Efficiently Determining the Starting Sample Size for Progressive Sampling

Baohua Gu[1], Bing Liu[2], Feifang Hu[3], and Huan Liu[4]

[1] Dept. of Computer Science, National Univ. of Singapore
gubh@comp.nus.edu.sg
[2] Dept. of Computer Science, National Univ. of Singapore
liub@comp.nus.edu.sg
[3] Dept. of Stat. & Applied Prob. National Univ. of Singapore
stahuff@nus.edu.sg
[4] Dept. of Computer Sci. and Eng., Arizona State Univ
hliu@asu.edu

Abstract. Given a large data set and a classification learning algorithm, Progressive Sampling (PS) uses increasingly larger random samples to learn until model accuracy no longer improves. It is shown that the technique is remarkably efficient compared to using the entire data. However, how to set the starting sample size for PS is still an open problem. We show that an improper starting sample size can still make PS expensive in computation due to running the learning algorithm on a large number of instances (of a sequence of random samples before achieving convergence) and excessive database scans to fetch the sample data. Using a suitable starting sample size can further improve the efficiency of PS. In this paper, we present a statistical approach which is able to efficiently find such a size. We call it the *Statistical Optimal Sample Size* (*SOSS*), in the sense that a sample of this size sufficiently resembles the entire data. We introduce an information-based measure of this resemblance (Sample Quality) to define the SOSS and show that it can be efficiently obtained in one scan of the data. We prove that learning on a sample of SOSS will produce model accuracy that asymptotically approaches the highest achievable accuracy on the entire data. Empirical results on a number of large data sets from the UCIKDD repository show that SOSS is a suitable starting size for Progressive Sampling.

1. Introduction

Classification is an important learning task. It is often solved by the decision tree approach [19]. Most decision tree algorithms require the data to reside in main memory. This presents a major problem if the data set is too large to fit in the memory. Even if a large data set is able to fit into memory, running a tree-building algorithm on the whole data can be very expensive. A natural way to overcome this problem is to do sampling [2]. One of the recent works towards improving tree-building efficiency (both in terms of time and memory space required) by sampling is Boat by Gehrke et al [6]. [6] builds an initial decision tree using a small sample and

then refines it via bootstrap to produce exactly the same tree as that would be produced using the entire data. Boat is able to achieve excellent efficiency in tree construction. However, due to the large size of the data, the produced tree can be very complex and large, which makes it hard for human understanding. It has been observed that the tree size often linearly increases with the size of training data, and additional complexity in the tree results in no significant increase in model accuracy [15]. Another recent work on improving the efficiency of tree building for large data sets is Progressive Sampling (PS for short) proposed by Provost et al [17]. By means of a learning curve (see an example learning curve in Figure 1, and related concepts in [4][17]) which depicts the relationship between sample size and model accuracy, PS searches for the optimal model accuracy (the highest achievable on the entire data) by feeding a learning algorithm with progressively larger samples. Assuming a well-behaved learning curve, it will stop at a sample size equal to or slightly larger than the *Optimal Sample Size* (*OSS* for short) corresponding to the optimal model accuracy[1]. [17] shows that PS is more efficient than using the entire data. Besides, it can in fact produce a less complex tree or simpler model (if the real *OSS* is far less than the total data size). Therefore, PS is able to achieve two goals by means of sampling: first, to avoid loading the whole data set into memory; second, to produce good as well as simple model from the data. In this paper, we restrict our attention to PS and aim to improve it further by finding a good starting sample size for it, which is still an open problem.

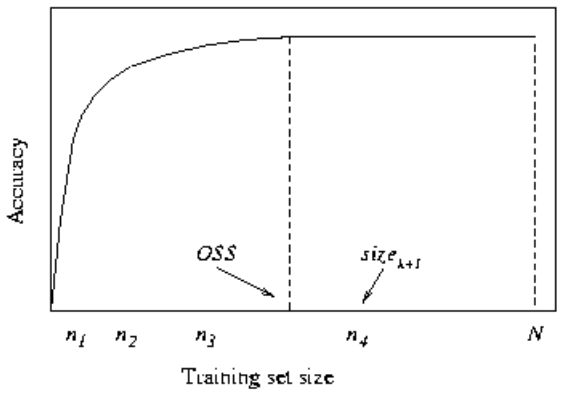

Fig. 1. Learning Curve and Progressive Samples

Obviously, the efficiency of PS gets to the highest when the starting sample size is equal to the *OSS*; and the smaller the difference between the two, the higher the efficiency. The following analysis shows how much benefit can be gained from a proper starting sample size. Suppose we use the simple geometric sampling schedule suggested by [17]. We denote the sample size sequence by $\{n_0, a\times n_0, a^2\times n_0, a^3\times n_0,$

[1] Upon each sample size, PS uses a special procedure to evaluate if the learning accuracy has converged [17]. The evaluation is independent of the previous samples.

