Alcom-FT Technical Report Series
ALCOMFT-TR-02-116

Sequential Sampling Algorithms: Unified Analysis and Lower Bounds

Ricard Gavaldà*
Universitat Politècnica de Catalunya
Barcelona, Spain

Osamu Watanabe†
Tokyo Institute of Technology
Tokyo, Japan

May 23, 2002

Abstract

Sequential sampling algorithms have recently attracted interest as a way to design scalable algorithms for Data mining and KDD processes. In this paper, we identify an elementary sequential sampling task (estimation from examples), from which one can derive many other tasks appearing in practice. We present a generic algorithm to solve this task and an analysis of its correctness and running time that is simpler and more intuitive than those existing in the literature. For two specific tasks, frequency and advantage estimation, we derive lower bounds on running time in addition to the general upper bounds.

Key words: Random sampling, sequential sampling, adaptive sampling, Chernoff bounds, Data mining.

1 Introduction

It is a universal phenomenon that data repositories in all domains are growing at an exponential rate. In many applications, the amount of available input data is already so large that not all data can be processed, maybe not even read, within reasonable time. *Random sampling* is one possible strategy that algorithm designers can use to deal with on massive amounts of data.

Typically, a sampling algorithm obtains a random sample from a large database, performs some computation on the sample, and hopes that the result is not too different from what would be obtained by computing over the whole database. Of course, for some computation tasks this

*Address: Dept. de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain. E-mail: `gavalda@lsi.upc.es`. Partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT), by EU EP27150 (Neurocolt II), by CIRIT 1997SGR-00366, TIC2000-1970-CE, and by the Spanish Government PB98-0937-C04-04 (project FRESCO).

†Address: Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro-ku Ookayama, Tokyo 152-8552, Japan. E-mail: `watanabe@is.titech.ac.jp`. Supported in part by Grant-in-Aids for Scientific Research on Priority Areas "Discovery Science" (1998–2000) and for Scientific Research (C) (2001–2002) from the Ministry of Education, Science, Sports and Culture of Japan.

approach fails completely; either the result is required to be exact, or there is no way of obtaining even an approximate answer without looking at all data. Often, however, an approximate answer is acceptable, and can be obtained with some confidence by looking at samples of reasonable size.

To make the discussion concrete, we assume from now on that the task is that of estimating some quantity μ implicit in the database. As an easiest example, μ could be the fraction of database records having a certain property. Then our task is to find some approximation $\hat{\mu}$ of μ up to ϵ that is correct with probability at least $1-\delta$, for given parameters ϵ and δ. Depending on the application, we may require *absolute* approximation, that is, $\mu-\epsilon \leq \hat{\mu} \leq \mu+\epsilon$, or *relative* one, that is, $(1-\epsilon)\mu \leq \hat{\mu} \leq (1+\epsilon)\mu$. We call this type of approximation (especially, the relative one) a *(ϵ, δ)-approximation* of μ.

In these situations, large deviation bounds (such as Chernoff or Hoeffding type bounds, or those given by the Central Limit Theorem) may be used to determine a sample size that guarantees such approximations. These bounds have often been used, more or less rigorously, to support sampling strategies in different domains, such as machine learning and data mining [4, 7, 8, 13, 18, 19]. In some scenarios, remarkable speedup has been reported, sometimes by orders of magnitude. In other works, however, researchers have considered sampling either impractical, or difficult to use. There are, in our opinion, two causes for the negative conclusions:

(1) Hoeffding (additive) type bounds are worst case, in the sense that they apply to all parameter settings. Generally this leads to prohibitive sample size. Furthermore, Hoeffding bounds can only be applied for absolute, not relative approximation.

(2) Chernoff (multiplicative) type bounds provide relative approximations and smaller bounds, but cannot be applied to our problem because they depend on the unknown quantity to be estimated. To make this point clear, suppose that μ is simply the frequency of some event in the database. Chernoff bound states that sample size $O((1/(\epsilon^2\mu)) \cdot \ln(1/\delta))$ suffices for obtaining a relative (ϵ, δ)-approximation of μ. Since μ is the unknown, it is not clear how to apply this formula to determine sample size.

Sequential sampling algorithms try to overcome these difficulties in the following way. Instead of obtaining the whole sample from the start, they collect database records sequentially; from time to time (in the extreme case, at every step) they decide whether they have already seen enough records to reach a safe conclusion, and if so, they stop. Since, in practice, we are not often in the worst-case situation, these algorithms usually stop much before their worst-case bound. They have also been called *adaptive* [5, 6] because their performance adapts to the complexity of the particular problem instance being solved.

Several researchers in the machine learning and the KDD communities have used heuristics based on stopping sampling when some conclusion seems very likely [13, 18, 19]. On the other hand, statisticians have worked extensively on sequential sampling since the 40s (Wald's work is classical [23]; see, e.g., [10] for recent works). But they focused mostly on minimizing the number of experiments for hypothesis testing.

Only very recently, some sequential sampling algorithms have been developed that (i) can be applied to current, general, KDD tasks and that (ii) have theoretical guarantees of correctness and performance [5, 6, 20, 21, 22]. In these works, the algorithms have been experimentally tested against those using non-sequential sampling, or no sampling at all, and have systematically outperformed them. Typically, the performance of these algorithms depends on the quality of the data rather than the size of the database; so their advantage is expected to be more apparent as databases grow.

We thus advocate for trying to apply sequential sampling algorithms in more and more situations. This is difficulted by at least two factors. First, the analyses in the literature (e.g., [6, 21, 22]) are somewhat complicated and for the particular form of the task that is being solved, instead of reflecting clearly the intuition of why adaptive sampling works; thus, when a new task has to be solved most of the analysis has to be done again. Second, there are few theoretical tools to determine how good a given sequential sampling algorithm is for some task, that is, whether it is about as fast as possible or whether a large improvement is possible by more clever design; but see [2] for an exception.

This paper contributes to these issues as follows:

1. We isolate a basic sequential sampling task from which many others can be derived; we present a very generic sequential estimator and provide an analysis of the correctness and efficiency of this procedure much more intuitive and simple than those in the literature (e.g., [6, 21, 22]). The analysis is given in an almost "recipe" form for the use of algorithm designers.
2. We present a couple of case studies (already discussed in the literature) showing the conciseness of analyses obtained this way.
3. We show how this analysis technique may lead to lower bounds on the running time of some sequential sampling techniques. In particular, we prove lower bounds for the running time for (ϵ, δ)-approximation in so-called *frequency estimation* and *advantage estimation* problems; these lower bounds match (up to doubly log factors) the upper bounds given by our implementations of the generic estimator, which therefore are almost optimal.

