Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 348 (2005) 3 – 14
Theoretical Computer Science
www.elsevier.com/locate/tcs

Sequential sampling techniques for algorithmic learning theory

Osamu Watanabe

Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo 152-8552, Japan

Abstract

A *sequential sampling algorithm* or *adaptive sampling algorithm* is a sampling algorithm that obtains instances sequentially one by one and determines from these instances whether it has already seen enough number of instances for achieving a given task. In this paper, we present two typical sequential sampling algorithms. By using simple estimation problems for our example, we explain when and how to use such sampling algorithms for designing *adaptive* learning algorithms.

Keywords: Sequential sampling; Adaptive sampling; Adaptive learning algorithm; Chernoff bound; Hoeffding bound

1. Introduction

Random sampling is an important technique in computer science for developing efficient randomized algorithms. A task such as estimating the proportion of instances with a certain property in a given data set can often be achieved by randomly sampling a relatively small number of instances. *Sample size*, i.e., the number of sampled instances, is a key factor for sampling, and for determining appropriate sample size, so-called concentration bounds or large deviation bounds have been used (see, e.g., [9]). In particular, the Chernoff bound [2] and the Hoeffding bound [14] have been used commonly in theoretical computer science because they derive a theoretically guaranteed sample size sufficient for achieving a given task with given accuracy and confidence. There are some cases, however, where these bounds can provide us with only overestimated or even unrealistic sample size. In this paper, we show that "sequential sampling algorithms" are applicable for some of such cases to design *adaptive* randomized algorithms with theoretically guaranteed performance.

A *sequential sampling algorithm* or *adaptive sampling algorithm* is a sampling algorithm that obtains instances sequentially one by one and determines from these instances whether it has already seen enough number of instances for achieving a given task. Intuitively, from the instances seen so far, we can more or less obtain some knowledge on the input data set, and it may be possible to estimate an appropriate sample size. Recently, we have proposed [7,8] a sequential sampling algorithm for a general hypothesis selection problem (see also [6] for some preliminary versions). Our main motivation was to scale up various known learning algorithms for practical applications such as data mining (see, e.g., discussions in [8]). While some applications and extensions of our approach towards this direction have been reported [1,4,20], it has been also noticed [3,5] that sequential sampling allows us to add "adaptivity" to learning algorithms while keeping their worst-case performance. In this paper, we use some simple examples and explain when and how to use sequential sampling for designing such adaptive learning algorithms.

E-mail address: watanabe@is.titech.ac.jp.

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.09.003

The idea of "sampling on-line" is quite natural, and it has been studied in various contexts. First of all, statisticians made significant accomplishments on sequential sampling during World War II [21]. In fact, from their activities, a research area on sequential sampling—sequential analysis—has been formed in statistics. Thus, it may be quite likely that some of the algorithms explained here have been already found in their contexts. (For recent studies on sequential analysis, see, e.g., [10,11].) In computer science, sequential sampling techniques have been studied in the database community. Lipton and Naughton [17] and Lipton et al. [16] proposed adaptive sampling algorithms for estimating query size in relational databases. Later Haas and Swami [13] proposed an algorithm that performs better than the Lipton–Naughton algorithm in some situations. More recently, Lynch [18] gave a rigorous analysis to the Lipton–Naughton algorithm. Roughly speaking, the spirit of sequential sampling is to use instances observed so far for reducing a current and future computational task. This spirit can be found in some of the learning algorithms proposed in machine learning community. For example, the Hoeffding race proposed by Maron and Moore [19] attempts to reduce a search space by removing candidates that are determined hopeless from the instances seen so far. A more general sequential local search has been proposed by Greiner [12].

All the above approaches more or less share the same motivation. That is, they attempt to design "adaptive algorithms" that can make use of the advantage of the situation to reduce sample size (or in general, computation time) whenever such reduction is indeed possible. We believe that some of these approaches can be formally discussed so that we can propose *adaptive* learning algorithms with theoretically guaranteed performance.

This paper has some overlap with the author's previous survey paper on sequential sampling [22].

2. Our problem and statistical bounds

In this paper, we fix one simple estimation problem for our basic example, and discuss sampling techniques on this problem or its variations. Let us specify our 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. Our problem is to estimate the probability p_B that $B(x) = 1$ when x is given at random from D; in other words, the ratio of instances x in D such that $B(x) = 1$ holds.

Clearly, the probability p_B can be computed by counting the number of instances x in D for which $B(x) = 1$ holds. In fact, this is only the way if we are asked to compute p_B *exactly*. 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 a situation is random sampling. That is, we pick up some instances of D randomly and estimate the probability p_B on these selected instances. Without seeing all instances, we cannot hope for computing the exact value of p_B. Also due to the "randomness", we cannot always obtain a desired answer. Therefore, we must be satisfied if our sampling algorithm yields a *good approximation* of p_B with *reasonable probability*. In this paper, we will discuss this type of approximate estimation problem.

Our estimation problem is completely specified by fixing an "approximation goal" that defines the notion of "good approximation". We consider the following one for our first approximation goal. (In the following, we will use $\widetilde{p_B}$ to denote the output of a sampling algorithm (for estimating p_B); thus, it is a random variable and the probability below is taken w.r.t. this random variable.)

Approximation Goal 1 (*Absolute error bound*). *For given* $\delta > 0$ *and* ε, $0 < \varepsilon < 1$, *the goal is to have*

$$\Pr[|\widetilde{p_B} - p_B| \leqslant \varepsilon] > 1 - \delta. \tag{1}$$

As mentioned above, the simplest sampling algorithm for estimating p_B is to pick up instances of D randomly and estimate the probability p_B on these selected instances. Fig. 1 gives the precise description of this simplest sampling algorithm, which we call *Batch Sampling* algorithm. Here the only assumption we need (for using the statistical bounds explained below) is that we can easily pick up instances from D uniformly at random and independently.

The description of Batch Sampling algorithm of Fig. 1 is still incomplete since we have not specified the way to determine n, the number of iterations or sample size. Of course, to get an accurate estimation, the larger the n the better; on the other hand, for the efficiency, the smaller the n the better. We would like to achieve a given accuracy with as small sample size as possible.