

# Inductive Learning in Less Than One Sequential Data Scan

Wei Fan, Haixun Wang, and Philip S. Yu

IBM T.J.Watson Research  
Hawthorne, NY 10532

{weifan, haixun, psyu}@us.ibm.com

Shaw-Hwa Lo

Statistics Department, Columbia University  
New York, NY 10027

slo@stat.columbia.edu

## Abstract

Most recent research of scalable inductive learning on very large dataset, decision tree construction in particular, focuses on eliminating memory constraints and reducing the number of sequential data scans. However, state-of-the-art decision tree construction algorithms still require multiple scans over the data set and use sophisticated control mechanisms and data structures. We first discuss a general inductive learning framework that scans the dataset exactly once. Then, we propose an extension based on Hoeffding's inequality that scans the dataset less than once. Our frameworks are applicable to a wide range of inductive learners.

## 1 Introduction

Most recent research on scalable inductive learning over very large dataset focuses on eliminating memory-constraints and reducing the number of sequential data scans (or the total number of times that the training file is accessed from secondary storage), particularly for decision tree construction. State-of-the-art decision tree algorithms (SPRINT [Shafer *et al.*, 1996], RainForest [Gehrke *et al.*, 1998], and later BOAT [Gehrke *et al.*, 1999] among others) still scan the data multiple times, and employ rather sophisticated mechanisms in implementation. Most recent work [Hulten and Domingos, 2002] applies Hoeffding inequality to decision tree learning on streaming data in which a node is reconstructed iff it is statistically necessary. Besides decision tree, there hasn't been much research on reducing the number of data scans for other inductive learners. The focus of this paper is to propose a *general approach* for a wide range of inductive learning algorithms to scan the dataset *less than once* from the secondary storage. Our approach is applicable not only to decision trees but also to other learners, e.g., rule and naive Bayes learners.

Ensemble of classifiers has been studied as a general approach for scalable learning. Previously proposed meta-learning [Chan, 1996] reduces the number of data scans to 2. However, empirical studies have shown that the accuracy of the multiple model is sometimes lower than respective single model. Bagging [Breiman, 1996] and boosting [Freund and Schapire, 1997] are not scalable since both methods scan the dataset multiple times. Our proposed method scans the

dataset less than once and has been empirically shown to have higher accuracy than a single classifier.

Based on averaging ensemble, we propose a statistically-based multiple model inductive learning algorithm that scans the dataset less than once. Previous research [Fan *et al.*, 2002b] on averaging ensemble has shown that it is more efficient and accurate than both bagging and meta-learning. In this paper, we apply Hoeffding inequality to estimate the probability that the partial and complete models are equal in accuracy. When the probability is higher than a threshold, the algorithm stops model construction and returns the current model, resulting in less than one sequential scan of the dataset. Our objective is completely different from [Hulten and Domingos, 2002] on determining whether to change the shape of a decision tree. Unlike previous research [Hulten and Domingos, 2002; Gehrke *et al.*, 1999], our algorithm is not limited to decision tree, but is applicable to a wide range of inductive learners. When it is applied to decision tree learning, the accuracy is higher than a single decision tree. Another advantage is that the ensemble reduces the asymptotic complexity of the algorithm besides scanning less data. One important distinction is that we are interested in reducing sequential scan from secondary storage. Once the data can be held entirely in memory, the base learning algorithm is allowed to scan the data *in memory* multiple times.

## 2 One Scan

We first describe a strawman algorithm that scans the data set exactly once, then propose the extension that scans the data set less than once. The strawman algorithm is based on probabilistic modeling.

