Proceedings of the Eighteenth International Conference on Machine Learning. 2001.

Incremental Maximization of Non-Instance-Averaging Utility Functions with Applications to Knowledge Discovery Problems

Tobias Scheffer[1,2] SCHEFFER@IWS.CS.UNI-MAGDEBURG.DE
Stefan Wrobel[1] WROBEL@IWS.CS.UNI-MAGDEBURG.DE
[1]University of Magdeburg, FIN/IWS, Universitätsplatz 2, 39106 Magdeburg, Germany
[2]SemanticEdge, Kaiserin-Augusta-Allee 10-11, 10553 Berlin, Germany

Abstract

Data-independent sample bounds are known to grossly overestimate the amount of data needed for most individual problem instances. This has led to significant recent interest in *sequential* algorithms which also give precise guarantees about the quality of results, but determine the amount of data needed based on characteristics of the actual problem instance at hand, and thus need significantly fewer examples. In this paper, we present a practical sequential sampling algorithm which (a) is capable of quickly identifying the n best hypotheses and (b) works for *all* utility functions that can be estimated with bounded error. The algorithm thus can be used not only for predictive learning, but also for tasks such as association rule finding or subgroup discovery. Our experiments with two real-world domains show that our algorithm can be orders of magnitude faster than non-sequential sampling algorithms.

1. Introduction

In many machine learning settings, an agent has to find a hypothesis which maximizes a given utility criterion. This criterion can be as simple as classification accuracy, or it can be a combination of generality and accuracy of, for instance, an association rule. The utility of a hypothesis can only be estimated based on data; it cannot be determined exactly (this would generally require processing very large, or even infinite amounts of data). Algorithms can still give stochastic guarantees on the optimality of the returned hypotheses, but providing guarantees that hold for all possible problems usually requires impractically large samples.

Past work on algorithms with stochastic guarantees has concentrated on predictive learning with *instance-averaging* utility functions, and has pursued two approaches — either processing a fixed amount of data and making the *guarantee* dependent on the observed empirical utility values (*e.g.,* Freund, 1998; Langford & McAllester, 2000), or demanding a certain fixed quality and making the number of *examples* dependent on the observed utility values (Wald, 1947; Maron & Moore, 1994; Greiner, 1996; Domingo et al., 1999) (this is often referred to as *sequential sampling*).

In this paper, we generalize known sampling results in two respects. Firstly, in many cases, it is more natural for a user to ask for the n best solutions instead of the single best or all hypotheses above a threshold. Secondly, and more importantly, many popular utility functions for tasks other than predictive learning (see, *e.g.,* Klösgen, 1996) cannot be expressed as instance-averaging functions. Improving on (Scheffer & Wrobel, 2000), we therefore present a *practical* algorithm that works for *all* utility functions that can be estimated with bounded error, and thus can be used not only for predictive learning, but also for tasks such as association rule finding or subgroup discovery. Our experiments with two real-world domains show that our algorithm can be orders of magnitude faster than non-sequential sampling algorithms.

In Section 2, we discuss the problem setting and related research. We generalize the problem setting in Section 3. Section 4 describes our sampling algorithm and presents the stochastic quality guarantee. In Section 5, we instantiate the algorithm for most of the utility functions popular in KDD and prove that there is no algorithm with similar guarantees for one utility function. We discuss experimental results on large databases in Section 6; Section 7 concludes.

2. Problem Setting and Prior Work

The most common instantiation of utility-maximizing search is predictive learning from examples. Here, the utility function to be maximized is the probability of correctly classifying a random instance drawn from the instance space X according to a fixed, unknown distribution D. Since we are only given a sample S of classified instances, we can only *estimate* the error rate and thus run the risk of over- or underestimating particular hypotheses and returning a suboptimal solution.

While many practical learning algorithms heuristically try to limit this risk, it is clearly desirable to arrive at learning algorithms that can give precise guarantees about the quality of their solutions. If the learning algorithm is not allowed to look at any data before specifying the guarantee or fixing the required sample size ("data-independent"), we arrive at impractically large bounds as they arise, for instance, when applying PAC learning (*e.g.,* Haussler, 1992) in a data-independent way. Researchers have therefore turned to algorithms that are allowed to look at (parts of) the data first.

We can then ask two questions. Knowing that our sample will be of size m, we can ask about the quality guarantee that results. On the other hand, knowing that we would like a particular quality guarantee, we can ask how large a sample we need to draw to ensure that guarantee. The former question has been addressed for predictive learning in work on self-bounding learning algorithms (Freund, 1998) and shell decomposition bounds (Haussler et al., 1996; Langford & McAllester, 2000).

