

Simple Sampling Techniques for Discovery Science

Osamu Watanabe[†], *Member*

SUMMARY We explain three random sampling techniques that are simple but widely applicable for various problems involving huge data sets. The first technique is an immediate application of large deviation bounds. The second and the third ones are sequential sampling or adaptive sampling techniques. We fix one simple problem and explain these techniques by demonstrating algorithms for this problem and discussing their correctness and efficiency.

key words: random sampling, the Chernoff bound, the Hoeffding bound, the Central Limit Theorem, sequential sampling, adaptive sampling

1. Introduction

One of the important topics of *Discovery Science* is data mining or knowledge discovery that aims to obtain some sort of feature or law from a given data set (see, e.g., [1], [9]). In some of the data mining problems, we need to deal with a huge input data set, so huge that it is impractical even just going through all instances in the data set. One possible approach we can take to such largeness is random sampling. That is, we take random examples from the data set and do data mining on these examples. However, several researchers claim (see, e.g., [15]) that this approach is less recommendable due to the difficulty of determining appropriate *sample size* (i.e., the number of examples) needed. Here we explain simple but widely applicable sampling techniques for which we do not need to worry so much about sample size.

In statistics, various formulas have been developed to determine appropriate sample size for given accuracy and confidence parameters, and some of them have been used in computer science. In particular, so called concentration bounds or large deviation bounds like the Chernoff bound have been used commonly in theoretical computer science (see, e.g., [12]) as well as application areas such as data mining (see, e.g., [9]). While these bounds allow us to calculate “safe” sample size in many situations, it is usually the case that resulting sample size is too large compared with what might be really necessary. That is, we often overestimate sample size.

One crucial reason for this overestimating problem

is that these bounds require some parameters that are unknown in many situations. To use these bounds to calculate sample size, we need to know such parameters. A typical case is, as we will see below, that we need to know the very parameter that we want to estimate! In such situations, we need to calculate sample size by considering the worst case, which often gives overestimated sample size.

This problem may be avoidable if we perform sampling in an *on-line* or *sequential* fashion. That is, a sampling algorithm obtains examples sequentially one by one, and it determines from these examples whether it has already seen enough number of examples. Intuitively, from the examples seen so far, we can more or less obtain some knowledge on the input data set, and it may be possible to estimate the parameters required by the statistical bounds. Thus, we do not fix sample size in advance. Instead sample size is determined *adaptively*. Then we may be able to use more appropriate sample size for the current input data set.

With this motivation, adaptive sampling techniques have been proposed in computer science. (See *Remark 1* below for statistics.) Lipton et al [10], [11] proposed adaptive sampling algorithms for relational database. More recently, Domingo et al [3]–[5] proposed more general adaptive sampling algorithms for rule selection problems. This paper explain key ideas of these two types of adaptive sampling techniques.

In this paper, we fix one simple problem and explain sampling techniques. Let us specify this problem. We consider the following simple estimation problem. Let D be an input data set; here it is simply a set of instances. Let B be a Boolean function defined on instances in D . That is, for any $x \in D$, $B(x)$ takes either 0 or 1. Now our problem is to estimate the average value p_B of B on D ; in other words, the ratio of instances x in D such that $B(x) = 1$ holds. Also we consider the situation where we only need to get the average “approximately”. That is, it is enough to compute p_B within a certain error bound required in each context.

Clearly, the average p_B is obtained by counting the number of instances x in D for which $B(x) = 1$ holds. But we consider the situation where D is *huge* and it is impractical to go through all instances of D for computing p_B . A natural strategy that we can take in such

Manuscript received ?

Manuscript revised ?

[†]Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo 152-8552

a situation is random sampling. That is, we pick up some elements of D randomly and estimate the average value of B on these selected examples. Here we assume that we can easily pick up instances from D uniformly at random and independently. Due to the “randomness nature”, we cannot always obtain a desired answer, and we must be satisfied if our sampling algorithm yields a *good approximation* of p_B with *reasonable probability*. We discuss this type of estimation problem.

Remark 1. Statisticians also have studied sequential sampling. In fact, they made significant accomplishments on this topic during World War II [14], from which a research area on sequential sampling — sequential analysis — has been formed in statistics. However, their main goal has been to test statistical hypotheses, which is slightly different, though related, from what we want to do here. Also I personally feel some strategic difference between statistic approach and our approach in computer science. Statisticians have tried to reduce sample size even by, say, 100, and they have developed various specific techniques for this purpose. This is because the cost of obtaining each example is usually high. On the other hand, we are given enough number of examples anyway. Thus, we do not care so much about the ± 100 difference of sample size, while we want to reduce sample size by, say, $1/10$. Instead, we would like to have techniques that are applicable in many situations and that can be used “automatically”. The sampling techniques explained in this paper are of this type. For using these techniques, we only need to assume that it is possible to obtain each example uniformly at random and independently. Nevertheless, “sequential estimation” have been also studied as one topic of sequential analysis [16], and it may be the case that the techniques and their analysis explained here have been already reported in the literature.