### 2.1 Probabilistic Modeling

Suppose  $p(\ell_i|x)$  is the probability that  $x$  is an instance of class  $\ell_i$ . In addition, we have a benefit matrix  $b[\ell_{i'}, \ell_i]$  that records the benefit received by predicting an example of class  $\ell_{i'}$  to be an instance of class  $\ell_i$ . For traditional accuracy-based problems,  $\forall i, b[\ell_i, \ell_i] = 1$  and  $\forall i' \neq i, b[\ell_{i'}, \ell_i] = 0$ . For cost-sensitive application such as credit card fraud detection, assume that the overhead to investigate a fraud is \$90 and  $y(x)$  is the transaction amount, then  $b[\text{fraud}, \text{fraud}] = y(x) - \$90$  and  $b[\neg \text{fraud}, \text{fraud}] = -\$90$ . Using benefit matrix and probability, the expected benefit received by pre-

dicting  $x$  to be an instance of class  $\ell_i$  is

$$\text{Expected Benefit: } e(\ell_i|x) = \sum_{\ell_{i'}} b[\ell_{i'}, \ell_i] \cdot p(\ell_{i'}|x) \quad (1)$$

Based on optimal decision policy, the best decision is the label with the highest expected benefit:

$$\ell_{\max} = \operatorname{argmax}_{\ell_i} e(\ell_i|x) \quad (2)$$

Assuming that  $\ell(x)$  is the true label of  $x$ , the accuracy of the decision tree on a test data set ST is

$$\text{Accuracy: } A = \sum_{x \in \text{ST}} b[\ell(x), \ell_{\max}] \quad (3)$$

For traditional accuracy-based problems,  $A$  is always normalized by dividing  $|\text{ST}|$ ; for cost-sensitive problems,  $A$  is usually represented in some measure of benefits such as dollar amount. For cost-sensitive problems, we sometimes use ‘‘total benefits’’ to mean accuracy.

## 2.2 The Strawman Algorithm

The strawman algorithm is based on averaging ensemble [Fan *et al.*, 2002b]. Assume that a data set  $S$  is partitioned into  $K$  disjoint subsets  $S_j$  with equal size. A base level model  $\mathcal{C}_j$  is trained from each  $S_j$ . Given an example  $x$ , each classifier outputs individual expected benefit based on probability  $p_j(\ell_{i'}|x)$

$$e_j(\ell_i|x) = \sum_{\ell_{i'}} b[\ell_{i'}, \ell_i] \cdot p_j(\ell_{i'}|x) \quad (4)$$

The averaged expected benefit from all base classifiers is therefore

$$E_K(\ell_i|x) = \frac{\sum_1^K e_j(\ell_i|x)}{K} \quad (5)$$

We then predict the class label with the highest expected return as in Eq[2].

$$\text{Optimal Decision: } L_K = \operatorname{argmax}_{\ell_i} E_K(\ell_i|x) \quad (6)$$

The obvious advantage is that the strawman algorithm scans the dataset exactly once as compared to two scans by meta-learning and multiple scans by bagging and boosting. In previous research [Fan *et al.*, 2002b], the accuracy by the strawman algorithm is also significantly higher than both meta-learning and bagging. [Fan *et al.*, 2002b] explains the statistical reason why the averaging ensemble is also more likely to have higher accuracy than a single classifier trained from the same dataset.

## 3 Less Than One Scan

The less-than-one-scan algorithm returns the current ensemble with  $k$  ( $< K$ ) number of classifiers when the accuracy of current ensemble is the same as the complete ensemble with high confidence. For a random variable  $y$  in the range of  $R = a - b$  with observed mean of  $\bar{Y}$  after  $n$  observations, without any assumption about the distribution of  $y$ , Hoeffding’s inequality states that with probability  $\geq p$ , the error of  $\bar{Y}$  to the true mean is at most

$$\epsilon_n = R \left( \frac{1}{2n} \ln \left( \frac{1}{1-p} \right) \right)^{\frac{1}{2}} \quad (7)$$

**Train**( $S, S_V, K, p$ )

**Data** : training set  $S$ , validation set  $S_V$ , partition number  $K$ , confidence  $p$

**Result** : multiple model with size  $k \leq K$

**begin**

partition  $S$  into  $K$  disjoint subsets of equal size

$\{S_1, \dots, S_K\}$ ;

train  $\mathcal{C}_1$  from  $S_1$ ;

test  $\mathcal{C}_1$  on  $S_V$ ;