For our purposes here, the latter question is more interesting. We assume that samples can be requested incrementally from an oracle ("incremental learning"). We can then dynamically adjust the required sample size based on the characteristics of the data that have already been seen; this idea has originally been referred to as sequential analysis (Dodge & Romig, 1929; Wald, 1947). Note that even when a (very large) database is given, it is useful to assume that examples are drawn incrementally from this database, potentially allowing termination before processing the entire database (referred to as *sampling* in KDD; Toivonen, 1996).

For predictive learning, the idea of sequential analysis has been developed into the Hoeffding race algorithm (Maron & Moore, 1994). It processes examples incrementally, updates the utility values simultaneously, and outputs (or discards) hypotheses as soon as it becomes very likely that some hypothesis is near-optimal (or very poor, respectively). The incremental greedy learning algorithm PALO (Greiner, 1996) has been reported to require many times fewer examples than the worst-case bounds suggest. In a KDD context, similar improvements have been achieved with the sequential algorithm of (Domingo et al., 1999).

3. Generalized Problem Setting

We generalize these above results in two respects. First, in many cases, it is more natural for a user to ask for the n best solutions instead of the single best or all hypotheses above a threshold. For instance, a user might find a small number of the most interesting patterns in a database, as is the case for association rule (Agrawal et al., 1996) or subgroup discovery (Klösgen, 1996; Wrobel, 1997). We thus arrive at the following generalized problem statement and quality guarantee.

Definition 1 (Approximate n-best hypotheses problem) *Let D be a distribution on instances, H a set of hypotheses, $f : H \rightarrow \mathbb{R}^{\geq 0}$ a function that assigns a utility value to each hypothesis and n a number of desired solutions. Then let δ, $0 < \delta \leq 1$, be a user-specified confidence, and $\varepsilon \in \mathbb{R}^+$ a user-specified maximal error. The* approximate n-best hypotheses problem *is to find a set $G \subseteq H$ of size n such that*

> *with confidence $1-\delta$, there is no $h' \in H$: $h' \notin G$ and $f(h', D) > f_{min} + \varepsilon$, where $f_{min} := min_{h \in G} f(h, D)$.*

Secondly, and more importantly, the work mentioned above has focused on the particular class of *instance-averaging* utility functions where the utility of a hypothesis h is the average of utilities defined locally for each instance. While prediction error clearly is an instance-averaging utility function, popular utility functions for other learning or discovery tasks often combine the generality of hypotheses with distributional properties in a way that cannot be expressed as average over the data records (Klösgen, 1996).

A popular example of such a discovery task is *subgroup discovery* (Klösgen, 1996). Subgroups characterize subsets of database records within which the average value of the target attributes differs from the global average value, without actually conjecturing a value of that attribute. For instance, a subgroup might characterize a population which is particularly likely (or unlikely) to buy a certain product. The generality of a subgroup is the fraction of all database records that belong to that subgroup. The term *statistical unusualness* refers to the difference between the default probability p_0 (the target attribute taking value one in the whole database) and the probability p of a target value of one within the subgroup. Usually, subgroups are desired to be both general (large g) and statistically unusual (large $|p - p_0|$). There are many possible utility functions (Klösgen, 1996) for subgroup discovery, none of which can be expressed as the average (over all instances) of an instance utility function.

Consequently, in order to avoid unduly restricting our algorithm, we will not make syntactic assumptions about f. In particular, we will not assume that f is based on averages of instance properties. Instead, we only assume that it is possible to determine a two-sided *confidence interval* f that bounds the possible difference between true utility and estimated utility (on a sample) with a certain confidence. As we will show in Section 5 below, finding such confidence intervals is straightforward for classification accuracy, and is also possible for all but one of the popular utility functions from association rule and subgroup discovery.

Definition 2 (Utility confidence interval) *Let f be a utility function, let $h \in H$ be a hypotheses. Let $f(h)$ denote the true utility of h on the instance distribution D, $\hat{f}(h, Q_m)$ its estimated quality computed based on a sample Q_m of size m, drawn* iid *from the distribution D. Then $E : \mathbb{N} \times \mathbb{R} \rightarrow \mathbb{R}$ is a* utility confidence bound *for f iff for any δ, $0 < \delta < 1$,*

$$Pr_{Q_m}[|\hat{f}(h, Q_m) - f(h)| \leq E(m, \delta)] \geq 1 - \delta \qquad (1)$$

If, in addition, for any $\delta, 0 < \delta \leq 1$ and any ε there is a number m such that $E(m, \delta) \leq \varepsilon$ we say that the confidence interval *vanishes*. We will see that we can only guarantee termination when the confidence interval vanishes.

We sometimes write the confidence interval for a specific hypothesis h as $E_h(m, \delta)$. Thus, we allow the confidence interval to depend on characteristics of h, such as the variance of one or more random variables that the utility of h depends on. We will discuss confidence intervals for different functions in Section 5.

