Finding Statistically Optimal Sample Size By Measuring Sample Quality

Baohua Gu * Feifang Hu † Huan Liu ‡

Abstract

Sampling is a useful means to handling large data sets in data mining. Given a data set and a learning algorithm, an *Optimal Sample Size* (OSS) can be approximately found by progressively increasing sample size until the learning accuracy no longer increases. However, if the starting sample size is too far away from the OSS, the extra computation done before arriving at the OSS could be very expensive due to repeatedly running the learning algorithm on a sequence of samples. Thus directly starting from a size *close* to the OSS will greatly reduce computational cost. In this paper, we attempt to find such a size without a learning algorithm, but via a statistical approach. We name this size *Statistically Optimal Sample Size* (SOSS), in the sense that a sample of this size statistically sufficiently "resembles" its mother data. We define an information-based measure of sample quality that can be efficiently calculated in only one scan of a data set. We show that learning on a sample of SOSS will produce accuracy very close to that learned on a sample of OSS. We present an efficient algorithm that calculates a "quality curve" of sample quality with respect to sample size, based on which we determine a SOSS by detecting convergence. Experiments on artificial data sets and UCI data sets show that learning accuracy obtained on a sample of SOSS is fairly close to that on an sample of OSS as well as that of the whole data set. We conclude that SOSS is an effective measure determining sample size in the sense that it can find OSS but with much less computational cost.

Keywords: *data mining, sampling, learning curve, optimal sample size.*

1 Introduction

When mining knowledge from large data sets, users sometimes care more about computational cost than about learning accuracy. In such cases, mining a sample often provides comparable accuracy but with far less computational cost than mining the whole data. Thus sampling is useful in Knowledge Discovery and Data Mining [3]. However, it is usually difficult to decide a correct sample size for sampling. Given a large data set and a learning algorithm, some latest approaches (particularly "Dynamic Sampling" [9], and "Progressive Sampling" [17]) suggest that one not set a fixed sample size at the beginning but start learning with small samples and dynamically or progressively increase their sizes until learning accuracy does not improve any more. The resulted size, we call in this paper the *optimal sample size* (OSS), corresponds to a convergence point on a learning curve. Such an approach is usually preferable to analyzing all instances in a data set. However, if the starting

*Dept. of Computer Sci., National Univ. of Singapore, gubh@comp.nus.edu.sg
†Dept. of Stat. & Applied Prob., National Univ. of Singapore, stahuff@nus.edu.sg
‡Dept. of Computer Sci., National Univ. of Singapore, liuh@comp.nus.edu.sg

size is too far away from the OSS, the computational cost spent on smaller samples before convergence could be rather significant, as shown in the following example.

Suppose we use a geometric schedule for sample sizes, which is shown often more efficient than an arithmetic schedule [17], denoted as $S_g = \{n_0, a*n_0, a^2*n_0, a^3*n_0, ...\}$, where $n_0 > 0$ is the size of the first sample, $a > 1$ is the increment ratio. If the convergence of learning accuracy is detected at the $(k+1)$th sample (its size being $size_{k+1} = a^k * n_0$), then the total number of instances of the previous k samples will be $size_{1..k} = n_0 * (1 + a + a^2 + ... + a^{k-1}) = n_0 * (a^k - 1)/(a - 1) = (size_{k+1} - n_0)/(a - 1)$. If $n_0 \ll size_{k+1}$ and $a = 2$, then $size_{1..k}$ is about the same as $size_{k+1}$. If the learning algorithm has a super-linear run-time complexity $O(n^c)$ (n is the number of instances used in learning, $1 < c$), this extra computation could be significantly high (e.g., when $c = 2$). Moreover, since in practice, the ratio a can not be too large (usually $a \leq 2$), if n_0 is very apart from the OSS, then the number of samples needed before convergence (k) will be large. Note that in the case when data is sequentially accessed from disk, in order to generate a random sample from a data set, it requires to go through the data set once. If this happens, a large k will result in considerably high I/O cost (see our analysis in Section 4 for detail).

Note that when a is fixed, $size_{1..k}$ will actually depend on the distance from the starting size n_0 to the stopping size $size_{k+1}$ (i.e., the OSS). Thus, to reduce this extra cost, a viable way is to use a n_0 which is *approximate* to the true OSS. Therefore, it is better to find a suitable starting size. However, this task could be as hard as finding the OSS, because many factors may involve. In this paper, we approach to this task statistically.

Intuitively, a sample with the OSS is very likely being similar to its mother data, and if this similarity, which we will call "sample quality" thereafter in this paper, can be efficiently measured, then the size of a sample with very high quality can be taken as the desired size that approximates to the true OSS. We realize this intuition in three steps. First, we define a measure of sample quality based on information theory. By its statistical property, we present an efficient algorithm that computes sample quality in only one scan of a large data set. Second, we define a *Statistically Optimal Sample Size* (SOSS) based on sample quality, and prove that the SOSS is close to the OSS in the asymptotic sense that learning on samples of the two sizes will produce sufficiently close learning accuracy. Third, we present an efficient algorithm to figure out a *"quality curve"*, the curve of sample quality with respect to sample size, on which we can determine a SOSS by finding a convergence point. Our approach is statistical and does not involve any data mining algorithm, therefore it saves computational cost of running a data mining algorithm. Experiments on some artificial data sets and UCI data sets show that learning accuracy achieved by a sample of SOSS is quite close to that of a sample of OSS. In other words, SOSS can be used as a suitable starting size to speed up the established *progressive sampling*.

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 first define SOSS and prove that it is asymptotically close to OSS in terms of learning accuracy. We conduct experiments to test our method and propose an algorithm to efficiently calculate a quality curve, then we show how to find SOSS on it. We discuss when SOSS will be useful and related research in Section 4. We conclude in Section 5.

2 Information-Based Sample Quality Measure

2.1 Underlying Information Theory

A sample with the optimal size should inherit property of its mother data as much as possible. This property intuitively suggests using the concept of information and its theory. Shannon's definition of information only depends on one probability distribution and describes the uncertainty of possible outcomes. Kullback's definition, generally called a *divergence* or *deviation*, as it depends on two probability distributions and describes the divergence between the two [6], is more suitable for measuring the similarity between a sample and its mother data. Below we briefly describe related definitions and conclusions of Kullback's information measure [11].

