

Outlier Detection by Sampling with Accuracy Guarantees *

Mingxi Wu
Department of Computer and Information
Sciences and Engineering
University of Florida
Gainesville, FL, USA, 32611
mwu@cise.ufl.edu

Christopher Jermaine
Department of Computer and Information
Sciences and Engineering
University of Florida
Gainesville, FL, USA, 32611
cjermain@cise.ufl.edu

ABSTRACT

An effective approach to detecting anomalous points in a data set is distance-based outlier detection. This paper describes a simple sampling algorithm to efficiently detect distance-based outliers in domains where each and every distance computation is very expensive. Unlike any existing algorithms, the sampling algorithm requires a fixed number of distance computations and can return good results with accuracy guarantees. The most computationally expensive aspect of estimating the accuracy of the result is sorting all of the distances computed by the sampling algorithm. The experimental study on two expensive domains as well as ten additional real-life data sets demonstrates both the efficiency and effectiveness of the sampling algorithm in comparison with the state-of-the-art algorithm and the reliability of the accuracy guarantees.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications — Data mining

Keywords: Outliers, distance-based, sampling algorithms, statistical modeling, approximate algorithms, nearest neighbor

1. INTRODUCTION

One effective approach for detecting anomalous data points in a set is *distance-based* (DB) outlier detection [6, 11, 1]. In DB-outlier detection, each data point is represented as a point in a multi-dimensional feature space and a distance function is chosen based on domain-specific requirements. Data points that are significantly far away from all of the others are flagged as outliers.

Though it may seem that DB-outlier detection is a solved problem, it turns out that even using advanced algorithms, the method is computationally prohibitive. This is particularly true in domains that require expensive distance functions. For example, the edit distance function for strings

*This paper is based upon work supported by the National Science Foundation under Grant No. 0347408.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.
Copyright 2006 ACM 1-59593-339-5/06/0008 ...\$5.00.

[12], the ERP distance function for time series [3], the quadratic-form distance function for color histograms [5] and various scoring matrices for aligning bioinformatics sequences [4] all are computationally expensive. Given two input points that are both of dimension n , the aforementioned distance functions require $\Theta(cn^2)$ time. In these domains, state-of-the-art algorithms may take days to detect DB-outliers, even in data sets of small sizes.

The goal of this paper is to define an algorithm that can provide users with interactive-speed performance over expensive domains. The question we consider is: How can we reduce the required number of distance computations in DB-outlier mining? For certain distance measures and data sets, indexing and pruning techniques can be used to reduce the number of distance computations. Unfortunately, indexing [2] is not useful in the domains mentioned above due to high data dimensionality, and pruning [1] tends only to reduce the required number of distance computations to a few hundred or a few thousand per data point; as we will show experimentally, this is still too costly in expensive domains.

In this paper, we consider a simple sampling algorithm for mining the k th nearest neighbor (k th-NN) DB-outliers [11]. Given integer parameters k and γ , the k th-NN outliers are the top γ points whose distance to its k th-NN is greatest. Sampling has been considered before as an option for detecting outliers [7], but never along with a rigorous treatment of the effect on result quality. If the user is able to tolerate some rigorously-measured inaccuracy, our algorithm can give arbitrarily fast response times.

Algorithm 1 Sampling Algorithm

Input: A data set G of size n ; k , specifying which NN distance as the criterion; α ($\alpha > k$), the sample size; γ , the number of outliers to return; a distance function

Output: γ points as the k th-NN outliers

```
1: for each point  $i \in G$  do  
2:   Draw a random sample of size  $\alpha$  from  $G$  (not including  
   point  $i$ )  
3:   Calculate point  $i$ 's  $k$ th-NN distance in its sample  
4: end for  
5: Return the top  $\gamma$  points whose  $k$ th-NN distance in its sample  
   is the greatest
```

Our algorithm (Algorithm 1) works as follows. For each data point i , we randomly sample α points from the data set. Using the user-specified distance function, we find the k th-NN distance of point i in those α samples. After repeating the above process for each point, the sampling algorithm returns the γ points whose sampled k th-NN distance

is greatest. Aside from the algorithm’s obvious simplicity, its biggest benefit is that it allows a user to control the total number of distance computations as $\Theta(\alpha n)$, thus bounding the time required to run the algorithm to completion.

