Efficient Biased Sampling for Approximate Clustering and Outlier Detection in Large Data Sets

George Kollios, *Member*, *IEEE Computer Society*, Dimitrios Gunopulos, Nick Koudas, and Stefan Berchtold, *Member*, *IEEE*

Abstract—We investigate the use of biased sampling according to the density of the data set to speed up the operation of general data mining tasks, such as clustering and outlier detection in large multidimensional data sets. In density-biased sampling, the probability that a given point will be included in the sample depends on the local density of the data set. We propose a general technique for density-biased sampling that can factor in user requirements to sample for properties of interest and can be tuned for specific data mining tasks. This allows great flexibility and improved accuracy of the results over simple random sampling. We describe our approach in detail, we analytically evaluate it, and show how it can be optimized for approximate clustering and outlier detection. Finally, we present a thorough experimental evaluation of the proposed method, applying density-biased sampling on real and synthetic data sets, and employing clustering and outlier detection algorithms, thus highlighting the utility of our approach.

Index Terms—Data mining, sampling, clustering, outlier detection.

1 Introduction

Clustering and detecting outliers in large multidimensional data sets are important data analysis tasks. A variety of algorithmic techniques have been proposed for their solution. The inherent complexity of both problems however is high and the application of known techniques on large data sets is time and resource consuming. Many data intensive organizations, such as AT&T, collect lots of different data sets of cumulative size in the order of hundreds of gigabytes over specific time frames. Application of known techniques to analyze the data collected even over a single time frame would require a large resource commitment and would be time consuming. Additionally, from a data analysis point of view not all data sets are of equal importance. Some data sets might contain clusters or outliers, others might not. To date, one has no way of knowing that unless one resorts to executing the appropriate algorithms on the data. A powerful paradigm to improve the efficiency of a given data analysis task is to reduce the size of the data set (often called *Data Reduction* [4]), while keeping the accuracy loss as small as possible. The difficulty lies in designing the appropriate approximation technique for the specific task and data sets.

- *G. Kollios is with the Department of Computer Science, Boston University, Boston, MA 02215. E-mail: gkollios@cs.bu.edu.*
- *D. Gunopulos is with the Department of Computer Science and Engineering, University of California at Riverside, Riverside, CA 92521. E-mail: dg@cs.ucr.edu.*
- *N. Koudas is with AT&T Labs Research, Shannon Laboratory, Florham Park, NJ 07932. E-mail: koudas@research.att.com.*
- *S. Berchtold is with stb software technologie beratung gmbh, 86150 Ausburg, Germany. E-mail: Stefan.berchtold@stb-gmbh.de.*

Manuscript received 13 Feb. 2001; revised 11 Oct. 2001; accepted 19 Nov. 2001.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 113620.

In this paper, we propose using *biased sampling* as a data reduction technique to speed up the operations of cluster and outlier detection on large data set collections. Biased sampling is a generalization of uniform random sampling, which has been used extensively to speed up the execution of data mining tasks. In uniform random sampling, every point has the same probability to be included in the sample [37]. In contrast, in *biased* sampling, the probability that a data point is added to the sample is different from point to point. In our case, this probability depends on the value of prespecified data characteristics and the specific analysis requirements. Once we derive a sampling probability distribution that takes into account the data characteristics and the analysis objectives, we can extract a biased sample from the data set and apply standard algorithmic techniques on the sample. This allows the user to find approximate solutions to different analysis task and gives a quick way to decide if the data set is worthy of further exploration.

Taking into account the data characteristics and the analysis objectives in the sampling process results in a technique that can be significantly more effective than uniform sampling. We present a technique that performs these tasks accurately and efficiently.

Our main observation is that the density of the data set contains enough information to derive the desired sampling probability distribution. Identifying clusters benefits from a higher sampling rate in areas with high density. For determining outliers, we increase the sampling probability in areas with low density. For example, consider a data set of postal addresses in the north east US consisting of $130,000$ points (Fig. 1a). In this data set, there are three large clusters that correspond to the three largest metropolitan areas, New York, Philadelphia, and Boston. There is also a lot of noise, in the form of uniformly distributed rural areas