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 probability measure μ_i, $i = 1, 2$, then the mean (average) information for discrimination in favor of H_1 against H_2 for all possible x and μ_1 is

$$I(1:2) = \int log\frac{P(H_1|x)}{P(H_2|x)}d\mu_1(x) - log\frac{P(H_1)}{P(H_2)},$$

where $P(H_i)$, $i = 1, 2$, is the prior probability of H_i and $P(H_i|x)$ is the posterior probability of H_i, or the conditional probability of H_i given $X = x$. Similarly, the mean (average) information for discrimination in favor of H_2 against H_1 for all possible x and μ_2 is

$$I(2:1) = \int log\frac{P(H_2|x)}{P(H_1|x)}d\mu_2(x) - log\frac{P(H_2)}{P(H_1)},$$

The information divergence is defined by $J(1,2) = I(1:2) + I(2:1)$, i.e.,

$$J(1,2) = \int log\frac{P(H_1|x)}{P(H_2|x)}d\mu_1(x) + \int log\frac{P(H_2|x)}{P(H_1|x)}d\mu_2(x).$$

According to [11], $J(1,2)$ is a measure of the divergence between hypotheses H_1 and H_2, or between μ_1 and μ_2, and is a measure of the difficulty of discriminating between them.

For multinomial populations, suppose we have two hypotheses H_1 and H_2 that specify the probabilities of two hypothetical c-values populations (c categories or classes), H_i : $p_{i1}, p_{i2}, ..., p_{ic}$, where p_{ij} is the probability of occurrence of the jth value, $j = 1, 2, ..., c$, and $p_{i1} + p_{i2} + ... + p_{ic} = 1$, $i = 1, 2$, then the information divergence between H_1 and H_2 is,

$$J(1:2) = I(1:2) + I(2:1) = \sum_{j=1}^{c}(p_{1j} - p_{2j})log\frac{p_{1j}}{p_{2j}}.$$

For multi-variate normal populations, suppose we have two r-variate normal populations $N(\mu_i, \Sigma_i)$, $i = 1, 2$, with $\mu_i = (\mu_{i1}, \mu_{i2}, ..., \mu_{ir})$, the one-row vectors of mean values, and $\Sigma_i = (\sigma_{ijk})_{r\times r}$, $i = 1, 2$; $j, k = 1, 2, ..., r$, the covariance matrices, Σ_i^{-1}, $i = 1, 2$, is the inverse of Σ_i, and $tr()$ is a calculation of trace, then the divergence between the two populations is,

$$J(1:2) = \frac{1}{2}tr(\Sigma_1 - \Sigma_2)(\Sigma_2^{-1} - \Sigma_1^{-1}) + \frac{1}{2}tr(\Sigma_1^{-1} + \Sigma_2^{-1})(\mu_1 - \mu_2)'(\mu_1 - \mu_2),$$

The information divergence has a good limiting property described in Theorem 1 below (for proof, refer to [11]), based on which we will prove an important theorem about *Statistical Optimal Sample Size* 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 $I(1^{(n)} : 1)$, then we have, $\lim_{n\to+\infty} I(1^{(n)} : 1) = 0$, if and only if $\lim_{n\to+\infty}(f_n(x)/f(x)) = 1[\lambda]$, uniformly.*

Here the $[\lambda]$ means that the limitation and the fraction hold in a probability measure λ, because $f_n(x)$ and $f(x)$ are probability functions in probability measurable space.

2.2 Defining Sample Quality

We extend the above definition of information divergence between two populations to a new measure of sample quality. The idea is to compute the information divergence between a sample and its mother data. The intuition here is that the degree of the similarity between a sample and its mother data should correspond to the divergence between the two, which is the mean information needed to discriminate them. In what follows, we assume that a large data set D has a simple "flat" structure, consisting of a very large number N of rows (or instances), each consisting of a fixed number w of attributes (or variables). The attributes are either categorical (nominal scale) or numerical (continuous scale). We assume there are u numerical attributes $A_1, A_2, ..., A_u$ and v categorical attributes $B_1, B_2, ..., B_v$, $w = u + v$.

Let S be a sample of D. To calculate its sample quality, we calculate the divergence of S from D by all numerical attributes and each categorical attribute respectively. To simplify the calculation, we use the formulation of divergence of multi-variate normal distributions for all numerical attributes, and the formulation of divergence of multi-nomial distributions for each categorical attribute. The sum of the obtained information divergences is the total divergence of S from D.

For all numerical attributes of D and S, denote their mean vector with μ_D and μ_S, covariance matrix with Σ_D and Σ_S, the inverse of Σ_D and Σ_S with Σ_D^{-1} and Σ_S^{-1}, then the divergence for the numerical part is:

$$J_{num} = \frac{1}{2}tr(\Sigma_D - \Sigma_S)(\Sigma_S^{-1} - \Sigma_D^{-1}) + \frac{1}{2}tr(\Sigma_D^{-1} + \Sigma_S^{-1})(\mu_D - \mu_S)'(\mu_D - \mu_S); \quad (1)$$

For each categorical attribute i, $i = 1, 2, ..., v$, suppose there are totally c_i different values in D on the ith attribute, the divergence of D and S for the attribute is:

$$J_i = \sum_{j=1}^{c_i}(p_S^j - p_D^j)log\frac{p_S^j}{p_D^j}; \quad (2)$$

The total divergence between D and S by adding up all the above divergence is:

$$J = J_{num} + \sum_{i=1}^{v} J_i. \quad (3)$$

Finally, normalize J to be the sample quality q of the sample S by $q = exp(-J)$.

According to the definition of information divergence, J will be a non-negative real number, thus q, the value of sample quality, will fall in $(0, 1]$, where $q = 1$ means that there is no information divergence between the two data sets. The larger the information divergence J, the lower the sample quality q; and vice versa.

For categorical attributes, p_S^j may be zero because some values in the mother data set D may not appear in the sample S. In such cases, we simply set the p_S^j to $1/N$, which is very

close to zero if N is very large, in order to make the logarithm computable. This setting is almost equivalent to adding one instance in S, which should not make significant change on it. Therefore, this small change should not make great difference in the total divergence.