4. Algorithm

In our algorithm (Table 1), we combine sequential sampling with the popular "loop reversal" technique found in many KDD algorithms. In step 3b, we collect data incrementally and apply these to all remaining hypotheses simultaneously (step 3c). This strategy allows the algorithm to be easily implemented on top of database systems (assuming they are capable of drawing samples), *and* enables us to terminate earlier. After the statistics of each remaining hypothesis have been updated, the algorithm checks all remaining hypotheses and (step 3(e)i) outputs those where it can be sufficiently certain that the number of better hypotheses is no larger than the number of hypotheses still to be found (so they can all become solutions), or (Step 3(e)ii) discards those hypotheses where it can be sufficiently certain that the number of better other hypotheses is at least the number of hypotheses still to be found (so it can be sure the current hypothesis does not need to be in the solutions). When the algorithm has gathered enough information to distinguish the good hypotheses that remain to be found from the bad ones with sufficient probability, it exits in step 3. Indeed it can be shown that this strategy leads to a total error probability less than δ as required.

Theorem 1 *The algorithm will output a group G of exactly n hypotheses (assuming that $|H| > n$) such that, with confidence $1 - \delta$, no other hypothesis in H has a utility which is more than ε higher than the utility of any hypothesis that has been returned:*

$$Pr[\exists h \in H \setminus G : f(h) > f_{min} + \varepsilon] \leq \delta \qquad (2)$$

where $f_{min} = \min_{h' \in G}\{f(h')\}$.

The proof (which can be found in the full paper; Scheffer & Wrobel, 2001) has two parts. We can first prove that at any time step i $(1 \leq i \leq M)$ and for any hypothesis $h \in H_i$, the two-sided difference between $\hat{f}(h, Q_i)$ and $f(h)$ is at most $E_h(i, \frac{\delta}{2M|H_i|})$. Using this Lemma, we can then prove that the algorithm never outputs a hypothesis for which there are at least n other hypotheses with a utility value that is ε or more higher, and never discards a hypothesis which is among the n best hypotheses. Secondly, we can show that after M examples have been seen, we can be certain that the two-sided difference between $\hat{f}(h, Q_i)$ and $f(h)$ lies below $\frac{\varepsilon}{2}$. It is then safe to output the hypotheses with highest estimated utility.

Theorem 2 (Termination) *If for any δ $(0 < \delta \leq 1)$ and $\varepsilon > 0$ there is a number m such that $E(m, \delta) \leq \varepsilon$, then the algorithm can be guaranteed to terminate. Moreover, the number of required examples is at most the smallest number m for which $E(m, \frac{\delta}{2|H|}) \leq \frac{\varepsilon}{2}$.*

Correctness of Theorem 2 follows immediately from Step 3 of the algorithm. Theorem 2 says that we can guarantee termination if the confidence interval vanishes for large numbers of examples.

5. Instantiations

In order to implement the algorithm for a given interestingness function we have to find a confidence bound $E(m, \delta)$ that satisfies Equation 1 for that specific f. We will in the following present a list of confidence intervals. We will start with the easiest case – instance averaging functions such as classification accuracy – and then discuss the functions that are most commonly used for knowledge discovery tasks. We ask the reader to refer to the full paper (Scheffer & Wrobel, 2001) for the proofs.

Table 1. Generic sequential sampling algorithm for the n-best hypotheses problem

Algorithm. Input: n (number of desired hypotheses), ε and δ (approximation and confidence parameters).
Output: n approximately best hypotheses (with confidence $1-\delta$).

1. **Let** $n_1 = n$ (the number of hypotheses that we still need to find) and **Let** $H_1 = H$ (the set of hypotheses that have, so far, neither been discarded nor accepted). **Let** $Q_0 = \emptyset$ (no sample drawn yet). **Let** $i = 1$ (loop counter).
2. **Let** M be the smallest number such that $E(M, \frac{\delta}{2|H|}) \leq \frac{\varepsilon}{2}$.
3. **Repeat until** $n_i = 0$ **Or** $|H_{i+1}| = n_i$ **Or** $E(i, \frac{\delta}{2|H_i|}) \leq \frac{\varepsilon}{2}$
   (a) **Let** $H_{i+1} = H_i$.
   (b) Query a random item of the database q_i. **Let** $Q_i = Q_{i-1} \cup \{q_i\}$.
   (c) Update the empirical utility $\hat{f}$ of the hypotheses in H_i.
   (d) **Let** H_i^* be the n_i hypotheses from H_i which maximize the empirical utility $\hat{f}$.
   (e) **For** $h \in H_i$ **While** $n_i > 0$ **And** $|H_i| > n_i$
      i. **If** $\hat{f}(h, Q_i) \geq E_h(i, \frac{\delta}{2M|H_i|}) + \max_{h_k \in H_i \setminus H_i^*} \left\{ \hat{f}(h_k, Q_i) + E_{h_k}(i, \frac{\delta}{2M|H_i|}) \right\} - \varepsilon$ **And** $h \in H_i^*$ (h appears good) **Then Output** hypothesis h and then **Delete** h from H_{i+1} and **Let** $n_{i+1} = n_i - 1$. **Let** H_i^* be the new set of empirically best hypotheses.
      ii. **Else If** $\hat{f}(h, Q_i) \leq \min_{h_k \in H_i^*} \left\{ \hat{f}(h_k, Q_i) - E_{h_k}(i, \frac{\delta}{2M|H_i|}) \right\} - E_h(i, \frac{\delta}{2M|H_i|})$ (h appears poor) **Then Delete** h from H_{i+1}. **Let** H_i^* be the new set of empirically best hypotheses.
   (f) **Increment** i.
4. **Output** the n_i hypotheses from H_i which have the highest empirical utility.