Remark 2. In order to make explanation easy to understand, we will simplify some of the arguments in this paper. Firstly we consider the simple estimation problem defined above. Some of the techniques and arguments explained here are applicable to more general cases. For example, the Hoeffding bound, hence the technique explained in Section 4, can be used in the case that B is not a Boolean function. Secondly we time to time will ignore small difference and use approximate relations. For example, we ignore to take ceiling to make a real number integral. We ask the reader to refer the original papers for details. The algorithm explained in Section 3 is from the query size estimation algorithm of Lipton et al [10]. The algorithm explained in Section 4 is based on the adaptive rule selection algorithm of Domingo et al [4], [5].

Batch Sampling Algorithm

```

begin
  m ← 0;
  for n times do
    get x uniformly at random from D;
    m ← m + B(x);
  output m/n as an approximation of pB;
end.
```

Fig. 1 Batch Sampling

2. Batch Sampling and Statistical Bounds

The simplest random sampling algorithm for estimating p_B is to pick up some elements of D randomly and estimate the average value of B on these selected examples. This is our first sampling algorithm and we call it *Batch Sampling* (Figure 1). Even this simple algorithm suffices if we want to estimate p_B within some *absolute* error bound; that is, for the following approximation goal. (In the following, we use \widetilde{p}_B to denote the output of a sampling algorithm; thus, it is a random variable and the probability is taken w.r.t. this random variable.)

Approximation Goal 1: For given $\delta > 0$ and ϵ , $0 < \epsilon < 1$, we want to have

$$\Pr[|\widetilde{p}_B - p_B| \leq \epsilon] > 1 - \delta. \quad (1)$$

Batch Sampling of Figure 1 is incomplete until we specify the way to determine n , the number of iterations, or the number of examples, which we call *sample size*. Of course, to get an accurate estimation, the larger n is the better; on the other hand, from the efficiency, the smaller n is the better. We would like to achieve a given accuracy with as small sample size as possible.

To determine appropriate sample size, we can use several statistical bounds, upper bounds of the probability that a random variable deviates far from its expectation. Here we explain typical three bounds: the Chernoff bound [2] the Hoeffding bound [7], and the bound from the Central Limit Theorem.

For explaining these bounds, let us prepare some notations. Let X_1, \dots, X_n be independent trials, which are called *Bernoulli trials*, such that, for $1 \leq i \leq n$, we have $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1 - p$ for some p , $0 < p < 1$. Let X be a random variable defined by $X = \sum_{i=1}^n X_i$. Then its expectation $E[X] = np$; hence, the expected value of X/n is p . The above three bounds respectively give an upper bound of the probability that X/n differs from p , say, ϵ . Below we use $\exp(x)$ to denote e^x , where e is the base of the natural logarithm.

We state the above three bounds from the Chernoff bound.

Theorem 2.1 (The Chernoff Bound): For any ϵ , $0 < \epsilon < 1$, we have the following relations.

$$\Pr[X/n > (1 + \epsilon)p] \leq \exp(-pn\epsilon^2/3). \quad (2)$$

$$\Pr[X/n < (1 - \varepsilon)p] \leq \exp(-pn\varepsilon^2/2). \quad (3)$$

The proof of this bound is not so difficult, and the reader can find a good explanation of this bound including its proof in [12].

Note that the Chernoff bound is stated in terms of relative error, i.e., multiplicative difference from the expected value. On the other hand, the other two bounds are stated more naturally in terms of absolute error, i.e., additive difference. (In this paper, we distinguish relative and absolute error bounds by using symbols ε and ϵ for relative and additive error bounds respectively.)

Theorem 2.2 (The Hoeffding Bound): For any ϵ , $0 < \epsilon < 1$, we have the following relations.

$$\Pr[X/n > p + \epsilon] \leq \exp(-2n\epsilon^2). \quad (4)$$

$$\Pr[X/n < p - \epsilon] \leq \exp(-2n\epsilon^2). \quad (5)$$

The third bound is derived from the Central Limit Theorem, one of the basic theorems in probability theory. Let us first state the Central Limit Theorem. (The following theorem is a special case of the Central Limit Theorem; see, e.g., [6] for more explanation.)

Theorem 2.3 (The Central Limit Theorem): For any a , as n goes infinity, we have

$$\Pr \left[\frac{X - np}{\sqrt{np(1-p)}} \leq a \right] \rightarrow \Phi(a),$$

where $\Phi(a) = (1/\sqrt{2\pi}) \int_{-\infty}^a e^{-x^2/2} dx$ is the normal distribution function.

This theorem says that for any a , if n is large enough, then we may consider

$$\Pr \left[\frac{X - np}{\sqrt{np(1-p)}} \leq a \right] \approx \Phi(a),$$

which we say that *the central limit approximation applies* (for a) [10]. How large should n be? It is said in statistics that if n exceeds 30, then we may be able to assume that the central limit approximation applies for a wide range of a . Then we can use the following bounds. (These bounds follow immediately from the above theorem and the property that $\Phi(-a) = 1 - \Phi(a)$.)

