, it can be expensive in case of learning large and complex tree structures. Steaming decision tree is a decision tree learner for extremely large and potentially infinite datasets and this learner should require each example to be read at most once, and only a small constant time to process it. This will make it possible to directly mine online data sources without ever storing the examples, and to build potentially very complex trees with acceptable computational cost.

Decision tree learning uses decision tree structure as a predictive model mapping observations about an item to conclusions about the item's target value. Decision tree learning is a common method used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables. Decision tree construction algorithms generally use top-down approach by choosing an attribute at each phase to split the given data set. Split-

ting is based on the best attribute chosen at each phase and the process keeps on repeating on each resultant subset recursively until the next splitting no longer adds value to the predictions. Once the labeled example reaches a leaf v the model estimates the probabilities for each class as

$$Pr_k(v) = \frac{n_k}{\sum_{j=1}^{|C|} n_j} \tag{1}$$

where C is the total number of classes and n_k is the number of training instances of the class k at a leaf.

However, the probability estimates are weak since when calculating the estimated probability as the formulation above, where n is the number of training instances of the specified class at a leaf and the denominator is the total number of training instances, nodes with a small number of training instances will be assigned the same probability as nodes with a large number of training instances. This is especially problematic for ranking, since it will result in many ties, and training instances with the same probability can not be ordered. The problem is usually addressed by smoothing the probability estimates to less extreme values. The most commonly used smoothing method is the Laplace estimate, which calculates the expected probability as

$$Pr_k(v) = \frac{n_k + 1}{\sum_{j=1}^{|C|} n_j + |C|} \tag{2}$$

where C is the total number of classes and n_k is the number of instances of the specified class at a leaf.

Different algorithms use different formulae for predicting “the best attribute" to split a node. There are two commonly used heuristics which are applied to each candidate subset, and the resulting values are combined to provide a measure of the quality of the split. First common heuristic is Gini index that measures heterogeneity or homogeneity of a node and it selects the best attribute that yields a class distribution where the majority of training instances belong to

the same class. Gini index is measured as

$$GI(v) = 1 - \sum_{k=1}^{|C|} Pr_k(v)^2 \tag{3}$$

where $Pr_v(k)$ is the number of samples that belong to class k divided by total number of samples within a node v, see equation 1. Gini index is maximal if the classes are perfectly mixed i.e. the instances at a node is uniformly distributed.

Second common heuristic is information gain that is based on the concept of entropy used in information theory. In this work, we use information gain heuristic to choose the best attribute for a split node. Entropy of a node v is measured as

$$H(v) = \sum_{k=1}^{|C|} -Pr_k(v) \log_2 Pr_k(v) \tag{4}$$

Conditional entropy after splitting a node v based on an attribute A is measured as

$$H(v|A) = \sum_{i \in A} Pr(A = i) H(l) \tag{5}$$

where A is attribute value, $Pr(A = i)$ is the number of samples that has the attribute value equal to i divided by total number of samples within a node and $H(l)$ is entropy measure of a new leaf l which includes samples that have attribute A value equal to i.

Using information gain heuristic, we would like to select an attribute to split the decision tree further such that the model gains knowledge as much as possible. Information gain is measured as

$$G(v, A) = H(v) - H(v|A) \tag{6}$$

where $H(v)$ is entropy measure before a split and $H(v|A)$ is entropy measure after the split based on attribute A.

Visual Example for Decision Tree Construction

Consider Table 1 as input dataset. The dataset has four attribute variables that are used to predict whether an athlete plays a golf game. This task is a binary classification problem. These features are weather related attributes and they are all categorical attributes. The fifth column in Table 1 is the target variable i.e. true labels.

Table 1: **The Weather Data.**

Outlook	Temperature	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

In order to start building a decision tree model we need a root node. At the root node we need to decide which attribute would be the most efficient to classify i.e. the one that would give the maximum information about data. Instead of selecting an attribute at random, the model uses the information gain heuristic that was discussed above. Entropy measure before splitting the initial dataset is $H(root) = -(\frac{5}{14}\log_2\frac{5}{14}) - (\frac{9}{14}\log_2\frac{9}{14}) = 0.94$ (see equation 4). Once we measure the current entropy value we need to measure entropy after a split. Let the model to consider first "outlook" attribute. In Figure 1, the model computes the entropy value after splitting the root node based on "outlook" attribute.

$$H(root|outlook) = Pr(outlook = "sunny")H(leaf = "sunny")$$
$$+Pr(outlook = "overcast")H(leaf = "overcast")$$

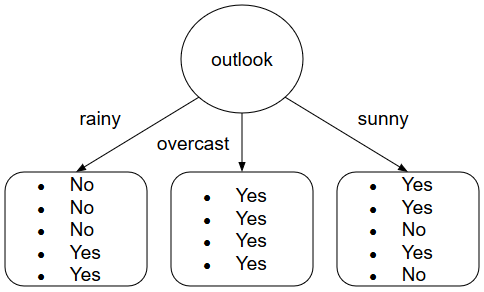

Figure 1: **Decision tree construction. First level. Attribute selection for the root node.**

$$+Pr(outlook = "rainy")H(leaf = "rainy")$$
$$= \frac{5}{14}(-(\frac{3}{5}\log_2\frac{3}{5}) - (\frac{2}{5}\log_2\frac{2}{5}))$$
$$+\frac{4}{14}(-(\frac{4}{4}\log_2\frac{4}{4}) - (\frac{0}{4}\log_2\frac{0}{4}))$$
$$+\frac{5}{14}(-(\frac{2}{5}\log_2\frac{2}{5}) - (\frac{3}{5}\log_2\frac{3}{5})) = 0.69 \text{ (see equation 5).}$$

Having computed the entropy measures before and after the split based on "outlook" attribute, we compute information gain which is $G(root, outlook) = H(root) - H(root|outlook) = 0.25$ (see equation 6). If we would compute information gain value for each attribute, we would get the following values in Table 2.