$k \leftarrow 1$ ;

**while**  $k \leq K$  **do**

train  $\mathcal{C}_k$  from  $S_k$ ;

test  $\mathcal{C}_k$  on  $S_V$ ;

$\forall \ell_i$ , compute Hoeffding error  $\epsilon_k(\ell_i)$  (Eq[8]);

confidence\_satisfied  $\leftarrow$  true;

**for**  $x \in S_V$  **do**

$\forall \ell_i$ , compute  $E(\ell_i|x)$ ;

$E(\ell_a|x)$  is the highest and  $E(\ell_b|x)$  is the second highest;

**if**  $E(\ell_a|x) - E(\ell_b|x) \leq \epsilon_k(\ell_a) + \epsilon_k(\ell_b)$  **then**

confidence\_satisfied  $\leftarrow$  false;

**break**;

**end**

**end**

**if** confidence\_satisfied **then**

**return**  $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$

**else**

$k \leftarrow k + 1$

**end**

**end**

**return**  $\{\mathcal{C}_1, \dots, \mathcal{C}_K\}$ ;

**end**

**Algorithm 1:** Less than one data scan

For finite population of size  $N$ , the adjusted error is

$$\epsilon_n = R \left( \frac{1-f}{2n} \ln \left( \frac{1}{1-p} \right) \right)^{\frac{1}{2}} \quad \text{where } f = \frac{n}{N} \quad (8)$$

The range  $R$  of expected benefit for class label  $\ell_i$  can be found from the index to the data, or predefined. When  $k$  base models are constructed, the Hoeffding error  $\epsilon_k$  can be computed by using Eq[8]. For data example  $x$ , assume that  $E(\ell_a|x)$  is the highest expected benefit and  $E(\ell_b|x)$  is the second highest,  $\epsilon_k(\ell_a)$  and  $\epsilon_k(\ell_b)$  are the Hoeffding errors. If  $E(\ell_a|x) - \epsilon_k(\ell_a) > E(\ell_b|x) + \epsilon_k(\ell_b)$  or  $E(\ell_a|x) - E(\ell_b|x) > \epsilon_k(\ell_a) + \epsilon_k(\ell_b)$ , with confidence  $\geq p$ , the prediction on  $x$  by the complete multiple model and the current multiple model is the same. Otherwise, more base models will be trained. The algorithm is summarized in Algorithm 1.

### Scan of validation set $S_V$

If an example  $x$  satisfies the confidence  $p$  when  $k$  classifiers are computed, there is no utility to check its satisfaction when more classifiers are computed. This is because that an ensemble with more classifiers is likely to be a more accurate model. In practice, we can only read and keep one example  $x$  from the validation set in memory at one time. We only read a new

instance from the validation set if the current set of classifiers satisfy the confidence test. In addition, we keep only the predictions on one example at any given time. This guarantees that the algorithm scans the validation dataset once with nearly no memory requirement.

### Training Efficiency

The extra overhead of the Hoeffding-based less than one scan algorithm is the cost for the base classifiers to predict on the validation set and calculate the statistics. All these can be done in main memory. As discussed above, we can predict on one example from the validation set at any given time. Assume that we have  $k$  classifiers at the end and  $n$  is the size of the validation set, the total number of predictions is approximately  $\frac{n \times k}{2}$  on average. The calculation of both mean and standard deviation can be done incrementally. We only need to keep  $\sum X_i$  and  $\sum X_i^2$  for just one example at anytime and calculate as follows:

$$\bar{X} = \frac{\sum X_i}{k} \quad (9)$$

$$\sigma^2(X) = \frac{\sum X_i^2 - k \cdot \bar{X}^2}{k - 1} \quad (10)$$

The average number of arithmetic operation is approximately  $3 \times \frac{n \times k}{2}$ .