Theorem 2.4: For any ϵ , $0 < \epsilon < 1$, if the central limit approximation applies, then we have the following relations.

$$\Pr[X/n > p + \epsilon] \approx 1 - \Phi \left(\frac{\epsilon\sqrt{n}}{\sqrt{p(1-p)}} \right). \quad (6)$$

$$\Pr[X/n < p - \epsilon] \approx 1 - \Phi \left(\frac{\epsilon\sqrt{n}}{\sqrt{p(1-p)}} \right). \quad (7)$$

Now by using these three bounds, we can calculate “safe” sample size, the number n of examples, so that Batch Sampling satisfies our approximation goal.

Theorem 2.5: For any $\delta > 0$ and ϵ , $0 < \epsilon < 1$, if Batch Sampling uses sample size n satisfying one of the following inequalities, then it satisfies (1). (In order to use the third inequality, we need to assume the central limit approximation.)

$$n > \frac{3p_B}{\epsilon^2} \ln \left(\frac{2}{\delta} \right). \quad (8)$$

$$n > \frac{1}{2\epsilon^2} \ln \left(\frac{2}{\delta} \right). \quad (9)$$

$$n > \frac{p_B(1-p_B)}{\epsilon^2} \left(\Phi^{-1} \left(1 - \frac{\delta}{2} \right) \right)^2. \quad (10)$$

The derivations of these inequalities are easy. Here, as one example, we give a proof for (8).

Proof. Consider an execution of Batch Sampling, and let x_1, \dots, x_n be the instances used in the execution. We can regard $B(x_1), \dots, B(x_n)$ as the Bernoulli trials X_1, \dots, X_n with $\Pr[X_i = 1] = p_B$; then X/n corresponds to the output of the algorithm. Thus, in order to estimate p_B within the ϵ additive error bound, it suffices to have $|X/n - p_B| \leq \epsilon$. Hence, it is enough to show $\Pr[|X/n - p_B| \leq \epsilon] > 1 - \delta$; in other words, $\Pr[X/n > p_B + \epsilon] + \Pr[X/n < p_B - \epsilon] < \delta$. Note that $\Pr[X/n > p_B + \epsilon] = \Pr[X/n > (1 + \epsilon/p_B)p_B]$ and that $\Pr[X/n < p_B - \epsilon] = \Pr[X/n < (1 - \epsilon/p_B)p_B]$. Then by the Chernoff bound, the former and the latter probabilities are bounded respectively by $\exp(-p_B n(\epsilon/p_B)^2/3) = \exp(-n\epsilon^2/3p_B)$ and by $\exp(-p_B n(\epsilon/p_B)^2/2) = \exp(-n\epsilon^2/2p_B)$. (Note also that $\exp(-n\epsilon^2/2p_B)$ is smaller than $\exp(-n\epsilon^2/3p_B)$.)

Now if n satisfies the inequality (8), i.e., $n > 3p_B/\epsilon^2 \ln(2/\delta)$, then we have $\exp(-n\epsilon^2/3p_B) < \delta/2$. Therefore, $\Pr[X/n > p_B + \epsilon] + \Pr[X/n < p_B - \epsilon]$ is less than δ as we desired. \square

Let us examine the above sample size bounds (8) \sim (10). Although (8) and (10) depend on p_B , we can simply assume that $p_B = 1$ in (8) and $p_B = 1/2$ in (10), which still give us safe bounds. Thus, roughly speaking, all bounds are proportional to $1/\epsilon^2$ (when δ is fixed to some constant).

Next compare them in more detail. The difference between (8) and (9) is clear. We had better use (8) if $p_B < 1/6$, and (9) otherwise. Notice, on the other hand, that if p_B is close to 1, we do not need large sample size by using any bound; that is, the difference is not so essential when p_B is close to 1. Thus, considering small p_B , we compare (8) and (10). Then the difference is $3 \ln(2/\delta)$ and $(\Phi^{-1}(1 - \delta/2))^2$. Table 2 illustrates the difference for some typical values of δ . (The table in [6] is used for computing Φ^{-1} .) From this table, we see that the sample size of (10) is twice to three times

δ	(a)	(b)
0.1	8.99	2.72
0.05	11.07	3.84
0.01	15.89	6.66
0.005	17.97	7.90

Fig. 2 Comparison of (a) $3\ln(2/\delta)$ and (b) $(\Phi^{-1}(1 - \delta/2))^2$

better than that of (8). On the other hand, the sample size bound (8) (or (9)) is much easier to use than (10) when we cannot assume that δ is constant.

3. Adaptive Sampling

While the simplest sampling algorithm — Batch Sampling — works to estimate p_B within an *absolute* error bound, it is often the case that we need to consider *relative* error instead of absolute error. That is, the following approximation goal is required.