2.3 Some Remarks

In the above definition, we have actually assumed a multi-variate normal distribution for numeric attributes and independence among categorical attributes. This is only for theoretical derivation, i.e., to make use of the involved formula. By the definition, only mean vector, co-variance matrix and categorical values' frequencies of a studied data set are involved in calculating the measure. Thus the measure can be widely applied to many types of data without checking their distributions. However, whether it is effective in depicting the similarity (or difference) between a sample and its mother data will depend on the actual distribution of the given data.

From its information base, it is reasonable to expect that the measure can well characterize the similarity for numerical attributes following multi-variate normal or similar distributions, and for categorical attributes with little inter-dependency. This is much like the popular *Linear Discriminant Analysis* by Fisher [7], which assumes linear separator but has proved effective in many cases that are not strictly linear.

However, this characterization may not be guaranteed so well when the assumption were grossly violated. Further research should address this problem in detail with the progress made in this work. With some prior knowledge, we could alleviate the problems as follow. If there is significant inter-dependency among categorical attributes, we can apply weights to these attributes in the summation of all categorical information divergence. If there exist multi-clusters in numerical data, Gaussian distribution can be assumed for each cluster, then the information divergence of each cluster can be summed to get the total divergence. In this work, we use the original definition without assuming any prior knowledge and it seems working well in our experiments.

In order to make the measure computable, it is required that the inverses for all involved covariance matrix exist. This is not necessarily always true if there exists linear dependence among numerical attributes. In such cases, we can solve this problem by *feature selection*, i.e., filtering out those irrelevant or redundant attributes before running data mining algorithms. There are many feature selection methods available, from which users can select a suitable one according to guide lines provided in [4]. However, based on the rule of thumb in Statistics, for a data set consisting of thousands of instances, the probability of existing singular inverse of the covariance matrix is empirically very small. Therefore, this relaxation should be working in large data sets.

2.4 Calculating Sample Quality

By their statistical property, all the intermediate quantities such as μ_D, μ_S, Σ_D, Σ_S, P_D^j, and P_S^j can be incrementally calculated when scanning D sequentially. Thus we design an efficient algorithm SQ listed below. To simplify the description of the algorithm, we denote that $D = \{r_i \in D, i = 1, 2, ..., N | r_i =< a_1, a_2, ..., a_u, b_1, b_2, ..., b_v >\}$, where R_i is the ith row of D, $\{a_1, a_2, ..., a_u\}$ are values of the corresponding numerical attributes $\{A_1, A_2, ..., A_u\}$, and $\{b_1, b_2, ..., b_v\}$ are values of the corresponding nominal attributes $\{B_1, B_2, ..., B_v\}$. A sample S of size n (i.e., n rows or instances) is denoted as $S = \{r_i \in D, i = i_1, i_2, ..., i_n | 1 \leq i_j \leq N, j = 1, 2, ..., n, n \leq N\}$. Denote the mean of the

numerical attribute A_i for D as μ_{Di}, for S as μ_{Si}, $i = 1, 2, ..., u$; the joint mean of the numerical attributes A_i and A_j for D as μ_{Dij}, for S as μ_{Sij}, $i, j = 1, 2, ..., u$. Denote the frequency of the jth value of the ith nominal attribute B_i in D as fre_{Dij}, in S as fre_{Sij}, $i = 1, 2, ..., v$, and $j = 1, 2, ..., c_i$. We have the following algorithm SQ to calculate sample quality.

PSEUDO ALGORITHM *SQ*
Input: a large data set D and sample size n
Output: a random sample S and its sample quality q.
begin
1. $SampIndex \longleftarrow \{ i_1, i_2, ..., i_n \}$;
/* generate a group of n uniformly distributed random numbers from $[1 : N]$ and sort them*/

2. initialize all means, joint means and frequency counts to be zero;

3. for $(i = 1; i \leq$N$; i++)$ { /* read the ith row from D */
$r_i \longleftarrow < a_1, a_2, ..., a_u, b_1, b_2, ..., b_v >$;
if $(i \in SampIndex)$ save r_i into S;
for $(k = 1; k \leq u; k++)$ { /* update the means and the joint means for D*/
$\mu_{Dk} \longleftarrow \mu_{Dk} + a_k$;
for $(h = 1; h \leq u; h++)$ { $\mu_{Dkh} \longleftarrow \mu_{Dkh} + a_k * a_h$; }
if $(i \in SampIndex)$ { /* if it is in the sample S, update likewise */
$\mu_{Sk} \longleftarrow \mu_{Sk} + a_k$;
for $(h = 1; h \leq u; h++)$ { $\mu_{Skh} \longleftarrow \mu_{Skh} + a_k * a_h$; } } }
for $(k = 1; k \leq v; k++)$ { /* update frequencies for nominal attributes */
for $(h = 1; h \leq c_k; h++)$ {
update fre_{Dkh}; if $(i \in SampIndex)$ update fre_{Skh}; } } }
4. for $(i = 1; i \leq u; i++)$ /* calculate Σ_D and Σ_S */
for $(j = 1; j \leq u; j++)$ {
$\Sigma_{Dij} \longleftarrow \mu_{Dij} - \mu_{Di} * \mu_{Dj}$; $\Sigma_{Sij} \longleftarrow \mu_{Sij} - \mu_{Si} * \mu_{Sj}$; }
5. $\Sigma_D^{-1} \longleftarrow inverse(\Sigma_D)$; $\Sigma_S^{-1} \longleftarrow inverse(\Sigma_S)$; /* calculate Σ_D^{-1} and Σ_S^{-1} */

6. Calculate J by equations (1), (2) and (3), then $q = exp(-J)$;
end

2.5 Computational Complexity Analysis

Now we analyze the computational complexity and efficiency of the algorithm *SQ*. In Step 1 of algorithm SQ, generating n uniformly distributed random numbers in $[1, N]$ will take $O(n)$ time and sorting them will take $O(nlogn)$ time. Step 2 of SQ will take $O(u^2+v))$ time. Step 3 of SQ consists of N repeated operations: loading an instance and calculating some intermediate quantities, each of which take 1 unit of I/O time and $O(u^2 + v)$ calculation. Since u and v are usually far less than N, each operation can be assumed to take a constant time dominated by I/O operation. Thus Step 3 of SQ will take $O(N)$ time. Step 4 of SQ takes $O(u^2)$ time. Step 5 will take $O(u^3)$ time [16]. In practice, I/O operation is much slower than calculation in CPU, and N is much larger than n, u, and v, therefore, the overall run-time complexity of SQ is $O(N)$ dominated by I/O time in Step 3.

