Intelligent Sampling for Big Data Using Bootstrap Sampling and Chebyshev Inequality

Ashwin Satyanarayana
N-913, Computer Systems Technology
New York City College of Technology
300 Jay Street, Brooklyn, NY - 11201
asatyanarayana@citytech.cuny.edu

***Abstract*—The amount of data being generated and stored is growing exponentially, owed in part to the continuing advances in computer technology. These data present tremendous opportunities in data mining, a burgeoning field in computer science that focuses on the development of methods that can extract knowledge from data. In many real world problems, these data mining algorithms have access to massive amounts of data. Mining all the available data is prohibitive due to computational (time and memory) constraints. Much of the current research is concerned with scaling up data mining algorithms (i.e. improving on existing data mining algorithms for larger datasets). An alternative approach is to scale down the data. Thus, determining a smallest sufficient training set size that obtains the same accuracy as the entire available dataset remains an important research question. Our research focuses on selecting how many (sampling) instances to present to the data mining algorithm. The goals of this paper is to study and characterize the properties of learning curves, integrate them with Chebyshev Bound to come up with an efficient general purpose adaptive sampling schedule, and to empirically validate our algorithm for scaling down the data.**

I. INTRODUCTION

When dealing with large finite populations, generally there are two accepted options. The first option is to evaluate every unit of the population; such a process is sometimes called a census. However for a large population, this option will be time consuming and expensive. In practice, therefore, as the other option we may study the characteristics of a population by examining only a part of it. Such a technique is known as *sampling*, one of the most popular techniques of statistics.

The theory of sampling in statistics [1], developed during the past several decades, provides us with various kinds of reasonable scientific tools for drawing samples and making valid inferences about the population parameters of interest [2]. Therefore, sampling has been widely used in almost every domain of the real world as well as data mining. As a fast developing area, data mining is facing the challenge of large volume and high dimensional data sets. We believe that sampling methodology, with its base on statistics, should be theoretically and empirically useful to improve performance while mitigating computational requirements in data mining.

Compared to the complete enumeration, many practical sampling methods possess one or more of the following advantages: reduced cost, greater speed, greater scope, or greater accuracy. With a small number of observations in sampling, it is possible to provide results much faster but with much less cost than a complete population. However, it is a misunderstanding that sampling can reveal the true characteristics of a population. There is no known method of sampling selection and estimation which will ensure with certainty that the sampling estimates will be equal to the unknown population characteristics. It should be understood that relying on a sample nearly always involves a risk of reaching incorrect conclusions. Sampling theory can assist in reducing that risk, but a certain risk is always present in every sampling [3].

One way of evaluating a sampling strategy is the Probably Close Enough (PCE) criterion. The key is that the sampling decision should occur in the context of the data mining algorithm we plan to use. The PCE idea is to think about taking a sample that is probably good enough, meaning that there is only a small chance that the mining algorithm could do better by using the entire database instead. We would like the smallest sample size $|D_i|$ such that:

$$\Pr[\ |acc(D) - acc(D_i)| \geq \varepsilon\ ] <= \delta \qquad (1)$$

where $acc(D_i)$ refers to the accuracy of our mining algorithm after seeing a sample of size D_i (where D_i is a subset of D), $acc(D)$ refers to the accuracy after seeing all records in the database, ε is a parameter to be specified describing what "close enough" means, and δ is a parameter describing what "probably" means. PCE is similar to the Probably Approximately Correct bound in computational learning theory.

Our contributions in this paper include a Generalized Dynamic Adaptive Sampling approach that estimates the number of instances required for learning curve convergence at each iteration. We then implement our and other approaches for incremental and non-incremental learners using different sized data sets.

The format of the paper is as follows. We begin by surveying prior work on the learning curve phenomenon. In Section III we introduce our hill climbing approach with bootstrap sampling. We then perform our experiments on real

978-1-4799-3010-9/14/$31.00 ©2014 IEEE

datasets in the next section. We finally summarize our results and conclude.