Approximation Goal 2: For given $\delta > 0$ and ε , $0 < \varepsilon < 1$, we want to have

$$\Pr[|\widetilde{p}_B - p_B| \leq \varepsilon p_B] > 1 - \delta. \tag{11}$$

Let us first try with Batch Sampling. Since the Chernoff bound is stated in terms of relative error, we can easily use it to obtain the following sample size bound. (We can get more or less similar sample size bounds by using the other probability bounds.)

Theorem 3.1: For any $\delta > 0$ and ε , $0 < \varepsilon < 1$, if Batch Sampling uses sample size n satisfying the following inequality, then it satisfies (11).

$$n > \frac{3}{p_B \varepsilon^2} \ln \left(\frac{2}{\delta} \right).$$

The above inequality is quite similar to (8). Nevertheless, it is inconvenient in practice. Recall that we do not know p_B ; in fact, this is what we want to estimate. On the other hand, we cannot determine n without knowing p_B . In some cases, we may be able to guess some appropriate value p for p_B , but to be safe, we need to use p such that $p < p_B$. Then if we underestimate p_B and use much smaller p , we have to use unnecessarily large sample size. (Cf. In the case of (8), we can safely assume that $p_B = 1$, which may be again too large if p_B is small. But usually this overestimation is not so serious as the above case.)

One way to avoid this problem is to perform presampling. By running our sampling algorithm, e.g., Batch Sampling, with small sample size and obtain some “rough” estimate of p_B . Although it may not be a good approximation of p_B , we can use it to determine appropriate sample size for main sampling. This is the strategy often suggested in statistics texts, and in fact, this idea leads to our “adaptive sampling” techniques.

Note first that we do not have to separate presampling and main sampling. On the course of sampling,

Adaptive Sampling Algorithm

```

begin
   $m \leftarrow 0$ ;  $n \leftarrow 0$ ;
  while  $m < A$  do
    get  $x$  uniformly at random from  $D$ ;
     $m \leftarrow m + B(x)$ ;  $n \leftarrow n + 1$ ;
    output  $m/n$  as an approximation of  $p_B$ ;
  end.

```

Fig. 3 Adaptive Sampling

we can improve our knowledge on p_B . Why don't we use it! This is the motivation or the intuitive idea of adaptive sampling. That is, we do not specify sample size in advance. Instead, an adaptive sampling algorithm calculates it, or it determines whether it has already seen enough number of examples by using the current estimation of p_B .

Lipton et al [10], [11] realized this intuitive idea and proposed adaptive sampling algorithms for query size estimation and related problems for relational database. Estimating query sizes is essentially the same as our problem of estimating p_B . Thus, we explain the key idea of their algorithms with our problem.

Figure 3 is the outline of the adaptive sampling algorithm of [10]. Though it is simplified for the problem of estimating p_B and one stopping condition for the skewed case is removed, the adaptive sampling part is essentially the same. As we can see, the structure of the algorithm is simple. It runs until it sees more than A examples x with $B(x) = 1$. (This outline, i.e., performing random sampling until the number of “positive observations” exceeds a certain limit, has been studied in depth in statistics [16]. It may be possible that the theorem explained below has been already obtained in statistics literature.)

We have to specify the way to determine A . Here for simplyfing our explanation, we use the Chernoff bound to compute A . (In [10] the bound from the Central Limit Theorem is used, which gives better sample size.)

Theorem 3.2: For any $\delta > 0$ and ε , $0 < \varepsilon < 1$, if Adaptive Sampling uses the following A , then it satisfies (11).

$$A > \frac{3(1 + \varepsilon)}{\varepsilon^2} \ln \left(\frac{2}{\delta} \right).$$

Remark. Note that A does not depend on p_B . Thus, we can execute Adaptive Sampling without knowing p_B .

In the following discussion, let t denote the number of execution of the while-iterations until Adaptive Sampling halts. In other words, the algorithm has seen t examples and then the while-condition breaks. (In the following, we simply call this situation “the algorithm halts at the t th step”.) Note that t is a random variable that varies depending on the examples drawn from D .

Since the while-condition breaks at the t th step, it holds that $A \leq \widetilde{m}_t$. On the other hand, $\widetilde{m}_t < A + 1$ holds because the while-condition holds before the t th step. Hence we have $A/t \leq \widetilde{p}_t < (A + 1)/t$. Here in order to simplify our discussion, we consider that $\widetilde{p}_t \approx A/t$. In fact, we will see below that t is larger than $1/(\varepsilon^2 p_B)$ with high probability; thus, the difference $(A + 1)/t - A/t (= 1/t)$ is negligible compared with the error bound εp_B . Now assuming $\widetilde{p}_t \approx A/t$, it is easy to see that \widetilde{p}_t is within the desired range $[(1 - \varepsilon)p_B, (1 + \varepsilon)p_B]$ (i.e., $|\widetilde{p}_t - p_B| \leq \varepsilon p_B$) if and only if

$$\frac{A}{(1 + \varepsilon)p_B} \leq t \leq \frac{A}{(1 - \varepsilon)p_B}.$$

holds for t . Therefore, the theorem follows from the following two lemmas.