...}, where n_0>0 is the starting sample size, a>1 is the increment ratio. If the convergence is detected at the (k+1)-th sample (its size being $size_{k+1}$= $a^k \times n_0 \approx OSS$), then the total size of the previous k samples will be $size_{1..k}$= $n_0 \times (1 + a + a^2 + ... + a^{k-1})$ = $n_0 \times (a^k-1)/(a-1)$ = $(size_{k+1} - n_0)/(a-1)$. When a=2, this is almost the same as the $size_{k+1}$. Given a learning algorithm whose runtime complexity is typically $O(n^c)$ (c>1), in terms of the instance number n used in learning, the computation spent on these pre-convergence samples could be very expensive. Obviously, this extra cost could be significantly reduced if n_0 is set near to the *OSS*. Moreover, as $k = log_a (size_{k+1} / n_0) \approx log_a (OSS / n_0)$, if n_0 is much less than *OSS*, then k, the number of samples needed before convergence, will be large. Note that generating a random sample from a single-table database typically requires scanning the entire table once [18]. Thus a large k will result in considerably high disk I/O cost. Therefore setting a good starting sample size can further improve the efficiency of PS by cutting the two kinds of costs.

In this paper, we find such a size via a statistical approach. The intuition is that a sample with the *OSS* should sufficiently resemble its mother data (the entire data set). We implement this intuition via three steps. First, we define an information-based measure of the resemblance (we call *Sample Quality*). Based on this measure, we then define a *Statistical Optimal Sample Size* (*SOSS* for short). We prove that learning on a sample of the SOSS will produce a model accuracy that is sufficiently close to that of the *OSS*. We show that the *SOSS* can be efficiently determined in one scan of the mother data. Our experiments on a number of UCIKDD [1] data sets show that our approach is effective.

The remainder of this paper is organized as follows. In Section 2, we introduce the measure of sample quality and its calculation. In Section 3, we define *SOSS*, prove a useful theorem and show its calculation. Experimental results and discussions are given in Section 4. Related work is introduced in Section 5. Section 6 concludes the paper.

2. Sample Quality Measure

A sample with the *OSS* should inherit the "property" of its mother data as much as possible. This property can be intuitively interpreted as information. In this work, we make use of Kullback's information measure [11], which is generally called the divergence or deviation, as it depends on two probability distributions and describes the divergence between the two. Below we first briefly describe its related definitions and conclusions, and then introduce our measure of sample quality.

2.1 Underlying Information Theory

Suppose x is a specific value (i.e., an observation) of a generic variable X, and H_i is the hypothesis that X is from a statistical population with generalized probability densities (under a probability measure λ) $f_i(x)$, i = 1, 2, then the information divergence is defined by

$$J(1,2) = \int (f_1(x) - f_2(x)) \log(f_1(x) / f_2(x)) d\lambda(x) \quad (1)$$

According to [11], $J(1,2)$ is a measure of the divergence between hypotheses H_1 and H_2, and is a measure of the difficulty of discriminating between them. Specifically, for two multinomial populations with c values (c categories), if p_{ij} is the probability of occurrence of the j-th value in population i (i=1, 2, and j=1, 2,..., c), then the information divergence is

$$J(1,2) = \sum_{j=1}^{c} (p_{1j} - p_{2j}) \log(p_{1j} / p_{2j}) \quad (2)$$

The information divergence has a good limiting property described in Theorem 1 below (see [11] for its proof), based on which we will prove an important theorem about *SOSS* in Section 3.

Theorem 1 Given a probability density function $f(x)$ and a series of probability density functions $f_n(x)$, where $n \to \infty$, denote the information divergence from $f_n(x)$ to $f(x)$ as $J(f_n(x), f(x))$, we have, if $J(f_n(x), f(x)) \to 0$, then $f_n(x)/f(x) \to 1$ [λ], uniformly. Here [λ] means that the limitation and the fraction hold in a probability measure λ. #

2.2 Definition and Calculation

We borrow the concept of information divergence between two populations and define our sample quality measure below. The idea is to measure the dissimilarity of a sample from its mother data by calculating the information divergence between them.