II. PRIOR WORK IN SAMPLING LARGE DATA

A. Learning Curve Phenomenon

A learning curve (Fig. 1) depicts the relationship between sample size and model accuracy. The horizontal axis represents *n*, the number of instances in a given dataset, which can vary between *0* and *N*, the total number of available instances. The vertical axis represents the accuracy of the model produced by an induction algorithm when given a sample of size *n*. Learning curves typically have a steeply sloping portion early in the curve, a more gently sloping middle portion, and a plateau late in the curve [6]. The plateau occurs when adding additional data instances does not improve accuracy. When a learning curve reaches its final plateau, we say it is converged. We denote the training set size at which convergence occurs as n_{min}.

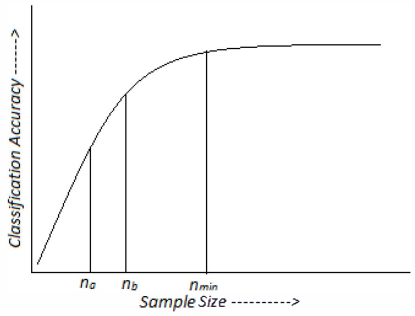

Fig 1: A hypothetical learning curve

B. PROGRESSIVE SAMPLING

John and Langley [5] define a method called *Arithmetic Sampling*, using a schedule $S_a = \langle n_0, n_0+n_\delta, n_0+2n_\delta, \ldots\ldots N\rangle$, where n_0 is the starting sample size, and n_δ is the fixed difference between successive terms. For example, one arithmetic schedule is *<100, 200, 300, …… N>*. As argued by Provost et al [7], the main drawback of arithmetic sampling is that if n_{min} is a large multiple of n_δ, then the approach will require many runs of the underlying algorithm to reach convergence.

One way to escape the limitations of arithmetic sampling is to use progressive sampling with a geometric schedule [7]. That is, $S_g = \langle n_0, a.n_0, a^2.n_0, a^3.n_0, \ldots\ldots N\rangle$ where a is the common ratio. One of the limitations of this approach is *overshooting*. For example, in the KDD CUP dataset, where n_{min} = 56,600, the geometric schedule is as follows: *<100, 200, 400, 800, 1600, 3200, 6400, 12800, 25600, 102400>*. Notice here that the last sample has overshot n_{min} by *45,800* instances.

C. LIMITATIONS OF PROGRESSIVE SAMPLING

There are certain drawbacks of Progressive sampling

a. *Sampling Schedules determined apriori*: One of the main problems with progressive sampling is that it is not dependent on the dataset at hand.

b. *Overshooting:* As described in the previous section, progressive sampling overshoots n_{min} as the sample size increases exponentially with each iteration.

c. *Sample measure*: There is no measure of uncertainty such as bias or variance used for picking any sample. A good sampling algorithm is expected to have low bias and low sampling variance. There characteristics have not been explored in progressive sampling

d. *Convergence tests*: The tests of convergence used in progressive sampling are not full proof, and the algorithm could converge at a local optimum (if the learning curve is not smooth) instead of the actual global optimum (plateau region)

D. DYNAMIC ADAPTIVE SAMPLING USING CHERNOFF INEQUALITY

In order to overcome some of the limitations of progressive sampling, in our prior work we introduced a Dynamic Adaptive Sampling [8][9] schedule, which selects instances to be included in the sample that depends on data characteristics obtained from the current sample. The primary purpose of adaptive (that varies with the problem) sampling is to take advantage of data such as classification accuracy in order to obtain more precise estimates of the next sample. In progressive sampling, the sampling schedule was determined apriori independent of the dataset without taking data characteristics into account.

Dynamic Adaptive sampling can be considered as sampling and mining being performed side by side to take advantage of the result of preliminary mining for more effective sampling. We used Chernoff inequality [12] to derive an expression which estimates the total number of instances needed at each iteration of the algorithm [8]. The number of instances to add at each iteration was derived to be:

$$m \geq \frac{2}{\frac{1}{|D_i|}\sum_{i=1}^{|D_i|} acc(x_i)}\left[\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right] \quad (2)$$

where D_i is the sample under consideration,
$acc(x_i)$ is the classification accuracy of the instance x_i.
ε is the approximation parameter and
δ is the probability of failure.