2 Previous Work

We review in this section some existing work on sequential sampling applied to Knowledge Discovery in databases. Of course, other fields of computer science have used sampling extensively, too; see, e.g., [2].

Example 1. Lipton et al. [15, 16] provide algorithms for estimating the size of queries to the database. Their core algorithm is a sequential sampling method for estimating the proportion p of records in a database that satisfy a certain property. That is, the task is essentially estimating a probability p that a certain event occurs in a sequence of i.i.d. Bernoulli trials, a task we call *frequency estimation*. The target μ is the probability p in this frequency estimation. Their algorithm achieves relative (ϵ, δ)-approximation in time $O((1/(\epsilon^2\mu)) \cdot \ln(1/\delta))$.

Example 2. In [5, 6] we presented an algorithm for a task appearing often in the context of supervised learning: *advantage estimation* for a Boolean classifier. A boolean classifier predicts a 0/1 value on any example. The advantage γ of a Boolean classifier is its classification accuracy on a set of examples minus 1/2, i.e., how much better it predicts than random guessing. The target μ is this advantage γ (of a given classifier) in advantage estimation. The algorithm in [6] achieves relative (ϵ, δ)-approximation in time $O(s \cdot (\ln\ln s + \ln(1/\delta)))$, where $s = 1/(\epsilon^2\mu^2)$.

Example 3. In [6], the algorithm in [5] is generalized to approximating any μ that can be obtained applying a smooth function to the average of bounded variables. The algorithm is applied to the *hypothesis selection problem:* selecting a hypothesis from a set H with utility μ that is at most within $(1+\epsilon)$ factor of the best one. Its running time is $O(s \cdot (\ln\ln s + \ln(|H|/\delta))$, where s is as above.

Example 4. Scheffer and Wrobel [21, 22] observed that similar techniques work for any μ for which one can prove deviation bounds that tend to 0 as sample size grows. They use it for the *n-best hypotheses selection* task, i.e., for given ϵ and δ, select n hypotheses from H such that none of the selected hypothesis is more than (absolute) ϵ worse than any discarded hypothesis. The running time of their algorithm is in $O((1/\epsilon^2) \cdot \ln(1/\delta))$, but note that it achieves absolute rather than relative approximation. Additionally, their algorithm includes Hoeffding races [18] to discard soon hypothesis that are unlikely to be among the n best.

3 Our Framework and a Generic Estimator Algorithm

We describe in this section a basic task that we want to solve by means of sequential sampling.

We want to estimate some unknown quantity μ, of which we know only that $0 < \mu < 1$. To do this, we have available an infinite sequence of *examples* $X_1, X_2, \ldots$, drawn independently from example space X and some function $F : X^\star \mapsto R$ such that, for any t, $E[F(X_1, \ldots, X_t)] = \mu$.

An *estimator algorithm* A performs as follows: At the start, it reads input parameter δ. Then at each (time-)step $t \geq 1$, it reads the value of X_t and produces a pair of real numbers (μ_t, ϵ_t). Note that, strictly speaking, A is not an algorithm because it never stops.

We say that A is an *estimator for* μ if the following condition holds for all inputs δ,

(a) with probability at least $1 - \delta$, it holds for all t that

$$(1-\epsilon_t)\mu \ \leq \ \mu_t \ \leq \ (1+\epsilon_t)\mu, \quad \text{and}$$

(b) the sequence of ϵ_t tends to 0 as t grows.

For condition (a), we should be a bit careful on the order of quantification of δ and t. It requires that only with probability less than δ, there may be *some* step where approximation is not good.

We are interested in efficient estimator algorithms, for which the quantity ϵ_t tends to 0 as quickly as possible. Quite naturally, an estimator algorithm for μ can be used to obtain an

(ϵ, δ)-approximation to μ if both ϵ and δ are given: It is enough to run the estimator with parameter δ, monitor its output until ϵ_t goes below ϵ, and then produce μ_t as the estimation.

In this paper, we consider one generic estimator algorithm, which is stated in Figure 1. This algorithm is quite straightforward; it just computes μ_t by using the function F (that is given when specifying the problem) for observed set of examples $X_1, ..., X_t$. It is so natural that one may not be able to think of any other algorithm for solving the type of estimation problem that we are considering. Below we will show "a recipe" to prove the correctness and analyze the running time of this algorithm, depending on the convergence properties of the estimations for μ. We consider only relative approximation, but, if necessary, it is routine to develop another algorithm for absolute approximation. This would correspond to the algorithms in [21, 22].

procedure GenericEstimator
  input δ;
  $t := 0$; $S_0 := \emptyset$;
  while (true) **do**
    $t := t + 1$;
    $S_t := S_{t-1} \cup \{X_t\}$;
    compute $\mu_t := F(S_t)$;
    compute ϵ_t (by some formula obtained in the analysis);
    output (μ_t, ϵ_t);
  end-while

Figure 1: The Generic Estimator Algorithm

3.1 Implementation and Analysis

Here we explain how to implement our generic algorithm for solving a given estimation problem and how to prove its correctness / efficiency for that problem.

Step 1. Review the literature on large deviation bounds and find a good bound B for which we can prove that, for all α and t,

$$\Pr[\,(1-\alpha)\mu \;\le\; \mu_t \;\le\; (1+\alpha)\mu\,] \;\ge\; 1 - B(\alpha, \mu, t, \ldots).$$

Let us just write $B(\alpha, \mu, t)$ even though B may depend on other parameters (such as $\mathrm{Var}(\mu)$, or even S_t itself).

Step 2. Observe that for the sequence α_t defined by

$$B(\alpha_t, \mu, t) \;=\; \frac{\delta}{t(t+1)},$$

the total probability that there is an error at some t is below δ. This is because

$$\begin{aligned}&\Pr[\,\exists t:\ \neg[\ (1-\alpha)\mu\ \le\ \mu_t\ \le\ (1+\alpha)\mu\ ]\,]\\ &\quad\le\ \sum_{t\ge 1} B(\alpha,\mu,t)\ \le\ \sum_{t\ge 1}\frac{\delta}{t(t+1)}\ =\ \delta.\end{aligned}$$

Unfortunately, this definition of α_t depends on unknown μ, so we cannot simply define $\epsilon_t \stackrel{\rm def}{=} \alpha_t$ in the algorithm. But we can use some kind of "bootstrapping" approach: if we assume that μ_t is quite near μ, plugging it in the equation instead of μ should give good enough results.