Definition 1 Given a large data set D (with r attributes) and its sample S, denote the information divergence between them on attribute k as $J_k(S, D)$, (k=1, 2, ..., r), then the sample quality of S is

$$Q(S) = exp(-J) \quad (3)$$

where the averaged information divergence J is

$$J = \frac{1}{r} \sum_{k=1}^{r} J_k(S, D) \quad (4)$$

In calculating $J_k(S, D)$, we treat a categorical (nominal) attribute as a multinomial population. For a continuous (numerical) attribute, we build its histogram (e.g., using a simple discretization), and then treat it also as a multinomial population by taking each bin as a categorical value and the bin size as its frequency. According to [14], the information divergence $J > 0$, therefore $0 < Q \leq 1$, where $Q = 1$ means that no information divergence exists between S and D. The larger the information divergence J, the smaller the sample quality Q; and vice versa. For numerical attributes, if we have prior knowledge about their distributions, further improvement on the calculation of J can be achieved by directly applying them to formula (1) to get the information divergence.

The calculation of Q is straightforward. In one scan of the data, both the occurrences of categorical values and the frequencies of numerical values that fall in

the bins can be incrementally gathered. At the end of the scan, Q can be worked out by using formula (2) for each attribute and then using formula (4) for the averaged J. Therefore the time complexity of calculating sample quality is $O(N)$ (N is the total number of instances or records of the mother data), while the space complexity is $O(r*v)$ (v is the largest number of distinct values or bins of each attribute). Note that a random sample can be obtained also in one scan. Thus we can calculate a sample's quality while generating it.

3. Statistical Optimal Sample Size

From the definition of sample quality, we can observe that the larger the sample size, the higher the sample quality. This is because as sample size increases, a sample will have more in common with its mother data, therefore the information divergence between the two will decrease. We can imagine that when a sample is sufficiently large, its sample quality should be sufficiently close to the mother data thus should sufficiently "resemble" the mother data in distribution. Below we define the *SOSS* and introduce its calculation.

3.1 Definition

We define the Statistical Optimal Sample Size (*SOSS*) as follows:

Definition 2 Given a large data set D, its *SOSS* is the size at which its sample quality Q is sufficiently close to 1. That is, $|1-Q| < \delta$, where $\delta>0$ is given by user. #

Clearly, the *SOSS* only depends on the data set D, while the *OSS* depends on both the data set and the learning algorithm. Therefore, the *SOSS* is not necessarily the *OSS*. However, by the following theorem, we can see that their corresponding model accuracies can be very close.

Given a learning algorithm L and a sufficiently large data set D with probability density function $f(x)$, we take L as an operator mapping $f(x)$ to a real number (namely the model accuracy), i.e., $L : f(x) \rightarrow Acc^*$, where Acc^* is the maximum accuracy obtained on D. Assume that a random sample of the size *OSS* has the probability density function $f_{OSS}(x)$, and a random sample of the size *SOSS* has $f_{SOSS}(x)$. Let the model accuracies on the two samples be Acc_{OSS} and Acc_{SOSS} respectively. We have the following theorem.

Theorem 2 If a random sample S of D has a probability density function $f_S(x)$ and L satisfies: $f_S(x) / f(x) \rightarrow 1[\lambda] \Rightarrow |L(f_S(x)) - L(f(x))| \rightarrow 0$, then $Acc_{OSS} \rightarrow Acc_{SOSS}$.

Proof (Sketch): Suppose we have a series of n random samples of D with incrementally larger sample sizes. Denote the size of the i-th sample as S_i , and the corresponding sample quality of the sample with $Q(S_i)$, i=1, 2, ..., n. According to the definition of *SOSS*, when $S_i \rightarrow SOSS$, $Q(S_i) \rightarrow 1$, i.e., the information divergence of S_i from D is $J(S_i, D) \rightarrow 0$. Applying Theorem 1, we have, $f_{Si}(x) / f(x) \rightarrow 1[\lambda]$. From the condition, we have $|L(f_{Si}(x)) - L(f(x))| \rightarrow 0$. In an asymptotic sense, $L(f_{Si}(x)) \rightarrow L(f_{SOSS}(x)) \rightarrow Acc_{SOSS}$, and $L(f(x)) \rightarrow Acc^* \rightarrow Acc_{OSS}$, that is, $Acc_{SOSS} \rightarrow Acc_{OSS}$. #

The premise of the theorem means that if two data sets are quite similar in their probability distributions, then the learning algorithm should produce quite close model accuracy on them. This is reasonable for a typical learning algorithm. Based on this theorem, we can search for the *SOSS* by measuring sample quality instead of directly searching for the *OSS* by running an expensive learning algorithm. We can also expect that the two are close in terms of model accuracy.