At each step, we check for convergence using the equation below. If not converged, we take another sample of size $|D_{i+1}|$ and compute the classification accuracy.

$$\left|\frac{1}{|D_i|}\sum_{i=1}^{|D_i|} acc(x_i) - \frac{1}{|D_{i-1}|}\sum_{i=1}^{|D_{i-1}|} acc(x_i)\right| < \varepsilon \quad (3)$$

Our empirical results showed that adaptive sampling outperforms progressive sampling techniques with respect to the number of instances required and the computation time in order to obtain the same level of accuracy, where samples are drawn independently from one another. Our work has been used in other real world applications such as nano-scale CMOS invertors [11] and chemical engineering [10]. However, all these applications use Artificial Neural networks which is a non-incremental algorithm.

E. LIMITATIONS OF OUR PRIOR WORK

a. *Bias and Variance*: Although our Dynamic, adaptive sampling technique used classification accuracy as the data characteristic, it did not consider sample bias or variance of the sample.
b. *Not useful for incremental learners*: We showed empirically [8] that adaptive sampling works well for non-incremental learners (such as neural networks) where the samples are drawn independent from one another, but fails to work for incremental learners.

Our current work addresses the limitations mentioned above and provides a more general purpose algorithm which would work for *all* learners.

III. CURRENT WORK: ADAPTIVE SAMPLING USING BOOTSTRAP SAMPLING AND CHEBYSHEV INEQUALITY

Our contributions in this paper include presenting a Generalized Dynamic Adaptive Sampling (GDAS) with an incremental approach that hill climbs to the plateau region using utility metrics that are estimated by sampling. The samples can be subsets of one another and are incrementally added. We will also address the limitation of not considering sample variance in our previous approach by using non-parametric bootstrap sampling along with Chebyshev Inequality to reach the plateau region of the learning curve. We now provide some background to our work.

Definition 1: (Utility confidence interval): *Let* u *be the utility function. Let* u(D) *denote the true quality when using all of the data, and let* $\hat{u}(D_i)$ *denote its estimated quality based on a sample* $D_i \subseteq D$ *of size m. Then θ is a utility confidence bound for u iff for any δ, 0 < δ ≤ 1,*

$$\Pr[\,|u(D) - \hat{u}(D_i)| \le \theta\,] \ge 1 - \delta \quad (4)$$

Equation (4) says that θ provides a two-sided confidence interval on $\hat{u}(D_i)$ with confidence δ. In other words, the probability of drawing a sample D_i, such that the difference between the true and estimated utility of any hypothesis disagree by θ or more (in either direction) lies below δ. If, in addition for any δ, $0 < \delta \le 1$, and any $\varepsilon > 0$ there is a number m such that $\theta \le \varepsilon$, we say that the confidence interval vanishes. In this case, we can shrink the confidence interval (at any confidence level δ) to arbitrarily low nonzero values by using a sufficiently large sample.

The utility function we consider is the average over all instances, of some instance function $f(x_i)$, where $x_i \in D$. The utility is then defined as:

$$u(D) = \frac{1}{|D|}\sum_{i=1}^{|D|} f(x_i) \quad (5)$$

(average over the entire dataset) and

$$\hat{u}(D_i) = \frac{1}{|D_i|}\sum_{i=1}^{|D_i|} f(x_i) \quad (6)$$

(average over the sample D_i)

For the rest of this paper, the utility function that we will be using is the classification accuracy $acc(x_i)$ (i.e. $f(x_i)=acc(x_i)$).