The problem that the proposed algorithm solves is one in which the training set is very large and the I/O cost of data scan is the major overhead. When I/O cost is the bottleneck, the extra cost of prediction and statistical analysis is minimum.

## 4 Experiment

In empirical evaluation, we first compare the accuracy of the complete multiple model (one scan as well as less than one scan) and the accuracy of the single model trained from the same data set. We then evaluate the amount of data scan and accuracy of the less than one scan algorithm as compared to the one scan models. Additionally, we generate a dataset with biased distribution and study the results of the less than one scan algorithm.

### 4.1 Datasets

The first one is the well-known donation data set that first appeared in KDDCUP'98 competition. Suppose that the cost of requesting a charitable donation from an individual  $x$  is \$0.68, and the best estimate of the amount that  $x$  will donate is  $Y(x)$ . Its benefit matrix is:

	predict <i>donate</i>	predict $\neg$ <i>donate</i>
actual <i>donate</i>	$Y(x) - \$0.68$	0
actual $\neg$ <i>donate</i>	-\$0.68	0

As a cost-sensitive problem, the total benefit is the total amount of received charity minus the cost of mailing. The data has already been divided into a training set and a test set. The training set consists of 95412 records for which it is known whether or not the person made a donation and how much the donation was. The test set contains 96367 records for which similar donation information was not published until after the KDD'98 competition. We used the standard training/test set splits to compare with previous results. The feature subsets were based on the KDD'98 winning submission.

To estimate the donation amount, we employed the multiple linear regression method.

The second data set is a credit card fraud detection problem. Assuming that there is an overhead \$90 to dispute and investigate a fraud and  $y(x)$  is the transaction amount, the following is the benefit matrix:

	predict <i>fraud</i>	predict $\neg$ <i>fraud</i>
actual <i>fraud</i>	$y(x) - \$90$	0
actual $\neg$ <i>fraud</i>	-\$90	0

As a cost-sensitive problem, the total benefit is the sum of recovered frauds minus investigation costs. The data set was sampled from a one year period and contained a total of 5M transaction records. We use data of the last month as test data (40038 examples) and data of previous months as training data (406009 examples).

The third data set is the adult data set from UCI repository. For cost-sensitive studies, we artificially associate a benefit of \$2 to class label **F** and a benefit of \$1 to class label **N**, as summarized below:

	predict <b>F</b>	predict <b>N</b>
actual <b>F</b>	\$2	0
actual <b>N</b>	0	\$1

We use the natural split of training and test sets, so the results can be easily replicated. The training set contains 32561 entries and the test set contains 16281 records.

### 4.2 Experimental Setup

We have selected three learning algorithms, decision tree learner C4.5, rule builder RIPPER, and naive Bayes learner. We have chosen a wide range of partitions,  $K \in \{8, 16, 32, 64, 128, 256\}$ . The validation dataset *SV* is the complete training set. All reported accuracy results were run on the test dataset.

### 4.3 Experimental Results

In Tables 1 and 2, we compare the results of the single classifier (which is trained from the complete dataset as a whole), one scan algorithm, and the less than one scan algorithm. We use the original "natural order" of the dataset. Later on in Section 4.4, we use a biased distribution. Each data set under study is treated both as a traditional and cost-sensitive problem. The less than one scan algorithm is run with confidence  $p = 99.7\%$ .

#### Accuracy Comparison

The baseline traditional accuracy and total benefits of the single model are shown in the two columns under "single" in Tables 1 and 2. These results are the baseline that the one scan and less than one scan algorithms should achieve. For the one scan and less than one scan algorithm, each reported result is the average of different multiple models with  $K$  ranging from 2 to 256. In Tables 1 and 2, the results are shown in two columns under accuracy and benefit. As we compare the respective results in Tables 1 and 2, the multiple model either significantly beat the accuracy of the single model or have very similar results. The most significant increase in both accuracy and total benefits is for the credit card data set. The total benefits have been increased by approximately \$7,000 ~ \$10,000; the accuracy has been increased by approximately 1% ~ 3%. For the KDDCUP'98 donation data set, the total