Lemma 3.3:

$$\Pr \left[t < \frac{A}{(1 + \varepsilon)p_B} \right] < \frac{\delta}{2}.$$

Remark. Note that t is a random variable. The probability is taken w.r.t. this random variable.

Proof. We would like to estimate the above probability, and for this purpose, we want to regard the B value of chosen examples as the Bernoulli trials and to use the statistical bounds of the previous section. There is, however, one technical problem. These statistical bounds are valid for fixed number of trials, i.e., examples in this case. On the other hand, the number of examples t itself is a random variable. Here we can get around this problem by arguing in the following way.

Let $t_0 = A/((1 + \varepsilon)p_B)$. Then our goal is to show that the algorithm halts within t_0 steps with high probability. Now we modify our algorithm so that it *always* sees exactly t_0 examples. That is, this new algorithm just ignores the while-condition and repeats the while-iteration exactly t_0 times. Consider the situation that the original algorithm does halt at the t th step for some $t < t_0$. Then we have $\widetilde{m}_t \geq A$ at the t th step, where \widetilde{m}_t denotes the value of m at the t th step. Though the algorithm stops here, if we continued the while-iteration after the t th step, we would clearly have $\widetilde{m}_{t_0} \geq A$ at the t_0 th step. From this observation, we have

$$\begin{aligned} & \Pr[\widetilde{m}_t \geq A \text{ for some } t < t_0] \\ & \leq \Pr[\widetilde{m}_{t_0} \geq A \text{ in the modified algorithm}]. \end{aligned}$$

On the other hand, the modified algorithm always sees t_0 examples; that is, it is Batch Sampling. Thus, we can use the statistical bounds to analyze the right-hand side probability.

Let x_1, \dots, x_{t_0} denote t_0 examples that the algorithm sees. Here we regard their B values $B(x_1), \dots, B(x_{t_0})$ as the Bernoulli trials X_1, \dots, X_{t_0} with $\Pr[X_i = 1] = p_B$. Then $\widetilde{m}_{t_0} = \sum_{i=1}^{t_0} X_i$. Hence, we have

$$\begin{aligned} & \Pr[\widetilde{m}_{t_0} \geq A \text{ in the modified algorithm}] \\ & = \Pr[\sum_{i=1}^{t_0} X_i \geq A] = \Pr[\sum_{i=1}^{t_0} X_i/t_0 \geq A/t_0] \\ & = \Pr[\sum_{i=1}^{t_0} X_i/t_0 \geq \frac{A}{A/((1 + \varepsilon)p_B)}] \\ & = \Pr[\sum_{i=1}^{t_0} X_i/t_0 \geq (1 + \varepsilon)p_B] \\ & \leq \exp(-p_B t_0 \varepsilon^2/3) \quad (\text{by (2)}) \\ & = \exp\left(\frac{\varepsilon^2 A}{3(1 + \varepsilon)}\right) < \frac{\delta}{2}. \end{aligned}$$

Thus, the desired bound is proved. The reason that we could argue by considering only the t_0 th step is because the stopping condition " $m \geq A$ " is *monotonic*. \square

Lemma 3.4:

$$\Pr \left[t > \frac{A}{(1 - \varepsilon)p_B} \right] < \frac{\delta}{2}.$$

Proof. Let $t_1 = A/((1 - \varepsilon)p_B)$. We want to bound the probability that the algorithm does not halt after the t_1 th step. Note that this event implies that $\widetilde{m}_{t_1} < A$. Thus, it suffices to bound $\Pr[\widetilde{m}_{t_1} < A]$ as follows. (Here again we consider the modified algorithm that sees exactly t_1 examples.)

$$\begin{aligned} & \Pr[\widetilde{m}_{t_1} < A] \\ & = \Pr[\sum_{i=1}^{t_1} X_i/t_1 < A/t_1] \\ & = \Pr[\sum_{i=1}^{t_1} X_i/t_1 < (1 - \varepsilon)p_B] \\ & \leq \exp(-p_B t_1 \varepsilon^2/2) \quad (\text{by (3)}) \\ & = \exp\left(\frac{\varepsilon^2 A}{2(1 - \varepsilon)}\right) < \frac{\delta}{2}. \end{aligned}$$

\square

The above lemma also shows that the algorithm needs no more than $A/((1 - \varepsilon)p_B)$ examples with high probability.