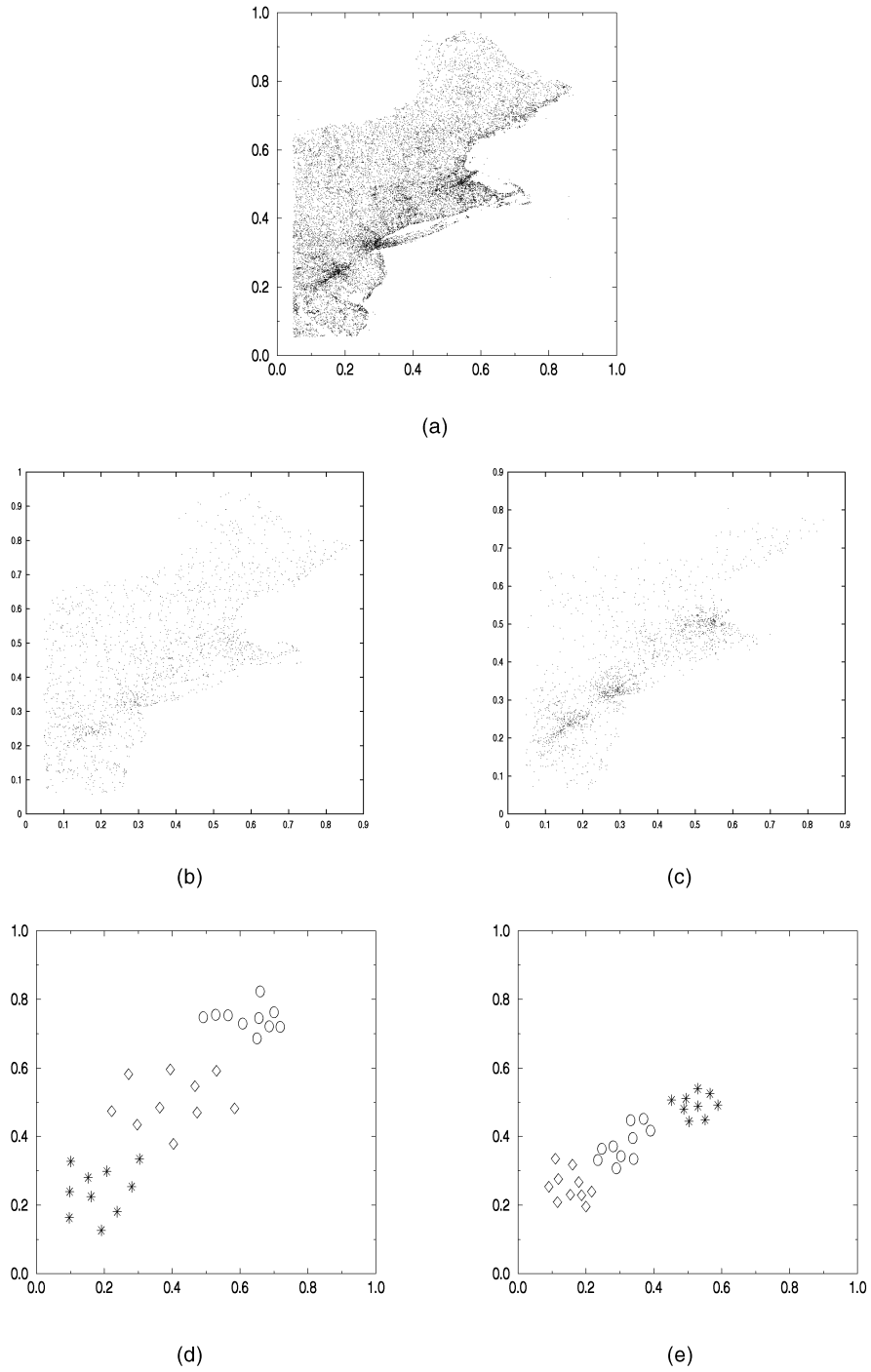


Fig. 1. (a) A data set of postal addresses ($130,000$ points) in the north east US. (b) Random sample of 1,000 points. (c) A biased sample based on data set density. (d) Clustering of (b) using a hierarchical algorithm and 10 representative points. (e) Clustering of (c) using a hierarchical algorithm and 10 representative points.

and smaller population centers. Fig. 1b presents the result of a random sample of size 1,000 points taken from this data set. Fig. 1c presents the result of a 1,000 point sample after we impose a higher sampling rate in areas of high density. We run a hierarchical clustering algorithm on both samples. Fig. 1d shows 10 representative points for each of the clusters found on the sample of Fig. 1b. Fig. 1e shows 10 representative points for each of the clusters found on the sample of Fig. 1c. The clusters found in Fig. 1e are exactly those present in the original data set. In contrast, in Fig. 1d, two of the clusters have been merged and a spurious third cluster has been introduced. The above example shows that sampling according to density can make a difference and be in fact more accurate for cluster detection than uniform sampling.

1.1 Our Contribution

We propose a new, flexible, and tunable technique to perform density-biased sampling. In this technique, we calculate, for each point, the density of the local space

around the point and we put the point into the sample with probability that is a function of this local density. For succinct (but approximate) representation of the density of a data set, we choose kernel density estimation methods. We apply our technique to the problems of cluster and outlier detection and report our experimental results. Our approach has the following benefits:

- **Flexibility**. Depending on the data mining technique, we can choose to sample at a higher rate the denser regions or the sparser regions. To identify the clusters when the level of noise is high or to identify and collect reliable statistics for the large clusters, we can oversample the dense regions. To identify all possible clusters, we can oversample the sparser regions. It now becomes much more likely to identify sparser or smaller clusters. On the other hand, we can make sure that denser regions in the original data set continue to be denser in the sample so that we do not lose any of the dense clusters. To identify outliers, we can sample the very sparse regions.
- **Effectiveness**. Density-biased sampling can be significantly more effective than random sampling as a data reduction technique.
- **Efficiency**. It makes use of efficient density estimation techniques (requiring one data set pass) and requires one or two additional passes to collect the biased sample.

Section 2 reviews related work. In Section 3, we analytically evaluate the benefits of the proposed approach over random sampling. Section 4 presents efficient techniques to estimate the density of a data set, followed by Section 5, which presents the details of our approach. In Section 7, we discuss the issues involved when combining our sampling technique with clustering and outlier detection algorithms in order to speed up the corresponding tasks. In Section 8, we present the results of a detailed experimental evaluation using real and synthetic data sets, analyzing the benefits of our approach for various parameters of interest. Finally, Section 9 concludes the paper and points to problems of interest for further study.

2 Related Work

Clustering is a descriptive task that seeks to identify homogeneous groups of objects based on the values of their attributes (dimensions). The groups may be mutually exclusive and exhaustive or consist of a richer representation such as hierarchical or overlapping groupings.

Current clustering techniques can be broadly classified into two categories: *partitional* and *hierarchical*. Given a set of objects and an objective function, partitional clustering obtains a partition of the objects into clusters such that overall the objects in the clusters commonly minimize the objective function. The popular K-means and K-medoid methods determine K cluster representatives and assign each object to the cluster that its representative is closest to the object. These techniques attempt to minimize the sum of the distances squared between the objects and their representatives (other metrics, such as the minimum of the distances is minimized, etc. are possible) over all choices of cluster representatives. CLARANS [28], EM [13], [5], and BIRCH [43] can be viewed as extensions of this approach to work against large databases. Both BIRCH [43] and EM [5] use data summarization to minimize the size of the data set. In BIRCH, for example, a structure called the CF-tree is built after an initial random sampling step. The leaves store statistical representations of space regions. Each new point is either inserted in an existing leaf or a new leaf is created to specifically store it. In EM [5], the data summarization is performed by fitting a number of Gaussians to the data. CLARANS uses uniform sampling to derive initial candidate locations for the clusters.

Very recently, efficient approximation algorithms for the Minimum Spanning Tree [9], k-medians [20], and 2-Clustering [19] where presented. All these algorithms can be used as building blocks for the design of efficient sampling-based algorithms for a variety of clustering problems. The approach presented herein can be utilized by these algorithms for the extraction of a sample that is better than the uniform random sample they extract.

Mode-seeking clustering methods identify clusters by searching for regions in the data space in which the object density is large. Each dense region, called mode, is associated with a cluster center and each object is assigned to the cluster with the closest center. DBSCAN [14] finds dense regions that are separated by low density regions and clusters together the objects in the same dense region. CLIQUE [1] automatically discovers those subspaces (projections into a subset of dimensions) that contain high-density clusters in addition to finding the clusters themselves.

A hierarchical clustering is a nested sequence of partitions. An agglomerative, hierarchical clustering starts by placing each object in its own cluster and then merges these atomic clusters into larger and larger clusters until all objects are in a single cluster. The recently proposed CURE [17] clustering algorithm is a hierarchical technique that also uses uniform random sampling of the data set to speed up the execution.

The notion of the outlier has various interpretations in different scientific communities. In statistics [32], [8], outliers are data points that deviate from a specific model assumption. The notion of distance-based outliers was recently introduced in databases [23], [24], [33]. According to this notion, a point, P, in a multidimensional data set is an outlier if there are less than (a user-specified constant) p points from the data in the ϵ-neighborhood of P. A related notion of outliers was introduced in [6]. Deviant points [21] are special forms of outliers.

Sampling [39], [40] is a well-recognized and widely used statistical technique. In the context of clustering in databases, both partitional [28] and hierarchical [17] clustering techniques employ uniform random sampling to pick a set of points and guide the subsequent clustering effort. In data mining, Toivonen [38] examined the problem of using sampling during the discovery of association rules. Sampling has also been recently successfully applied in query optimization [11], [10], as well as approximate query answering [16], [2].