3.2 Calculation

To calculate the *SOSS* of D, we can set m sample sizes S_i spanning the range of $[1, N]$ and compute the corresponding qualities Q_i (i=1,2,...,m). We then draw a sample quality curve (the relationship between sample size and sample quality) using these (S_i , Q_i) points. The *SOSS* is estimated using the curve. To be efficient, we can calculate all samples' qualities at the same time in one sequential scan of D by using the idea of Binomial Sampling [16]. That is, upon reading in each instance or data record, a random number x uniformly distributed on [0.0, 1.0) is generated. If $x<S_i/N$, then corresponding statistics (by counting a categorical value or binning a numerical value) are gathered for the i-th sample. This is repeated for all samples before reading the next instance. We describe the procedure using the pseudo algorithm below.

Pseudo Algorithm *SOSS*
input: a large data set D of size N, m sample sizes $\{S_i \mid i = 1, 2, ..., m\}$;
output: m pairs of (S_i , Q_i);
begin
1. for each instance k in D ($k \in [1, N]$)
   {
     update corresponding statistics for D;
     for each sample $i \in [1, m]$
     {
       $r \leftarrow$ *UniformRandomNumber*(0.0,1.0);
       if $r < S_i/N$, then update corresponding statistics for sample i;
     }
   }
2. for each sample $i \in [1, m]$
   {
     calculate its Q_i and output (S_i , Q_i);
   }
end

It is easy to see that its runtime complexity and the memory needed are m times that of computing sample quality for a single sample. With these (S_i , Q_i) points in hand, we draw the quality curve and decide the *SOSS* in the following way: starting from the first point, we do a linear regression on every l consecutive points (we set l=5 in our experiment), if the 95% confidence interval of the slope of the regressed line includes zero, then the size of the middle point is the SOSS.

4. Experimental Results and Discussions

We evaluate our measure on four large UCIKDD data sets: *adult*, *led*, *census*, and *covtype*. The number of instances in training data/testing data are, 36k/12.8k for *adult*, 100k/50k for *led*, 199.5k/99.8k for *census*, and 400k/181k for *covtype*. The classification algorithm we use is C5.0, the latest improvement of C4.5 [19]. We first draw the learning curve for each data set by running C5.0 on a number of sample sizes from 1% to 100% of all training data. All accuracies are tested against the corresponding testing data. Then we draw the quality curve for each data set by calculating multiple sample qualities in one scan. We set the bin number to be 20 for discretizing numerical attributes[2]. We use 50 sample sizes equally covering [1, *N*].