To provide accuracy guarantees for Algorithm 1, we formally analyze the statistical properties of the algorithm, and describe an effective technique that gives the user a statistical indication of the algorithm’s result quality. Specifically, we treat the number of the true top γ *kth*-NN outliers in Algorithm 1’s return set as a random variable whose characteristics can be used to measure the quality of the outliers returned by the algorithm. Thus, if we tell the user that the expected number of true outliers returned is 20 for a given data set, then this implies that if the sampling algorithm were run many times on that data set, on average 20 out of the 30 returned points will be among the true top 30 *kth*-NN outliers. Finally, extensive experiments were performed in comparison with the state-of-the-art algorithm.

The remainder of the paper is organized as follows. Section 2 describes the overall process of providing quality guarantees for the sampling algorithm. In Section 3, a statistical analysis of the sampling algorithm is performed. Section 4 illustrates how the analysis can be applied to a constructed distance database efficiently. Section 5 details the experimental study. The paper is concluded in Section 6.

2. PROVIDING QUALITY GUARANTEES

Algorithm 1 (hereafter referred to as the “sampling algorithm”) is so simple that there is little benefit in discussing it in greater detail. However, the issue that clearly *does* deserve more attention is the question of how to give the user an understanding of exactly how accurate this algorithm is likely to be in practice; it is this question that we consider in detail in the remainder of the paper.

The paper describes a two-step process for analyzing the quality of the sampling algorithm, so that the estimated result quality can be returned to the user along with the discovered outliers. In the first step, the distances between each point and their samples are used to (logically) construct a distance database D' , where the set of pairwise distances in D' is a reasonable approximation of the actual distance database D . By *distance database*, we refer to a matrix that stores the pairwise distance from point i to point j for every i and j . During the construction of D' , we wish to avoid additional distance computations beyond those required by the sampling algorithm. Thus, for a data point i , the sampled distances $\{d'_1, d'_2, \dots, d'_\alpha\}$ between i and the other points in its sample are replicated as needed to serve as a surrogate for the actual neighbor distances $\{d_1, d_2, \dots, d_{n-1}\}$ of i . This reconstruction is identical to a type of reconstruction advocated when making use of the without-replacement bootstrap from statistics [10].

In the second step, an exact, statistical analysis of the quality of the sampling algorithm for the database D' is performed, and the result is returned to the user as an indication of the accuracy of the algorithm when it is applied to D . While it is true that bounds that are returned rely on the supposition that D' is in fact a reasonable surrogate for D , such assumptions are generally unavoidable in statistical analysis. An analogous assumption in classical sampling theory is that the sample variance can be used as a reasonable surrogate for the population variance. In order to address this concern, Section 5 shows that over twelve real-

life data sets, for various sample sizes, the bounds derived by our method do in fact reliably predict the accuracy of the sampling algorithm.

3. STATISTICAL ANALYSIS

This section describes a formal, probabilistic analysis of the quality of the sampling algorithm’s return set.

3.1 Quality Measure N

Our analysis needs to be performed with respect to some measure of the algorithm’s quality. Let A be the set of the true top γ *kth*-NN outliers, and let A' be the return set of the sampling algorithm. Then the size of the intersection $A \cap A'$ is a reasonable measure of the sampling algorithm’s quality. Let N be a random variable denoting the size of this intersection set for a single run of the sampling algorithm. To characterize the sampling algorithm’s quality, we characterize the expectation and variance of N (denoted by $E[N]$ and $Var(N)$ respectively). They describe the average number of correct outliers returned by the algorithm, as well as how much this number is expected to vary.