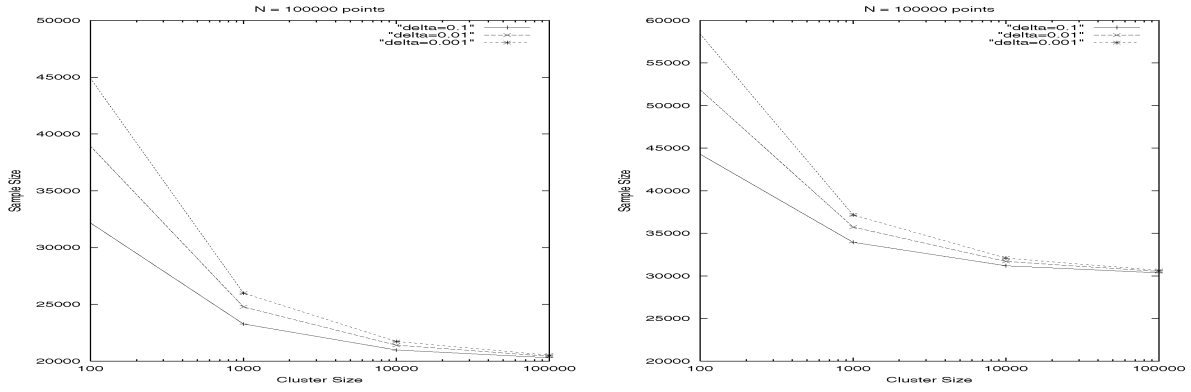


Fig. 2. Required sample sizes to guarantee that a fraction ϕ of the cluster is in the sample for various δs. (a) $\phi = 0.2$. (b) $\phi = 0.3$.

Independently, Palmer and Faloutsos [30] developed an algorithm to sample for clusters by using density information under the assumption that clusters have a zipfian distribution. Their technique is designed to find clusters when they differ a lot in size and density, and there is no noise. Their approach tightly couples the extraction of density information and biased sampling based on a hash-based approach, and the quality of the sample degrades with collisions implicit to any hash-based approach. The approach presented herein decouples density estimation and biased sampling. Our use of kernels for density estimation is orthogonal to our framework, which is independent on any assumptions about cluster distribution. Any density estimation technique can be used instead and we suggest a hash free approach to eliminate the impact of collisions to density estimation. In Section 8, we present an experimental comparison of our method against the approach by Palmer and Faloutsos.

3 Biased Sampling

The focus of this paper is in the design of alternative forms of sampling to expedite the process of cluster and outlier detection. More specifically, we are interested in designing sampling techniques that are biased either toward clusters or outlier points. Let D be a data set; we are interested in a sampling technique that can be suitably tuned by the user to meet one of the following objectives:

Objective 1. *If u is a cluster in D, then, with very high probability, a* very large fraction *of the points in u will be included in the sample.*

Objective 2. *If p is an outlier point in D, then, with very high probability, p will be included in the sample.*

3.1 The Problem with Uniform Random Sampling

A natural question arises regarding the potential benefits of such sampling techniques. In other words, why is uniform random sampling not sufficient for such a purpose? Consider a data set D of size n which is a single cluster, that is, all n points are members of the cluster. In this case, by employing uniform random sampling, we will do an excellent job in identifying the cluster; the fraction of points in the sample belonging to the cluster is one. In addition, this happens with probability one. In such cases, one expects, for a fixed sample size, the distribution of the points in the sample to precisely track the point distribution in D.

Consider D again, but this time let us remove points from the cluster and introduce them as Gaussian noise. Although the size of the data set remains the same, intuitively we expect that a larger sample size would be needed in this case in order to guarantee that the same fraction of points in the sample belong to the cluster with the same probability as before. In a sense, every noise point included in the sample penalizes the sampling technique; an additional sample in the cluster for every noise point is required to achieve the same fraction of cluster points in the sample. We formalize these intuitions in the sequel.

Guha et al. [17] presented a theorem linking the sample size with the probability that a fraction of the cluster is included in the sample. Let D be a data set of size n, and u a cluster of size $|u|$. Guha et al. consider a cluster u to be included in the sample when more than $\phi * |u|$ points from the cluster are in the sample for $0 \leq \phi \leq 1$. Then, the sample size s, required by uniform random sampling to guarantee that u is included in the sample with probability no less than δ, $0 \leq \delta \leq 1$ is:

$$s \geq \phi * n + \frac{n}{|u|}\log\left(\frac{1}{\delta}\right) + \frac{n}{|u|}a\sqrt{\left(\log\left(\frac{1}{\delta}\right)\right)^2 + 2 * \phi * |u| * \log\left(\frac{1}{\delta}\right)}. \quad (1)$$

Let us obtain some quantitative feel about the implications of this equation in practice. Consider a database of $n = 100,000$ points. Let us use (1) and examine the sensitivity of s to various parameters. Figs. 2a and 2b presents the minimum sample size required to guarantee that a fraction of $\phi = 0.2$ ($\phi = 0.3$) of point in $|u|$ is included in the sample for a range of probabilities, for $|u|$ ranging from n to $100,000$ points.

It is evident from Figs. 2a and 2b that, using uniform random sampling, we cannot guarantee with high

probability that a large fraction of a small cluster is included in the sample, unless we use very large samples. For example, in order to guarantee with probability 90 percent that a fraction $\phi = 0.2$ of a cluster with 1,000 points is in the sample, we need to sample 25 percent of the data set. As the fraction of the cluster required in the sample increases to 0.3 (Fig. 2b), the required sample size increases to almost 40 percent of the data base size.

Clearly, one must be willing to include a very large fraction of the data set in the sample in order to provide good guarantees with uniform random sampling. This requirement, however, defies the motivation for sampling. If the decision of whether a data set should be explored further for clusters or outliers is based on the quality of the sample, then high confidence and large representation of the cluster in the sample is what is needed.

3.2 Advantages of Biased Sampling

Instead of employing uniform sampling, let us assume that we sample according to the following rule: Let $\pi_1, \ldots, \pi_n$ be the points in the data set. Let u be a cluster. Assume that we had the ability to bias our sampling process according to the following rule, R:

$$P(\pi_I \; is \; included \; in \; the \; sample) = \begin{cases} p \; if \; \pi_i \in u, \\ 1 - p \; otherwise \end{cases} \quad (2)$$

for $0 \leq p \leq 1$. We state and prove the following theorem:

Theorem 1. *Let D be a data set of size n, u a cluster of size $|u|$. A cluster u is included in the sample when more than $\phi * |u|$ points from the cluster are in the sample for $0 \leq \phi \leq 1$. Let s_R be the sample size required by sampling according to R to guarantee that u is included in the sample with probability more than $1 - \delta$, $0 \leq \delta \leq 1$, and s the sample size required by uniform random sampling to provide the same guarantee. Then,*

$$s_R \leq s \text{ if } p \geq \frac{|u|}{n}. \quad (3)$$

Proof. Let $D = \{\pi_1, \ldots \pi_N\}$ be a data set of N points, u a cluster of size $|u|$, and assume we are sampling according to the following rule R:

$$P(\pi_I \; is \; included \; in \; the \; sample) = \begin{cases} p \; if \; \pi_i \in u, \\ 1 - p \; otherwise. \end{cases} \quad (4)$$