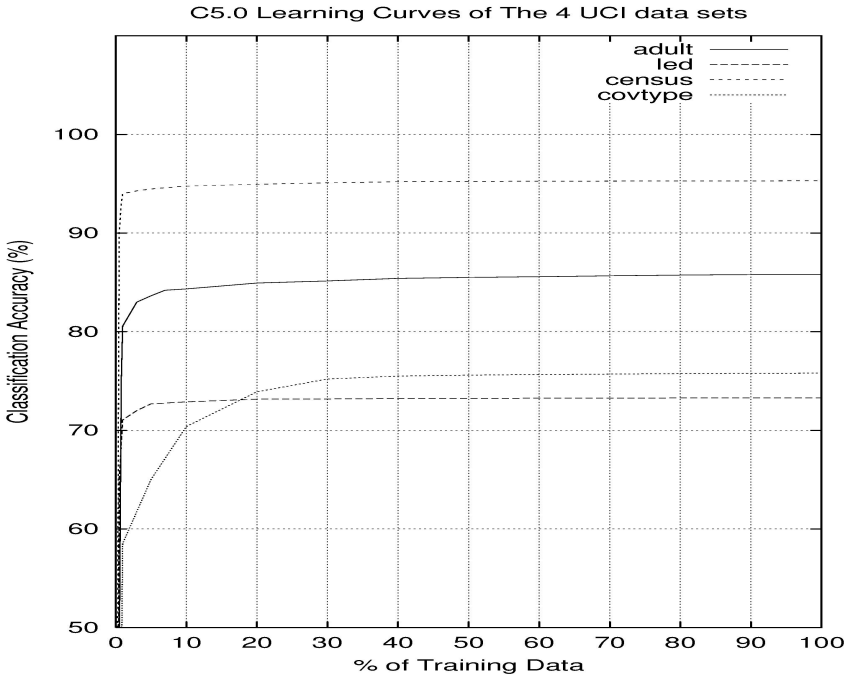

Fig. 2. C5.0 Learning Curves of the four data sets

The learning curves and quality curves (averaged on 10 runs for the four data sets) are shown in Figure 2 and Figure 3 respectively. We decide the *OSS* as the size corresponding to 99.5%×*Acc**, where *Acc** is the model accuracy on all instances. The *SOSS* is determined by the method in Section 3. The resulting *OSS* and *SOSS* with their corresponding tree sizes (S_{tree}) and tested accuracies (*Acc*) are listed in Table 1.

[2] In our experiments, we also test other bin numbers. We find that a too small bin number (e.g., 5, or 10) is not good in representing distributions, while a too large bin number (e.g., 50 or 100) increases the needs of memory but improves little in representing distributions.

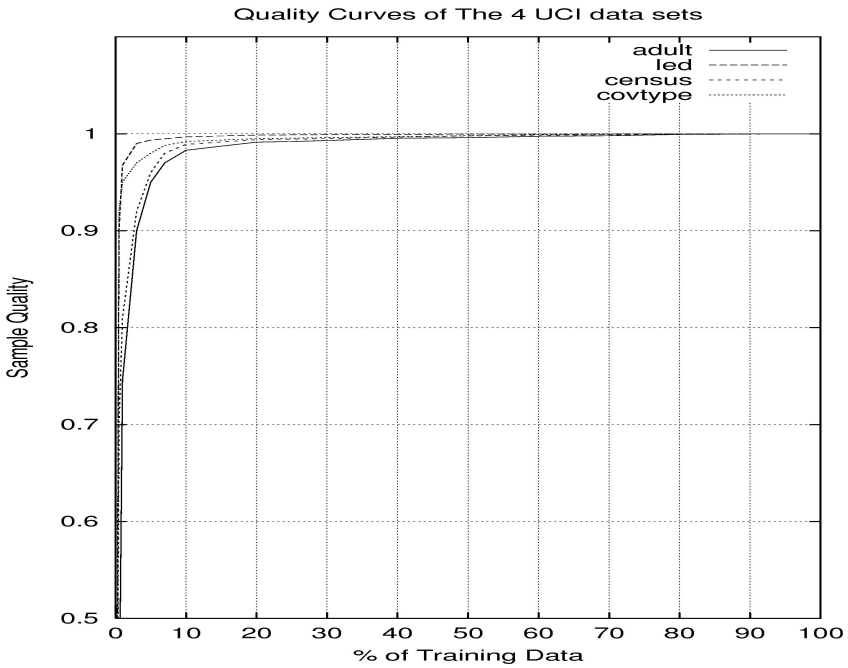

Fig. 3. Quality Curves of the four data sets

Table 1. Results of the 4 data sets

dataset	N (×1k)	S_{tree}	Acc* (%)	OSS (×1k)	S_{tree}	Acc (%)	$SOSS$ (×1k)	S_{tree}	Acc (%)	$t_{PS+SOSS}$ (t_{PS} /t_N)(sec)
adult	36	344	86.2	14	170	85.8	8	127	85.6	18 (20/22)
Led	100	3896	73.3	10	480	72.9	10	480	72.9	14 (38/41)
census	199.5	848	95.3	30	91	94.8	35	108	94.9	94 (152/176)
covtype	400	10970	75.8	140	4402	75.4	70	2417	73.0	668(766/1330)

We can see that for *led* and *census*, the *SOSS* is equal to or very close to the *OSS*[3]. If starting from this size, PS will reach the optimal model accuracy at once. For *adult* and *covtype* data sets, the SOSS is about half of the OSS, which means only one more