Definition 2: (Chebychev Inequality* [12]*): *For any probability distribution, the probability of the estimated mean p' being more than ε far away from the true mean p after m independently drawn points is bounded by:*

$$\Pr[|p - p'| \ge \varepsilon] \le \left(\frac{\sigma^2}{\varepsilon^2 p'^2}\right)\frac{1}{m} \quad (7)$$

By the property of learning curves and the weak law of large numbers, we know that:

$$\lim_{n\to\infty} \sigma^2 = 0 \quad (8)$$

Using the above in Equation (8) we get:

$$\lim_{n\to\infty} Pr[|p - p'| \ge \varepsilon] = 0 \quad (9)$$

Here are two reasons why we use Chebyshev Inequality for our generalized sampling algorithm:

a. Chebyshev bound give a better bound than Markov Inequality and Chernoff bound because they use more information about the distribution. Specifically, they use information about the standard deviation of the random variable.
b. Chebyshev bound applies to a class of random variables and does give exponential fall-off of probability with distance from the mean.

We will now briefly explain our *hill climbing sampling algorithm*. From an initial sample D_1, whose size is (say) $1/10^{th}$ the size of the entire dataset, we apply some learning algorithm (e.g. Decision trees, Support vector machines, Neural networks, etc) to it, gather utility measures – classification accuracy and variance– of that initial sample, and then determine whether to add in more number of instances. When to stop this process is a sequential decision problem. The challenge is that we do not know the true utility p as that would require using all the data, which defeats the purpose of sampling. In order to address this challenge we consider pairs of samples, and look one step at a time.

Let D_{opt} be the oracle specified optimal sample whose size is n_{min} (i.e. $|D_{opt}| = n_{min}$) where n_{min} is the smallest sufficient sample size to reach the plateau region. Ideally we would stop when $|u(D_{opt}) - u(D_i)| < \varepsilon$ (for some D_i), that is when we are within ε distance from the optimal utility measure $u(D_{opt})$ (in either direction). Since we would not know D_{opt} for any given dataset, we use a "myopic" strategy of looking one step at a time. In order to use this myopic strategy, we need to define the population p and the estimate p' as follows.

Definition 3: The population we will consider are all distinct pairs of training set sizes (a,b), where a>b>1, obtained from two consecutive sampling steps.

The destination of our hill climbing algorithm is the plateau region of Fig.1. At the plateau region, the Probably close enough criterion (1) is satisfied as $u(D_a) \approx u(D_b)$.

Definition 4: We define the true value p as follows:

$$p = u(D_a) - u(D_b), where\ |D_a| > |D_b| > n_{min} \quad (10)$$

where $u(D_a)$ and $u(D_b)$ are utility measures of samples of sizes $|D_a|$ and $|D_b|$ respectively. For all practical purposes, we make the assumption that once we are at the plateau region (i.e. when $u(D_a) \approx u(D_b)$), the **true value $p = 0$**.

Definition 5: The estimated value p' is defined as follows

$$p' = u(D_a) - u(D_b), where\ |D_a| > |D_b| > 1 \quad (11)$$

Note that this definition is akin to the definition of the true value p except that the constraint $|D_b| > n_{min}$ may not be satisfied in this case (i.e. we may not have reached the plateau region).

We can then use the values of p and p' to determine the number of instances needed at each iteration of our algorithm using Chebyshev Inequality. With each hill climbing step, the estimated value p' approaches closer to the true value (i.e. p =0). We converge when the difference between the two consecutive utilities becomes smaller than ε (i.e. when we have reached the plateau region).

The Chebyshev inequality gives us an estimate for the total number of instances (m) to ensure that the difference will be within distance ε of the optimum utility. For any two distinct consecutive steps of the hill climbing algorithm of sample sizes $|D_i|$ and $|D_{i-1}|$, by substituting the true value p (which is zero by definition above), the estimated value $p' = u(D_i)-u(D_{i-1})$, the bootstrapped variance σ^2_{BOOT} (discussed in section B), the given approximation parameter ε and the given confidence parameter δ, we reduce equation (7) to the following:

$$Pr[|0 - (u(D_i) - u(D_{i-1}))| \geq \varepsilon] \leq \left(\frac{\sigma^2_{BOOT}}{\varepsilon^2.(u(D_i) - u(D_{i-1}))^2}\right)\frac{1}{m} \leq \delta \quad (12)$$

We then solve for m from the above equation to obtain:

$$m \geq \left(\frac{\sigma^2_{BOOT}}{\varepsilon^2.(u(D_i) - u(D_{i-1}))^2}\right)\frac{1}{\delta} \quad (13)$$