The memory requirement for holding means is $O(u)$, for joint means is $O(u^2)$, for frequencies is $O(v)$, and for keeping sample index is $O(n)$. As n is usually much larger than both u and v, the memory requirement is dominated by $O(n)$ sample index.

Note that if S is obtained by a sequential sampling method such as sequential random sampling and systematic sampling, then even the sampling procedure itself can be done in the same one scan of the original data set D as the above-mentioned calculations. This means that we can calculate the quality of a sample while we generate the sample. An excessive cost besides generating the sample is that we have to go through the original data set D once, while a procedure of generating a sequential sample usually finishes before reaching the last row of D. In general, the algorithm is very efficient in calculating sample quality.

3 Finding Statistically Optimal Sample Size

Based on the sample quality definition, it is easy to see 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. Now we define a *Statistically Optimal Sample Size* (SOSS) as the size at which the sample quality is sufficiently close to 1. Although a SOSS is in itself not necessarily the OSS, below we will prove that the SOSS should be quite close to the OSS in the sense that their accuracy rates of learning can be asymptotically close.

Provided a learning algorithm L and a sufficiently large data set D, we can take L as an operator mapping a probability density function $f(x)$ of D to a real number a (namely learning accuracy), i.e., $L : f(x) \to [0, acc^*]$, where acc^* is the maximum learning accuracy on D. We denote a random sample of OSS has probability density function $f_{OSS}(x)$, and a random sample of SOSS has $f_{SOSS}(x)$, denote the learning accuracy on the two samples as acc_{OSS} and acc_{SOSS}, then we have the following theorem.

Theorem 2 *If L satisfies that $\lim_{n\to\infty} f_n(x)/f(x) = 1[\lambda] \Longrightarrow \lim_{n\to\infty} |L(f_n(x)) - L(f(x))| = 0$, then in an asymptotic meaning, $acc_{SOSS} \to acc_{OSS}$.*

Proof: Suppose there exist a series of random samples $\{S(n)\}$ in the given data set D, n is sample size, denoted their corresponding sample quality as $q(S_n)$. According to the definition of SOSS, when $n \to SOSS$, $q(S(n)) \to 1$, that is, the information divergence of $S(n)$ from D, $I(S(n) : D) \to 0$. Applying the Theorem 1, we have, $\lim_{n\to\infty} f_{S(n)}(x)/f(x) = 1[\lambda]$, therefore, $\lim_{n\to\infty} |L(f_{S(n)}(x)) - L(f(x))| = 0$. In an asymptotic meaning, $L(f_{S(n)}(x)) \to acc_{SOSS}$ and $L(f(x)) \to acc_{OSS}$, that is, $acc_{SOSS} \to acc_{OSS}$. #

The premise of the theorem means that if two probability distributions are quite similar, then a learning algorithm should produce quite close learning accuracy on them. This is reasonable for a stable learning algorithm. With the support of the *law of large numbers* and the *central limit theorem*, the asymptotic property should apply to sufficient large data sets. Based on this theorem, we can search for an SOSS by measuring sample quality instead of directly search for an OSS by running a learning algorithm and can expect the two are close in terms of learning accuracy.

3.1 Experimental Evaluation

First, we test the relation of sample quality with respect to sample size by using artificial data sets and UCI data sets [1]. We follow the method in [19] to generate 3 artificial data sets with multi-variate normal distributions. We set their sizes to be 10,000 instances with 10 attributes. Each attribute has values randomly generated from a standard normal