Table 2: **Informatin gain values for each attribute in the Weather dataset.**

Attribute	Information gain
Outlook	$G(root, outlook) = 0.25$
Temperature	$G(root, temperature) = 0.05$
Humidity	$G(root, humidity) = 0.17$
Windy	$G(root, windy) = 0.07$

According to Table 2, the model chooses the "outlook" attribute since it provides the maximum value in terms of information gain heuristic. In another words, the "outlook" attribute provides the maximum information i.e. it groups the input samples in a way that the input samples with the same target values mostly gathered together.

Having set the root node, the model will look for the next best attribute to split in order to further grow the tree. According to Figure 2, for the left node at the second level of the tree, attribute "windy" is the best to split on.

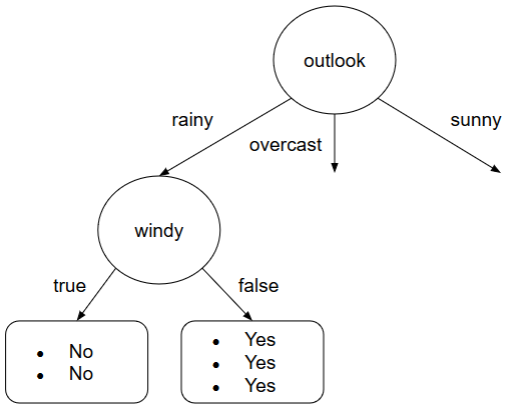

Figure 2: **Decision tree construction. Second level. Attribute selection for the left node.**

After recursively splitting the leaf nodes, eventually the model gets the tree structure shown in Figure 3. As it shown in Figure 3, for the right node at the second level, attribute "humidity" provides the maximum value in terms of information gain.

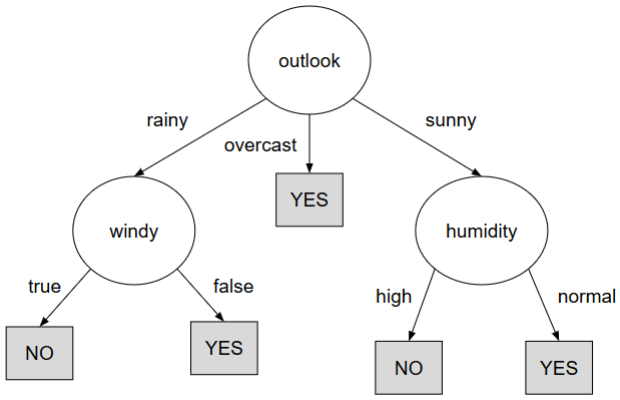

Figure 3: **Decision tree construction. Full tree.**

Hoeffding Tree Algorithm

Hoeffding tree model uses Hoeffding bound [9] for construction and analysis of decision tree. The Hoeffding tree is capable of learning from massive data streams with assumption that the distribution generating examples do not change over time i.e. data distribution is stationary. Hoeffding bound is used to decide the number of instances to be run in order to achieve a certain

level of confidence. Consider a real-valued random variable r whose range is $[0, R]$. Suppose we have made n independent observations of this variable, and computed their mean $\hat{r}$. The Hoeffding bound states that, with probability $1-\delta$, the true mean of the variable is at least $\hat{r}-\epsilon$, where

$$\epsilon = \sqrt{\frac{R^2 \ln \frac{1}{\delta}}{2n}} \tag{7}$$

Using the above concept, the goal is to ensure that, with high probability, the attribute chosen using n examples, where n is as small as possible, is the same that would be chosen using infinite examples. Let G be true information gain and be information gain measured on a subset of data. Assume G is to be maximized, and let X_a be the attribute with highest observed estimated $\hat{G}$ after seeing n examples, and X_b be the second-best attribute. Let $\triangle\hat{G} = \hat{G}(X_a) - \hat{G}(X_b) \geqslant 0$ be the difference between their observed heuristic values. Then given desired δ, the Hoeffding bound guarantees that X_a is the correct choice with probability $1-\delta$ if n examples have been seen at this node and $\triangle\hat{G} > \epsilon$. In other words, if the observed $\triangle\hat{G} > \epsilon$ then the Hoeffding bound guarantees that the true $\triangle G \geqslant \triangle\hat{G} - \epsilon > 0$ with probability $1-\delta$ and therefore that X_a is indeed the best attribute with probability $1-\delta$.