5.1 Instance-Averaging Functions

This simplest form of a utility function is the average, over all example queries, of some instance utility function $f_{inst}(h, q_i)$. The utility is then defined as $f(h) = \int f_{inst}(h, q_i) D(q_i) dq_i$ (the average over the instance distribution) and the estimated utility is $\hat{f}(h, Q_m) = \frac{1}{m} \sum_{i=1}^{m} f_{inst}(h, q_i)$ (average over the example queries). An easy example of an instance-averaging utility is the classification accuracy. We assume that the possible range of utility values lies between 0 and Λ ($\Lambda = 1$ for classification accuracy).

We can derive a confidence interval $E(m, \delta)$ which satisfies Equation 1 from the Hoeffding inequality (which is a general probability tail bound). Equation 1 is satisfied when we choose $E(m, \delta) = \sqrt{\frac{\Lambda^2}{2m} \log \frac{2}{\delta}}$. In Equation 3 we insert our definition of E into Equation 1. We apply the Hoeffding inequality in Equation 4 and obtain the desired result.

$$Pr\left[|\hat{f}(h, Q_m) - f(h)| > E(m, \delta)\right]$$

$$= Pr\left[|\hat{f}(h, Q_m) - f(h)| > \sqrt{\frac{\Lambda^2}{2m} \log \frac{2}{\delta}}\right] \quad (3)$$

$$\leq 2 \exp\left\{ -2m \frac{\left(\sqrt{\frac{\Lambda^2}{2m} \log \frac{2}{\delta}}\right)^2}{\Lambda^2} \right\} \leq \delta \quad (4)$$

For implementation purposes, the Hoeffding inequality is less suited since it it not very tight. For large m, we can replace the Hoeffding inequality by the normal distribution, referring to the central limit theorem. $\hat{f}(h, Q_m) - f(h)$ is a random variable with mean value 0; we further know that $\hat{f}(h, Q_m)$ is bounded between zero and Λ. In order to calculate the normal distribution, we need to refer to the variance of our random variable. In step 3, the variance is not known since we do not refer to any particular hypothesis. We can only bound the variance as $s \leq \frac{\Lambda}{2\sqrt{m}}$. Random variable $\frac{2\sqrt{m}(\hat{f}(h, Q_m) - f(h))}{\Lambda}$ is governed by the standard normal distribution which implies that a confidence interval of $E(m, \delta) = z_{1-\frac{\delta}{2}} \cdot \frac{\Lambda}{2\sqrt{m}}$ satisfies Equation 1. This interval is a little tighter than Hoeffding's inequality. We can look up the inverse normal distribution z.

In Steps 3(e)i and 3(e)ii, we refer to specific hypotheses h and can therefore determine the empirical variance