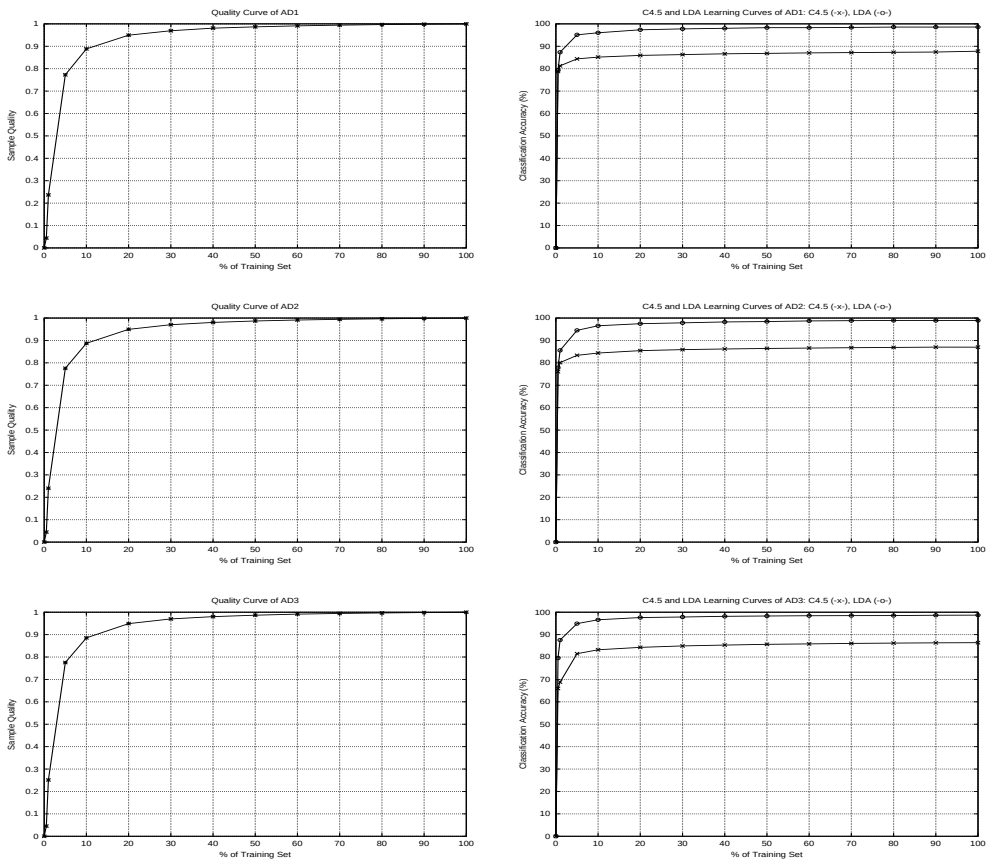

Figure 1: Quality Curves, C4.5 and LDA Learning Curves of 3 Artificial Data Sets

distribution $N(0,1)$. The covariance between the ith and jth attribute is set to $(-1)^{(i-j)} \times \sigma$, with the sign reflecting positive or negative correlation between the two and the constant $\sigma \in (0,1)$ reflecting the extent of the correlation. For each data set, one extra attribute is made for each instance as a class label by a given rule. The rule we use is: for each instance, if the sum of the 10 attributes values is larger than 0, then label it as class '1'; otherwise, class '2'. By setting σ to be 0.2, 0.5 and 0.8, we create 3 data sets: $AD1$, $AD2$, and $AD3$. These 3 data sets are used for training. Similarly, we generate an independent testing data set containing 5000 instances for each of them.

For each of the 3 data sets, we select random samples with their sizes being 0.5%, 1%, 5%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90% of the total, then calculate their sample qualities. For each size, we repeat 100 times and take average. We draw 3 quality curves in the left column of Figure 1. The quality curves of the 3 artificial data sets consistently show that sample quality tends to increase with the sample size and steadily approaches to 1.

The 3 quality curves show shapes similar to the famous learning curve. For comparison, we also draw the corresponding learning curves of the 3 data sets. We choose two classification algorithms: a *Linear Discriminant Analysis* LDA [12] and a decision tree algorithm C4.5 [18]. We use Matlab implementation of LDA and C4.5 Release 8. Classification accuracy is tested against independent test data sets and averaged on 100 runs. The learning curves of the 3 artificial data sets are listed in the right column of Figure 1.

For testing our measure on real world data, we conduct similar experiments on 4 UCI data sets: *Adult*, *Letter*, *Nursery* and *Pendigits* and draw their quality curves, C4.5 and LOG learning curves of in Figure 2. LOG is a Fortran implementation of logistic discriminant analysis, which is a variation of LDA and can handle categorical attributes [13]. The description about the 4 data sets is listed in Table 1. The quality curves on the UCI data sets show similar trend as the artificial data sets. The results also show that our quality measure works well without assuming multi-variate normal distributions and also show shapes similar to corresponding learning curves. Like a learning curve, a quality curve also has a steep slope in the early portion, a more gentle slope in the middle portion, and a plateau in the late portion. The plateau occurs when increasing of sample size does not improve sample quality. This phenomenon suggests that we use the quality curve to determine the SOSS by detecting convergence on a quality curve. On a learning curve, an OSS is determined at the convergence point; Similarly, an SOSS can be set as the convergence point on a quality curve.

Table 1: Description of 4 UCI Data Sets

Data Set Name	Numerical Attributes	Categorical Attributes	Cases for training	Cases for testing
Adult	6	8	36000	12842
Letter	16	0	15000	5000
Nursery	0	8	10000	2960
Pendigits	16	0	8000	2992

Given n pairs of *(sample size, sample quality)* on a quality curve, to detect a convergence more accurately, we first fit the points with a model, then decide the SOSS to be the size at which the slope of tangent line is smaller than a given value δ very close to 0.

We use $y = a - b * x^{-c}$ to fit a learning curve [8], where a, b and c are parameters to be estimated, x is sample size (in percentage) and y is learning accuracy. More interestingly, this model also fits the quality curves well. We list results of fitting for learning curves and quality curves of the 3 artificial data sets in Table 2 and the 4 UCI data sets in Table 3. In both tables, S denotes the *standard error* of the estimates and r denotes the *correlation coefficient*.

In our experiments, we decide the minimum OSS as the size on the fitted learning curve from which the learning accuracy can only increase no more than 1%, the SOSS as the size on the fitted quality curve at which the slope of tangent line is less than 0.001. We list the corresponding OSS's and SOSS's for the 3 artificial data sets in Table 4and the 4 UCI data sets in Table 5. We also list the maximum learning accuracy that is learned on full size of each of the 7 data sets.

From Table 4, we find that for the 3 artificial data sets, SOSS is quite close to OSS and the corresponding learning accuracy (of both C4.5 and LDA) are also quite close. This may be due to the fact that the data sets themselves are generated from multi-variate normal distributions, so that the quality measure works very well. From Table 5, we find that for