Step 3. Assume that statement (1) holds:

$$\forall t:\quad (1-\alpha_t)\mu\ \le\ \mu_t\ \le\ (1+\alpha_t)\mu. \tag{1}$$

Assuming that B is decreasing in μ, we have

$$B(\alpha_t, \mu_t/(1+\alpha_t), t)\ \ge\ (\ B(\alpha_t,\mu,t)\ =\ )\ \delta/t(t+1),$$

and, intuitively, this inequality should be quite tight. Assuming that B is also decreasing in α (which is most often the case), solve this inequation for α_t, and obtain an upper bound for α_t in terms of μ_t and t (and possibly other parameters), which we denote by $B'(\mu_t,t)$. Define ϵ_t to be that upper bound; that is,

$$\alpha_t\ \le\ B'(\mu_t,t) \stackrel{\rm def}{=} \epsilon_t. \tag{2}$$

Step 4 (Conclusion). With probability at least $1-\delta$, the bound (1) holds for *any* $t \ge 1$. Then, so holds bound (2) (for any $t \ge 1$), and therefore

$$\forall t:\quad (1-\epsilon_t)\mu\ \le\ \mu_t\ \le\ (1+\epsilon_t)\mu. \tag{3}$$

Also, for reasonable bounds B, it should be immediate to check that the sequence ϵ_t tends to 0. Then we have proved that the algorithm, with this definition of ϵ_t, is an estimator for μ.

3.2 Efficiency

For given ϵ and δ, we can use our algorithm to obtain an (ϵ,δ)-approximation of μ. That is, we run the algorithm until ϵ_t becomes less than ϵ. The following procedure can be used to estimate the time used by the algorithm.

(1) We have a definition of ϵ_t using μ_t and t.

(2) We also have upper and lower bounds for μ_t in terms of μ, ϵ_t, and t, that hold with probability at least $1-\delta$.

(3) Solve for ϵ_t in these two equations. This gives bounds for ϵ_t in terms of μ and t that hold with probability $1-\delta$.

(4) Now demand that ϵ_t is at most ϵ, and solve for t. This gives a bound on the time t required/sufficient to reach $\epsilon_t \leq \epsilon$ with probability $1-\delta$.

An algorithmic trick to improve this bound on convergence time was introduced in [6]. The idea is to produce a new pair (μ_t, ϵ_t) not at each iteration, but update the estimation only at time steps t of the form $t = \lfloor s^i \rfloor$, for some constant $s > 1$. Equivalently, we may view this as follows: the algorithm is not taking one example X_i at each time step, but rather a batch of $s^i - s^{i-1}$ examples at time i. This way, new errors can be introduced at exponentially increasing time steps only, and it is enough to define sequence α_t as $B(\alpha_t, \mu, t) = \delta/(i(i+1))$. This results in a new definition of ϵ_t and in a smaller running time. Typically, this reduces a $\log t$ factor in the running time to some $\log\log t$ factor.

In [6], algorithms using this trick are referred to as *geometric* algorithms, by comparison with *arithmetic* algorithms that check some stopping condition at every step, or after each constant number of steps as in [13]. In [20] sampling schedules for machine learning problems are considered. Independently, the terms *arithmetic* and *geometric* schedules are proposed there to distinguish between essentially the same strategies. Also, geometric schedules are proved to be optimal in some precise sense.

4 Case Study (1): Frequency Estimation

In the following two sections, we present a couple of instances of the estimator and analysis above. The interested reader may want to compare this analysis with the lengthier ones in [5, 6, 21, 22, 24].

First we consider frequency estimation, i.e., Example 1 mentioned in Section 2. More specifically, we assume that we have access to a series of i.i.d. variables X_1, X_2, ..., such that each $X_i \in \{0,1\}$ takes value 1 with probability p, and our goal is to estimate p. That is, our target μ is the probability (or frequency) p of X_i being 1. Clearly, we have $\mu = \mathrm{Exp}[(X_1 + \cdots + X_t)/t]$. Thus, the formula we would use for estimating μ_t is $\mu_t = (X_1 + \cdots + X_t)/t$. (In other words, the function F is defined by $F(X_1, ..., X_t) \stackrel{\mathrm{def}}{=} (X_1 + \cdots + X_t)/t$ here.)

We show that, in this case, the generic estimator leads to a procedure for (ϵ, δ)-approximation that runs in time $\tilde{O}((1/(\epsilon^2\mu)) \cdot \ln(1/\delta))$. (In this paper, we use $\tilde{O}(f)$ to denote $O(f \log f)$.) We also provide a similar lower bound. Recall that the log factor can be reduced to log log by the trick discussed above, but we omit its use here for simplicity.

Both for applying our basic estimator algorithm and for deriving a lower bound, we use bounds provided by the Central Limit Theorem (see, e.g., [9]). Here we recall some basic facts about the Central Limit Theorem. Consider any sufficiently large t, and let X_1, X_2, ..., X_t be a series of i.i.d. random variables that take value either 1 or 0 with probability μ and $1-\mu$ respectively. Define $Y_t = \sum_{i=1}^{t} X_i$. Then the expectation and variance of Y_t are $\rho_t = t\mu$ and $\sigma_t = \sqrt{t\mu(1-\mu)}$ respectively. Now the Central Limit Theorem for sums of Bernoulli variables implies the following approximation (if t is large enough).

$$\Pr\left[\ \left|\frac{Y_t-\rho_t}{\sigma_t}\right| \ \le\ x\ \right] \ \approx\ 1-2(1-\Phi(x)), \tag{4}$$

where $\Phi(x) = \int_{-\infty}^{-x} exp(-z^2)\,dz$. In our analysis, we will assume that this approximation holds. (We remark that the Chernoff bound provides a very similar upper bound, which is in fact provable for all t.)

We use the following upper and lower approximations of $1-\Phi(x)$ [9].

$$\frac{1}{\sqrt{2\pi}}\frac{\exp(-x^2/2)}{x}\left(1-\frac{1}{x^2}\right)\ <\ 1-\Phi(x)\ <\ \frac{1}{\sqrt{2\pi}}\frac{\exp(-x^2/2)}{x}. \tag{5}$$

For $x \ge 1.1$, the $1-1/x^2$ factor can be replaced with 0.17.