Algorithm: Hoeffding Tree

Input: Let $\hat{G}$ be information gain function

Let ϵ be Hoeffding bound

Let HT be streaming Hoeffding Tree

Let S be a small set of labeled data

Initialize $HT(S)$ in batch mode

For each sample s:

Sort s into leaf l using HT

Update counters of each attribute in l

Let X_i be ith attribute

Compute $\hat{G}_l(X_i)$ for each attribute

Let X_a be attribute with highest $\hat{G}_l$

Let X_b be attribute with second-highest $\hat{G}_l$

If $\hat{G}_l(X_a)$ - $\hat{G}_l(X_b) > \epsilon$

Replace l with decision node that splits on feature variable X_a

Add a leaf for each branches of the split

The algorithm above presents high-level algorithm for Hoeffding tree and this algorithm forms the basis for our work. The algorithm uses information gain heuristic and the hyper parameters can be chosen by cross validation approach.

CHAPTER 4. FRUGAL HOEFFDING TREES

In order to achieve a high accuracy, supervised machine learning models need a lot of data and supervised streaming machine learning models are not the exception. This means that a lot of labeling effort is needed to achieve a high accuracy permormance.

One of the approach to reduce labeling complexity is utilizing confidence of prediction that the model provides. For instance, classification model provides probability estimate for each class given an input sample. Looking at these estimates, the model can provide very high probability estimate for a certain class. This knowledge can be used as an indicator of prediction confidence. Practically, probability estimates with high probability difference tend to be true answer. Therefore asking for true answers to compare is unnecessary work. This concept can be used in other supervised ML models as long as the model provides similar performance.

In this section we present a modification to the Hoeffding tree shown in the previous section. As an instance arrives, the original Hoeffding tree model produces probability estimates for each class given an instance. According to these probabilities a given instance is classified to a class where it has the highest probability value. Looking at these probability estimates we are also interested in the difference of first highest probability value and second highest probability value. This difference can be used as a numerical measurement for prediction confidence.

Let's consider the following example. Assume a binary classification problem and for an incoming sample "A" the algorithm predicts that 80% that the sample belongs to $class_1$ and 20% belongs to $class_2$. Also, for a sample "B" the model provides 40% and 60% probability estimates respectively. The probability difference of sample "A" is 60% and 20% for sample "B". If the difference is high enough then the input sample is being correctly classified. Updating the model statistics by input samples that have high probability difference has less influence on the model's behavior. Instead we are interested in input samples that are difficult to classify and therefore such input samples most likely to have less probability difference.

Algorithm: Frugal Hoeffding Tree

Input: L - true labels

α - probability threshold

Let HT be streaming Hoeffding Tree

Let S be a small set of labeled data

Initialize $HT(S)$ in batch mode

For each unlabeled sample s:

$P =$ predict $HT(s)$, vector of probabilities with one element for each class

let p_1 be first highest probability in P

let p_2 be second highest probability in P

if $p_1 - p_2 < \alpha$:

$label = L(s)$

train $HT(s, label)$

Algorithm below illustrates abstract algorithm of the Frugal Hoeffding tree. In this study we selected several values for α and set α equally for all leaves in order to demonstrate effect of α in reducing labeling complexity however further work is needed for efficient α selection. Note choosing α is important since it is the hyper parameter to control the number of labeled samples to be needed. One can note that if $\alpha = 1$ Frugal Hoeffding tree would become the original Hoeffding tree. Generally we would like to minimize α and it can be chosen with respect to the number of classes. Alternative way of α selecting is updating the value of α with respect to classification error rate in the corresponding leaf.

In order to reduce the labeling effort in Frugal Hoeffding tree, the model is updated by only uncertain observations. As we defined above uncertain observations are the ones that are difficult to classify and therefore those inputs need to be labeled by experts. Probability threshold α can be selected using cross validation and in the experimental study we show that as the model trains the number of observations that need to be labeled decreases dramatically.

CHAPTER 5. EXPERIMENTAL STUDY

Datasets

Adult Data Set [10]. This dataset is for predicting wage of a person. The dataset is taken from UC Irvine Repository and consists of around 45,000 labeled records The problem can be formulated as a classification problem by prediction whether income is above fifty thousand dollars based on given attributes. The dataset consist of 14 (fourteen) attributes including categorical and numeric features.

Covertype Data Set [11]. The dataset is predicting forest cover type using cartographic variables. The dataset includes areas in Roosevelt National Forest of northern Colorado and is available from UC Irvine Repository. The classification problem can be established by predicting 6 types of covers using 54 (fifty four) numerical and categorical attributes. The total amount of records is around 495,000.

KDD Cup 1999 Data Set [12]. This is a large dataset consisting of benign connections and attack intrusions simulated in a military network environment. The dataset was used for The Third International Knowledge Discovery and Data Mining Tools Competition and available at UC Irvine Repository. The size of dataset is about 4,898,000 records consisting of 42 (forty two) categorical and numeric attributes.

Experimental Setup

Experiment 1. We compare the Hoeffding tree with CART [3] and Random Forest [13] models. We measure their performances and compare them in terms of precision, recall and error rate. These algorithms are run in a batch mode in order to keep fairness in comparison.

Experiment 2. We run the original Hoeffding tree where it trains incrementally. During the training, before we feed the model with new labeled train data we evaluate current performance

of the model. We evaluate the model according to precision, recall and error rate. Also, we run Frugal Hoeffding tree and evaluate its performance according to precision, recall and error rate.

Experiment 3. In the third experiment we evaluate the labeling effort. We measure the number of labeled data used during the training stage in order to achieve similar performance as it would use all available labeled training data. By evaluating in this way, we can show that with less labeled data we can achieve similar performance.