Table 2. Utility functions and the corresponding utility confidence bounds

$f(h)$	$E(m,\delta)$	sample bound
instance-averaging	$E(m,\delta) = -\frac{z_{1-\frac{\delta}{2}}\Lambda}{2\sqrt{m}}$; $\quad E_h(m,\delta) = -z_{1-\frac{\delta}{2}} s_h$	$\frac{2\Lambda^2}{\varepsilon^2}\log\frac{\|H_i\|}{2\delta}$
$g(p-p_0)$ $g\|p-p_0\|$ $g\frac{1}{c}\sum_{i=1}^{c}\|p_i-p_{0_i}\|$	$E(m,\delta) = \frac{z_{1-\frac{\delta}{4}}}{\sqrt{m}} + \frac{(z_{1-\frac{\delta}{4}})^2}{4m}$ $E_h(m,\delta) = z_{1-\frac{\delta}{4}}(s_g + s_p + z_{1-\frac{\delta}{4}} s_g s_p)$	$\frac{18}{\varepsilon^2}\log\frac{8\|H_i\|}{\delta}$
$g^2(p-p_0)$ $g^2\|p-p_0\|$ $g^2\frac{1}{c}\sum_{i=1}^{c}\|p_i-p_{0_i}\|$	$E(m,\delta) = \frac{3}{2\sqrt{m}}z_{1-\frac{\delta}{2}} + \frac{m+\sqrt{m}}{4m\sqrt{m}}(z_{1-\frac{\delta}{2}})^2 + \frac{1}{8m\sqrt{m}}(z_{1-\frac{\delta}{2}})^3$ $E_h(m,\delta) = (2s_g+s_p)z_{1-\frac{\delta}{2}} + (s_g^2+2s_g s_p)(z_{1-\frac{\delta}{2}})^2 + s_p s_g^2 (z_{1-\frac{\delta}{2}})^3$	$\frac{98}{\varepsilon^2}\log\frac{8\|H_i\|}{\delta}$
$\sqrt{g}(p-p_0)$ $\sqrt{g}\|p-p_0\|$ $\sqrt{g}\frac{1}{c}\sum_{i=1}^{c}\|p_i-p_{0_i}\|$	$E(m,\delta) = \sqrt{\frac{z_{1-\frac{\delta}{4}}}{2\sqrt{m}}} + \frac{z_{1-\frac{\delta}{4}}}{2\sqrt{m}} + \sqrt{\frac{z_{1-\frac{\delta}{4}}}{2\sqrt{m}}}\frac{z_{1-\frac{\delta}{4}}}{2\sqrt{m}}$ $E_h(m,\delta) = \sqrt{s_g z_{1-\frac{\delta}{4}}} + s_p z_{1-\frac{\delta}{4}} + \sqrt{s_g z_{1-\frac{\delta}{4}}}s_p z_{1-\frac{\delta}{4}}$	$\frac{648}{\varepsilon^2}\log\frac{8\|H_i\|}{\delta}$

of $\hat{f}(h, Q_m)$. We can define $E_h(m,\delta)$ in Equation 6.

$$E(m,\delta) = z_{1-\frac{\delta}{2}} \cdot s_h \quad (5)$$

$$= \quad z_{1-\frac{\delta}{2}}\frac{1}{m}\sqrt{\sum_{i=1}^{m}(f_{inst}(h,q_i) - \hat{f}(h,Q_i))^2} \quad (6)$$

The algorithm exits the for loop (at latest) when $E\left(m, \frac{\delta}{2|H|}\right) \leq \frac{\varepsilon}{2}$. We can show that this is the case with certainty when $m \geq \frac{2\Lambda^2}{\varepsilon^2}\log\frac{|H|}{2\delta}$. It follows from inserting this bound and our definition of E into the Hoeffding equality.

But note that our algorithm will generally terminate much earlier; firstly, because we use the normal distribution rather than the Hoeffding approximation and, secondly, our sequential approach will terminate much earlier when the n best hypotheses differ considerably from many of the "bad" hypotheses. The worst case occurs only when all hypotheses in the hypothesis space are equally good which makes it much more difficult to identify the n best ones.

5.2 Utility Functions used for KDD Problems

Table 2 summarizes our results on more general utility functions that are popular in KDD (Klösgen, 1996). See our full paper (Scheffer & Wrobel, 2001) for proofs and a more detailed discussion.

The first class of nontrivial utility functions weight the generality g of a subgroup and the deviation of the probability of a certain feature p from the default probability p_0 equally (Piatetski-Shapiro, 1991). Hence, these functions multiply generality and distributional unusualness of subgroups. Alternatively, we can use the absolute distance $|p - p_0|$ between probability p and default probability p_0. The multi-class version of this function is $g\frac{1}{c}\sum_c |p_i - p_{0_i}|$ where p_{0_i} is the default probability for class i.

The third row of Table 2 summarizes our results for utility functions with squared terms (Wrobel, 1997) which are introduced to put more emphasis on the the difference between p and the default probability.

The final class of utility functions that we study is derived from the binomial test heuristic (Klösgen, 1992) which is based on elementary considerations. Suppose that the probability p is really equal to p_0 (*i.e.,* the corresponding subgroup is really uninteresting). How likely is it that the subgroup with generality g displays a frequency of $\hat{p}$ on the sample Q with a greater difference $|\hat{p} - p_0|$? For large $|Q| \times g$, $(\hat{p} - p_0)$ is governed by the normal distribution with mean value of zero and variance at most $\frac{1}{2\sqrt{m}}$. The probability density function of the normal distribution is monotonic, and so the resulting confidence is order-equivalent to $\sqrt{m}(p - p_0)$ (m being the support) which is factor equivalent to $\sqrt{g}(p - p_0)$. Several variants of this utility function have been used. The worst-case sample size bound given in Table 2 is not very tight. In the empirical studies (see Section 6) we observed quite a reasonable behavior of our algorithm for this utility.