4.1 An Upper Bound

With the bound given by the Central Limit Theorem, we follow the steps in our "recipe" to instantiate the generic algorithm for frequency estimation.

Step 1. We use the bound derived (or more specifically the function $B(\alpha,\mu,t)$) from the righthand inequality of (5). For defining our $B(\alpha,\mu,t)$, we first obtain an appropriate x so that the following relation holds.

$$[\,(1-\alpha)\mu\ \le\ \mu_t\ \le\ (1+\alpha)\mu\,]\ \iff\ \left[\ \left|\frac{Y_t-\rho_t}{\sigma_t}\right| \le x\ \right].$$

Recall the following definitions:

$$\begin{array}{lllllll} \mu_t &=& (\sum_{i=1}^{t} X_i)/t, & & Y_t &=& \sum_{i=1}^{t} X_i \ \ (\text{hence, } Y_t = t\mu_t), \\ \rho_t &=& t\mu, & \text{and} & \sigma_t &=& \sqrt{t\mu(1-\mu)}. \end{array}$$

Thus, by using the fact that $(1-\alpha)\mu \le \mu_t \le (1+\alpha)\mu$ if and only if $|\mu_t-\mu| \le \alpha\mu$, the above relation can be restated as

$$[\,|\mu_t-\mu|\ \le\ \alpha\mu\,]\ \iff\ \left[\ |\mu_t-\mu|\ \le\ x\sqrt{\frac{\mu(1-\mu)}{t}}\ \right].$$

Hence, this relation holds by letting $x = \alpha\sqrt{t\mu/(1-\mu)}$. Then, since we assume the approximation (4), we have

$$\Pr[\,(1-\alpha)\mu \;\le\; \mu_t \;\le\; (1+\alpha)\mu\,] \;\approx\; 1 - 2(1-\Phi(\alpha\sqrt{t\mu/(1-\mu)})). \tag{6}$$

Furthermore, using the bound in (5) and assuming that $x \ge 1$, we have

$$\begin{aligned}
\Pr[\,(1-\alpha)\mu \;\le\; \mu_t \;\le\; (1+\alpha)\mu\,] \;&>\; 1 - \frac{2}{\sqrt{2\pi}}\frac{\exp(-\alpha^2 t\mu/2(1-\mu))}{x} \\
&>\; 1 - \frac{2}{\sqrt{2\pi}}\exp(-\alpha^2 t\mu/2).
\end{aligned}$$

Thus, $B(\alpha,\mu,t) \stackrel{\rm def}{=} \sqrt{2/\pi}\exp(-\alpha^2 t\mu/2)$ satisfies our purpose.

Step 2. Now define α_t so that the following relation holds.

$$B(\alpha_t,\mu,t) \;\le\; \frac{\delta}{t(t+1)}.$$

For this, it is enough to define

$$\alpha_t \stackrel{\rm def}{=} c\sqrt{\frac{1}{t\mu}\left(\ln\frac{t(t+1)}{\delta} + d\right)},$$

for some $c = 2$ and $d = \ln(\sqrt{\pi/2}) = 0.225\cdots$. (For simplifying our later discussion, we ignore d by using a slightly larger constant for c.)

Step 3. Assume here that $\mu_t \le (1+\alpha_t)\mu$. Then we have

$$\alpha_t \;=\; c\sqrt{\frac{1}{t\mu}\ln\frac{t(t+1)}{\delta}} \;\le\; c\sqrt{\frac{1}{t\frac{\mu_t}{(1+\alpha_t)}}\ln\frac{t(t+1)}{\delta}},$$

or

$$\frac{\alpha_t}{\sqrt{1+\alpha_t}} \;\le\; c\sqrt{\frac{1}{t\mu_t}\ln\frac{t(t+1)}{\delta}}.$$

Since $\alpha_t < 1$ in all cases of interest, we have $\sqrt{1+\alpha_t} < 1.41\cdots$, and we can ignore the $\sqrt{1+\alpha_t}$ factor by taking a slightly larger constant for c. Then, define ϵ_t to be the right-hand side of this inequality and we have $\alpha_t \le \epsilon_t$.

4.2 A Lower Bound

Here again we assume the approximation (4) due to the Central Limit Theorem. Then we prove that if we use our generic algorithm for frequency estimation, it is impossible to bound ϵ_t by any function smaller than $c_1\sqrt{(1/(t\mu_t))\cdot\ln(1/\delta)}$ for some constant c_1. That is, our definition of $\epsilon_t = \mathcal{O}(\sqrt{(1/(t\mu_t))\cdot\ln(t^2/\delta)})$ obtained above achieves almost optimal convergence speed (ignoring the difference in the log factor). (We note here that a more general result, including this one as a special case, was proved in [2] with a more complicated technique.)

Theorem 1. For frequency estimation, suppose that we use the generic algorithm with the formula $\mu_t = (\sum_{i=1}^t X_i)/t$. Then there is some constant c_1 such that for any $\delta \le 0.27$, any μ, and any t, it must hold that either $\epsilon_t > 3/4$ or

$$\epsilon_t \;>\; c_1\sqrt{\frac{1}{t\mu_t}\ln\frac{1}{\delta}}$$

in order to guarantee that μ_t is an (ϵ_t,δ)-approximation of μ.

Remark. 1. This lower bound result is for our generic algorithm, and it does not deny all possibility of a better algorithm for frequency estimation. But, on the other hand, our generic algorithm is quite general, and in fact, this bound applies so long as the estimation μ_t is calculated as the observed frequency.
2. Note that we discuss the error probability at the tth step. Since it may be possible that errors occurs in earlier steps, the total error probability is larger than our estimation. We might be able to get a better lower bound if we also considered this point.
3. In the following, though very rough estimation, we prove this bound with $c_1 = 1/4$.
Proof. Consider any algorithm obtained from the generic one for frequency estimation. Suppose that, for a given $\delta \le 0.27$, it outputs (ϵ_t, μ_t) at the tth step. We assume that $\epsilon_t \le 3/4$ (since otherwise, the theorem trivially holds). Furthermore, we assume that μ_t is an (ϵ_t,δ)-approximation of μ, or more precisely, the following holds.

$$\delta \;\ge\; \Pr[\,\neg[\,(1-\epsilon_t)\mu \;\le\; \mu_t \;\le\; (1+\epsilon_t)\mu\,]\,].$$

From the approximation (6), this implies that

$$\delta \;\ge\; 2(1-\Phi(\epsilon_t\sqrt{t\mu/(1-\mu)})).$$

Let $x_t = \epsilon_t\sqrt{t\mu/(1-\mu)}$.