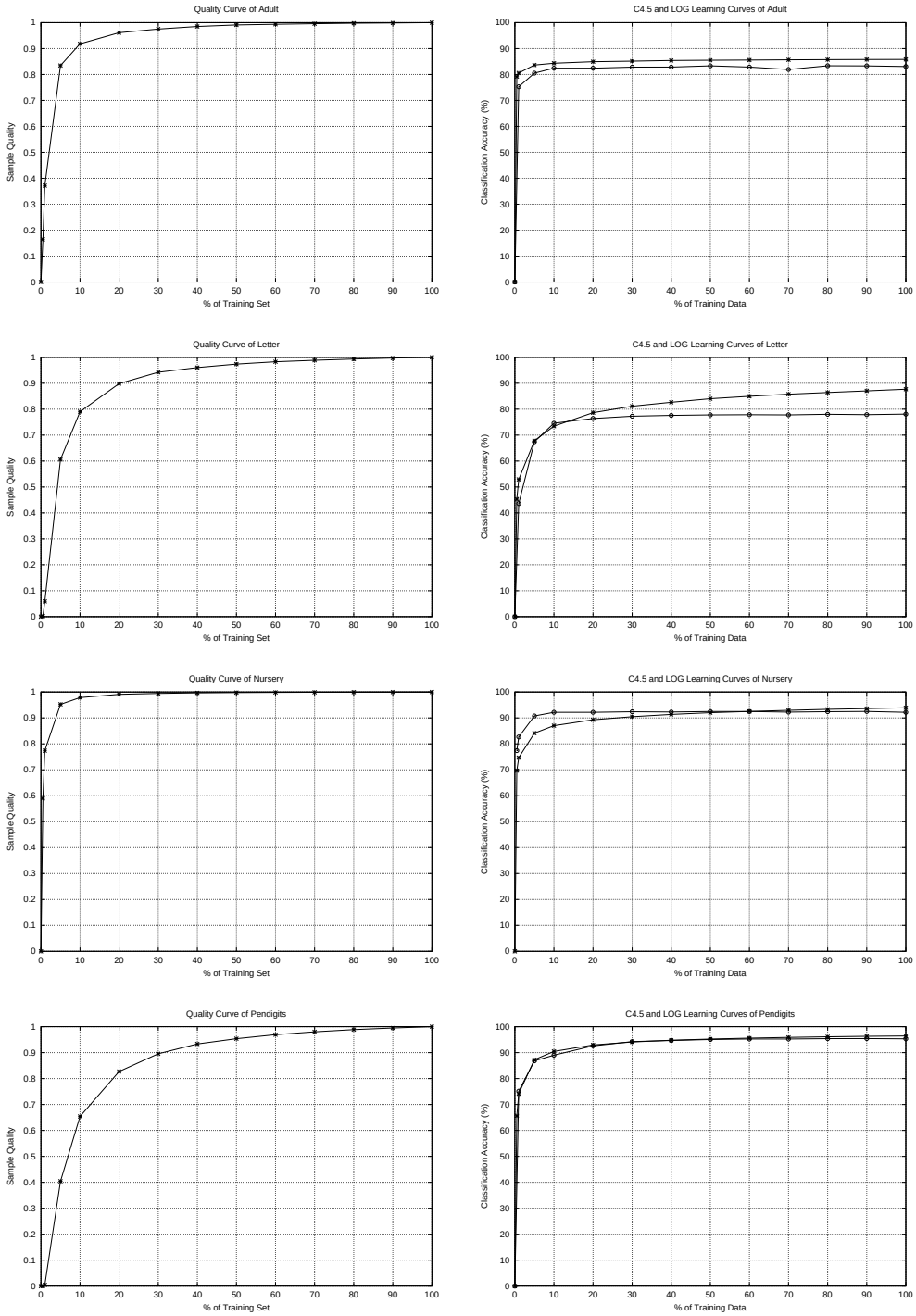

Figure 2: Quality Curves and C4.5 Learning Curves of 4 UCI Data Sets

Table 2: Fitting Learning Curves and Quality Curves of the 3 Artificial Data Sets

Data Set	C4.5 Learning Curve	LDA Learning Curve	Quality Curve
AD1	a =89.73	a =98.97	a =1.10
	b =8.77	b =12.03	b =0.77
	c =0.29	c =0.70	c =0.50
	S =0.20	S =0.24	S =0.04
	r =0.998	r =0.999	r =0.995
AD2	a =88.44	a =99.72	a =1.09
	b =9.13	b =14.17	b =0.77
	c =0.39	c =0.63	c =0.50
	S =0.33	S =0.11	S =0.04
	r =0.996	r =0.999	r =0.995
AD3	a =88.87	a =99.03	a =1.09
	b =17.70	b =11.85	b =0.76
	c =0.44	c =0.71	c =0.51
	S =0.96	S =0.18	S =0.03
	r =0.992	r =0.999	r =0.996

Table 3: Fitting Learning Curves and Quality Curves of the 4 UCI Data Sets

Data Set	C4.5 Learning Curve	LOG Learning Curve	Quality Curve
Adult	a=87.08	a=83.35	a=1.06
	b=6.27	b=8.08	b=0.62
	c=0.35	c=0.75	c=0.57
	S=0.10	S=0.47	S=0.03
	r=0.999	r=0.98	r=0.996
Letter	a=119.97	a=79.54	a=1.33
	b=66.95	b=36.05	b=1.14
	c=0.16	c=0.76	c=0.29
	S=0.17	S=0.75	S=0.05
	r=0.999	r=0.998	r=0.990
Nursery	a=102.82	a=92.88	a=1.01
	b=27.99	b=9.24	b=0.23
	c=0.25	c=0.78	c=0.87
	S=0.13	S=0.49	S=0.002
	r=0.999	r=0.995	r=0.999
Pendigits	a=100.05	a=99.87	a=2.65
	b=25.68	b=24.83	b=2.55
	c=0.43	c=0.40	c=0.10
	S=0.09	S=0.53	S=0.06
	r=0.999	r=0.997	r=0.989

Table 4: The OSS and the SOSS of 3 Artificial Data Sets

Data Set	Full Size & Accuracy	The OSS & Accuracy	The SOSS & Accuracy
AD1	10000, 87.8% (C4.5)	5000, 86.8% (C4.5)	4000, 86.4% (C4.5)
	10000, 98.6% (LDA)	3000, 97.6% (LDA)	4000, 98.0% (LDA)
AD2	10000, 87.0% (C4.5)	3500, 86.0% (C4.5)	3500, 86.0% (C4.5)
	10000, 98.9% (LDA)	3300, 97.9% (LDA)	3500, 98.0% (LDA)
AD3	10000, 86.4% (C4.5)	4200, 85.4% (C4.5)	3500, 85.1% (C4.5)
	10000, 98.7% (LDA)	2800, 97.7% (LDA)	3500, 98.0% (LDA)

Table 5: The OSS and the SOSS of 4 UCI Data Sets

Data Set	Full Size & Accuracy	The OSS & Accuracy	The SOSS & Accuracy
Adult	36000, 85.8% (C4.5)	9200, 84.8% (C4.5)	11000, 85.2% (C4.5)
	36000, 83.1% (LOG)	3500, 82.2% (LOG)	11000, 82.8% (LOG)
Letter	15000, 87.7% (C4.5)	12000, 86.7% (C4.5)	7350, 83.5% (C4.5)
	15000, 78.1% (LOG)	4000, 77.1% (LOG)	7350, 77.8% (LOG)
Nursery	10000, 93.9% (C4.5)	7200, 92.9% (C4.5)	1600, 89.0% (C4.5)
	10000, 92.2% (LOG)	800, 91.2% (LOG)	1600, 92.1% (LOG)
Pendigits	8000, 96.4% (C4.5)	5500, 95.4% (C4.5)	6300, 96.0% (C4.5)
	8000, 95.3% (LOG)	2800, 94.3% (LOG)	6300, 95.2% (LOG)

the LOG algorithm, the learning accuracy on the SOSSs are quite close to that on the OSSs of all the 4 UCI data sets (the differences less than 1%), although their SOSSs are not close to the corresponding OSSs. And similar results can be found on *Adult* and *Pendigits* for the C4.5 algorithm.