C4.5				
	Single	OneScan	Less Than One	
			accuracy	datascan
Donation	94.94%	94.94%	94.94%	0.61
Credit Card	87.77%	90.37%	90.41%	0.62
Adult	84.38%	85.6%	85.0%	0.76

RIPPER				
	Single	OneScan	Less Than One	
			accuracy	datascan
Donation	94.94%	94.94%	94.94%	0.45
Credit Card	90.14%	91.46%	91.42%	0.56
Adult	84.84%	86.1%	86.0%	0.59

NB				
	Single	OneScan	Less Than One	
			accuracy	datascan
Donation	94.94	94.94%	94.94%	0.51
Credit Card	85.46%	88.64%	88.71%	0.57
Adult	82.86%	84.94%	84.6%	0.61

Table 1: Comparison of single model, one scan ensemble, and less than one scan ensemble for accuracy-based problems

benefit has been increased by \$1400 for C4.5 and \$250 for NB.

We next study the trends of accuracy when the number of partitions  $K$  increases. In Figure 1, we plot the accuracy and total benefits for the credit card data sets, and the total benefits for the donation data set with increasing number of partitions  $K$ . C4.5 was the base learner for this study. As we can see clearly that for the credit card data set, the multiple model consistently and significantly improve both the accuracy and total benefits over the single model by at least 1% in accuracy and \$40000 in total benefits for all choices of  $K$ . For the donation data set, the multiple model boosts the total benefits by at least \$1400. Nonetheless, when  $K$  increases, both the accuracy and total benefits show a slow decreasing trend. It would be expected that when  $K$  is extremely large, the results will eventually fall below the baseline.

Another important observation is that the accuracy and total benefit of the less than one scan algorithm are very close to the one scan algorithm. Their results are nearly identical.

#### Data Scan

In both Tables 1 and 2, we show the amount of data scanned for the less than one scan algorithm. It ranges from 40% (0.4) to about 70% (0.7). The adult dataset has the most amount of data scanned since the training set is the smallest and it requires more data partitions to compute an accurate model. C4.5 scans more data than both RIPPER and NB. This is because we generate the completely unpruned tree for C4.5, and there are wide variations among different models.

In Table 3, we compare the differences in accuracy and amount of training data when the validation set is either read completely by every classifier (under “Batch”) or sequentially only by newly computed base classifiers (under “Seq”) (as discussed in Section 3). Our empirical studies have found that “Batch” mode usually scans approximately 1% to 2% more training data, and the models computed by both methods are nearly identical in accuracy. The extra training data from the

C4.5				
	Single	OneScan	Less Than One	
			benefit	datascan
Donation	\$13292.7	\$14702.9	\$14828	0.71
Credit Card	\$733980	\$804964	\$804914	0.65
Adult	\$16443	\$16435	\$16205	0.77

RIPPER				
	Single	OneScan	Less Than One	
			benefit	datascan
Donation	\$0	\$0	\$0	0.47
Credit Card	\$712541	\$815612	\$815310	0.57
Adult	\$19725	\$19875	\$19615	0.62

NB				
	Single	OneScan	Less Than One	
			benefit	datascan
Donation	\$13928	\$14282	\$14278	0.55
Credit Card	\$704285	\$798943	\$799104	0.59
Adult	\$16269	\$19169	\$16102	0.63

Table 2: Comparison of single model, one scan ensemble, and less than one scan ensemble for cost-sensitive problems

“batch” method is due to the fact that some examples satisfied by previously learned classifiers have high probability, but may not necessarily be satisfied by more base classifiers. However, our empirical studies have shown that the difference in how the validation set is handled doesn’t significantly influence the final model accuracy.