First we note that $x_t \ge 1.1$. Because otherwise $2(1-\Phi(x_t)) > 2(1-\Phi(1.1)) = 0.27133\ldots > 0.27$, contradicting our assumption that $\delta \le 0.27$. Now since $x_t \ge 1.1$, by using the left-side inequality of (6), we have

$$\delta \;\geq\; 2(1-\Phi(x_t)) \;\geq\; \frac{1}{\sqrt{2\pi}}\frac{\exp(-x_t^2/2)}{x_t}\left(1-\frac{1}{x_t^2}\right) \;\geq\; \frac{1}{0.17\sqrt{2\pi}}\frac{\exp(-x_t^2/2)}{x_t}.$$

Hence, for some constant d $(= 1/(0.17\sqrt{2\pi} = 2.34\ldots)$, we have

$$\frac{d}{\delta} \;\leq\; x_t \cdot \exp(x_t^2/2).$$

Since we may assume that $d/\delta > 8$, it is then easy to see that x_t must satisfy the following bound.

$$x_t \;>\; 2\sqrt{\ln\frac{d}{\delta} - 2\ln\ln\frac{d}{\delta}} \;>\; \frac{1}{2}\sqrt{\ln\frac{1}{\delta}}.$$

Hence,

$$\epsilon_t \;=\; \frac{x_t}{\sqrt{t\mu(1-\mu)}} \;>\; \frac{1}{2}\sqrt{\frac{1}{t\mu(1-\mu)}\ln\frac{1}{\delta}} \;>\; \frac{1}{2}\sqrt{\frac{1}{t\mu}\ln\frac{1}{\delta}}$$

Here we may assume that $\mu_t \geq (1-\epsilon_t)\mu$ holds. (Otherwise, μ_t clearly does not satisfy the approximation condition.) Then since $\mu \leq \mu_t/(1-\epsilon_t)$, by using the assumption that $\epsilon_t \leq 3/4$, we have

$$\epsilon_t \;>\; \frac{1}{2}\sqrt{\frac{1}{t\mu}\ln\frac{1}{\delta}} \;>\; \frac{1}{2}\sqrt{\frac{1-\epsilon_t}{t\mu_t}\ln\frac{1}{\delta}} \;>\; \frac{1}{4}\sqrt{\frac{1}{t\mu_t}\ln\frac{1}{\delta}}.$$

This completes the proof. $\square$

4.3 Efficiency

For given ϵ and δ, let us estimate the time sufficient/necessary for obtaining an (ϵ, δ)-approximation of frequency μ by using our generic algorithm.

First consider the algorithm we instantiated in Section 4.1 and derive an upper bound. From the definition of ϵ_t obtained in Section 4.1 and under the assumption that $\mu_t \geq (1-\epsilon_t)\mu$, it is easy to show that $\epsilon_t \leq \epsilon$ holds when

$$\frac{t}{\ln(t^2/\delta)} \;\approx\; \frac{t}{\ln(t(t+1)/\delta)} \;\geq\; \frac{c^2}{\epsilon^2\mu}.$$

We observe that $x \geq 2z\ln(zy) \geq z\ln(z^2y)$ implies $x/\ln(x^2y) \gtrsim z$; using this approximation, we can roughly bound the time t sufficient for having $\epsilon_t < \epsilon$ as

$$t \;\leq\; \frac{2c^2}{\epsilon^2\mu}\ln\frac{c^2}{\epsilon^2\mu\delta} \;=\; \frac{2c^2}{\epsilon^2\mu}\left(\ln\frac{1}{\delta} + \ln\frac{c^2}{\epsilon^2\mu}\right).$$

That is, with probability $\geq 1-\delta$, we have $\epsilon_t \leq \epsilon$ by the time t reaches the bound above.

Here it should be noted that with an analysis quite specific to frequency estimation, Lipton and Naughton [16] gave an algorithm that runs in time $O((1/(\epsilon^2\mu))\cdot\ln(1/\delta))$, i.e., a log factor faster than this algorithm. See [24] for some details and reasoning of this difference.

Next we show some lower bound from Theorem 1. Consider any instantiation of the generic algorithm, and let (ϵ_t, μ_t) be its output at the tth step. The theorem claims that, for any step t, it must hold that $\epsilon_t \geq c_1(\sqrt{(1/(t\mu_t))\cdot\ln(1/\delta)})$ in order to guarantee that μ_t is an (ϵ_t,δ)-approximation of μ. Then clearly, to obtain an (ϵ,δ)-approximation of μ, it must hold that

$$\epsilon \;\geq\; \epsilon_t \;\geq\; c_1\sqrt{\frac{1}{t\mu_t}\ln\frac{1}{\delta}} \;\geq\; c_1\sqrt{\frac{1}{t(1+\epsilon)\mu}\ln\frac{1}{\delta}}.$$

From this and using the trivial bound $\epsilon < 1$, we have the following lower bound.

$$t \;\geq\; \frac{c_1^2}{2\epsilon^2\mu}\ln\frac{1}{\delta}.$$

5 Case Study (2): Advantage Estimation

In advantage estimation, i.e., Example 2 stated in Section 2, an algorithm has access to a series of i.i.d. variables X_1, X_2, ... such that each X_i takes only values 0 or 1 and $\mathrm{Exp}[X_i] = p$. Here we assume that $p > 1/2$. Our goal is to estimate the advantage $\mu = p - 1/2$. The formula we use in the generic algorithm is $\mu_t = (X_1 + \cdots + X_t)/t - 1/2$.

5.1 An Upper Bound

Step 1. Observe that we cannot apply Chernoff or (simple) Hoeffding bounds directly because μ is not the average of a sum of 0/1 variables. On the other hand, defining $p_t = (\sum_{i=1}^{t} X_i)/t$, we have

$$(1-\alpha)\mu \;\leq\; \mu_t \;\leq\; (1+\alpha)\mu \quad\Longleftrightarrow\quad p-\alpha\mu \;\leq\; p_t \;\leq\; p+\alpha\mu.$$

Thus, by applying Hoeffding bound now, we have

$$\Pr[\,(1-\alpha)\mu \;\leq\; \mu_t \;\leq\; (1+\alpha)\mu\,] \;=\; \Pr[\,p-\alpha\mu \;\leq\; p_t \;\leq\; p+\alpha\mu\,] \;\geq\; 1 - 2\exp(-2\alpha^2\mu^2 t).$$

Hence we define $B(\alpha,\mu,t) \stackrel{\text{def}}{=} 2\exp(-2\alpha^2\mu^2 t)$.