Let X_i be a random variable with value 1 if the ith sampled point belongs to the cluster (with probability p) and 0 otherwise (probability $1 - p$). Let X be a random variable containing the sum of the X_is after s_R sampling steps, thus $X = X_1 + X_2 + \ldots + X_{s_R}$, where the X_is, $1 \leq i \leq s_R$, are independent identically distributed random variables. By sampling s_R points according to rule R, the expected number of cluster points in the sample is:

$$\mu = \sum_{i=1}^{s_R} X_i = s_R * p. \quad (5)$$

We wish to compute the probability that, after s_R sampling steps, $\phi * |u|$, $0 \leq \phi \leq 1$, points from the cluster are in the sample with probability $\delta, 0 \leq \delta \leq 1$. Thus, we wish:

$$P(X < \phi * |u|) < \delta. \quad (6)$$

Equation (6) can be rewritten as:

$$P\left(X < \left(1 - \left(1 - \frac{\phi * |u|}{\mu}\right)\right) * \mu\right) < \delta. \quad (7)$$

By applying Chernoff bounds, we have:

$$P(X < (1 - \epsilon)\mu) < e^{\frac{\mu * (1-\epsilon)^2}{2}}. \quad (8)$$

Using (7) and (8), after algebraic manipulation, we get:

$$s_R > \frac{\phi * |u| + \log\left(\frac{1}{\delta}\right) + \sqrt{\left(\log\frac{1}{\delta}\right)^2 + 2 * \phi * |u| * \log\left(\frac{1}{\delta}\right)}}{p}. \quad (9)$$

Contrasting (9) and (1), the result follows. □

Theorem 1 has a very intuitive explanation: Points in a cluster are included with higher probability, thus achieving the requirement that a fraction ϕ of cluster points are included in the sample much sooner. Theorem 1 provides a means to overcome the limitations of uniform random sampling since we can assure, with the same probability, that the same fraction of the cluster is in the sample by taking a sample of smaller size. The serious limitation, however, is that we have no clear way to bias our sampling toward points in the clusters since we do not know these points a priori.

We observe that knowledge of the probability density function of the underlying data set can provide enough information to locate those points. Clusters are located in dense areas of the data set and outliers in sparse areas. By adapting our sampling, according to the density of the data set we can bias our sampling probabilities toward dense areas in the case of clusters and sparse areas in the case of outliers. All that is needed is a mapping function that maps density to sampling probability in a way that depends on the objective of our sampling process. For the case of clusters, this function can be as simple as sampling according to the probability density function of the underlying data set. For outliers, it can be as simple as sampling according to the inverse of this function. Since the sample is biased toward specific areas, it has the potential due to Theorem 1 to require smaller sample sizes to give certain guarantees.

The viability of such an approach depends on our ability to derive quick estimates of the density function from the underlying data set. This is the topic of the next section.

4 Density Estimation Techniques

A nonparametric density estimator technique attempts to define a function that approximates the data distribution. It is imperative, for performance reasons, to be able to derive the approximate solution quickly. Thus, description of the function must be kept in memory. The success of the various density estimator techniques hinges upon several parameters, which include the simplicity of the function, the time it takes to find the function parameters, and the

number of parameters we need to store for a given approximation quality.

Let D be a data set with d attributes and n tuples. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_d\}$ be the set of attributes. The domain of each attribute A_i is scaled to the real interval $[0, 1]$. Assuming an ordering of the attributes, each tuple is a point in the d-dimensional space $[0, 1]^d$.

Formally, a *density estimator* for D is a d-dimensional, nonnegative function

$$f(x_1, \ldots x_d), f : [0, 1]^d \rightarrow \mathcal{R}$$

with the property:

$$\int_{[0,1]^d} f(x_1, \ldots x_d) dx_1 \ldots dx_d = 1.$$

The value of f at a specific point $\mathbf{x} = (x_1, \ldots x_d)$ of the d-dimensional space approximates the limit of the probability that a tuple exists in an area U around $\mathbf{x}$, over the volume of U, when U shrinks to $\mathbf{x}$. For a density estimator f and a given region

$$Q(a_1 \leq D.A_1 \leq b_1) \wedge \ldots \wedge (a_d \leq D.A_d \leq b_d),$$

the approximation of Q using f is:

$$sum(f, Q) = \int_{[a_1,b_1]\times\ldots\times[a_d,b_d]} f(x_1, \ldots x_d) dx_1 \ldots dx_d.$$

Given D and f, we say that f is a *good* estimator of D if, for *any* region Q, the difference between the actual value $sum(D, Q)$ and the estimated value $n \times sum(f, Q)$ is small.

Biased sampling according to data density can use any density estimation technique. There is a number of methods suggested in the literature employing different ways to find the density estimator for multidimensional data sets. These include, computing multidimensional histograms [31], [11], [22], [3], using various transforms, like the wavelet transformation [41], [27], SVD [31], or the discrete cosine transform [25] on the data, using kernel estimators [7], [34] [35], as well as sampling [29], [26], [18].

Although our biased-sampling technique can use any density estimation method, using kernel density estimators is a good choice. Work on query approximation [15] and in the statistics literature [35], [34] shows that kernel functions always estimate the density of the data set better than just using a random sample of the data set. The technique is also very efficient. Finding a kernel density estimator is as efficient as taking a random sample and can be done in one data set pass. There are other techniques of similar accuracy (for example, multidimensional histograms), but computing the estimator typically requires a lot more processing time.

4.1 Using Multidimensional Kernels for Density Estimation

Kernel density estimation is based on statistics and, in particular, on kernel theory [34], [12], [42]. Kernel estimation is a generalized form of sampling. Each sample point has weight one, but it distributes its weight in the space around it. A *kernel function* describes the form of the weight distribution.

Given a multi-dimensional dataset D,

1. In one pass take a random sample of size β (input parameter), and approximate the standard deviation in each dimension.
2. Set a kernel function centered on each of the β points selected in step 1, and set the bandwidths of the kernel functions using Scott's rule and the standard deviation computed in step 1.

Fig. 3. Computing a kernel density estimator.

For a data set D, let S be a set of tuples drawn from D uniformly at random (with $|S| = \beta$). Assume there exists a d-dimensional function $k(x_1, \ldots, x_d)$, the *kernel function*, with the property that

$$\int_{[0,1]^d} k(x_1, \ldots, x_d) dx_1 \ldots dx_d = 1.$$

The approximation of the underlying probability distribution of D is:

$$f(x) = \frac{1}{b} \sum_{t_i \in S} k(x_1 - t_{i_1}, \ldots, x_d - t_{i_d}).$$

It has been shown that the exact shape of the kernel function does not affect the approximation a lot [12]. A polynomial or a Gaussian function can work equally well. What is important is the standard deviation of the function, that is, its bandwidth. Thus, one should choose a kernel function that is easy to integrate. The Epanechnikov kernel function has this property [12]. The d-dimensional Epanechnikov kernel function centered at 0 is:

$$k(x_1, \ldots, x_d) = \left(\frac{3}{4}\right)^d \frac{1}{B_1 B_2 \ldots B_d} \prod_{1 \leq i \leq d} \left(1 - \left(\frac{x_i}{B_i}\right)^2\right)$$

if, for all i, $|\frac{x_i}{B_i}| \leq 1$, and $k(x_1, \ldots, k_d) = 0$ if $|\frac{x_i}{B_i}| > 1$. The d parameters $B_1, \ldots, B_d$ are the bandwidth of the kernel function along each of the d dimensions. The magnitude of the bandwidth controls how far from the sample point we distribute the weight of the point. As the bandwidth becomes smaller, the nonzero diameter of the kernel becomes smaller.