5.3 Negative Results

Several independent impurity criteria have led to utility functions which are factor-equivalent to $f(h) = \frac{g}{1-g}(p - p_0)^2$; *e.g.,* Gini diversity index and twoing criterion (Breiman et al., 1984), and the chi-square test (Piatetski-Shapiro, 1991). Note that it is also order-equivalent to the utility measure used in Inferrule (Uthurusamy et al., 1991). Unfortunately, this utility function is not bounded and a few examples that have not been included in the sample can impose dramatic changes on the values of this function. This motivates our negative result.

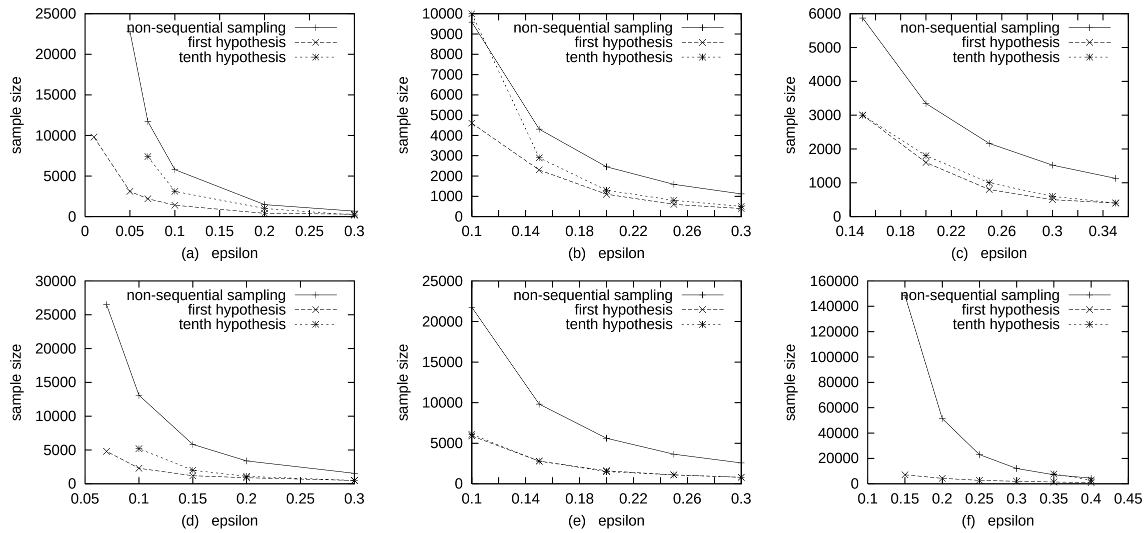

Figure 1. Sample sizes for the juice purchases database. (a) $f = g|p - p_0|$, $k = 1$, $\delta = .1$; (b) $k = 2$; (c) $k = 3$; (d) $f = g^2|p - p_0|$, $k = 1$, $\delta = .1$; (e) $k = 2$; (f) $f = \sqrt{g}|p - p_0|$, $k = 1$, $\delta = .1$

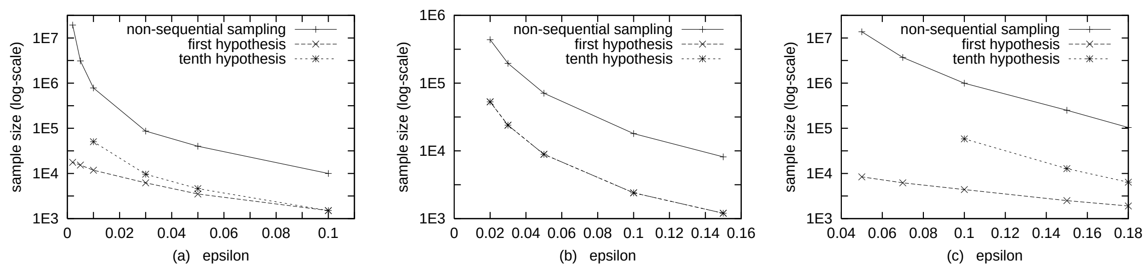

Figure 2. $\log_{10}$ of the sample sizes for the KDD cup data of 1998. (a) $f = g|p - p_0|$, $k = 1$, $\delta = .1$; (b) $f = g^2|p - p_0|$, (c) $f = \sqrt{g}|p - p_0|$.

Theorem 3 *There is no algorithm that satisfies Theorem 1 when* $f(h) = \frac{g}{1-g}(p - p_0)^2$.

Sketch of proof. We need to show that $\hat{f}(h, Q_m) - f(h)$ is unbounded for any finite m. This is easy since $\frac{g+\varepsilon}{1-(g+\varepsilon)} - \frac{g}{1-g}$ goes to infinity when g approaches 1 or $1 - \varepsilon$ (Equation 7).

$$\frac{g+\varepsilon}{1-(g+\varepsilon)} - \frac{g}{1-g} = \frac{\varepsilon}{(g+\varepsilon-1)(g-1)} \qquad (7)$$

This implies that, even after an arbitrarily large sample has been observed (that is smaller than the whole database), the utility of a hypothesis with respect to the sample can be arbitrarily far from the true utility. The picture does not change when we require $\hat{f}(h, Q)$ only to be within a multiplicative constant. When a sampling algorithm uses all but very few database transactions as sample, then the few remaining examples may still impose huge changes on $\hat{f}(h, Q_m)$ which renders the use of sampling algorithms prohibitive.