Notice here that the argument from now on works for any other μ for which a similar Hoeffding bounding can be applied. This is the case considered in [6].

Step 2. To have $B(\alpha_t, \mu, t)$ $(= 2\exp(-2\alpha_t^2\mu^2 t)) = \delta/t(t+1)$, we define

$$\alpha_t \stackrel{\text{def}}{=} \sqrt{\frac{1}{2\mu^2 t}\ln\frac{t(t+1)}{\delta}}.$$

Step 3. Assuming that $\mu_t \le (1+\alpha_t)\mu$, we have

$$\alpha_t \;\le\; \sqrt{\frac{1}{2\left(\frac{\mu_t}{(1+\alpha_t)}\right)^2 t}\ln\frac{t(t+1)}{\delta}},$$

hence,

$$\frac{\alpha_t}{(1+\alpha_t)} \;\le\; \sqrt{\frac{1}{2\mu_t^2 t}\ln\frac{t(t+1)}{\delta}}.$$

Now define β_t to be the right-hand side of this inequality. Then by simple calculations, we have

$$\alpha_t/(1+\alpha_t) \;\le\; \beta_t \quad\Longleftrightarrow\quad \alpha_t \;\le\; \beta_t/(1-\beta_t).$$

Thus, we define $\epsilon_t \stackrel{\text{def}}{=} \beta_t/(1-\beta_t)$. This gives us an instance of the generic algorithm.

5.2 A Lower Bound

Theorem 2. For advantage estimation, suppose that we use the generic algorithm with the formula $\mu_t = (\sum_{i=1}^{t} X_i)/t - 1/2$. Then there is a constant c_2 such that for any $\delta \le 0.27$, any μ, and any t, it must hold that either $\epsilon_t > 1/4$ or

$$\epsilon_t \;>\; c_1\sqrt{\frac{1}{t\mu_t^2}\ln\frac{1}{\delta}}$$

in order to guarantee that μ_t is an (ϵ_t, δ)-approximation of μ.

Proof. We work by contradiction. Assume that the theorem is false, i.e., some instantiation of the generic algorithm for advantage estimation does better than stated in the theorem. Then we will show that some instantiation of the generic algorithm for frequency estimation contradicts the lower bound in Theorem 1.

In order to distinguish the pairs (μ_t, ϵ_t) produced by the two instances of the algorithm, we will use symbol μ to denote an advantage, μ_t and ϵ_t for the output of the advantage estimator, p for a frequency, and p_t and θ_t for the output of the frequency estimator.

Assume that the body of the advantage estimator computes μ_t and ϵ_t as:

compute $\mu_t := (\sum_{i=1}^t X_i) - 1/2$;
compute $\epsilon_t := f(S_t)$;

for some function f; then instantiate the frequency estimator algorithm with the assignments:

compute $p_t := (\sum_{i=1}^t X_i)$;
compute $\theta_t :=$ if $f(S_t) > 1/4$ then 1 else $4 \cdot (p_t - 1/2) \cdot f(S_t)$;

Observe that, for any fixed sequence of random examples, we have $p_t = \mu_t + 1/2$, and either $\theta = 1$ (if $\epsilon_t > 1/4$), or $\theta_t = 4\mu_t\epsilon_t$ (if $\epsilon_t \leq 1/4$).

We show first that, with probability $1 - \delta$, p_t is a (θ_t, δ)-approximation of p. Indeed, by assumption μ_t is a (ϵ_t, δ)-approximation of μ, i.e., with probability $1 - \delta$,

$$(1 - \epsilon_t)\mu \leq \mu_t \leq (1 + \epsilon_t)\mu$$

As noted in the start of Section 5.1, this is equivalent to

$$p - \epsilon_t\mu \leq p_t \leq p + \epsilon_t\mu.$$

Suppose first $\epsilon_t > 1/4$. Then $\theta_t = 1$ and $(1-\theta_t)p \leq p_t \leq (1+\theta)p$ holds because we are assuming $p > 1/2$ for this proof. Otherwise, $\epsilon_t \leq 1/4$; then $\theta = 4\epsilon_t\mu_t$ and also $\mu_t \geq (1 - \epsilon_t)\mu > \mu/2$. Therefore, applying the definition of θ_t and $p > 1/2$,

$$(1 - \theta_t)p = p - \theta_t p = p - 4\mu_t\epsilon_t \cdot p \leq p - 4(\mu/2)\epsilon_t \cdot (1/2) = p - \mu\epsilon_t \leq p_t$$

and, similarly, $(1 + \theta_t)p \geq p_t$.

On the other hand, assume that for constant $c_2 = c_1/4$ and some $\delta \leq 0.27$, μ, and t, we have both $\epsilon_t \leq 1/4$ and $\epsilon_t \leq c_2\sqrt{(1/(t\mu_t^2)) \cdot \ln(1/\delta)}$. Then, $\theta_t = 4\mu_t\epsilon_t \leq \mu_t < 3/4$. Also, $\theta_t = 4\mu_t\epsilon_t \leq 4c_2\sqrt{(1/t) \cdot \ln(1/\delta)} \leq 4c_2\sqrt{(1/(tp_t)) \cdot \ln(1/\delta)}$. This contradicts Theorem 1. □

5.3 Efficiency

Proceeding as in Section 4.3, we can show that the time t sufficient/necessary to get an (ϵ, δ)-approximation of advantage μ by our generic algorithm is

$$\frac{c_2^2}{2\epsilon^2\mu^2} \ln\frac{1}{\delta} \leq t \leq \frac{1}{2\epsilon^2\mu^2} \ln\frac{1}{\epsilon^2\mu^2\delta}.$$

This is about $1/\mu$ times bigger than the bound obtained for frequency estimation. Again, the log factor can be reduced to log log by the trick described in Section 3.2.

This upper bound was presented in [5], and is implied also by results in [6].