3.2 Average Number of Correct Outliers

We begin with a mathematical expression for N . Let y_i evaluate to one if the i th point in the data set G is a true *kth*-NN outlier, and zero otherwise. We define a series of Bernoulli (zero/one) random variables of the form M_i , where M_i is one if the i th point is declared as an outlier by the sampling algorithm, and zero otherwise. Then:

$$N = \sum_{i=1}^n y_i M_i \quad (1)$$

Taking the expectation of both sides, we have:

$$E[N] = \sum_{i=1}^n y_i E[M_i] \quad (2)$$

Note that for a given input of the sampling algorithm, y_i is a constant. Thus, deriving an expression for $E[N]$ reduces to the problem of deriving an expression for $E[M_i]$. Since M_i is a Bernoulli random variable, this is equivalent to computing the probability that M_i evaluates to one.

3.2.1 How Often Is Point i Declared an Outlier?

M_i evaluates to one if point i is reported as an outlier by the sampling algorithm. Obviously, point i is flagged as an outlier only if there are at most $\gamma - 1$ points in G whose *kth*-NN distance in its sample is larger than point i ’s. Let T_i be a random variable denoting the total number of such points. Then M_i is one if and only if $T_i \leq \gamma - 1$. Noting that T_i is asymptotically normally distributed¹, we have:

$$\begin{aligned} E[M_i] &= Pr[M_i = 1] \\ &= Pr[T_i \leq \gamma - 1] \\ &= Pr\left[\frac{T_i - E[T_i]}{\sqrt{Var(T_i)}} \leq \frac{\gamma - 1 - E[T_i]}{\sqrt{Var(T_i)}}\right] \\ &= \Phi\left(\frac{\gamma - 1 - E[T_i]}{\sqrt{Var(T_i)}}\right) \end{aligned} \quad (3)$$

¹In general, T_i is normally distributed due to the Lyapounov Central Limit Theorem. Since T_i is the sum of $n - 1$ independent Bernoulli random variables, this Theorem applies; see Lehmann [8], Corollary 2.7.1.

In Equation 3, $\Phi(x)$ is the standard normal cumulative distribution function. In order to make use of $\Phi(x)$, we need to calculate the mean and variance of T_i . Before we do this, for each point $i \in G$, we introduce a random variable N_i , which denotes the sampled k th-NN distance of point i . In addition, we define a function $one(expr)$ that evaluates to one if and only if the boolean-type argument $expr$ is true, and zero otherwise. Given these notations, $T_i = \sum_{j,j \neq i} one(N_j > N_i)$. If $D_i = \langle d_1, d_2, \dots, d_{n-1} \rangle$ refers to the vector of point i 's neighbor distances ordered from the smallest to the largest, then the following lemma gives the formulas for computing the mean and variance of T_i . This lemma provides the base formulas that we will use to compute $E[N]$ in Section 4.

Lemma 1. *The mean and variance of T_i are given by:*

$$E[T_i] = \sum_{d \in D_i} Pr[N_i = d] \sum_{j,j \neq i} Pr[N_j > N_i | N_i = d]$$

$$Var(T_i) = E[T_i] - E^2[T_i] + sum1 - sum2$$

In Lemma 1, $sum1$ and $sum2$ are given by:

$$sum1 = \sum_{d \in D_i} Pr[N_i = d] \left[\sum_{j,j \neq i} Pr[N_j > N_i | N_i = d] \right]^2$$

$$sum2 = \sum_{d \in D_i} Pr[N_i = d] \sum_{j,j \neq i} Pr^2[N_j > N_i | N_i = d]$$

We will not present the proof due to space limits.

3.2.2 Probability Formulas in Lemma 1

- *Computing $Pr[N_i = d]$.* Recall that steps (2)-(3) of the sampling algorithm draw a random sample (without replacement) of size α from point i 's $n - 1$ neighbors and compute i 's k th-NN distance in this sample. This process can be modelled by the hypergeometric distribution. In the hypergeometric distribution, a jar has N balls, M red, $N - M$ green. If a person were to reach in, blindfolded, and select K balls at random *without replacement*, the probability that exactly x balls of the K balls retrieved were red is given by $H(X = x | M, N, K)$. Analogously, suppose that point i 's q th-NN distance (denoted by d_q) is chosen to be N_i . Then we can regard the $q - 1$ NNs of point i as red balls, and those NNs who are further away than i 's q th-NN as green balls. Given this:

$$Pr[N_i = d_q] = \frac{\alpha}{n-1} H(X = k-1 | q-1, n-2, \alpha-1) \quad (4)$$

In Equation 4, $\frac{\alpha}{n-1}$ is the probability that the q th-NN of point i is included in the sample during a run of the sampling algorithm; k and α are the parameters of the sampling algorithm; n is the size of the input data set. Note that it is the index q that solely determines the probability of $N_i = d_q$, given n, α, k fixed.

- *Computing $Pr[N_j > N_i | N_i = d]$.* Computing this conditional probability is nothing but finding out all the distances in N_j 's domain that are greater than d , and then summing the probabilities of observing those distances. Given point j 's sorted vector of neighbor distances $D_j = \langle d_1, d_2, \dots, d_{n-1} \rangle$, suppose that d_{min_1} is

the smallest distance in this array that is larger than d . Then by making use of Equation 4:

$$Pr[N_j > N_i | N_i = d] = \sum_{q=min_1}^{n-1} Pr[N_j = d_q] \quad (5)$$

3.3 Variance In Correct Outliers

Knowing on expectation how many correct outliers in the return set of the sampling algorithm is not enough. It is crucial to help the user to understand how the number of correct outliers is going to vary around its mean. The variance of N gives such a measure. This quantity is $Var(N) = E[N^2] - E^2[N]$. Since we have already discussed how to compute $E[N]$ in Section 3.2, the remaining task is to calculate the second moment of N :

$$\begin{aligned} E[N^2] &= E\left[\left(\sum_i y_i M_i\right)^2\right] \\ &= E[N] + \sum_i \sum_{j,j \neq i} y_i y_j E[M_i M_j] \end{aligned} \quad (6)$$

Note that in Equation 6, y_i and y_j are constants for a given input of the sampling algorithm. Therefore, deriving an expression for $Var(N)$ reduces to deriving an expression for $E[M_i M_j]$. Since $M_i M_j$ is a Bernoulli random variable, this is equivalent to computing $Pr[M_i M_j = 1]$.

3.3.1 How Often Is $M_i M_j$ One?

$M_i M_j$ evaluates to one if and only if both point i and point j are reported as outliers by the sampling algorithm. Point i and point j are both flagged as outliers only if there are at most $\gamma - 2$ points in G whose sampled k th-NN distances are larger than the smaller of i 's and j 's. Otherwise, either point i or j (or both) will not be flagged as outliers. In a manner analogous to the way that we use T_i to help calculate $E[M_i]$, we will use U_{ij} to help calculate $E[M_i M_j]$. Let U_{ij} be a random variable denoting the total number of the points whose sampled k th-NN distance is larger than the smaller of i 's and j 's sampled k th-NN distance. Given this, $M_i M_j$ is one if and only if $U_{ij} \leq \gamma - 2$. Noting that U_{ij} is asymptotically normally distributed (for reasons identical to T_i 's normality), we have:

$$\begin{aligned} E[M_i M_j] &= Pr[M_i M_j = 1] \\ &= Pr[U_{ij} \leq \gamma - 2] \\ &= Pr\left[\frac{U_{ij} - E[U_{ij}]}{\sqrt{Var(U_{ij})}} \leq \frac{\gamma - 2 - E[U_{ij}]}{\sqrt{Var(U_{ij})}}\right] \\ &= \Phi\left(\frac{\gamma - 2 - E[U_{ij}]}{\sqrt{Var(U_{ij})}}\right) \end{aligned} \quad (7)$$

To make use of $\Phi(x)$, we need to calculate the mean and variance of U_{ij} . To express U_{ij} in a mathematical form, we again introduce a series of random variables, where S_{ij} denotes the smaller of point i 's and point j 's k th-NN distances in their samples. Obviously, the domain of S_{ij} is $D_i \cup D_j$. Given these notations, we have $U_{ij} = \sum_{l,l \neq i,j} one(N_l > S_{ij})$. Lemma 2 gives the formulas for computing the mean and variance of U_{ij} . It provides the base formulas for computing $Var(N)$ in Section 4.