In our experiments, we choose the bandwidths according to Scott's rule [34] in d-dimensional space: $B_i = \sqrt{5}\, s_i\, |S|^{-\frac{1}{d+4}}$, where s_i is the standard deviation of the data set on the ith attribute.

Computing a kernel estimator consists of taking a random sample of size S from the data set and computing the standard deviation of the data set in each dimension. This second step can be done in the same pass that the sampling is done, using the technique of [36]. We summarize the technique in Fig. 3.

4.2 Computing the Selectivity of a Range Query Using Kernels

Since the d-dimensional Epanechnikov kernel function is the product of d one-dimensional degree-2 polynomials, its integral within a rectangular region can be computed in $O(d)$ time:

$$sel(f,[a_1,b_1]\times\ldots\times[a_d,b_d]) = \frac{1}{n}\int_{[a_1,b_1]\times\ldots\times[a_d,b_d]} \left(\sum_{1\le i\le b} k_i(x_1,\ldots,x_d)dx_1\ldots dx_d\right) =$$

$$\frac{1}{n}\int_{[a_1,b_1]\times\ldots\times[a_d,b_d]} \sum_{1\le i\le b}\left(\frac{3}{4}\right)^d \frac{1}{B_1B_2\ldots B_d}\prod_{1\le j\le d}$$

$$\left(1-\left(\frac{x_j-X_{ij}}{B_j}\right)^2\right)dx_1\ldots dx_d = \frac{1}{n}\left(\frac{3}{4}\right)^d\frac{1}{B_1B_2\ldots B_d}$$

$$\sum_{1\le i\le b}\int_{[a_1,b_1]}\left(1-\left(\frac{x_1-X_{i1}}{B_1}\right)^2\right)\ldots\int_{[a_d,b_d]}$$

$$\left(1-\left(\frac{x_d-X_{id}}{B_d}\right)^2\right)dx_d\ldots dx_1.$$

It follows that, for a sample of $|S|$ tuples, $sel(f,Q)$ can be computed in $O(d|S|)$ time.

5 The Proposed Biased Sampling Technique

In this section, we outline our technique for density-biased sampling. Let D be a d-dimensional data set with n points. For simplicity, we assume that the space domain is $[0,1]^d$; otherwise, we can scale the attributes. Let

$$f(x_1,\ldots,x_d):[0,1]^d\rightarrow\mathcal{R}$$

be a density estimator for the data set D. That is, for a given region $R\subset[0,1]^d$, the integral

$$n\int_R f(x_1,\ldots,x_d)dx_1\ldots dx_d$$

is approximately equal to the number of points in R. We define the **local density** around a point $\mathbf{x}$ to be the value $LD(\mathbf{x})=\frac{I}{V}$, where $I=\int_B f(\mathbf{x})dx$, B is a ball around $\mathbf{x}$ with radius $\epsilon>0$, and V is the volume of this ball ($V=\int_B d\mathbf{x}$). We assume that the function that describes the density of the data set is continuous and smooth (e.g., it is a multiple differentiable function). The local density in a ball around a point represents the average value of the density function inside this ball. Since the function is smooth, we approximate the local density of the point $\mathbf{x}$ with the value of the function at this point, e.g., $f(\mathbf{x})$. For small values of ϵ, the error of the approximation is expected to be small.

Our goal is to produce a biased sample of D with the following property:

Property 1. *The probability that a point $\mathbf{x}$ in D will be included in the sample is a function of the local density of the data space around the point $\mathbf{x}$.*

Since the desired size of the sample is a user-defined parameter, we want the technique to satisfy the following property as well:

Property 2. *The expected size of the sample should be b, where b is a parameter set by the user.*

Let us consider the case of uniform sampling first. A point in D will be in the sample with probability $\frac{b}{n}$. By the definition of f, if $f(x_1,\ldots,x_d)>1$, then the local density around the point $(x_1,\ldots,x_d)$ is larger than the average density of the space, defined as

$$\frac{\int_{[0,1]^d} f(x_1,\ldots,x_d)dx_1\ldots dx_d}{Volume([0,1]^d)}=1.$$

In density biased sampling, we want the probability of a given point $(x_1\ldots x_d)$ to be included in the sample to be a function of the local density of the space around the point $(x_1,\ldots,x_d)$. This is given by the value of $f(x_1\ldots x_d)$. In addition, we want the technique to be tunable, so we introduce the function $f'(x_1,\ldots,x_d)=(f(x_1,\ldots,x_d))^a$ for some $a\in\mathcal{R}$. The basic idea of the sampling technique is to sample a point $(x_1,\ldots,x_d)$ with probability that is proportional to $\frac{b}{n}f'(x_1,\ldots,x_d)$. The value of a controls the biased sampling process.

Let $k=\sum_{(x_1,\ldots,x_d)\in D} f'(x_1,\ldots,x_d)$. We define

$$f^*(x_1,\ldots,x_d)=\frac{n}{k}f'(x_1,\ldots,x_d).$$

Thus, f^* has the additional property

$$\sum_{(x_{i1}\ldots x_{id})\in D} f^*(x_{i1}\ldots x_{id})=n.$$

In the sampling step, each point $(x_1,\ldots,x_d)\in D$ is included in the sample with probability

$$\frac{b}{n}f^*(x_1,\ldots,x_d).$$

Our technique for density-biased sampling is shown in Fig. 4.

Given a dataset D, a density estimator f for the dataset, and a sample size b set by the user,

1. Compute, for each point $\mathbf{x}$ in $\mathcal{P}$ the density biased probability that it is included in the sample.
2. Make one pass over the data and output each point with the appropriate probability.

Fig. 4. The proposed biased sampling technique.

It is easy to observe that this algorithm satisfies Property 1. Also, it satisfies Property 2 since the expected size of the sample is equal to

$$\sum_{(x_1,\ldots,x_d)\in D}\frac{b}{n}f^*(x_1,\ldots,x_d)=\frac{b}{n}\sum_{(x_1,\ldots,x_d)\in D} f^*(x_1,\ldots,x_d)=b.$$

Another observation is that the technique requires two data set passes, assuming that the density estimator f is available. The value of k can be computed in one pass. In the second pass, we can compute, for each point x in the data set, the value of $f^*(x)$. One of the main advantages of the proposed technique is that it is very flexible and can be tuned for different applications. Using different values of a, we can oversample either the dense regions (for $a>0$) or the sparse regions (for $a<0$). We consider the following cases:

1. $a=0$. Each point is included in the sample with probability $\frac{b}{n}$ and, therefore, we have uniform sampling.