A. Stopping Criterion: Four Special Cases of GDAS

An important ingredient for sampling algorithms is to determine when to stop at a particular sample. At each iteration of our hill climbing algorithm, we check for convergence, by performing the following two tasks:

a. Check if the difference in utilities of two consecutive steps to be greater than or less than the approximation parameter ε. (i.e. Compare $|u(D_i)-u(D_{i-1})|$ with ε), and
b. Compute the probability of event (a) occurring with Chebyshev inequality and compare this probability with the predefined confidence parameter δ.

Thus we would have four different criteria as follows:

Case (a): $|u(D_i)-u(D_{i-1})| < \varepsilon$ and $\left(\frac{\sigma}{\varepsilon}\right)^2 \frac{1}{m} < \delta$

We have reached convergence with probability of success $\geq 1-\delta$ and the error has stabilized.

Case(b): $|u(D_i)-u(D_{i-1})| < \varepsilon$ and $\left(\frac{\sigma}{\varepsilon}\right)^2 \frac{1}{m} > \delta$

Although it appears that we have reached convergence from $|u(D_i)-u(D_{i-1})| < \varepsilon$, we have not done so, as the second constraint shows us. We then discard this sample and start with a new sample.

Case (c): $|u(D_i)-u(D_{i-1})| \geq \varepsilon$ and $\left(\frac{\sigma}{\varepsilon}\right)^2 \frac{1}{m} < \delta$

We have reached convergence as the probability of exceeding ε is less than δ.

Case (d): $|u(D_i)-u(D_{i-1})| \geq \varepsilon$ and $\left(\frac{\sigma}{\varepsilon}\right)^2 \frac{1}{m} > \delta$

We solve for the number of instances m using the previously derived equation.

$$m \geq \left(\frac{\sigma^2_{BOOT}}{\varepsilon^2.(u(D_i) - u(D_{i-1}))^2}\right)\frac{1}{\delta} \quad (14)$$

And add m instances to form the next sample D_{i+1}.

B. Bootstrap Sampling

In order to use the Chebyshev Inequality to solve for the number of instances at each iteration, we need to compute the variance term σ^2. This variance σ^2 requires that the samples be independent of one another. However, in our method we add samples incrementally. This makes the samples dependent on one another as the old sample becomes a subset of the newly added incremental sample – $D_i \subseteq D_{i+1}$ (=D_i+m). Davidson [16] has shown that the posterior standard deviation obtained by bootstrapping σ_{BOOT} provides a good estimate for this standard deviation σ. Hence we use non-parametric bootstrapping (i.e. sampling with replacement) with our incremental algorithm, and determine the bootstrapped variance σ^2_{BOOT}, which would provide a good estimate for independent random samples.

1. Select B independent bootstrap samples $\chi^*_1, \chi^*_2, \ldots\ldots\ldots \chi^*_B$ each consisting of n data values drawn with replacement from the original sample.
2. Evaluate the variance corresponding to each bootstrap sample $s(\chi^*_b)$
3. Estimate the boostrap variance using the following:

$$\sigma_{BOOT} = \frac{\left\{\sum_{b=1}^{B}\left[s(\chi^*_b) - s(\overline{\chi^*})\right]^2\right\}}{(B-1)^{\frac{1}{2}}} \quad (15)$$

where

$$s(\overline{\chi^*}) = \frac{1}{B}\sum_{b=1}^{B} s(\chi^*_b) \quad (16)$$

To show that bootstrapped variance produces good estimates of the actual variance, we performed the following experiment. We took a sample from the CREDIT dataset, say of size n. We then took 300 independent samples of size n from the entire dataset and obtained the actual variance σ^2. We then took 300 bootstrapped samples from the original sample of size n and then computed the variance σ^2_{BOOT}. The results are shown in Table 1.2.