Performance Metrics

In order to evaluate the models, we use the following metrics: precision, recall, error rate and the number of labeled data. Error rate is the number of incorrect predictions made as a ratio of all predictions made. This is the most common evaluation metric for classification problems.

$$ErrorRate = \frac{FP + FN}{P + N} \tag{8}$$

Precision-Recall is a useful measure of success of prediction when the classes are very imbalanced. In information retrieval, precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. Precision is defined as the number of true positives (TP) over sum of the number of true positives and the number of false positives (FN) as

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

while recall is defined as the number of true positives over sum of the number of true positives and the number of false negatives (FN) and measured as

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

Hoeffding Tree Performance

As the first experimental case we use the Adult dataset. Table 3 shows the performance results of random forest, CART and the original Hoeffding tree. In order to compare the models fairly all three models firstly train on 35K of samples and then are tested on 10K of samples. Random forest model outperforms CART and Hoeffding tree. The precision value of random forest is around 74% whereas CART and Hoeffding tree are close to 62% and 71% respectively. One can notice that Hoeffding tree has more precision rate then CART. However in term of recall metric, Hoeffding tree is less efficient having only 53% whereas random forest and CART have similar results landing close to 63%. Accuracy of random forest is the lowest as 15% and Hoeffding tree has 17% outperforming CART that has 19% of error rate results.

Table 3: **Adult Data Set. Model Comparison. Average result of seven trees.**

	Precision	Recall	Error Rate
Random Forest	74.3%	63%	14.6%
CART	61.6%	63%	18.9%
Hoeffding Tree	71.1%	53%	16.9%

In the following three figures, we compare performances of the original Hoeffding tree and Frugal Hoeffding tree. Performing testing and training routine using 44 mini-batches and each mini-batch contains 1K of samples. After testing current performance of both models we then feed the models with the given mini-batch. However the proposed Frugal Hoeffding tree is feed only by samples that the model is uncertain in its prediction. Figure 4 shows that we have fairly similar precision rate results. Both models' results converge around 75%. We have fairly similar recall rate results in Figure 5. Both models' results converge close to 60%.

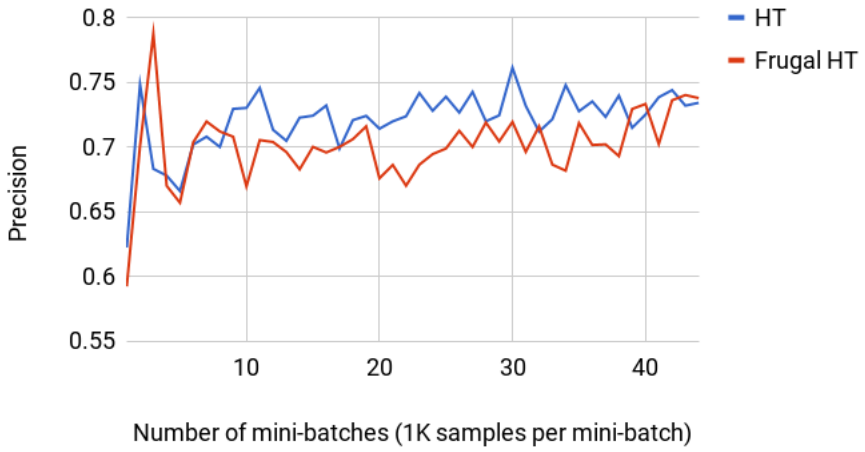

Figure 4: **Adult Data Set. Precision performance per 1K samples. Blue line is precision result of Hoeffding Tree. Red line is precision result of Frugal Hoeffding Tree.**

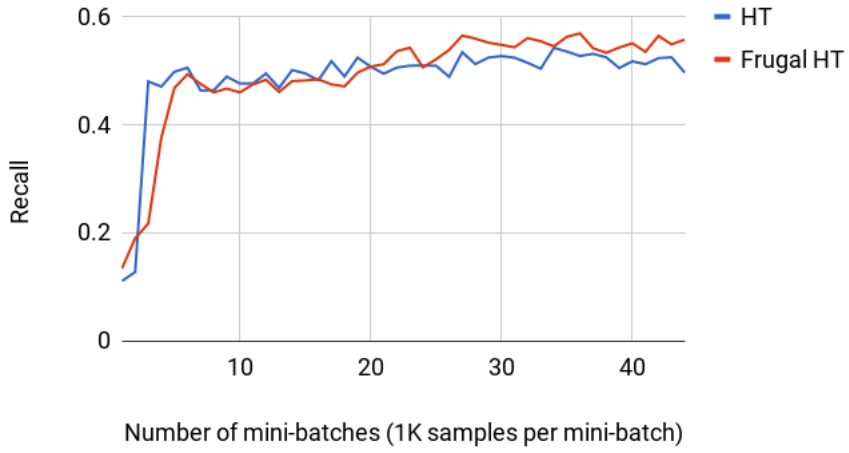

Figure 5: **Adult Data Set. Recall performance per 1K samples. Blue line is recall result of Hoeffding Tree. Red line is recall result of Frugal Hoeffding Tree.**

Figure 6 shows classification accuracy of the models. Here we have also similar drops in terms of classification error and they converge at around 15%.