2. $0 < a$. In this case, $f^a(\mathbf{x}) > f^a(\mathbf{y})$ if and only if $f(\mathbf{x}) > f(\mathbf{y})$. It follows that the regions of higher density are sampled at a higher rate than regions of lower density. In particular, regions with density that is higher than the average density in the data space are sampled at a rate that is higher than the uniform sampling rate and regions where the density is lower than the data space average are sampled at a rate that is lower than the uniform sampling rate.
3. $0 > a$. In this case, the opposite is true. Since $f^a(\mathbf{x}) > f^a(\mathbf{y})$ if and only if $f(\mathbf{x}) < f(\mathbf{y})$, the regions with lower density than average are sampled at a higher rate than uniform and the regions of higher density are sampled with a lower rate than uniform.
4. $a = -1$. There is the same expected number of sample points in any two regions of the same volume. Consider two regions A and B with volume V and assume that the density is uniform inside each region and is d_A and d_B, respectively. The number of points inside each region are $d_A V$ and $d_B V$. Each point in region A is sampled with probability $\frac{b}{k}\frac{1}{d_A}$ and, therefore, the expected number of points from this region is $\frac{b}{k}\frac{1}{d_A} * d_A V = \frac{b}{k} V$. Following the above reasoning, we get the same expected number of points for region B.

The sampling technique we described has an additional fundamental property, namely, it preserves, in the sample, the relative densities of different regions in D. If regions A and B in D are dense, they should appear dense in the sample too. Moreover, if A has a higher density than B, then the same should be true in the sample. We state and prove the following lemma:

Lemma 1. *If $a > -1$, relative densities are preserved with high probability. That is, if a region A has more data points than region B in the data set, then, with high probability, the same is true for the sample.*

Proof. Without loss of generality, assume the volumes of regions A and B are the same and equal to V. Also, assume that V is small and the density inside each region is d_A and d_B, respectively. Then, the number of points in A is approximately $d_A\, V$ and, in B, is approximately $d_B\, V$ (where $d_A\, V > d_B\, V$). The points in A are sampled with probability which is proportional to d_A^a, and the points in B are sampled with probability that is proportional to d_B^a. Therefore, the expected number of points in region A that will be included in the sample is proportional to d_A^{a+1} and those in region B is proportional to d_B^{a+1}. Since $d_A > d_B\; d_A^{a+1} > d_B^{a+1}$ if and only if $a + 1 > 0$. It follows that the expected density of region A in the sample is larger than the expected density in region B if $a > -1$. Using Chernoff bounds we can show that, if the number of the points is large, then this happens with high probability. □

Lemma 1 allows us to use density-biased sampling to discover small clusters that are dominated by very large and very dense clusters or to discover clusters in the presence of noise. In the case of $a > 0$, dense areas are oversampled. As a result, it is more likely to obtain a sample that contains points from clusters other than noise. If we set $-1 < a < 0$, we oversample areas that are less dense, but still sample the very dense areas enough so that they remain dense in the sample. This means that it is more likely to discover small and sparse clusters using such a sample.

Our choice of kernel estimators for the density in our presentation and experiments is based solely on the efficiency of computing kernel estimators. One can show that our sampling technique can work with any method that is capable of computing densities accurately and efficiently.

In our presentation of the sampling technique, we choose for simplicity to separate the step of density estimation and the computation of the biased sampling probability. It is possible to integrate both steps in one, thus deriving the sampling probability in a single pass over the database.

6 Biased Sampling in One Data Set Pass

Let $\mathcal{P}$ be a partitioning of the space $[0, 1]$ into nonoverlapping regions. Our technique for adaptive sampling works as follows.

Given a data set D, a partitioning $\mathcal{P}$ of the space into nonoverlapping buckets, a density estimator f, and a sample size b,

1. Compute, for each bucket in $\mathcal{P}$, the probability that a point in the bucket is in the sample. This probability is the same for all points in the bucket.
2. Make one pass over the data, for each point determines the bucket it belongs in and outputs the point with the appropriate probability.

It remains to show how to compute the probabilities for each bucket in the partitioning. In uniform random sampling, a point in bucket c will be in the sample with probability $\frac{b}{n}$. If there are $\int_c f$ points in the bucket c, an expected number of

$$\frac{b}{n} n \int_c f(x_1, \ldots, x_d) dx_1 \ldots dx_d = b \int_c f(x_1, \ldots, x_d) dx_1 \ldots dx_d$$

of them will be in the sample. By the definition of f, if $f(x_1, \ldots, x_d) > 1$, then the local density around the point $(x_1, \ldots, x_d)$ is larger than the average density of the space, defined as

$$\frac{\int_{[0,1]^d} f(x_1, \ldots, x_d) dx_1 \ldots dx_d}{Volume([0,1]^d)} = 1.$$

Let $f'(x_1, \ldots, x_d) = (f(x_1, \ldots, x_d)^a$ for some $0 \leq a$. Then, f' has the following property:

$$\begin{aligned} &\text{If, for regions } R_1, R_2, \int_{R_1} f(x_1, \ldots, x_d) dx_1 \ldots dx_d \\ &< \int_{R_2} f(x_1, \ldots, x_d) dx_1 \ldots dx_d, \end{aligned} \tag{10}$$

$$\begin{aligned} &\text{then } \int_{R_1} f'(x_1, \ldots, x_d) dx_1 \ldots dx_d \\ &< \int_{R_2} f'(x_1, \ldots, x_d) dx_1 \ldots dx_d. \end{aligned} \tag{11}$$

Let $\int_{[0,1]^d} f'(x_1, \ldots, x_d) dx_1 \ldots dx_d = k$. Then, we define

$$f^*(x_1, \ldots, x_d) = \frac{1}{k} f'(x_1, \ldots, x_d).$$

Thus, f^* has the additional property

$$\int_{[0,1]^d} f^*(x_1, \ldots, x_d) dx_1 \ldots dx_d = 1.$$

For a given bucket c, the expected number of points in the bucket that appear in the sample is

$$b \int_c f^*(x_1, \ldots, x_d) dx_1 \ldots dx_d.$$

Therefore, these points are sampled with probability

$$\frac{b}{n} \frac{\int_c f^*(x_1, \ldots, x_d) dx_1 \ldots dx_d}{\int_c f(x_1, \ldots, x_d) dx_1 \ldots dx_d}.$$

The expected size of the sample is equal to

$$\sum_{c \in \mathcal{P}} b \int_c f^*(x_1, \ldots, x_d) dx_1 \ldots dx_d =$$
$$b \sum_{c \in \mathcal{P}} \int_c f^*(x_1, \ldots, x_d) dx_1 \ldots dx_d = b$$

because $\mathcal{P}$ is a partitioning of the space. This method needs only one pass over the data set, assuming that the density estimator function f is available.

6.1 Computation of the Integral

To implement the one-pass method efficiently, we have to evaluate the integrals in each region. These integrals cannot be evaluated analytically in the general case, but they can be evaluated numerically, using Monte-Carlo techniques.