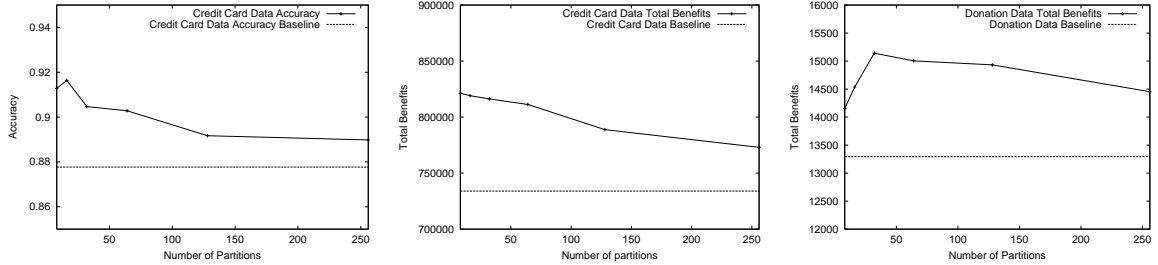
#### 4.4 Biased Distribution

When a data is biased in its distribution, the less than one scan algorithm need to scan more data than uniform distribution to produce an accurate model. With the same amount of datascan, it may not have the same accuracy as uniform distribution. We have created a “trap” using the credit card dataset. We sorted the training data with increasing transaction amount. The detailed results are shown in Table 4(a) and (b). The accuracy (and total benefits) in Table 4(a) are nearly identical to the results of “natural distribution” as reported in Tables 1 and 2. However, the amount of datascan by the less than one scan algorithm is over 0.9 as compared to approximately 0.6 for natural distribution. As shown in Table 4(b), when the datascan is less than 0.9 (the confidence is not satisfied and less one scan will continue to compute more model), the total benefits are much lower. When distribution is biased, the variations in base classifiers’ prediction are wider. It requires more data to compute an accurate model and the less than one scan algorithm is performing in the correct way.

#### 4.5 Training Efficiency

We recorded both the training time of the batch mode single model, and the training time of both the one scan algorithm and less than one scan algorithm *plus* the time to classify the validation set multiple times and statistical estimation. We then computed serial improvement, which is the ratio that the one scan and less than one scan algorithm are faster than training the single model. In Figure 2, we plot results for the credit card dataset using C4.5. Our

Figure 1: Plots of accuracy and total benefits for credit card data sets, and plot of total benefits for donation data set with respect to  $K$  when using C4.5 as the base learner



C4.5				
	Accuracy		Data Scan	
	Batch	Seq	Batch	Seq
Donation	94.94%	94.94%	0.64	0.61
Credit Card	90.39%	90.41%	0.62	0.62
Adult	85.1%	85.0%	0.78	0.76

RIPPER				
	Accuracy		Data Scan	
	Batch	Seq	Batch	Seq
Donation	94.94%	94.94%	0.48	0.45
Credit Card	91.44%	91.42%	0.56	0.55
Adult	85.9%	86.0%	0.62	0.59

NB				
	Accuracy		Data Scan	
	Batch	Seq	Batch	Seq
Donation	94.96%	94.94%	0.54	0.51
Credit Card	88.62%	88.71%	0.59	0.57
Adult	84.84%	84.6%	0.62	0.61

Table 3: Comparison of accuracy and amount of datascan using the batch (or all in memory) and sequential method for accuracy-based problems

training data can fit into the main memory of the machine. Any single classifier algorithm that reduces the number of data scan [Shafer *et al.*, 1996; Gehrke *et al.*, 1998; 1999; Hulten and Domingos, 2002] will not have training time less than this result. As shown in Figure 2, both one scan and less than one scan algorithm are significantly faster than the single classifier, and the less than one scan algorithm is faster than the one scan algorithm.