For *Letter* and *Nursery*, the C4.5 learning accuracy at the SOSS is already very high, however, the difference between SOSS and OSS is about 5000, the difference of their learning accuracy is about 3%. For *Letter*, this may be due to the essential differences between a learning curve and a quality curve: a learning curve depends on the learning algorithm and the data set, while a quality curve in itself depends only on the data set. Note that even at full size of the training data, the learning curve still has a trend to increase but the quality curve is getting to its maximum. From the quality curve, we can only know at which size the sample is sufficiently similar to its mother data, and the SOSS should not be far away from OSS in terms of their learning accuracy, but we cannot decide the learning accuracy from a quality curve. For C4.5 results of *Nursery*, besides the above reasons, the difference may also be due to our definition of sample quality for categorical attributes. Note that the *Nursery* only has categorical attributes and we calculate the information divergence for each categorical attribute then add them up. Thus the inter-attribute dependency may not be included in our definition. This should be one aspect for further research.

3.2 Calculating Quality Curve

The algorithm SQ computes a sample's quality in one scan of a data set. We employ the spirit of SQ to calculate several sample's qualities in parallel in the same scan of the dataset. Suppose we have sequential access to a large data set D whose size is N. We draw k samples with different sizes $n_1, n_2, ..., n_k$. For each size, we take a random sample from D. The samples are denoted as $S_1, S_2, ..., S_k$. We present the procedure as follows:

Pseudo Algorithm *SOSS*
Input: a large data set D and number of samples k;
Output: k pairs of *(sample size, sample quality)*;
begin
1. Generate k groups of sorted random index for the k samples; initialize means, joint means and frequencies for D and all the samples; $i = 1$;
2. Read the ith instance from D into memory;
3. Iteratively update means, joint means and frequencies for D; if the instance is to be in any of the samples, update means, joint means and frequencies for that sample;
4. $i++$; if $i \leq N$, go to Step 2; otherwise, go to Step 5.
5. Calculate the sample qualities for all the k samples.
end

Based on the complexity of SQ, it is easy to know *SOSS* will also take $O(N)$ time but using more memory than SQ. Its memory usage mainly depends on k. Obviously, the larger the k (number of samples), the more exact the quality curve, but the more memory requirement (to keep sample index numbers in memory). One can make his choice according to the trade-off of the two.

Besides, the selection of different sample sizes may also account for memory usage. Generally, an arithmetic scheme ($n_i = n_0 + (i-1) * d$) or a geometric scheme ($n_i = n_0 * a^{i-1}$) can be used. Given k, n_0, a and d, an arithmetic scheme will use $S_a = \sum_{i=1}^{k} n_k = k * n_0 + k * (k-1) * d/2$, while a geometric scheme will use $S_g = n_0 * (a^k - 1)/(a - 1)$. For example, let $N = 10^5$, $k = 20$, $n_0 = 10^3$, and the last sample size $n_k = 0.9 * N$, if one integer takes 4 bytes in memory, an arithmetic scheme will use 3.4MB, while a geometric scheme will use 1.5MB. This should not be a big burden even for a modern PC.

4 Discussion and Related Research

With SOSS, we can: (1) directly use it as a rough OSS for learning. This should generally be better than using a randomly guessed sample size, but it cannot guarantee the maximum learning accuracy, or (2) use it as a starting size for *Progressive Sampling*. Now we analyze in the second case, will SOSS be useful and when?

Given a large data set D with N instances saved in external memory and accessible via a DBMS, a learning algorithm L with run-time complexity $O(n^c)$ (usually $1 \leq c \leq 3$) in terms of the number of instances used for learning, and a learning task of running L on D to get the maximum (or the optimal) learning accuracy, typically we can do it in 4 ways: (1) Scheme 1: directly run L on whole D; (2) Scheme 2: if we happen to know an OSS (denoted as $n_\star$), run L on $n_\star$ number of instances randomly chosen from D; (3) Scheme 3: conduct a *Progressive Sampling* (using a geometric size scheme $n_i = n_0 * a^{(i-1)}$ with a randomly guessed n_0, and assume it will stop at the kth sample); (4) Scheme 4: find a SOSS first (denoted as n_*), then conduct the *Progressive Sampling* with $n_0 = SOSS$. And we assume that starting from the SOSS, we can soon detect the convergence with only $h-1$ more samples ($h \ll k$). For simplicity, we assume both the Scheme 3 and Scheme 4 take the same increment ratio a and happen to stop at the $n_\star$.

The involved computational cost mainly includes: Cost 1: the I/O operations to transfer instances from disk into main memory, usually in terms of the number of transferred instances; Cost 2: the CPU time for running L, in terms of the number of used instances; and Cost 3: disk operations on locating instances that will be loaded.

There may be some cases for the Cost 3. For Scheme 1, it will be $O(N)$, no matter random or sequential access. However, it may be different for the other 3 schemes. Given a random sample of size n, if we have random access to all records, then locating n instances needs $O(n)$. When we only have sequential access, if we can skip records in zero time (e.g., fixed size records on disk), then it needs $O(n)$, otherwise, we may be forced to scan every record in time $O(N)$ [15]. We list the related cost for the 4 Scheme in Table 6.