Let $f(x)$ be a density function computed on D. For purposes of exposition, we assume that f is one-dimensional and defined over $[0, 1]$. We wish to compute the integral $I = \int_0^1 f(x)dx$. We apply the Monte Carlo technique in the computation of the integral. We employ uniform random sampling on points $\xi_1, \ldots, \xi_N$ on $[0, 1]$. Let $f(\xi_1), \ldots, f(\xi_N)$ be the values of f at those points. We can approximate I by

$$S = \frac{\sum_{i=1}^N f(\xi_i)}{N},$$

where S is the arithmetic mean of $f(\xi_i), 1 \leq i \leq N$. Let μ be the mathematical expectation $E(f(\xi_i))$ and σ^2 the true variance of $f(\xi_i)s$. The error of the estimation is $|D - \mu|$. An estimation of this error can be made with certainty $1 - \epsilon$, by application of the Chebyshev's inequality:

$$\delta = |S - \mu| = \sigma * \sqrt{\left(\frac{1}{\epsilon * N}\right)}.$$

In practice, we don't know σ, but usually the variance of D can be used instead and get an estimate of the error while applying the technique. If the sample size is large enough (in the order of thousands), then we can apply the central limit theorem to the distribution of D and assume that is normal. In that case, addition bounds for δ can be obtained using the normal curve.

Using kernel density estimators, however, we can analytically evaluate the integral, assuming that the regions are hyperrectangles (Section 4.2). In addition, computing a kernel density estimator can be done in one data set pass. This makes the proposed biased sampling technique a two data set pass process.

7 Using Biased Sampling for Cluster and Outlier Detection

We describe how density biased sampling can be integrated with cluster and outlier detection algorithms. The emphasis is on the evaluation of biased sampling in conjunction with "off-the-self" clustering and outlier detection algorithms as opposed to come up with new algorithms to solve the problems. We focus on hierarchical clustering algorithms and distance-based outliers due to their popularity and discuss how our sampling technique can be used in conjunction with other classes of clustering algorithms.

7.1 Clustering

We run clustering algorithms on the density-biased sample to evaluate the accuracy of the proposed technique. To cluster the sample points, we use a hierarchical clustering algorithm. The algorithm starts with each point in a separate cluster and at each step merges the closest clusters together. The technique is similar to CURE in that it keeps a set of points as a representative of each cluster of points. The hierarchical clustering approach has several advantages over K-means or K-medoids algorithms. It has been shown experimentally that it can be used to discover clusters of arbitrary shape, unlike K-means or K-medoids techniques, which typically perform best when the clusters are spherical [17]. It has also been shown that a hierarchical clustering approach is insensitive to the size of the cluster. On the other hand, K-means or K-medoids algorithms tend to split up very large clusters and merge the pieces with smaller clusters.

The main disadvantage of the hierarchical approach is that the runtime complexity is quadratic. Running a quadratic algorithm even on a few thousand data points can be problematic. A viable approach to overcome this difficulty is to reduce the size of the data set in a manner that preserves the information of interest to the analysis without harming the overall technique.

A hierarchical clustering algorithm that is running off the biased sample is in effect an approximation algorithm for the clustering problem on the original data set. In this respect, our technique is analogous to the new results on approximation clustering algorithms [19], [20] since these algorithms also run on a (uniform random) sample to efficiently obtain the approximate clusterings. The techniques are not directly comparable, however, because these algorithms are approximating the optimal K-medoids partitioning, which can be very different from the hierarchical clusterings we find.

We note here that we can also use biased sampling in conjunction with K-means or K-medoids algorithms if this is desirable. However, K-means,or K-medoids algorithms optimize a criterion that puts equal weight to each point in the data set. For example, for the K-means clustering

algorithm, assuming K clusters the optimization criterion is minimization of the following function:

$$\sum_{\text{(cluster i)}} \sum_{\text{(j } \in \text{ i)}} dist(j, m(i)), \tag{12}$$

where, for the K-means algorithm, $m(i)$ is the mean of cluster i and, for the K-medoids algorithm, $m(i)$ is the medoid of cluster i. In density biased sampling, since some areas are underrepresented and some are overrepresented in the sample, we have to factor in the optimization process the probability that a single point is present in the sample. To take this into account, we have to weight the sample points with the inverse of the probability that each was sampled. The following lemma formalizes this process.

Lemma 2. *If each point is weighted by the inverse of the probability that it was sampled, the density of the weighted sample approximates the density of the original data set.*

The proof of Lemma 2 is similar to that of Lemma 1, but using $a = -1$ rather than $a > -1$, and so it is omitted here.

We conjecture that the use of biased sampling will result into good approximation results for the K-means or K-medoids clustering because the points in the dense areas are more likely to be close to cluster centroids or to be the cluster medoids.

7.2 Distance-Based Outliers

Distance-based outliers were introduced by Knorr and Ng [23], [24]. Following them, we give the following definition:

Definition 1. *An object O in a data set D is a $DB(p, k)$ outlier if at most p objects in D lie at distance at most k from O.*

Knorr and Ng [23] show that this definition is compatible with empirical definitions of what is an outlier and it also generalizes statistical definitions of outliers. Note that, in the definition of a $DB(p, k)$-outlier, the values of k and p are parameters set by the user. The number of objects that can be at distance at most k can also be specified as a fraction fr of the data set size, in which case, $p = fr\,|D|$. The technique we present has the additional advantage that it can estimate the number of $DB(p, k)$-outliers in a data set D very efficiently in one data set pass. This gives the opportunity for experimental exploration of k and p in order to derive sufficient values for the specific data set and the task at hand.

To facilitate the description of our algorithm, we also define the function $N_D(O, k)$ to be the number of points that lie in distance at most k from the point O in the data set D. Clearly, a point $O \in D$ is a $DB(p, k)$-outlier iff $N_D(O, k) < p$.

7.2.1 Finding $DB(p, d)$-Outliers

The basic idea of the algorithm is to sample the regions on which the data point density is very low. Since we are dealing with data sets with numerical attributes, we assume, in the description of the algorithm that the distance function between points is the Euclidean distance. However, different distance metrics (for example, the L_1 or Manhattan metric) can be used equally well.

Let D be a data set with $|D| = n$ points and d numerical dimensions. Let f be a density estimator of the data set D. Given input parameters k (the maximum distance of neighboring points) and p (the maximum number of the data set points that can be at distance at most k away), we compute, for each point O, the expected number of points in a ball with radius k that is centered at the point. Thus, we estimate the value of $N_D(O, k)$ by

$$N'_D(O, k) = \int_{\text{ball of radius k around O}} f(x_1, \ldots, x_d) dx_1 \ldots dx_d.$$

We keep the points that have a smaller expected number of neighbors than $O(k)$. These are the likely outliers. Then, we make another pass over the data and verify the number of neighbors for each of the likely outliers.