Lemma 2. The mean and variance of U_{ij} are given by:

$$E[U_{ij}] = \sum_{d \in D_i \cup D_j} Pr[S_{ij} = d] \sum_{l, l \neq i, j} Pr[N_l > S_{ij} | S_{ij} = d]$$

$$Var(U_{ij}) = E[U_{ij}] - E^2[U_{ij}] + sum3 - sum4$$

In Lemma 2, $sum3$ and $sum4$ are given by:

$$sum3 = \sum_{d \in D_i \cup D_j} Pr[S_{ij} = d] \left[\sum_{l, l \neq i, j} Pr[N_l > S_{ij} | S_{ij} = d] \right]^2$$

$$sum4 = \sum_{d \in D_i \cup D_j} Pr[S_{ij} = d] \sum_{l, l \neq i, j} Pr^2[N_l > S_{ij} | S_{ij} = d]$$

Again, we will not present the proof due to space limits.

3.3.2 Probability Formulas in Lemma 2

- *Computing $Pr[S_{ij} = d]$.* Since S_{ij} 's domain is $D_i \cup D_j$, we may use the conditional probability definition to compute the probability of $S_{ij} = d$.

When $d \in D_i$:

$$Pr[S_{ij} = d] = Pr[N_i = d] \times Pr[N_j > N_i | N_i = d] \quad (8)$$

When $d \in D_j$:

$$Pr[S_{ij} = d] = Pr[N_j = d] \times Pr[N_i > N_j | N_j = d] \quad (9)$$

For each $d \in D_i \cup D_j$, we can use Equation 4 and 5 and Equation 8 and 9 to compute $Pr[S_{ij} = d]$.

- *Computing $Pr[N_l > S_{ij} | S_{ij} = d]$.* This conditional probability can be computed as we did in Equation 5. Given point l 's sorted vector of neighbor distances $D_l = \langle d_1, d_2, \dots, d_{n-1} \rangle$, suppose that d_{min_2} is the smallest distance in this array that is larger than d , then by making use of Equation 4:

$$Pr[N_l > S_{ij} | S_{ij} = d] = \sum_{q=min_2}^{n-1} Pr[N_l = d_q] \quad (10)$$

4. SPEEDING UP THE COMPUTATION

Equipped with Section 3's theoretical foundation, we now follow the two steps discussed in Section 2 to perform the estimation on a constructed distance database.

4.1 Constructing the Distance Database D'

Since we wish to avoid new distance computations in the construction of D' , one strategy is to uniformly replicate the distances computed by the sampling algorithm to create a full-size distance database that can subsequently be analyzed. We describe such a construction with an example, illustrated in Figure 1. On the left-hand side of Figure 1, a two-dimensional matrix is used to represent the distance database, where each column of the matrix contains the sorted distances from a given point to all of the other data points. Given this data, our sampling algorithm randomly selects two neighbor distances for each point of the input data set. As a result, we obtain the sampled distance matrix shown in the middle of Figure 1. In order to build a complete distance matrix D' by making use of the sampled

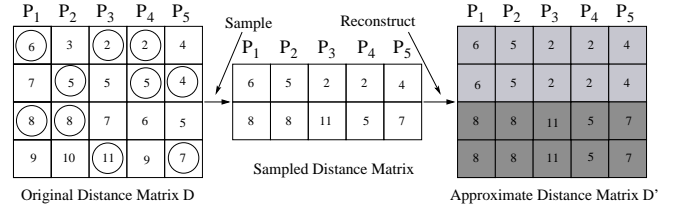


Figure 1: Replicating sample distances to construct D' .

distance matrix, we replicate each row of the sampled distance matrix twice, which gives us the approximate distance matrix D' shown on the right-hand side of Figure 1. Note that no additional distance computations are performed.