6 Applications

In this section we describe the application of our framework to various machine learning or KDD tasks and problems discussed in literature. The intent is to show how our framework helps in modular algorithm design: in a possibly complex KDD algorithm, it encapsulates sequential sampling neatly in a black box. Then, the rest of the argument concerning the algorithm usually becomes much more clear.

6.1 The Hypothesis Selection Problem

Many tasks in machine learning, statistics, or KDD can be seen as instances of the following task, that we call *hypothesis selection.* A prime example of this task would be selecting some predictor, chosen among a prespecified class of predictors, that (almost) minimizes the training error over the training dataset. Next we describe the task in general.

We have competing hypothesis $h^1, \ldots, h^H$, each h^i characterized by some "cost" quantity $\mu^{(i)}$. We want to choose an h^i with smallest $\mu^{(i)}$ up to ϵ, with probability $1-\delta$. That is, if $h^\star$ is the one with smallest $\mu^\star$, the chosen h^i must satisfy $\mu^{(i)} \le (1+\epsilon)\cdot\mu^\star$ with probability $1-\delta$.

In many situations, quantities $\mu^{(i)}$ can be estimated with controlled precision and confidence by means of random sampling. This would allow us to design an ad-hoc algorithm for the hypothesis selection problem.

On the other hand, in Figure 2 we give an algorithm that solves the hypothesis selection problem using, as black boxes, any estimator algorithms for the quantities $\mu^{(i)}$. Thus, the design effort can be focused on designing the most efficient estimators. Let us remark that the ad-hoc solution for the hypothesis selection problem has the same running time as the one we propose; no efficiency is lost by this "modular" approach.

Algorithm hypothesis_selection
  start parallel executions of H estimators for $\mu^{(1)}, \ldots, \mu^{(H)}$ with parameter δ/H;
  repeat
    get example X_t and pass it to the H estimators;
    obtain pairs $(\mu_t^{(i)}, \epsilon_t^{(i)})$ for each i;
    let k be such that $\mu_t^{(k)}$ is minimum;
    stopping_condition := $\left(\forall\, i \,:\, 1 \le i \le H \,:\, \dfrac{1+\epsilon_t^{(i)}}{1-\epsilon_t^{(k)}} \le 1+\epsilon\right)$;
  until stopping_condition;
  output h^k as winner and stop

Figure 2: An algorithm for the hypothesis selection problem

First, note that the algorithm terminates: since all sequences $\epsilon_t^{(i)}$ tend to 0, the ratio $(1+\epsilon_t^{(i)})/(1-\epsilon_t^{(k)})$ tends to 1; therefore, stopping_condition holds eventually.

We next argue that the algorithm solves the hypothesis selection problem with error probability at most δ. Observe that:

- with probability $1 - (\delta/H) \cdot H = 1 - \delta$, all estimators are correct at all times.
- for the winning h^k and any h^j we have $\mu_t^{(k)} \leq \mu_t^{(j)}$ at stop time t.
- then, with probability at least $1 - \delta$, if h^k is the winning hypothesis and t the stopping time, we have

$$\mu^{(k)} \leq \mu_t^{(k)} \cdot \frac{1}{1 - \epsilon_t^{(k)}} \leq \mu_t^\star \cdot \frac{1}{1 - \epsilon_t^{(k)}} \leq \mu^\star \cdot \frac{1 + \epsilon_t^\star}{1 - \epsilon_t^{(k)}} \leq \mu^\star \cdot (1 + \epsilon).$$

  In other words, h^k satisfies the condition required for a solution of the hypothesis selection problem.

Let us consider the efficiency of this algorithm. Consider first the number of iterations performed by the algorithm. We argue (not completely formally) that this number is determined essentially by the number of estimator steps required to obtain an (ϵ, δ)-estimation of $\mu^\star$; in particular, it depends only logarithmically, not linearly, on the number H of hypothesis.

Define $\epsilon^\star = \epsilon/(2 + \epsilon)$. Suppose that after some number of iterations t, it holds that $\epsilon_t^{(j)} \leq \epsilon^\star$ for all j. Then, some routine algebra shows that, for all i and k, it also holds $(1+\epsilon_t^{(i)})/(1-\epsilon_t^{(k)}) \leq 1 + \epsilon$, and therefore the stopping condition of the algorithm holds.

Let $T^\star$ be number of iterations required to obtain an $(\epsilon^\star, \delta/H)$-approximation of $\mu^\star$. Usually, one can prove that $T^\star$ becomes larger as $\mu^\star$ becomes smaller, i.e., approximation requires more steps for smaller values of $\mu^\star$. Then, for all fixed j, the probability that $\epsilon_t^{(j)} \geq \epsilon^\star$ is at most δ/H. Consequently, by time $T^\star$, the probability that there is *some* j such that $\epsilon_t^{(j)} \geq \epsilon^\star$ is less than δ, and so with probability $1 - \delta$ the stopping condition holds.

Note that typically $T^\star$ is a small constant times $\ln(H)$ times the time required to achieve an $(\epsilon^\star, \delta)$-estimation of $\mu^\star$. Hence, the dependence on H is logarithmic rather than linear.

On the other hand, the work performed at each iteration is proportional to H, because it involves simulating one step of each of H estimators. Later we will discuss Hoeffding races, an approach that tries to reduce this work, and therefore the running time of the algorithm.

6.2 The n-Best Hypothesis Selection Problem

Scheffer and Wrobel [21, 22] discuss this generalization of the Hypothesis Selection Problem. In the n-best hypothesis selection problem, we want to select n out of H competing hypothesis that are not substantially worse than the excluded hypothesis, with respect to their associated quantities μ. More precisely, if h^i is selected and h^j is not selected, it must happen that $\mu^{(i)} \leq \mu^{(j)} + \epsilon$ (recall that [21, 22] use additive approximations).

This problem can be solved probabilistically by an immediate extension of the algorithm in Figure 2. It is enough to change the stopping condition there so that it holds for the n hypothesis with largest $\mu_t^{(i)}$, and not only for the winner one. Details are omitted.

6.3 Hoeffding Races

Maron and Moore [18] and Moore and Lee [19] propose Hoeffding races as a way to speed up hypothesis selection. Recall that the number of iterations of the algorithm in Figure 2 depended logarithmically on the number of hypothesis H, but its running time depended linearly in H.

The idea in Hoeffding races is to detect soon that a hypothesis will not be the winner, and then discard it for the rest of the execution. Of course, the question is how to predict, as early as possible and with high confidence, that a hypothesis is not likely to win.