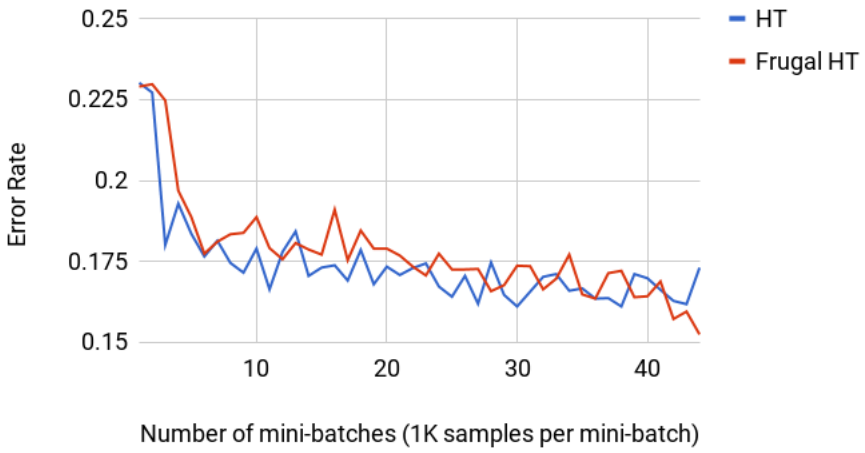

Figure 6: **Adult Data Set. Classification error per 1K samples. Blue line is error rate result of Hoeffding Tree. Red line is error rate result of Frugal Hoeffding Tree.**

Figure 7 shows the number of samples used in each mini batch. As you may note that in every mini batch only around 500 of samples are used.

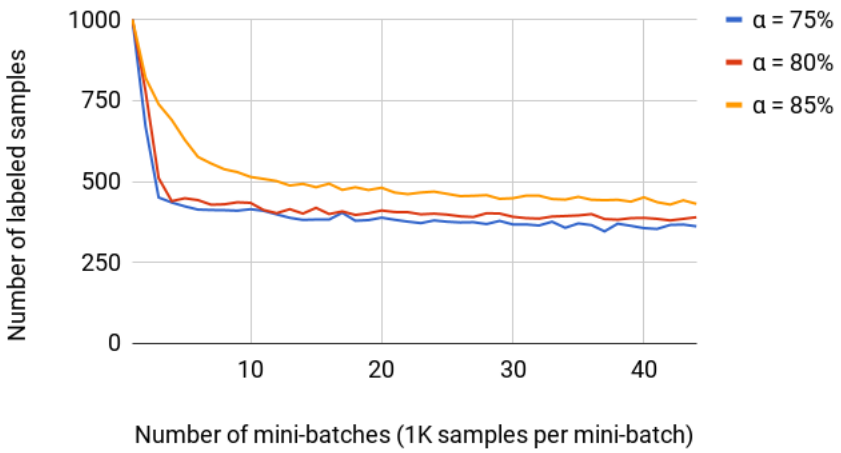

Figure 7: **Adult Data Set. Labeling Complexity. Number of labeled samples used per 1K samples. Blue line is labeling complexity result of Hoeffding Tree. Red line is labeling complexity result of Frugal Hoeffding Tree.**

As the second experimental case we use the Cover dataset. Table 4 shows the performance results of random forest, CART and the original Hoeffding tree. In order to compare the models fairly all three models firstly train on 400K of samples and then are tested on 95K of samples. In this experiment Hoeffding tree performance is below than the other two models' performance. The precision of random forest and CART are 96.2% and 95.9% respectively, whereas Hoeffding tree has 84.7%. In term of recall metric, Hoeffding tree is also less efficient having only 87.7% whereas random forest and CART have 97.9% and 96% respectively. Accuracy of random forest is the lowest as 3.4% and CART has 4.6% outperforming Hoeffding tree that has 16.1% of error rate result.

Table 4: **Covertype Data Set. Model Comparison. Average result of seven trees.**

	Precision	Recall	Error Rate
Random Forest	96.2%	97.9%	3.4%
CART	95.9%	96%	4.6%
Hoeffding Tree	84.7%	87.7%	16.1%

In the following three figures, we compare performances of the original Hoeffding tree and Frugal Hoeffding tree. Performing testing and training routine using 48 mini-batches and each mini-batch contains 10K of samples. Figure 8 shows that we have fairly similar precision rate results. Both models' results didn't converge and therefore it would most likely improve providing more data. We have fairly similar recall rate results in Figure 9. This dataset wasn't enough for the convergence thus we used even larger dataset in the next experiment.

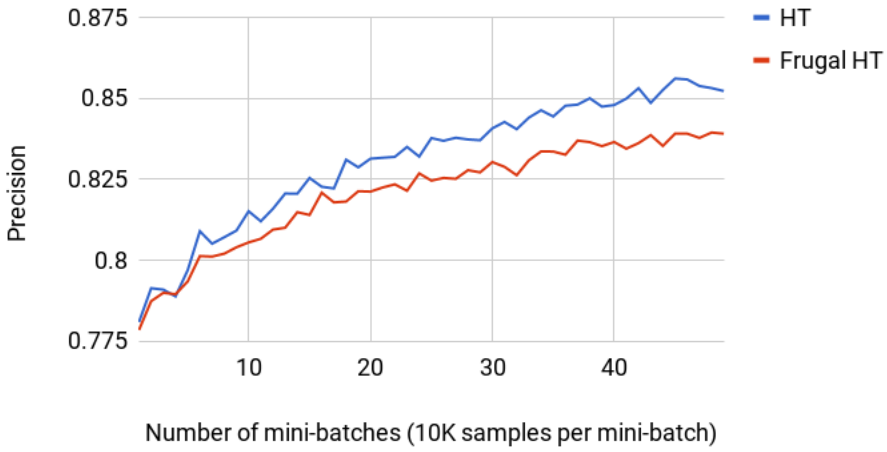

Figure 8: **Covertype Data Set. Precision performance per 10K samples. Blue line is precision result of Hoeffding Tree. Red line is precision result of Frugal Hoeffding Tree.**