Sample Size	*Accuracy*	$\sigma^2(acc_i)$	$\sigma^2_{BOOT}(acc_i)$
100	67.4	10.04	11.75
200	71.2	6.16	7.08
300	71.65	5.84	6.18
400	71.83	5.52	5.98
500	72.98	1.48	1.68
600	73.65	1.11	1.35
700	74.02	0.68	0.61

Table 1.2: Comparison of the actual and the bootstrapped variance for CREDIT_G dataset.

C. GDAS Algorithm

Algorithm GDAS*(D,ε ,δ)*

Input: Training dataset D, approximation parameter ε and the probability measure δ

Output: Total number of instances and mean computation time (for convergence)

Step 0: $\hat{u}(D_0) \leftarrow 0$

Step 1: Randomly select (1/10)|D| instances (=$|D_1|$). Apply the learner (e.g. Decision Tree), determine $\hat{u}(D_1)$ (using Equation (5))

Step 2: For each iteration *i* (≥1) do:

a. Check for convergence using the criteria:

Test	$\left(\frac{\sigma}{\varepsilon}\right)^2 \frac{1}{m} < \delta$	$\left(\frac{\sigma}{\varepsilon}\right)^2 \frac{1}{m} \geq \delta$
$\lvert\hat{u}(D_i) - \hat{u}(D_{i-1})\rvert < \varepsilon$	Yes, Exit	Discard the sample and start again
$\lvert\hat{u}(D_i) - \hat{u}(D_{i-1})\rvert \geq \varepsilon$	Yes, Exit	Add instances according to Chebyshev Bound

b. Add *m* instances using equation 14 to form the new sample D_{i+1}

c. Apply the classification mining algorithm on the sample and determine $\hat{u}(D_{i+1})$

Fig 2: Generalized Dynamic Adaptive Sampling Algorithm (GDAS)

Theorem 1: The GDAS(D,ε,δ) algorithm produces a series of utilities $\hat{u}(D_0)$, $\hat{u}(D_1)$, $\hat{u}(D_2)$,……$\hat{u}(D_m)$, such that,

(1) $\hat{u}(D_m) \geq \hat{u}(D_i)$, $0 \leq i \leq m-1$

(2) the sample mean $\frac{1}{n} * \sum_{i=1}^{m}[\hat{u}(D_{i+1}) - \hat{u}(D_i)]$ *converges to the population mean as* $n \rightarrow \infty$ *where n is the number of samples.*

Proof: Our approach moves from D_i to D_{i+1} iff the expected utility $u(D_{i+1})$ is better than $u(D_i)$ by at least ε, and hence (1) follows. We define $q_i=\hat{u}(D_{i+1})-\hat{u}(D_i)$. Let S be the sample mean over n samples given by:

$$S = \frac{1}{n} * \sum_{i=1}^{m} q_i$$

This average tends to the true population mean as $n \rightarrow \infty$ at the rate of convergence given by Chernoff Bounds: The probability that "q_i is more than $\mu+\gamma$" goes to 0 exponentially fast as *n* increases; and for a fixed *n*, exponentially as γ increases and hence (2) follows. Formally we have:

$$\Pr[q_i > \mu + \gamma] \leq e^{-2n\left(\frac{\gamma}{\Lambda}\right)^2}$$

where Λ is the range of possible values for $\hat{u}(D_{i+1}) - \hat{u}(D_i)$. □

IV. EMPIRICAL RESULTS

We now present from our experiments on three datasets from the UCI repository: LED, WAVEFORM and CENSUS (adult) which was also used by Provost et al [7] and two massive real world Web traces: KDD CUP 2000 and NASA-HTTP. We use C4.5- decision trees (incremental learner) as our classification algorithm for KDD CUP 2000 and use Neural Network (non-incremental learner) for NASA-HTTP. We compare our method with other methods (Full, Geo, Chernoff, Oracle), and the results are in Table 4.1 and 4.2

Web Applications: In order to evaluate the different comparisons on large real world problems, we ran it on two massive Web traces. The first was the dataset used in the KDD CUP 2000 competition. The second was a trace of all requests made to the website of the busy NASA-HTTP which consists of one month worth of all HTTP requests to the NASA Kennedy Space Center WWW server in Florida.