[3] It happens that the *SOSS* is slightly larger than the *OSS* for census. This may be due to the way we decide the *OSS*. Intuitively, the *SOSS* could not be far larger than the *OSS*, unless the learning algorithm produces very different learning accuracy on two similar distributions.

iteration is needed for PS (if the size increment ratio a=2) starting with the *SOSS*. In all data sets, the resulting tree sizes with both the *SOSS* and the *OSS* are significantly smaller than those with the full size, while their accuracy are very close. We also empirically compare the execution time used for PS starting with the found *SOSS* ($t_{PS+SOSS}$) (including the time for finding the *SOSS*) to that without the *SOSS* (t_{PS}) [4]. The execution time of directly learning on the entire data (t_N) is also given. They are all in the last column of Table 1. We can see that the *SOSS* indeed speeds up PS in all the data sets. All these results support our expectation that the *SOSS* can be used as a good starting sample size for PS.

As we have shown, the benefit of using the *SOSS* to start a PS actually lies in that it directly starts learning from a sample that sufficiently resembles the mother data, and thus saves the computation on those too dissimilar small samples. Of course, the procedure of finding the *SOSS* does involve some cost in computation. However, the time complexity of the procedure is $O(N)$. Compared with the time complexity of a learning algorithm, which is typically $O(N^c)$ (c>1), this cost is much smaller. As the total size of a data set increases, the use of *SOSS* becomes more attractive. Moreover, a practical data mining task often needs a pre-processing stage in which the raw data are filtered, extracted or cleaned from where they reside before mining. The parallel nature of the *SOSS* computation enables us to run it in the pre-processing stage, and thus can further enhance the benefit of using the *SOSS*.

5. Related Work

Besides being applied to classification [6][17], sampling has been used for association rule mining [21] and clustering [7]. Sample size is an important issue for sampling. Since only using a subset of the total data for learning will probably cause some loss of accuracy, a proper amount of data is necessary to gain efficiency while not lose too much learning accuracy. In terms of how the sample size is decided, sampling can be performed in two ways. One way is to find a proper sample size before extracting a sample and delivering it to data mining procedure. In [10], Kivinen and Mannila present some theoretical bounds for the sample size required to show the truth or falsity of a set of regularities in data. Toivonen [21] gives the sample size bounds induced from Chernoff bounds for mining association rules. Guha et al [7] present a sufficient sample size for clustering also using Chernoff bounds, which aims to produce a sample so that the probability of missing clusters is low. However, Chernoff bounds are found to be too conservative in practice [3][21].

The other way is not to fix the sample size *a priori*, but directly start learning from a small sample and continue trying larger samples until some stopping criterion is satisfied. Progressive Sampling works in this way. It directly uses the convergence of learning curve as the stopping criterion. Musick et al [14] use the expected loss in information gain, which is associated with choosing an attribute in decision tree construction, as the stopping criterion. Domingo et al [3] and Scheffer and Wrobel

[4] We use the geometric size scheme for PS as suggested in [17]. The sample size starts from 1% of the total data and doubles in the next iteration (a=2) and so on. All experiments run on SUN Sparc 450.

[20] propose some *utility functions*, which measure the "goodness" of rules or hypotheses on a given data set, as the stopping criterion. John and Langley [9] define a "*Probably Close Enough*" (PCE) criterion, which also makes use of the convergence of learning curve like Progressive Sampling [17].

The proposed *SOSS* can be viewed as the first way. It balances the need of efficiency and accuracy by directly measuring the similarity of a sample to the entire data. It also has a desirable property: the *SOSS* only depends on the data set and is independent of the learning algorithm. Although we present it for classification in Progressive Sampling, this property makes it applicable to other DM techniques.

The work that we find most similar to ours is that given in [5], where Ganti et al introduce a measure to quantify the difference between two data sets in terms of the models built by a given data mining algorithm. Our measure is different as it is based on statistical information divergence of the data sets. Although [5] also addresses the issue of building models using random samples and shows that bigger samples produce better models, it does not study how to determine a proper sample size.

Learning curve and similarity measure are essential to our work. They have long been addressed by machine learning researchers. Haussler et al [8] theoretically characterize the asymptotic behavior of learning curves (they call *sample complexity*) for Bayesian learning. Frey and Fisher [4] empirically study the behavior of learning curves for decision tree algorithm. Latorrette [12] measures the explanatory similarity between exemplars and applies it to nearest neighbor classification. Lin [13] proposes a general information-based definition of similarity and suggests its uses in different domains. Our similarity measure is different from [13] in that we do not measure the similarity of two instances, instead we measure the similarity between a sample and its mother data. Our measure is similar to that of [13] in that both have roots in information theory and require a probabilistic distribution. However, [13] does not associate similarity with a sufficient sample size, while we do.