In [18, 19], a criterion based on Hoeffding inequality is derived when the "cost" of a hypothesis is just its training error on a set of examples. Carlos Domingo ([3], p.55) observed already Hoeffding races fit naturally in sequential sampling algorithms for hypothesis selection. We basically redo his argument in our framework of estimator algorithms, but we generalize it to notions of "hypothesis cost" other than training error.

Indeed, as in Section 6.1, suppose we want to select a hypothesis whose $\mu^{(k)}$ value is at most $(1+\epsilon)$ times larger than the smallest one $\mu^\star$. If, at any time t of the algorithm, there is an i such that

$$\mu_t^{(k)} < \mu_t^{(i)} \cdot \frac{1-\epsilon_t^{(k)}}{1+\epsilon_t^{(i)}},$$

discard hypothesis h^i since μ^i is not likely to be minimum, i.e., h^i is probably not $h^\star$.

This is because with probability $1-\delta$,

$$\mu^{(k)} \le \mu_t^{(k)} \cdot \frac{1}{1-\epsilon_t^{(k)}} < \left(\mu_t^{(i)} \cdot \frac{1-\epsilon_t^{(k)}}{1+\epsilon_t^{(i)}}\right) \cdot \frac{1}{1-\epsilon_t^{(k)}} = \mu_t^{(i)} \cdot \frac{1}{1+\epsilon_t^{(i)}} \le \mu^{(i)}.$$

A similar argument can be made for the other trick suggested by Maron and Moore [18]: when two hypothesis seem to have almost the same value, discard one of them. By similar calculations, one can estimate the number of steps necessary so that this conclusion can be safely reached (with some confidence $1-\delta$). However, from these calculations, it appears that the required number of steps is roughly the same as for reaching the stopping conclusions, i.e., to determine a winner hypothesis. Therefore, we expect that this second rule will only have any effect when the algorithm is near to stopping, and thus it will not save by far as much time as the first rule.

References

[1] H. Chernoff, A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations, *Annals of Mathematical Statistics* 23, pp.493–509, 1952.

[2] P. Dagum, R. Karp, M. Luby, and S. Ross, An optimal algorithm for monte carlo estimation, *SIAM J. Comput.* Vol. 29(5), pp.1484–1496, 2000.

[3] C. Domingo, *Concept Learning: Theory, Applications, and Related Problems.* Ph.D. thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 1999.

[4] C. Domingo and O. Watanabe, Scaling up a boosting-based learner via adaptive sampling, in *Proc. of Knowledge Discovery and Data Mining* (PAKDD'00), Lecture Notes in Artificial Intelligence 1805, Springer-Verlag, pp.317–328, 2000.

[5] C. Domingo, R. Gavaldà, and O. Watanabe, Practical algorithms for on-line selection, in *Proc. of the First Intl. Conference on Discovery Science*, Lecture Notes in Artificial Intelligence 1532, Springer-Verlag, pp.150–161, 1998.

[6] C. Domingo, R. Gavaldà, and O. Watanabe, Adaptive sampling methods for scaling up knowledge discovery algorithms, in *Proc. of the Second Intl. Conference on Discovery Science*, Lecture Notes in Artificial Intelligence, Springer-Verlag, pp.172–183, 1999. The final version will appear in *J. Knowledge Discovery and Data Mining* and is also available as research report C-136, Dept. of Math. and Computing Sciences, Tokyo Institute of Technology, from `www.is.titech.ac.jp/research/research-report/C/`.

[7] P. Domingos and G. Hulten, Mining high-speed data streams, in *Proc. 6th Intl. Conference on Knowledge Discovery in Databases*, ACM Press, pp.71–80, 2000.

[8] P. Domingos and G. Hulten, A general method for scaling up machine learning algorithms and its applications to clustering, in *Proc. 8th Intl. Conference on Machine Learning*, Morgan Kaufmann, pp.106–113, 2001.

[9] W. Feller, *An Introduction to Probability Theory and its Applications* (Third Edition), John Wiley & Sons, 1968.

[10] B.K. Ghosh, M. Mukhopadhyay, P.K. Sen, *Sequential Estimation*, Wiley, 1997.

[11] P. Haas and A. Swami, Sequential sampling, procedures for query size estimation, *IBM Research Report*, RJ 9101 (80915), 1992.

[12] W. Hoeffding, Probability inequalities for sums of bounded random variables, *Journal of the American Statistical Association* 58, pp.13–30, 1963.

[13] G.H. John and P. Langley, Static versus dynamic sampling for data mining, in *Proc. of the Second Intl. Conference on Knowledge Discovery and Data Mining*, AAAI/MIT Press, pp.367–370, 1996.

[14] J. Kivinen and H. Mannila, The power of sampling in knowledge discovery, in *Proc. of the 14th ACM SIGACT-SIGMOD-SIGACT Symposium on Principles of Database Systems* (PODS'94), ACM Press, pp.77–85, 1994.

[15] R.J. Lipton, J.F. Naughton, D.A. Schneider, and S. Seshadri, Efficient sampling strategies for relational database operations, *Theoretical Computer Science* 116, pp.195–226, 1993.

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

[17] J.F. Lynch, Analysis and application of adaptive sampling, in *Proc. of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (PODS'99), ACM Press, pp.260–267, 1999.

[18] O. Maron and A. Moore, Hoeffding races: accelerating model selection search for classification and function approximation, in *Advances in Neural Information Processing Systems*, Morgan Kaufmann, pp.59–66, 1994.

[19] A.W. Moore and M.S. Lee, Efficient algorithms for minimizing cross validation error, in *Proc. of the 11th Intl. Conference on Machine Learning*, Morgan Kauffman, pp.190–198, 1994.

[20] F.J. Provost, D. Jensen, and T. Oates, Efficient Progressive Sampling, in *Proc. of the 5th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, ACM Press, pp.23–32, 1999.

[21] T. Scheffer and S. Wrobel, A sequential sampling algorithm for a general class of utility criteria, in *Proc. of the 6th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, ACM Press, 2000, to appear.

[22] T. Scheffer and S. Wrobel, Finding the most interesting patterns in a database quickly by using sequential sampling, Technical Report, University of Magdeburg, School of Computer Science, january 2001.

[23] A. Wald, *Sequential Analysis*, John Wiley & Sons, 1947.

[24] O. Watanabe, Simple sampling techniques for discovery science, *IEICE Trans. Info. & Systems*, E83-D (1), pp.19–26, 2000.