Corollary 3.5: Use Adaptive Sampling by setting A with the smallest integer satisfying the condition of Theorem 3.2. Then with probability $> 1 - \delta/2$, we have

$$\text{sample size} \leq \frac{3(1 + \varepsilon)}{(1 - \varepsilon)\varepsilon^2 p_B} \ln \left(\frac{2}{\delta} \right).$$

Compare this sample size bound with the one from Theorem 3.1. We see the difference is small. Note that the sample size bound from Theorem 3.1 is optimal; it is the bound computable if we *know* p_B in advance. Adaptive Sampling achieves almost the same sample size bound even if it can be used without knowing p_B at all.

4. Adaptive Sampling: Nonmonotonic Case

We have seen two ways for estimating p_B within either an absolute or a relative error bound. But in some applications, we may need the other closeness conditions, or in more general, we might want to estimate not p_B but some other value computed from p_B .

For example, consider the problem of determining whether p_B is greater or smaller than $1/2$. Clearly, the problem gets harder when p_B is closer to $1/2$, and in fact, no answer exists if $p_B = 1/2$. Thus, what we can reasonably ask for is to answer correctly (with high probability) in the case either $p_B > 1/2 + \sigma$ or $p_B < 1/2 - \sigma$ for a given $\sigma > 0$. Unfortunately, however, the previous two sampling algorithms seem inappropriate for solving this problem because the closeness of p_B to $1/2$ determines required precision; that is, the closer p_B is to $1/2$, the more accurate estimation is necessary. In other words, what we want to estimate is not p_B itself but the following value:

$$u_B = p_B - \frac{1}{2}.$$

More specifically, the above problem is easily solved if the following approximation goal is achieved. (In the following, we use \widetilde{u}_B to denote the output of a sampling algorithm for estimating u_B .)

Approximation Goal 3: For given $\delta > 0$ and ε , $0 < \varepsilon < 1$, we want to have

$$\Pr[|\widetilde{u}_B - u_B| \leq \varepsilon |u_B|] > 1 - \delta. \quad (12)$$

Remark. Note that u_B is not always positive.

Suppose that some sampling algorithm satisfies this goal. Then for solving the above problem, we run this algorithm to estimate u_B with relative error bound $\varepsilon = 1/2$. Then decide $p_B > 1/2$ if $\widetilde{u}_B > 1/2 + \sigma/2$ and $p_B < 1/2$ if $\widetilde{u}_B < 1/2 - \sigma/2$. It is easy to check that this method correctly determines whether $p_B > 1/2$ or $p_B < 1/2$ with probability $> 1 - \delta$ (when either $p_B > 1/2 + \sigma$ or $p_B < 1/2 - \sigma$ holds).

One might want to modify our Adaptive Sampling for achieving this new approximation goal. For example, by replacing its while-condition “ $m < A$ ” with “ $m - n/2 < B$ ” and by choosing B appropriately, we may be able to satisfy the new approximation goal. Unfortunately, though, this naive approach does not seem to work. In the previous case, the stopping condition (i.e., the negation of the while-condition “ $m < A$ ”) was monotonic; that is, once $m \geq A$ holds at some point, this condition is unchanged even if we keep sampling. On the other hand, even if $m - n/2 \geq B$ holds at some point, the condition may be falsified later if we keep sampling. Due to this nonmonotonicity, the previous proof (i.e., the proof of Lemma 3.3) does not work.

Fortunately, we can deal with this nonmonotonicity by using a slightly more complicated stopping con-

Nonmonotonic Adaptive Sampling Algorithm

```

begin
  m ← 0; n ← 0;
  u ← 0; α ← ∞;
  while |u| < α(1 + 1/ε) do
    get x uniformly at random from D;
    m ← m + B(x);
    u ← m/n - 1/2;
    α ← √((1/2n) ln(n(n + 1)/δ));
  output u as an approximation of u_B;
end.
```

Fig. 4 Nonmonotonic Adaptive Sampling

dition. In Figure 4, we state an adaptive sampling algorithm that estimates u_B and satisfies the new approximation goal. (For simplifying our discussion, we consider here only the problem of estimating u_B defined as $p_B - 1/2$, but we can use the same technique when u_B is defined by $f(p_B)$ for some “smooth” function. See [4], [5] for details.)

Theorem 4.1: For any $\delta > 0$ and ε , $0 < \varepsilon < 1$, Nonmonotonic Adaptive Sampling satisfies (12).

We argue in a similar way as Section 3. Again let t be a random variable whose value is the step when the algorithm terminates. For any $k \geq 1$, we use \widetilde{u}_k and α_k to denote respectively the value of u and α at the k th step. Define t_0 and t_1 by

$$t_0 = \min_k \{ \alpha_k \leq \varepsilon |u_B| \}, \text{ and}$$

$$t_1 = \min_k \{ \alpha_k \leq \varepsilon |u_B| / (1 + 2\varepsilon) \}.$$

Since α_k decreases monotonously in k , both t_0 and t_1 are uniquely determined, and $t_0 \leq t_1$.

Below we first show that if $t_0 \leq t \leq t_1$, that is, if the algorithm stops no earlier than the t_0 th step nor later than the t_1 th step, then its output \widetilde{u}_t is in the desired range. Then we will discuss the probability that the algorithm halts between the t_0 th and t_1 th step.