6. Experiments

In our experiments, we want to study the order of magnitude of examples which are required by our algorithm for realistic tasks. Furthermore, we want to measure how much an improvement our sequential algorithm achieves over a "static" algorithm that determines the sample size with worst-case bounds.

We implemented a simple subgroup discovery algorithm. Hypotheses consist of conjunctions of up to k attribute value tests, continuous attributes are discretized in advance. The non-sequential algorithm that we use determines a sample size M like our algorithm does in step 2, but using the full available error probability δ rather than only $\frac{\delta}{2}$. Hence, the non-sequential algorithm has a lower worst-case sample size than the sequential one but never exits or returns any hypothesis before that worst-case sample bound has been reached. Sequential and non-sequential sampling algorithm use the same normal approximation and come with identical guarantees on the quality of the returned solution.

For the first set of experiments, we used a database of 14,000 fruit juice purchase transactions. Each transaction is described by 29 attributes which specify properties of the purchased juice as well as customer attributes. The task is to identify subgroups of customers that differ from the overall average with respect to their preference for cans, recyclable bottles, or non-recyclable bottles. We studied hypothesis spaces of size 288 ($k = 1$, hypotheses test one attribute for a particular value), 37,717 ($k = 2$, conjunctions of two tests), and 3,013,794 ($k = 3$). Since δ has only a minor (logarithmic) influence on the resulting sample size, all results presented in Figure 1 were obtained with $\delta = 0.1$. We varied the utility function; the target attribute has three possible values, so we used the utility functions $f_1 = g\frac{1}{3}\sum_{i=1}^{3}|p_i - p_{0_i}|$, $f_2 = g^2\frac{1}{3}\sum_{i=1}^{3}|p_i - p_{0_i}|$, and $f_3 = \sqrt{g}\frac{1}{3}\sum_{i=1}^{3}|p_i - p_{0_i}|$.

Figure 1 shows the sample size of the non-sequential algorithm as well as the sample size required before the sequential algorithm returned the first (out of ten) hypothesis and the sample size that the sequential algorithm required to return the last (tenth) hypothesis and terminate. The sequential algorithm tends to terminate significantly earlier than the non-sequential one; as ε becomes small, the relative benefit of sequential sampling can reach orders of magnitude. Consider, for instance, the linear utility function and $k = 1, \epsilon = .1, \delta = .1$ where the sequential algorithm can return the first hypothesis after 9,800 examples whereas the non-sequential algorithm returns the solution only after 565,290 examples.

For the second set of experiments, we used the data provided for the KDD cup 1998. The data contains 95,412 records that describe mailings by a veterans organization. Each record contains 481 attributes describing one recipient of a previous mailing. The target fields note whether the person responded and how high his donation to the organization was. Our task was to find large subgroups of recipients that were particularly likely (or unlikely) to respond (we used the attribute "Target_B" as target and deleted "Target_D"). Our hypothesis space consists of all 4492 attribute value tests.

Figure 2 displays the sample sizes required by the sequential and the non-sequential sampling algorithm Note that we use a logarithmic ($\log_{10}$) scale on the y axis. Although it is fair to say that this is a large-scale problem, the sample sizes used by the sequential sampling algorithm are in a reasonable range for all three studied utility functions. Less than 10,000 examples are required when ε is as small as 0.002 for $f = g|p - p_0|$ and $f = g^2|p - p_0|$ and when ϵ is 0.05 for $f = \sqrt{g}|p - p_0|$. The relative benefit of sequential over non-sequential sampling is between one and three orders of magnitude. For instance, in Figure 2a ($\varepsilon = 0.002$) the non-sequential algorithm requires over 10^7 examples (of course, much more than are available) whereas the sequential one needs about $10^{4.2}$.

7. Discussion and Related Results

Sequential analysis is a very promising approach to reducing the sample size required to guarantee a high quality of the returned hypotheses. Sample sizes in the order of what the Chernoff and Hoeffding bounds suggest are only required when all hypotheses exhibit identical empirical utility values (in this case, identifying which one is really best is difficult). In all other cases, the single best, or the n best hypotheses can be identified much earlier.