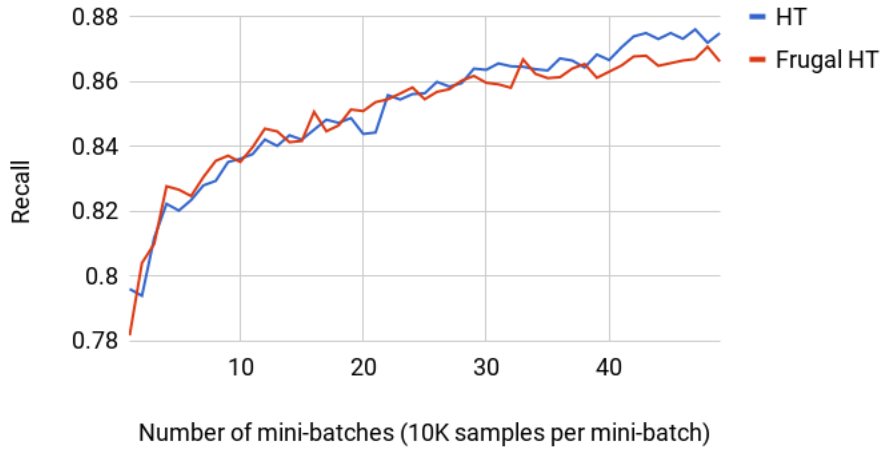

Figure 9: **Covertype Data Set. Recall performance per 10K samples. Blue line is recall result of Hoeffding Tree. Red line is recall result of Frugal Hoeffding Tree.**

Figure 10 shows classification accuracy of the models. Here we have also similar drops in terms of classification error. The error rate results for both models are below 20% after 40th mini-batch.

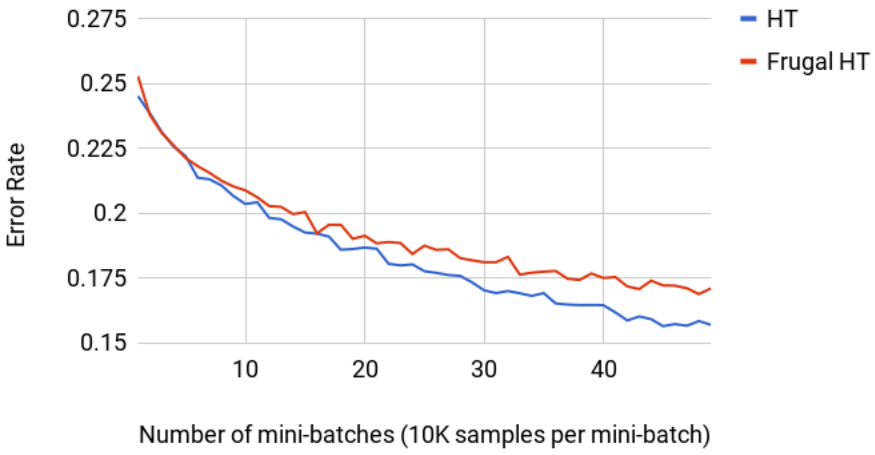

Figure 10: **Covertype Data Set. Classification error per 10K samples. Blue line is error rate result of Hoeffding Tree. Red line is error rate result of Frugal Hoeffding Tree.**

Figure 11 shows the number of samples used in each mini batch. As you may note that in every mini batch the number of used labeled samples monotonically decreases.

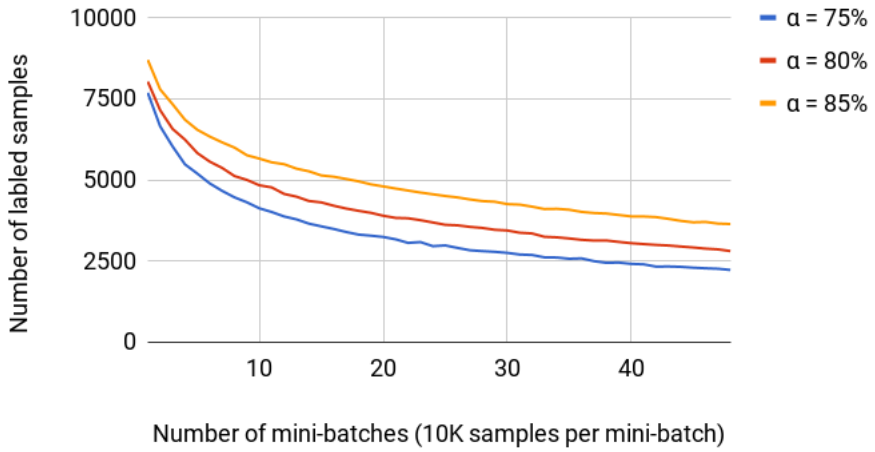

Figure 11: **Covertype Data Set. Labeling Complexity. Number of labeled samples used per 10K samples. Blue line is labeling complexity result of Hoeffding Tree. Red line is labeling complexity result of Frugal Hoeffding Tree.**

As the third experimental case we use the KDD dataset. Table 5 shows the performance results of random forest, CART and the original Hoeffding tree. In order to compare the models fairly all three models firstly train on 4M of samples and then are tested on 898K of samples. The performances of all models are very high and close to each other. One of the reasons is having very large dataset for the evaluation.