Lemma 4.2: If $t_0 \leq t \leq t_1$, then we have $|\widetilde{u}_t - u_B| \leq \varepsilon |u_B|$ with probability $> 1 - \delta / (2t_0)$.

Proof. Since the algorithm stops at the t th step, the while-condition holds at the $(t - 1)$ th step; that is, we have $|\widetilde{u}_{t-1}| < \alpha_{t-1}(1 + 1/\varepsilon)$. Here t is large enough so that we may assume that the difference between $|\widetilde{u}_t|$ and $|\widetilde{u}_{t-1}|$ and the difference between α_t and α_{t-1} are both negligible. Then we have

$$|\widetilde{u}_t| \lesssim \alpha_t \left(1 + \frac{1}{\varepsilon} \right) \leq \alpha_{t_0} \left(1 + \frac{1}{\varepsilon} \right) \quad (13)$$

$$\leq \varepsilon |u_B| \left(1 + \frac{1}{\varepsilon} \right) = (1 + \varepsilon) |u_B|,$$

where the last inequality is from the choice of t_0 .

Similarly by using $t \leq t_1$ and assuming that $\alpha_{t_1} \approx \varepsilon |u_B| / (1 + 2\varepsilon)$, we can prove that

$$|\tilde{u}_t| \gtrsim (1 - \varepsilon)|u_B|. \quad (14)$$

Now if \tilde{u}_t and u_B have the same sign, then the lemma follows from (13) and (14). On the other hand, if \tilde{u}_t and u_B have different signs, then they must be quite different because $|\tilde{u}_t| \geq \alpha_t(1 + 1/\varepsilon)$; the difference is far enough for us to prove, by using the Hoeffding bound, that such a situation occurs with probability $< \delta/(t(t+1)) \leq \delta/(2t_0)$ (since $t \geq t_0 \geq 1$). \square

Now by the following two lemmas, we give a lower bound to the probability that the algorithm halts in the desired interval. Then it is easy to check that the total error probability is bounded by δ , which proves Theorem 4.1.

Lemma 4.3:

$$\Pr[t < t_0] < \delta \left(1 - \frac{1}{t_0} \right).$$

Proof. In order to bound $\Pr[t < t_0]$, we first consider, for any k , $1 \leq k < t_0$, the probability P_k that the algorithm halts at the k th step.

Note that the algorithm halts at the k th step if and only if $|\tilde{u}_k| \geq \alpha_k(1 + 1/\varepsilon)$. Thus, we have

$$\begin{aligned} P_k &= \Pr \left[|\tilde{u}_k| \geq \alpha_k \left(1 + \frac{1}{\varepsilon} \right) \right] \\ &= \Pr \left[|\tilde{u}_k| \geq \frac{\alpha_k}{\varepsilon} + \alpha_k \right] \\ &\leq \Pr[|\tilde{u}_k| > |u_B| + \alpha_k], \end{aligned}$$

because $\alpha_k > \varepsilon|u_B|$ since $k < t_0$.

This means that $P_k \leq \Pr[\tilde{u}_k > u_B + \alpha_k]$ if $\tilde{u}_k \geq 0$, and $P_k \leq \Pr[\tilde{u}_k < u_B - \alpha_k]$ otherwise. Both probabilities are bounded by using the Hoeffding bound in the following way. (Here we only state the bound for the former case. Also although we simply uses the Hoeffding bound below, precisely speaking, the argument as in the proof of Theorem 3.3 is necessary to fix the number of examples. That is, we first modify the algorithm so that it always sees k examples.)

$$\begin{aligned} P_k &\leq \Pr[\tilde{u}_k > u_B + \alpha_k] \\ &= \Pr \left[\sum_{i=1}^k X_i/n - \frac{1}{2} > p_B - \frac{1}{2} + \alpha_k \right] \\ &\leq \exp(-2\alpha_k^2 k) = \frac{\delta}{k(k+1)}. \end{aligned}$$

Now summing up these bounds, we have

$$\Pr[t < t_0] \leq \sum_{k=1}^{t_0-1} P_k \leq \delta \left(1 - \frac{1}{t_0} \right). \quad \square$$

Lemma 4.4:

$$\Pr[t > t_1] < \frac{\delta}{2t_0}$$

Proof. Note that $t > t_1$ implies that $|\tilde{u}_{t_1}| < (1 + 1/\varepsilon)\alpha_{t_1}$. Hence we have

$$\begin{aligned} \Pr[t > t_1] &\leq \Pr \left[|\tilde{u}_{t_1}| < \alpha_{t_1} \left(1 + \frac{1}{\varepsilon} \right) \right] \\ &= \Pr \left[|\tilde{u}_{t_1}| < \left(\frac{1 + 2\varepsilon}{\varepsilon} \right) \alpha_{t_1} - \alpha_{t_1} \right] \\ &\leq \Pr[|\tilde{u}_{t_1}| < |u_B| - \alpha_{t_1}]. \end{aligned}$$

because $\alpha_{t_1} \leq \varepsilon|u_B|/(1 + 2\varepsilon)$.