In general, the process of constructing D' uses $\frac{n-1}{\alpha}$ replications of each distance computed by the original sampling algorithm. As a result, we regard each column D'_i in D' as an approximation of its corresponding column D_i in D .

Algorithm 2 EN Algorithm

Input: the sample distance array SD ; Algorithm 1's parameters
Output: the $E[N]$ for D'

- 1: Sort SD ascendingly
 - 2: **for** $j = \alpha n$ to 1 /*backward scan SD */ **do**
 - 3: **if** $SD[j]$ is a sample distance from an outlier column D'_i in D' **then**
 - 4: Follow Lemma 1 to update $E[T_i]$ and the $sum1$ and $sum2$ of $Var(T_i)$ by making use of the sufficient statistics maintained during the backward scan of SD
 - 5: **end if**
 - 6: Update the sufficient statistics
 - 7: **end for**
 - 8: Apply Lemma 1 and Equation 3 to calculate $E[M_i]$ for each outlier i , then sum them and return the sum as $E[N]$ for D'
-

4.2 Computing $E[N]$ and $Var(N)$ for D'

In this section, we show efficient algorithms that can obtain $E[N]$ and $Var(N)$ for D' in $\Theta(\alpha n \log(\alpha n))$ time.

To compute $E[N]$, the main task is to compute $E[T_i]$ and $Var(T_i)$ for each outlier column i in D' (those with the γ largest $D'_i[k]$ values in D').

To calculate $E[T_i]$, the following two observations of Equation 4 and the $E[T_i]$ formula in Lemma 1 are critical:

1. Each distance $d \in D'$ is associated with a fixed probability $Pr[N_i = d]$ which is determined by d 's row position in D' .
2. The main task of computing $\sum_{j, j \neq i} Pr[N_j > N_i | N_i = d]$ is nothing but finding out all of those distances in D' that are larger than d and from a column other than column i , then summing the probabilities associated with those distances.

Furthermore, notice that for a given column D'_i of D' , the $\frac{n-1}{\alpha}$ rows resulting from a single sample distance can be represented by one row, if the probability associated with the representative row is the sum of the probabilities associated with its $\frac{n-1}{\alpha}$ replications. With this compression, for each distance d in column i , we must find all of those sample distances that are not from column i and are greater than d . A straightforward way to do this is to sort all of the

sample distances and scan the sorted distance array from back to front. During the scan, we maintain sufficient statistics so that when we encounter a sample distance $d \in D'_i$, $Pr[N_i = d] \times \sum_{j, j \neq i} Pr[N_j > N_i | N_i = d]$ can be computed. We find that the following statistics are sufficient for this purpose: (1) the sum of the probabilities associated with each sample distance that has been checked; (2) how many sample distances from column i have been passed. Then we can update $E[T_i]$ by following Lemma 1 during the backward scan. This requires $O(n)$ time. A similar idea applies to the calculation of $sum1$ and $sum2$ in Lemma 1. Therefore, $Var(T_i)$ requires $O(n)$ time too, provided the sample distance array is sorted (this setup requires $\Theta(\alpha n \log(\alpha n))$ time).

Algorithm 2 (hereafter referred to as the EN algorithm) presents the pseudo code for computing $E[N]$ on D' . Since Lemma 1 and Lemma 2 have similar structures, we can calculate $E[U_{ij}]$ and $Var(U_{ij})$ according to Lemma 2 similarly by maintaining some sufficient statistics during the backward scan of the sorted sample distance array. After obtaining $E[U_{ij}]$ and $Var(U_{ij})$, computing $Var(N)$ is trivial. We call the algorithm to compute $Var(N)$ for D' as the $VarN$ algorithm.

5. EMPIRICAL EVALUATION

This section details the results of two sets of experiments designed to test our methods.

5.1 Testing Expensive Distance Functions

The first set of experiments has two goals. First, we wish to test whether our algorithms are able to return a high-quality answer set in an acceptably short time in a domain requiring an expensive distance function. Second, we wish to check whether a representative, state-of-the-art algorithm for this purpose (Bay’s nested loop algorithm [1]) can also provide acceptable performance.