Table 6: Run-time Cost for the 4 Learning Schemes

Scheme	Cost 1	Cost 2	Cost 3
1	$O(N)$	$O(N^c)$	$O(N)$
2	$O(n_\star)$	$O(n_\star^c)$	$O(n_\star)$, or $O(N)$
3	$O(\sum_{i=1}^{k} n_i)$ $= O(a * n_\star - n_0)$	$O((\sum_{i=1}^{k} n_i)^c)$ $= O((a * n_\star - n_0)^c)$	$O(\sum_{i=1}^{k} n_i) = O(a * n_\star - n_0)$, or, $k * O(N)$
4	$O(\sum_{i=1}^{h} n_i) + O(N)$ $= O(a * n_\star - n_*) + O(N)$	$O((\sum_{i=1}^{h} n_i)^c)$ $= O((a * n_\star - n_*)^c)$	$O(\sum_{i=1}^{h} n_i) = O(a * n_\star - n_*)$ + O(N), or $h * O(N) + O(N)$

Note that for Scheme 4, we have to include the cost in finding the SOSS, which is mainly dominated by the time in locating and transferring N instances, taking $O(N)$ of Cost 1 and $O(N)$ of Cost 3. Considering this cost, we can see from Table 6 that: (1) Scheme 2 is surely the best of all schemes, while Scheme 1 is usually the worst, considering all the three costs; (2) Scheme 4 is worse than Scheme 3 in Cost 1, because it needs extra $O(N)$ in finding the SOSS; (3) Scheme 4 is always better than Scheme 3 in Cost 2, because usually $n_0 \ll n_*$; (4) when locating n instances of a random sample does not need to go through D, Scheme 4 is worse than Scheme 3 in Cost 3; otherwise, it is better than Scheme 3. Therefore, Scheme 4 should be used when the cost of running a learning algorithm is dominating, or repeatedly scanning a large data set is highly expensive.

Although little work on measuring sample quality has been reported in the literature, the research on sufficient sample sizes is abundant. It is closely related to this work. By defining two different error measures for sentences, Kivinen and Mannila [10] give almost-optimal upper and lower bounds for the necessary sample size for a given relation. They theoretically prove that sampling is a powerful method in data mining. Aiming to show that random sampling of transactions in large databases is an effective method for finding association rules, Zaki et al [21] and Toivonen [20] try to determine a sufficient sample size based on statistical bounds (e.g., *Chernoff* bounds). Nevertheless, in applications, these bounds may still be too loose to give an optimal size. To find an appropriate sample size, John and Langley [9] compare *Static Sampling* and *Dynamic Sampling* to find a *Probably Close Enough* size. Provost et al [17] present *Progressive Sampling* to search progressively the minimum sufficient sample size. The database community has also been interested in finding the optimal sample size. To estimate the size of a database query, Lipton and Naughton [14] propose *adaptive sampling*; to construct histogram for query optimization, Chaudhuri et al [2] suggest *adaptive page sampling*.

In our quality measure, statistics such as means and covariances are used to summarize information from data. Similar application of such statistics can be found in recent work on squashing data by DuMouchel et al [5], where moments of higher orders are also used as summarization of the original data. While their purpose is different from ours, their success does encourage us to consider higher order moments when means and covariances are not

enough to be sufficient statistics for some distributions and this should be one possible approach to improving our quality measure.

5 Conclusion and Future Research

In this paper, we address the approach to finding an Statistically Optimal Sample Size (SOSS) in a purely statistical way. We first introduce a novel measure of sample quality describing similarity between a sample and its mother data by measuring their information divergence. We design an algorithm to efficiently compute a sample's quality. We then introduce the concept of SOSS and prove that asymptotically, the SOSS will achieve learning accuracy very close to that achieved by the OSS. This claim is supported by our experimental results on some artificial data sets and UCI data sets. We then present an efficient algorithm to calculate a quality curve, on which the SOSS can be found by detecting convergence. We also discuss the use of SOSS and show when it will outperform existing "progressive sampling". To the best of our knowledge, this is the first attempt in the KDD literature to define such a measure for sample quality and apply it to finding a sample size approximate to the optimal one. Because of its soundness in statistical theory and low cost in computing, this approach is useful for data mining on large data sets.

Our future research will include several related aspects: (1) to give a more appropriate definition of sample quality for non-normal distributions numerical data; (2) to embed dependencies among categorical attributes into the definition of sample quality; (3) to reveal the underlying mechanism behind quality curves and learning curves; and (4) to explore new applications of the sample quality measure.

Acknowledgment

The authors wish to thank all anonymous referees for their valuable comments, constructive suggestions, and thorough reviews on the earlier version of the paper.

References

[1] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. (http://www.ics.uci.edu/~mlearn/MLRepository.html).

[2] S. Chaudhuri, R. Motwani, and V. Narasayya. Random sampling for histogram construction: How much is enough? In *Proceedings of ACM SIGMOD'98*, 1998.

[3] 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*, 8:866–883, 1996.

[4] M. Dash and H. Liu. Feature selection for classification. *Intelligent Data Analysis - An International Journal*, 17, 1997.

[5] W. DuMouchel, C. Volinsky, T. Johnson, C. Cortes, and D. Pregibon. Squashing flat files flatter. In *Proceedings of the 5th ACM Conference on Knowledege Discovery and Data Mining*. AIII/MIT Press, 1999.

[6] B. Ebanks, P. Sahoo, and W. Sander. *Characterizations of Information Measures*. World Scientific Publishing Co. Pte. Ltd, Singapore, 1998.

[7] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[8] L.J. Frey and D.H. Fisher. Modeling decision tree performance with the power law. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*. Morgan Kaufmann, 1999.

[9] G.H. John and P. Langley. Static versus dynamic sampling for data mining. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD'96)*. AAAI / MIT Press, 1996.

[10] J. Kivinen and H. Mannila. The power of sampling in knowledge discovery. In *Proceedings of the thirteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 77–85. ACM, 1994.

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

[12] T. Lim, W. Loh, and Y. Shih. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, 1999.

[13] Tjen-Sien Lim. Users' guide for logdiscr version 2.0, 1999. (http://recursive-partitioning.com/logdiscr/).

[14] R.J. Lipton and J.F. Naughton. Query size estimation by adaptive sampling. *Journal of Computer and System Science*, 51:18–25, 1995.

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

[16] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, USA, 1988.

[17] F. Provost, D. Jensen, and T. Oates. Efficient progressive sampling. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining(KDD'99)*, pages 23–32. AAAI/MIT Press, 1999.

[18] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993. (http://www.cse.unsw.edu.au/ quinlan/).

[19] P.D. Scott and E. Wilkins. Evaluating data mining procedures: techniques for generating artificial data sets. *Information Software Technology*, 41:579–587, 1999.

[20] H. Toivonen. Sampling large databases for association rules. In *Proceedings of the 22nd International Conference on Very Large Data Base (VLDB'96)*. Morgan Kaufmann, 1996.

[21] M.J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara. Evaluation of sampling for data mining of association rules. In *Proceedings of the 7th Worhshop on Research Issues in Data Engineering*, 1997.