The KDD cup data consists of 777,000 web page requests collected from an e-commerce site. Each request is annotated with a request session ID and a large collection of attributes describing the product in the requested page. We focused on one of the fields in the log, "Assortment Level 1". For each session we produced an example with one Boolean attribute – "True" if the session visited a page with that category. There were 235,000 sessions in the log which we used for our experiments.

The NASA-HTTP dataset was created from a log of every request made to the NASA Kennedy Space Center WWW server in Florida. We extracted dataset of size 461,612 instances from this log in a manner similar to the KDD cup dataset.

We compare the different approaches with our method using the following two performance criteria.

a. ***The mean computational time***: The runtimes averaged over 20 runs of each of the five datasets

b. ***The total number of instances needed to converge***: If the sampling schedule is $S=<|D_1|, |D_2|, |D_3|.....|D_k|>$, then the total number of instances would be $|D_1|+|D_2|+|D_3|+...+|D_k|$.

We compare our convergence method with other methods, which are as follows:

(Full): $S_N=\{N\}$, a single sample with all the instances. This is the most commonly used method. This method suffers from both speed and memory drawbacks.

(Geo): Geometric Sampling [7], in which the sample size is created geometrically, $S_g = \{|D_1|, a.|D_1|, a^2.|D_2|.....a^k|D_1|\}$. We use $|D_1|=100$ and $a=2$ as used by Provost et al [7].

(Chernoff): Adaptive sampling using Chernoff Bounds [8] where we use $\varepsilon = 0.001$ and $\delta = 0.05$ (95% probability).

(GDAS): Our Generalized Dynamic Adaptive Sampling approach where we use $\varepsilon = 0.001$ and $\delta = 0.05$.

(Oracle): $S_O=<n_{min}>$, the optimal sample size determined by the omniscient oracle; we determined n_{min} empirically by analyzing the full learning curve beforehand.

Dataset	Full: $S_N=<N>$	Geo: $S_g=a^k.\lvert D_i\rvert$	Chernoff	GDAS	Oracle $S_O=<n_{min}>$
LED	100,000	6,300	6,100	5,100	2,000
WAFEFORM	100,000	25,500	20,030	16,108	12,000
CENSUS	32,000	25,500	14,322	10,014	8,000
KDD CUP	235,000	204,700	304,322	67,800	56,600
NASA-HTTP	461,612	409,500	278,433	158,345	130,645

Table 4.1: Comparison of the total number of instances required for the different methods to reach convergence.

Dataset	Full: S_N=<N>	Geo: $S_g=a^k.\|D_i\|$	Chernoff	GDAS	Oracle S_O=<n_{min}>
LED	46.51	15.67	20.34	25.87	5.72
WAFEFORM	558.91	89.76	124.45	156.73	32.85
CENSUS	48.76	10.77	18.93	27.84	13.87
KDD CUP	17,870.59	5,616.89	7,198.24	3,116.84	1,826.16
NASA-HTTP	38,160.78	13,482.49	10,232.44	8,713.00	4,719.85

Table 4.2: Comparison of the mean computational time (in CPU seconds) required for the different methods to obtain the same accuracy (averaged over 20 runs of the experiment).

A. Discussion

1. Geometric Sampling: Geometric sampling is between three and six times faster in terms of the computation time than learning with all of the data *(Full)*

2. GDAS v/s other approaches:

(a) GDAS v/s Full: The adaptive scheduling approach is between two and four times faster than using the entire dataset to obtain the same accuracy.

(b) GDAS v/s Geo: For medium sized datasets (UCI repository) *GDAS* outperforms *Geo* by about 20% in terms of the number of instances required for convergence (see Table 4.1). For massive datasets, geometric sampling overshoots by a very large amount. For example in the KDD CUP dataset, where n_{min} = 56,600, the geometric schedule is as follows: *<100, 200, 400, 800, 3200, 6400, 12800, 25600, 51200, 102400>*. Notice here that the last sample has overshot n_{min} by 45,800 instances. In such massive datasets, *GDAS* outperforms *Geo* by more than 50%.