## 5 Related Work

To scale up decision tree learning, SPRINT [Shafer *et al.*, 1996] generates multiple sorted attribute files. Decision tree is constructed by scanning and splitting attribute lists, which eliminates the need for large main memory. Since each attribute list is sorted. for a data file with  $f$  attributes and  $N$  examples, the total cost to produce the sorted lists is  $f \cdot O(N \cdot \log(N))$ . External sort is used to avoid the need for main memory. Each attribute list has three columns—a unique record number, the attribute value and the class label; the total size of attribute lists is approximately three times the size of the original data set. When a split takes place at a node,

Figure 2: Serial improvement for credit card dataset using one scan and less than one scan



	Accuracy Based		Cost-sensitive	
	Accuracy	DataScan	Benefit	Datascan
C4.5	89.7%	0.95	\$794933	0.96
RIPPER	90%	0.93	\$769344	0.97
NB	86.9%	0.89	\$774854	0.93

(a). Performance for different classifier for biased distribution

DataScan	0.6	0.7	0.8	0.96
Total Benefits	\$561382	\$614575	\$728315	\$794933

(b). Performance of C4.5 with different amount of data scanned under the biased distribution

Table 4: Performance of less than one scan under biased distribution

SPRINT reads the attribute lists at this node and breaks them into  $r$  sets of attribute lists for  $r$  child nodes, for a total of  $f$  file read and  $r \cdot f$  file writes. Later, RainForest [Gehrke *et al.*, 1998] improves the performance of SPRINT by projecting attribute list on each unique attribute value into an AVC-set. The complete AVC-sets of all attributes are called AVC-group. When the AVC-group can be held in main memory, the selection of predictor attribute can be done efficiently. To construct the initial AVC-group, it incurs the same cost as SPRINT to construct initial attribute lists plus one more scan over the sorted lists to project into AVC-sets. Whenever a split happens, RainForest has to access the data file again and reconstruct the AVC-group for child nodes. The exact number of read and write is based on variations of RainForest that is chosen. In the best scenario where the AVC-group of every node in the tree fit in memory, the RF-read version still has to scan the whole data once at each level of the tree and write it into  $r$  files. When this condition is not met, RainForest solves the problem by multiple read and write. More recently, BOAT [Gehrke *et al.*, 1999] constructs a “coarse”

tree from a data sample that can fit into main memory. The splitting criterion in each node of the tree is tested against multiple decision trees trained from bootstrap samples of the sampled data. It refines the tree later by scanning the complete data set, resulting in a total of two complete data read.

Meta-learning [Chan, 1996] builds a tree of classifiers and combine class label outputs from base classifiers. It is based on heuristics and the total number of datascan is two. The improvements by our methods are in many folds. We combine probabilities instead of class labels. The combining technique is straightforward and can estimate the final accuracy prior to full construction. The total amount of sequential scan is less than once. There is a strong statistical reason to support why the multiple model method works [Fan *et al.*, 2002b]. Besides meta-learning, both bagging [Breiman, 1996] and boosting [Freund and Schapire, 1997] have been shown to have higher accuracy than a single model. However, bagging and boosting scan the dataset many times, and are not scalable for large dataset. Previous research [Fan *et al.*, 2002b] has shown the our combining method is more accurate than bagging.

One earlier work to use Hoeffding's inequality is online aggregation [Hellerstein *et al.*, 1997] to estimate the accuracy of aggregate queries. When the confidence about the accuracy of the partial query is sufficient, the user can terminate the query to avoid a complete scan. One recent work using Hoeffding inequality is on building decision tree from streaming data [Hulten and Domingos, 2002]. They keep the number of examples that satisfy or fail the predicate of a node in the tree. When it is confident that the current node is not accurate for the new data, it will be reconstructed. One limitation of [Hulten and Domingos, 2002] is that it is only applicable to decision tree learning. Our ensemble method using Hoeffding's inequality is applicable to a wide range of inductive learners. Empirically, we have applied it to decision tree, rule learner as well as naive Bayes learner. Beside its generality, the proposed one scan and less than one scan algorithms have been shown to have potentially higher accuracy than the single model (existing approaches are single model method [Hulten and Domingos, 2002; Gehrke *et al.*, 1999; Shafer *et al.*, 1996; Gehrke *et al.*, 1998]). Another advantage is the asymptotic complexity using the less than one scan algorithm is approximately  $k \cdot O(\frac{n}{K})$  while computing a single model is still  $O(n)$ . Our empirical studies have shown that both the one scan and less than one scan algorithms are significantly faster than learning a single model. In previous work [Fan *et al.*, 2002a], we have used Central limit theorem (CLT) to implement a concept of progressive modeling where we estimate the range of accuracy of the final model. The learning stops when the estimated accuracy of the final falls within a tolerable range with confidence  $p$ . CLT requires the data distribution to be uniform. Hoeffding inequality has a different implication, in which we estimate the probability that the partial and final models are exactly the same.