Table 5: **KDD Data Set. Model Comparison. Average result of seven trees.**

	Precision	Recall	Error Rate
Random Forest	99.924%	99.909%	0.1329%
CART	99.921%	99.908%	0.1363%
Hoeffding Tree	99.898%	99.877%	0.1796%

In the following three figures, we compare performances of the original Hoeffding tree and Frugal hoeffding tree. Performing testing and training routine using 103 mini-batches and each mini-batch contains 50K of samples. Figure 12 shows that we have fairly similar precision rate results. Both models' results converges similarly. We have fairly similar recall rate results in Figure 13. Both models' results converges close to 99%.

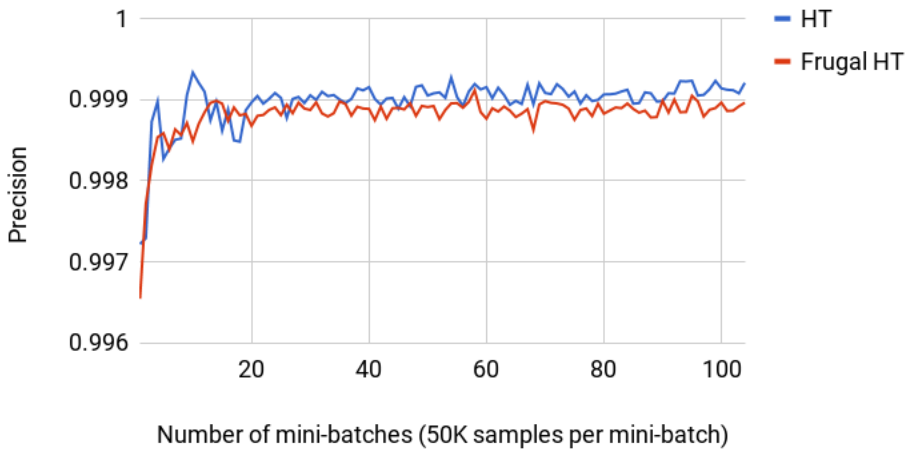

Figure 12: **KDD Data Set. Precision performance per 50K samples. Blue line is precision result of Hoeffding Tree. Red line is precision result of Frugal Hoeffding Tree.**

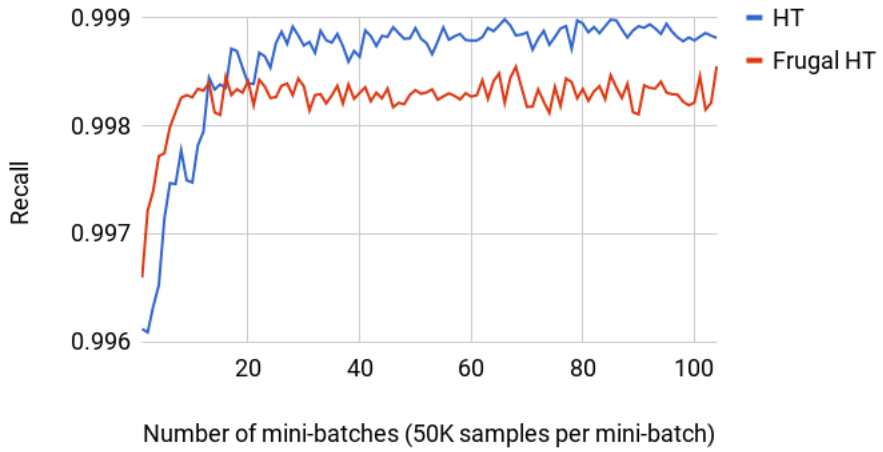

Figure 13: **KDD Data Set. Recall performance per 50K samples. Blue line is recall result of Hoeffding Tree. Red line is recall result of Frugal Hoeffding Tree.**

Figure 14 shows classification accuracy of the models. Here we have also similar drops in terms of error rates. They both converge similarly.

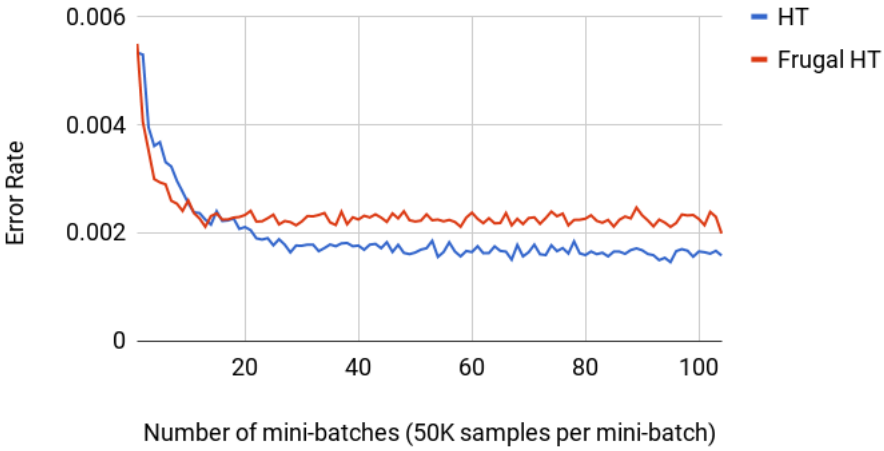

Figure 14: **KDD Data Set. Classification error per 50K samples. Blue line is error rate result of Hoeffding Tree. Red line is error rate result of Frugal Hoeffding Tree.**