To accomplish these goals, the first task we consider is detecting outlier nucleotide sequences in human chromosome 18 downloaded from NCBI ftp site. This data set was created by randomly selecting 4000 non-overlapping subsequences of length 2000 from human chromosome 18 (Chr18). We employed Needleman-Wunsch algorithm [9] to compute the Edit distance between two subsequences. The second task we consider is detecting outlier images in UCID version 2 [13], which is an image database containing 1338 uncompressed color images. We transformed each image to a 576 dimensional color histogram. The quadratic distance between two histograms was used [5].

The experiments were performed on a Linux workstation having an Intel Xeon 2.4GHz Processor with 1GB of RAM. All the algorithms were implemented in C++ and compiled with gcc version 4.02. Since we were interested in testing the scenarios that the distance computations dominate all the other costs, we loaded the entire data set into memory for all the algorithms that we tested. We report the wall clock time here. All experiments were run to return the top 30 5th-NN outliers.

We began our experiments by running Bay’s nested loop algorithm over both data sets. We recorded the time required, as well as the average distance computations per data point. The results are reported in Table 1. Next, we ran our implementations of the sampling algorithm and the EN and $VarN$ algorithms over the same data, and then in-

Data Set	Cont./Feature	Size	BA
Wisconsin cancer	30/31	569	165
Balance scale	4/4	625	625
Florida farm	188/188	811	140
California farm	188/188	1,373	175
FCAT Read	27/27	1,404	275
FCAT Math	28/28	1,405	274
California house	9/9	20,640	654
Baseball pitching	22/22	36,245	454
Baseball batting	17/17	85,979	745
Cover Type	10/55	581,012	4527

Table 3: Description of the 10 real data sets and the average number of distance computations per data point (denoted by BA) by Bay’s algorithm. Cont./Feature means the number of continuous features over the number of total features.

tersected the result set of our sampling algorithm with the result set of Bay’s algorithm to get the observed N . Meanwhile, we recorded the bound calculated by $E[N] - \sigma$, where $E[N]$ is the estimate of the expected value of N , and σ (the standard deviation of N) is the square root of the estimate of the variance of N . For each data set, we ran 10 trials with the sample size α set to 10. The performance results are also shown in Table 1.

Discussion. Even though the two data sets have just a few thousand points, Bay’s algorithm had relatively poor performance on both, taking nearly one day for the first one. In contrast, our algorithms did an excellent job in these two domains. Our algorithms provided between one and two orders of magnitude speedup, and still maintained relatively high result quality. The sampling algorithm returned more than half of the total true 5th-NN outliers in both cases (and nearly all of them in the UCID case) using only ten samples per data set point. Furthermore, in seven of the ten trials over the Chr18 data set and in ten of the ten trials over the UCID data set, the actual number of outliers returned either exceeded or was almost equal to $E[N] - \sigma$. This indicates that (at least for these two data sets), the reported $E[N] - \sigma$ constitutes a safe lower bound on qualitative performance.

5.2 Reliability of the Estimation

This subsection describes an additional experimental evaluation of our algorithms, aimed at evaluating the reliability of our estimation algorithms. We wish to obtain some experimental evidence that the results given by our estimation algorithms are reliable with various sample sizes and different sizes of data sets.

The test was designed as follows. We selected 10 real data sets with different characteristics summarized in Table 3. The experimental setup was identical to what we described in the previous subsection. We processed the data by normalizing all continuous variables to the range [0,1] and converting all categorical variables to an integer representation. Hamming distance was used for categorical features and Euclidean distance was used for continuous features. The average number of distance computations per data point required by Bay’s algorithm are reported in the last column of Table 3. For each of the 10 real data sets, we systematically tested the paper’s algorithms with the sample

Algorithm	Chr18		UCID	
	Distance computations/point	Time	Distance computations/point	Time
Bay’s algorithm	331	24h:9m:31s	180	3h:48m:26s
Sampling algorithm	10	47m:24s	10	14m:7s
Ratios	33.1	30.6	18	16.2

Table 1: Efficiency comparisons between Bay’s algorithm and the sampling algorithm on the Chr18 and the UCID data sets. The time reported for the sampling algorithm includes the running time of both the sampling algorithm and the EN and VarN algorithms.