## 6 Conclusion

In this paper, we propose two scalable inductive learning algorithms. The strawman multiple model algorithm scans the data set exactly once. We then propose a less than one scan

extension based on Hoeffding's inequality. It returns a partial multiple model when its accuracy is the same as the complete multiple model with confidence  $\geq p$ . Since Hoeffding inequality makes no assumption about the data distribution, the advantage of this method is that the data items can be retrieved sequentially. We have also discussed how to sequentially read the validation set exactly once using minimal memory. We have evaluated these methods on several data sets as both traditional accuracy-based and cost-sensitive problems using decision tree, rule and naive Bayes learners. We have found that the accuracy of all our methods are the same or far higher than the single model. The amount of data scan by the less than one scan algorithms range from 0.45 to 0.7 for the original natural distribution of data. For a significantly biased dataset, the amount of datascan by the less than one scan algorithm is over 0.9. It needs extra data to resolve the bias in data distribution in order to compute an accurate model. In addition, our empirical studies have shown that both methods are significantly faster than computing a single model even when the training data can be held in main memory, and the less than one scan algorithm is faster than the one scan algorithm.

## References

- [Breiman, 1996] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [Chan, 1996] P Chan. *An Extensible Meta-learning Approach for Scalable and Accurate Inductive Learning*. PhD thesis, Columbia University, Oct 1996.
- [Fan *et al.*, 2002a] Wei Fan, Haixun Wang, Philip S Yu, Shawhwa Lo, and Salvatore Stolfo. Progressive modeling. In *Proceedings of Second IEEE International Conference on Data Mining (ICDM2002)*, 2002.
- [Fan *et al.*, 2002b] Wei Fan, Haixun Wang, Philip S Yu, and Salvatore Stolfo. A framework for scalable cost-sensitive learning based on combining probabilities and benefits. In *Second SIAM International Conference on Data Mining (SDM2002)*, April 2002.
- [Freund and Schapire, 1997] Y Freund and R Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Computer and System Sciences*, 55(1):119–139, 1997.
- [Gehrke *et al.*, 1998] Johannes Gehrke, Venkatesh Ganti, and Raghu Ramakrishnan. RainForest: a framework for fast decision construction of large datasets. In *Proceeding of 24th International Conference on Very Large Databases (VLDB'1998)*, 1998.
- [Gehrke *et al.*, 1999] Johannes Gehrke, Venkatesh Ganti, Raghu Ramakrishnan, and Wei-Yin Loh. BOAT-optimistic decision tree construction. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD 1999)*, 1999.
- [Hellerstein *et al.*, 1997] Joseph M. Hellerstein, Peter J. Haas, and Helen J. Wang. Online aggregation. In *Proceedings of ACM SIGMOD International Conference on Management of Data (SIGMOD'97)*, 1997.
- [Hulten and Domingos, 2002] Geolff Hulten and Pedro Domingos. Learning from infinite data in finite time. In *Advances in neural information processing systems*. MIT Press, 2002.
- [Shafer *et al.*, 1996] J Shafer, Ramesh Agrawl, and M Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proceedings of Twenty-second International Conference on Very Large Databases (VLDB-96)*, pages 544–555, San Francisco, California, 1996. Morgan Kaufmann.