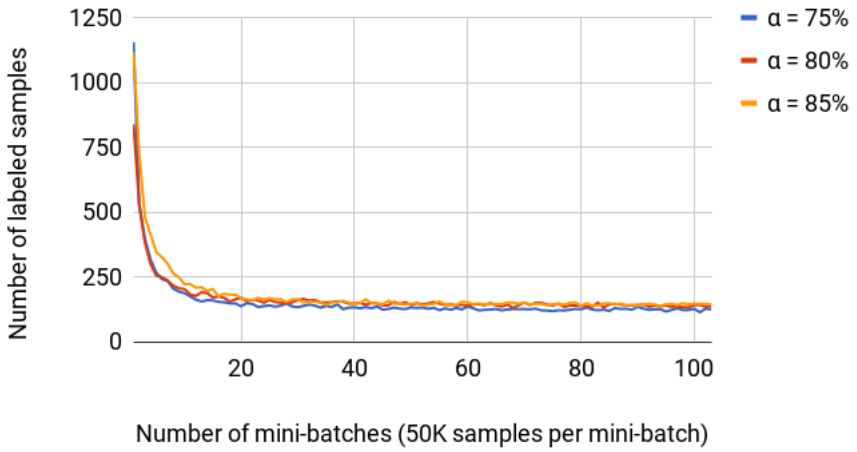

Figure 15: **KDD Data Set. Labeling Complexity. Number of labeled samples used per 50K samples. Blue line is labeling complexity result of Hoeffding Tree. Red line is labeling complexity result of Frugal Hoeffding Tree.**

Figure 15 shows the number of samples used in each mini batch. As you may note that in every mini batch less than 300 (three hundred) of samples are used whereas the original Hoeffding tree use 50K per mini batch.

To summarize the experiments, the main contribution is the observation that possibility of reduction of labeling complexity by using probability estimates of the Hoeffding tree. It is important to understand the impact of the smoothing (see equation 2) and the relationship between the quality of probability estimates produced by the original Hoeffding tree. By using the differences of the probability estimates we achieve the following. The performance of Frugal Hoeffding tree is the same as the Hoeffding tree. The Frugal Hoeffding tree utilizes lesser amount of labeled data to achieve the same performance result.

CHAPTER 6. CONCLUSION

This work evaluated the Hoeffding tree model, a method for learning online from the high-volume data streams. Hoeffding trees have strong guarantees of high asymptotic similarity to the corresponding batch trees. Empirical studies show its effectiveness in taking advantage of massive numbers of examples. In addition, we proposed Frugal Hoeffding tree, a modification of Hoeffding tree that uses less number of labeled data providing similar performance results as the original Hoeffding tree.

REFERENCES

[1] R. Quinlan, "Induction of decision trees," p. 81–106, Proc. Ninth ACM SIGKDD International Conference Knowledge Discovery and Data Mining, 1986.

[2] R. Quinlan, "C4.5: programs for machine learning," Morgan Kaufmann Publishers Inc, 1993.

[3] J. Breiman, L. Olshen, R. Friedman, and C. J. Stone, "Classification and regression trees," Wadsworth International Group, 1984.

[4] P. Domingos and G. Hulten, "Mining high-speed data streams," In Proceedings of the ACM Conference on Knowledge and Data Discovery (SIGKDD), 2000.

[5] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," In Proceedings of the ACM Conference on Knowledge and Data Discovery (SIGKDD), 2001.

[6] R. Jin and G. Agrawal, "Efficient decision tree construction on streaming data," Proc. Ninth ACM SIGKDD International Conference Knowledge Discovery and Data Mining, 2003.

[7] M. Mehta, A. Agrawal, and J. Rissanen, "Sliq:a fast scalable classifier for data mining," In Proceedings of the Fifth International Conference on Extending Database Technology, 1996.

[8] J.C.Shafer, R.Agrawal, and M.Mehta, "Sprint: A scalable parallel classifier for data mining," In Proceedings of the Twenty-Second International Conference on Very Large Databases, 1996.

[9] W. Hoeffding, "Probability inequalities for sums of bounded random variables," pp. 18–30, Journal of the American Statistical Association, 1963.

[10] U. Repository, "Adult data set." `https://archive.ics.uci.edu/ml/datasets/Adult`. accessed on March 2018.

[11] U. Repository, "Covertype data set." `https://archive.ics.uci.edu/ml/datasets/covertype`. accessed on March 2018.

[12] U. Repository, "Kdd cup 1999 data set." https://archive.ics.uci.edu/ml/datasets/KDD+Cup+1999+Data. accessed on March 2018.

[13] L. Breiman, "Random forests," p. 45:15–32, Machine Learning, 1984.