In terms of computational time, *Geo* outperforms better than *GDAS* for small and medium sized datasets (see Table 4.2) mainly due to the bootstrapping overhead of *GDAS*. However for very large datasets, *GDAS* performs better than Geo, because it minimizes overshooting n_{min}.

(c) In all datasets, *GDAS* comes closest to the optimal oracle in terms of the number of instances required for convergence. (see Table 4.1)

3) *GDAS v/s Chernoff Bound:*

(a) Incremental Learners: Chernoff inequality performs poorly (as expected) for KDD CUP data (incremental learner). *GDAS* outperforms Chernoff Inequality by 50%.

(b) Non-incremental learners:GDAS outperforms Chernoff by 22% in the number of instances needed for convergence.

We close this section with some general comments on the GDAS algorithm:

a. *GDAS* will process more samples using later elements than using the earlier ones (similar to geometric sampling), as its convergence tests are increasingly more difficult to pass. This is desirable, as the overall system is dealing with increasing number of samples using later, more better instances.

b. At any time, *GDAS* will provide a usable result, with the property that later models are clearly more better than earlier models i.e. for $i > j$, $u(D_i) > u(D_j)$ with high probability.

V. CONCLUSION

Computing the optimal sample size is fundamental question to answer in data mining. Prior work in this area showed that the progressive sampling literature allows for a fixed apriori specification of the number of instances for the sampling schedule. Our prior work using Chernoff bounds although performed better than progressive sampling, worked well only with non-incremental learners. Our contributions in this paper include a general purpose technique that works for all learners. We showed empirically that our approach outperforms other approaches with respect to number of instances required and computational time. Future work will look at exploring statistical bounds to other areas of data mining such as clustering and classification.

REFERENCES

[1] Vladimir N Vapnik: Statistical Learning Theory. John Wiley & Sons. New York, NY (1998)

[2] Valiant, L.G. 1984. A theory of the Learnable Communications of the ACM 27(11), 1134-1142 (1984)

[3] Tryfos, P. Sampling Methods for Applied Research: Text and Cases, John Wiley & Sons, Inc (1996)

[4] Haussler, D., Kearns, M., Seung, H.S., Tishby, N.: Rigorous Learning Curve Bounds from Statistical Mechanics. In: Proc. 7th ACM Workshop on Comp. Learning Theory (1994)

[5] John, G., Langley, P.: Static versus dynamic sampling for data mining. In: Proc. of the 2nd International Conference on Knowledge Discovery and Data Mining, pp. 367–370 (1996)

[6] Meek, C., Theisson, B., Heckerman, D.: The learning-curve sampling method applied to model- based clustering. The Journal of Machine Learning Research (2002)

[7] Provost, F., Jensen, D., Oates, T.: Efficient progressive sampling. In: Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining, pp. 23–32 (1999)

[8] Satyanarayana. A., Davidson. I.: A Dynamic Adaptive Sampling Algorithm (DASA) for Real world Applications: Finger Print Recognition and Face Recognition. In Proc.15th ISMIS 2005,pp631-640

[9] Satyanarayana. A., Intellingent Sampling and Filtering. In Proc of 21st AAAI conference (AAAI 2006), Doctoral Consortium,Boston, MA

[10] Nuchitprasittichai, Aroonsri, and Selen Cremaschi. "An algorithm to determine sample sizes for optimization with artificial neural networks." *AIChE Journal* (2012).

[11] Dhabak, Dipankar, and Soumya Pandit. "Adaptive sampling algorithm for ANN-based performance modeling of nano-scale CMOS inverter." *World Acad Sci Eng Technol* 80 (2011): 812-818.

[12] Mallows, C. L., and Donald Richter. "Inequalities of Chebyshev type involving conditional expectations." *The Annals of Mathematical Statistics* 40.6 (1969): 1922-1932.

[13] Chernoff, H.: A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations. Annals of Mathematical Statistics 23, 493–507 (1952)

[14] Davidson, I.: An Ensemble Approach for Stable Learners. The National Conference on A.I. (AAAI), San Jose (2004).