Data set	$\alpha = 10$			$\alpha = 60$			$\alpha = 110$		
	observed N	$E[N]$	σ	observed N	$E[N]$	σ	observed N	$E[N]$	σ
Wisconsin cancer	16.60	13.29	3.55	24.60	22.20	0.00	25.60	23.12	0.00
Balance scale	8.60	5.17	5.29	8.60	6.46	5.75	7.60	5.70	5.77
Florida farm	27.10	27.09	0.27	28.20	26.90	0.86	28.10	26.48	0.71
California farm	22.70	24.59	0.73	24.40	25.15	0.00	25.60	25.12	0.00
FCAT Read	15.90	13.86	3.75	19.90	18.48	1.43	19.50	17.76	2.28
FCAT Math	12.90	12.69	4.40	18.70	17.14	2.11	18.50	17.94	1.46
California house	10.40	14.30	5.09	11.30	17.94	3.00	12.10	18.40	2.86
Baseball pitching	14.70	17.04	2.32	14.70	19.01	1.32	14.10	19.05	0.00
Baseball batting	6.10	4.19	5.57	8.50	7.30	5.99	11.40	9.67	5.42
Cover Type	0.10	3.24	6.34	0.00	2.98	5.17	0.00	2.67	5.30

Table 2: The average estimation results and the average empirical results of N for 10 trials on each of the 10 real data sets. The sample sizes α is set to 10, 60 and 110 respectively.

size α set to 10, 60 and 110 respectively. For each experiment, we performed 10 trials. The average statistics of the 10 trials are reported in Table 2.

Discussion. We observed high reliability for our estimation algorithms, indicating that the constructed distance database D' is a reasonable surrogate for the original distance database D . Specifically, for the 300 totals runs that were performed in order to construct Table 2, the actual number of true 5th-NN outliers returned was larger than $E[N]$ 199/300 times, larger than $(E[N] - \sigma)$ 236/300 times, and larger than $(E[N] - 2\sigma)$ 253/300 times. The table depicts in detail the close correlation between the predicted $E[N]$ and the observed average N . Again, this shows the general utility of our analysis for predicting the accuracy of the algorithm.

6. CONCLUSIONS

We have considered the problem of how to efficiently detect DB-outliers when the distance function is expensive. A simple sampling algorithm requiring a fixed number of distance computations is proposed and the statistical characteristics of this sampling algorithm is formally analyzed. Based on the analysis, two estimation algorithms are proposed, requiring only sorting time of the sampled distances. As a result, this paper provides a practical tool to explore DB-outliers in expensive domains.

7. REFERENCES

- [1] S. D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *SIGKDD*, pages 29–38, 2003.
- [2] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *SIGMOD*, pages 93–104, May 2000.
- [3] L. Chen and R. T. Ng. On the marriage of lp-norms and edit distance. In *VLDB*, Aug 2004.
- [4] M. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, 5:345–352, 1978.
- [5] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, 1994.
- [6] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB Journal*, 8(3-4):237–253, February 2000.
- [7] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold. Efficient biased sampling for approximate clustering and outlier detection in large datasets. *IEEE TKDE*, 15(5), 2003.
- [8] E. Lehmann. *Elements of Large-Sample Theory*. Springer, 1998.
- [9] S. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *JMB*, 48:443–453, 1970.
- [10] B. Presnell and J. Booth. Resampling methods for sample surveys. *Technical Report 470, Department of Statistics, University of Florida*, 1994.
- [11] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *SIGMOD*, pages 427–438, May 2000.
- [12] E. S. Ristad and P. N. Yianilos. Learning string-edit distance. *IEEE TPAMI*, 20(5), 1998.
- [13] G. Schaefer and M. Stich. Ucid - an uncompressed colour image database. In *Proc. of SPIE*, 2004.