6. Concluding Remark

[17] shows that Progressive Sampling is more efficient than learning on the entire data, and eliminates the need to load the entire data into memory. In this paper, we showed that a proper starting sample size could further improve its efficiency. With the intuition that a sample "must" sufficiently resemble its mother data in order to produce a good model, we proposed a technique to find a suitable starting sample size by measuring the similarity of a sample and its mother data. An information-based sample quality or similarity measure was introduced. Based on this measure, we defined the *SOSS*, and proved that asymptotically, the *SOSS* will achieve a model accuracy very close to that of the *OSS*. This claim was supported by our experimental results on UCIKDD data sets. Furthermore, the *SOSS* can also be efficiently determined in one scan of the mother data and is independent of learning algorithms. All these clearly suggest that the proposed *SOSS* can be used to start a Progressive Sampling. It can save many runs on unnecessary small samples, which are too dissimilar to the entire data.

References

[1] S. D. Bay. The UCI KDD Archive [http://kdd.ics.uci.edu], 1999.

[2] M. S. Chen, J. W. Han, and P. S. Yu. Data Mining: An Overview from a Database Perspective. IEEE Transactions on Knowledge and Data Engineering, 1996.

[3] C. Domingo, R. Gavalda and O. Watanabe. Adaptive Sampling Methods for Scaling Up Knowledge Discovery Algorithms. Book Chapter in "Instance Selection and Construction for Data Mining", Edited by H. Liu and H. Motoda. Kluwer Academic Publishers, 2001.

[4] L. J. Frey, and D. H. Fisher. Modeling Decision Tree Performance With the Power Law. The 7th Workshop on AI and Stat (Uncertainty'1999).

[5] V. Ganti, J. Gehrke, R. Ramakrishnan, and W. Y. Loh. A Framework for measuring Changes in Data Characteristics. In Proceedings of PODS'1999.

[6] J. Gehrke, V. Ganti, R. Ramakrishnan, and W. Y. Loh. Boat—Optimistic Decision Tree Construction. In Proceedings of ACM SIGMOD'1999.

[7] S. Guha, R. Rastogi, and K. Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In Proceedings of SIGMOD'1998.

[8] D. Haussler, M. Kearns, and R. Schapire. Bounds on the Sample Complexity of Bayesian Learning Using Information Theory and the VC Dimension. In Proceedings of the Fourth Annual Workshop on Computational Learning Theory (COLT'1991).

[9] G. H. John and P. Langley. Static Versus Dynamic Sampling for Data Mining. In Proceedings of KDD'1996.

[10] J. Kivinen and H. Mannila. The Power of Sampling in Knowledge Discovery. In Proceedings of ACM SIGMOD/PODS'1994.

[11] S. Kullback. Information Theory and Statistics. John Wiley & Sons, Inc, New York, 1959.

[12] M. Latourrette. Toward an Explanatory Similarity Measure for Nearest Neighbor Classification. In Proceedings of ECML'2000.

[13] D. Lin. An Information-theoretic Definition of Similarity. In Proceedings of ICML'1998.

[14] R. Musick, J. Catlett, and S. Russell. Decision Theoretic Subsampling for Induction on Large Databases. In proceedings of ICML'1993.

[15] T. Oates and D. Jensen. The Effects Of Training Set Size on Decision Tree Complexity. In Proceedings of ICML'1997.

[16] F. Olken. Random Sampling from Databases. PhD thesis, Department of Computer Science, University of California Berkeley, 1993.

[17] F. Provost, D. Jensen, and T. Oates. Efficient Progressive Sampling. In Proceedings of KDD'1999.

[18] F. Provost and V. Kolluri. A Survey of Methods for Scaling Up Inductive Algorithms. Data Mining and Knowledge Discovery, 3(2), 131-169, 1999.

[19] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993.

[20] T. Scheffer and S. Wrobel. A Sequential Sampling Algorithm for a General Class of Utility Criteria. In Proceedings of KDD'2000.

[21] H. Toivonen. Sampling Large Databases for Association Rules. In Proceedings of VLDB'1996.