Again note that $|\tilde{u}_{t_1}| < |u_B| - \alpha_{t_1}$ implies either $\tilde{u}_{t_1} < u_B - \alpha_{t_1}$ or $\tilde{u}_{t_1} > u_B + \alpha_{t_1}$. Thus, by using the Hoeffding bound, we can bound the above probability by $\exp(-2\alpha_{t_1}^2 t_1) = \delta/(t_1(t_1 + 1)) \leq \delta/(2t_0)$ (since $t_1 \geq t_0 \geq 1$). \square

Here again we can estimate ‘‘average’’ sample size from the above lemma. The lemma says that Nonmonotonic Adaptive Sampling needs no more than t_1 examples, where t_1 is the smallest integer satisfying

$$t_1 \geq \frac{(1 + 2\varepsilon)^2}{2(\varepsilon u_B)^2} \ln \left(\frac{t_1(t_1 + 1)}{\delta} \right).$$

To get an approximate bound, we substitute t_1 in the righthand side by $1/(\varepsilon u_B)^2$ and obtain the following size bound.

$$\text{sample size} \lesssim \frac{2(1 + 2\varepsilon)^2}{(\varepsilon u_B)^2} \ln \left(\frac{1}{\varepsilon u_B \delta} \right).$$

5. Concluding Remarks

We explained three sampling techniques by taking a simple estimation problem as an example. While the first two techniques have been used and tested with various practical data, the third one has not been tested well. Such experiments are important in order to evaluate the tightness of our analysis and the choice of parameters. Note that the formulas used in the above algorithms are do not provide the best possible sample size. For example, by assuming the central limit approximation, we can achieve the same estimation with 2 to 3 times smaller sample size. Also we can improve Nonmonotonic Adaptive Sampling by reducing $\ln(1/(\varepsilon u_B \delta))$ factor in its sample size bound [4], but then we should be a bit careful to get the best sample size in a given parameter range.

Acknowledgment

This paper is based on a series of joint works [3]–[5] with Carlos Domingo and Ricard Gavaldà. I have been enjoying working with these talented researchers. I thank Carlos and the anonymous referee for pointing out errors in the early version of this paper. I also thank Professor Satoru Miyano and the editorial committee of this special issue for giving me the opportunity to write this article. This work is supported in part by Grant-in-Aid for Scientific Research on Priority Areas (Discovery Science), 1999, the Ministry of Education, Science, Sports and Culture.

References

- [1] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and I. Verkamo, "Fast discovery of association rules," in *Advances in Knowledge Discovery and Data Mining*, AAAI Press, pp.307–328, 1996.
- [2] 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.
- [3] C. Domingo, R. Gavaldà, and O. Watanabe, "Practical algorithms for on-line selection," in *Proceedings of the First International Conference on Discovery Science*, Lecture Notes in Artificial Intelligence 1532, Springer-Verlag, pp.150–161, 1998.
- [4] C. Domingo, R. Gavaldà, and O. Watanabe, "Adaptive sampling methods for scaling up knowledge discovery algorithms," Research Report C-136, Dept. of Math. and Computing Sciences, Tokyo Institute of Technology, 1999. (Available at www.is.titech.ac.jp/research/research-report/C/)
- [5] C. Domingo, R. Gavaldà, and O. Watanabe, "Adaptive sampling methods for scaling up knowledge discovery algorithms (the 2nd version)," in *Proceedings of the Second International Conference on Discovery Science*, to appear.
- [6] W. Feller, *An Introduction to Probability Theory and its Applications* (Third Edition), John Wiley & Sons, 1968.
- [7] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association* 58, pp.13–30, 1963.
- [8] 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*, AAAI/MIT Press, 1996.
- [9] J. Kivinen and H. Mannila, "The power of sampling in knowledge discovery," in *Proceedings of the ACM SIGACT-SIGMOD-SIGACT Symposium on Principles of Database Theory*, ACM Press, pp.77–85, 1994.
- [10] 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.
- [11] R.J. Lipton and J.F. Naughton, "Query size estimation by adaptive sampling," *Journal of Computer and System Science* 51, pp.18–25, 1995.
- [12] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, 1995.
- [13] D. Siegmund, *Sequential Analysis: Tests and Confidence Intervals*, Springer-Verlag, 1985.
- [14] A. Wald, *Sequential Analysis*, John Wiley & Sons, 1947.
- [15] M. Wang, B. Iyer, and J.S. Vitter, "Scalable mining for classification rules in relational databases," In *Proceedings of IDEAS'98*, pp.58–67, 1998.
- [16] G.B. Wetherill, *Sequential Methods in Statistics* (Second Edition), Chapman and Hall, 1975.