In essence, the problem settings of sequential analysis (Wald, 1947), incremental learning (*e.g.,* Greiner, 1996) and database sampling (*e.g.,* Toivonen, 1996) are equal. In each case, data is processed sequentially and the hypothesis space updated in each step. The learning algorithm is to determine when sufficient data has been processed and no further data needs to be collected. The setting differs from most definitions of active learning (*e.g.,* Cohn et al., 1996) in the fact that the algorithm determines the number of examples it needs, but it does not query specific (regions of) instances that it considers helpful. Also, the setting differs fundamentally from the online learning setting (*e.g.,* Ben-David et al., 1997) where the target hypothesis needs to be identified exactly after finitely many steps.

The main contribution of this paper is a generalization of sequential analysis to utility functions that cannot be expressed as an average over all instances. We can thus cover popular utility functions used in data mining such as $g|p - p_0|$ where both g and p are an average. Known machine learning and knowledge discovery algorithms that exploit the general idea of sequential analysis (*e.g.,* Maron & Moore, 1994; Greiner, 1996; Domingo et al., 1999 do not cover such utility functions. Our algorithm requires a utility confidence interval specific for each class of utility functions. We presented such intervals for most of the popular functions in KDD and showed that there is no such interval for one class of functions. Another small but useful generalization is that our algorithm finds the n approximately best hypotheses which is often appropriate for knowledge discovery tasks.

Note that the presented algorithm differs quite strongly from the less practical algorithm presented by Scheffer and Wrobel (2000). Both the use of confidence intervals and the use of empirical variances in determining the individual intervals for each hypothesis are important extensions, and are crucial for achieving the kind of empirical results reported in Section 5.

Our algorithm represents all hypotheses in the current hypothesis space explicitly. In its current form, it it therefore only applicable when the hypothesis space is relatively small, as is the case, for instance, with association rule and subgroup discovery. Note, however, that the core of our algorithm is the sequential mechanism which it employs to determine the sample size that suffices to guarantee a high quality of the solution for the given problem. This mechanism can in principle also be combined with implicit representations of the hypothesis space (used in most machine learning algorithms). Future work should address this issue.

References

Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Verkamo, A. (1996). Fast discovery of association rules. *Advances in Knowledge Discovery and Data Mining*.

Ben-David, S., Kushilevitz, E., & Mansour, Y. (1997). Online learning versus offline learning. *Machine Learning*, *29*, 45.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Pacific Grove.

Cohn, D., Ghahramani, Z., & Jordan, M. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, *4*, 129–145.

Dodge, H., & Romig, H. (1929). A method of sampling inspection. *The Bell System Technical Journal*, *8*, 613–631.

Domingo, C., Gavelda, R., & Watanabe, O. (1999). *Adaptive sampling methods for scaling up knowledge discovery algorithms* (Technical Report TR-C131). Dept. de LSI, Politecnica de Catalunya.

Freund, Y. (1998). Self-bounding learning algorithms. *Proceedings of the International Workshop on Computational Learning Theory (COLT-98)*.

Greiner, R. (1996). PALO: A probabilistic hill-climbing algorithm. *Artificial Intelligence*, *83*.

Haussler, D. (1992). Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, *100*, 78–150.

Haussler, D., Kearns, M., Seung, S., & Tishby, N. (1996). Rigorous learning curve bounds from statistical mechanics. *Machine Learning*, *25*.

Klösgen, W. (1992). Problems in knowledge discovery in databases and their treatment in the statistics interpreter explora. *Journal of Intelligent Systems*, *7*, 649–673.

Klösgen, W. (1996). Explora: A multipattern and multistrategy discovery assistant. In *Advances in knowledge discovery and data mining*, 249–271. AAAI.

Langford, J., & McAllester, D. (2000). Computable shell decomposition bounds. *Proceedings of the International Conference on Computational Learning Theory*.

Maron, O., & Moore, A. (1994). Hoeffding races: Accelerating model selection search for classification and function approximating. *Advances in Neural Information Processing Systems* (pp. 59–66).

Piatetski-Shapiro, G. (1991). Discovery, analysis, and presentation of strong rules. *Knowledge Discovery in Databases* (pp. 229–248).

Scheffer, T., & Wrobel, S. (2000). A sequential sampling algorithm for a general class of utility functions. *Proceedings of the International Conference on Knowledge Discovery and Data Mining*.

Scheffer, T., & Wrobel, S. (2001). *Finding the most interesting patterns in a database quickly by using sequential sampling* (Technical Report). University of Magdeburg. http://kd.cs.uni-magdeburg.de/~scheffer/papers/.

Toivonen, H. (1996). Sampling large databases for association rules. *Proc. VLDB Conference*.

Uthurusamy, R., Fayyad, U., & Spangler, S. (1991). Learning useful rules from inconclusive data. *Knowledge Discovery in Databases* (pp. 141–158).

Wald, A. (1947). *Sequential analysis*. Wiley.

Wrobel, S. (1997). An algorithm for multi-relational discovery of subgroups. *Proc. First European Symposion on Principles of Data Mining and Knowledge Discovery (PKDD-97)* (pp. 78–87). Berlin.