7.2.2 Running Time

Using multidimensional kernels for density estimation, it takes one pass over the data set to compute the density estimator function f. The time to find the likely outliers is also linear to the size of the data set since each point has to be read only once. We approximate $N'_D(O, k)$ for each $O \in D$, $O = (o_1, \ldots, o_d)$, by taking a $\mathrm{g}\Phi\mathrm{hyperrectangle}$ with side equal to $2k$ around O. Using an estimator f with b d-dimensional kernels, finding the approximation takes time to the size of the description of f, that is, $O(b\,d)$:

$$\int_{[o_1-k, o_1+k] \times \ldots \times [o_d-k, o_d+k]} f(x_1, \ldots, x_d) dx_1 \ldots dx_d =$$
$$\left(\frac{3}{4}\right)^d \frac{1}{B_1 B_2 \ldots B_d} \sum_{1 \le i \le b} \int_{[o_1-k, o_1+k]} \left(1 - \left(\frac{x_1 - X_{i1}}{B_1}\right)^2\right) \cdots,$$
$$\int_{[o_d-k, o_d+k]} \left(1 - \left(\frac{x_d - X_{id}}{B_d}\right)^2\right) dx_d \ldots dx_1$$

which can be computed in closed form.

The third, optional, part of checking if the likely outliers are indeed $DB(p, k)$-outliers can also be accomplished in one data set pass if we assume that the total number of likely outliers fits in memory. The total running time is $O(dmn)$, where d is the dimensionality, m is the number of outliers, and n is the number of the points. If we use an efficient data structure (e.g., kd-trees) to store the likely candidates, we can reduce the running time to $O(d \log m\, n)$. The algorithm makes $2 + \lceil \frac{m}{M} \rceil$ data set passes, where m is the likely number of outliers and M is the size of the memory. It is reasonable to expect that, in practice, m will fit in memory since, by definition, an outlier is an infrequent event. Therefore, the technique will typically perform three data set passes.

8 Experiments

We present an experimental evaluation of our approach. We run clustering and outlier detection tasks in both synthetic and real data sets and we intend to explore the following issues: 1) determine whether biased sampling indeed offers advantages over uniform random sampling for cluster and outlier detection and quantify this advantage and 2) determine the accuracy of the approximate solution that we obtain if we run cluster and outlier detection algorithms on only the biased sample, as opposed to running on the entire data set.

We use the same algorithm for clustering the density-biased sample and the uniform random sample in order to make the comparison objective. The clustering algorithm we use is hierarchical. The algorithm is an implementation

of CURE, as described in [17]. In this algorithm, each cluster is represented by a set of points that have been carefully selected in order to represent the shape of the cluster (well scattered points). Therefore, when the hierarchical algorithm is running on the uniform random sample, the clusters it produces are the ones CURE would have found. We compare these results with the ones obtained when the hierarchical algorithm is run on a density-biased sample (with different values of a). We also include the BIRCH [43] clustering algorithm in the comparison since it is generally considered one of the best clustering methods proposed so far for very large databases. We have not run experiments using CLARANS [28], but, based on experimental work in [43], [17], we expect CLARANS to perform approximately as well as BIRCH. We allow BIRCH to use as much space as the size of the sample to keep the CF-tree (Section 2) which describes the data set. Although we constrain the size of the CF-tree, we allow BIRCH to work on the entire data set for clustering, that is, we do not run BIRCH on our samples. By comparing with BIRCH, we intend to explore the effectiveness of our summarization technique for identifying clusters to that of BIRCH.

8.1 Description Of Data Sets

Synthetic Data Sets. We used a synthetic data generator to produce two and three-dimensional data sets. The number of clusters is a parameter, varied from 10 to 100 in our experiments. Due to space constraints, we report only the results for 10 clusters. The rest of our results with data sets containing a larger number of clusters are qualitatively similar with those presented herein, thus they are omitted for brevity. Each cluster is defined as a hyperrectangle and the points in the interior of the cluster are uniformly distributed. The clusters can have nonspherical shapes, different sizes (number of points in each cluster), and different average densities. We also generate and use in our experiments data sets with noise. We generate these data sets as follows: Let D be a data set of size $|D|$ containing k synthetically generated clusters. We add $l * |D|, 0 \leq l \leq 1$ uniformly distributed points in D as noise and we say that D contains $f_n = 100 * l$ noise. We vary f_n from 5 to 80 percent in our experiments. In addition, we used synthetic data sets from [17] (see Fig. 8).

Geospatial Data Sets. We also used real-life data sets from spatial databases. One such data set is presented in Fig. 1 and represents 130,000 postal addresses in the northeast states and we refer to it as the northeast data set. The other is a data set that represents postal addresses in California and has a size of 62,553 points. We refer to it as the California data set. Both data sets contain two-dimensional points. We employ both clustering and outlier detection algorithms on these sets. Another data set is the Forest Cover Data set from the University of California at Irvine (UCI) KDD archive.[1] This was obtained from the US Forest Service (USFS). It includes 59,000 points.

8.2 Implementation Details And Parameter Setting

For the density approximation function, we use the Kernel technique. In our experiments, we use $\beta = 1,000$ sample points to initialize the kernel centers and we use the Epanechnikov kernel function. In our experiments, the value of 1,000 gave very good results across a wide range of data sets with different statistical characteristics and sizes and we propose the use of this value in practice. We use the Scott's rule to select the kernel bandwidths (we compute the standard deviation using an approximation algorithm in one pass, the same pass that we get the sample for the kernels). We extract two types of samples:

- *Uniform Sampling*. We extract a uniform random sample of size b, by first reading the size, N, of the data set and then sequentially scanning the data set and choosing a point with probability $\frac{b}{N}$. Thus, the expected size of the sample is b.
- *Biased Sampling*. We implemented the algorithm described in Section 5. There are three parameters affecting this algorithm:
  1. the number of sample points that we use for the kernel estimation (ks),
  2. the number of sample points b, and
  3. the parameter α that controls the rate of sampling for areas with different densities.

  We experiment with those parameters in the sequel.

  We also compare our technique with the technique of [30].[2] Since this technique is designed specifically to identify clusters of different sizes and densities by oversampling sparse regions and undersample sparse regions, we run the comparison for this case only.

We employ the following clustering algorithms:

- *BIRCH*. We used an implementation of the BIRCH algorithm provided to us by the authors [42]. We set the parameter values as suggested in [42]. Thus, we set the page size to 1,024 bytes, initial threshold to 0, and input range to 1,000.
- *Hierarchical Clustering algorithm*. We follow the suggestion of the experimental study in [17] and we set the parameters accordingly. Parameter α (the shrink factor) is set to 0.3. We use one partition and the default value for the representative points is 10.

8.3 Clustering Experiments

8.3.1 Running Time Experiments

In this section, we show that the sampling part of our technique has linear running time to the size of the data set and the number of kernels used to approximate the density distribution. Fig. 5 plots the running time of the sampling phase of the algorithm with the number of kernels ranging from 100 to 3,000. We present results for two data sets (data set 1 from [17] and one of our data sets with $f_n = 20\%$). Fig. 6 shows that our method scales linearly as the size of the data set increases from 100k points to 1 million.

Fig. 7 plots the total running time of the clustering algorithm for biased (BS-CURE) and random sampling (RS-CURE). We used a data set with 1 million points and 1,000 kernels. Note that the running time of the hierarchical algorithm is quadratic. Also note that the time we spend for building the density estimator and for

1. Available from kdd.ics.uci.edu/summary.data.type.html.

2. The code is available in: www.cs.cmu.edu/People/crpalmer/dbs-current.tar.gz.

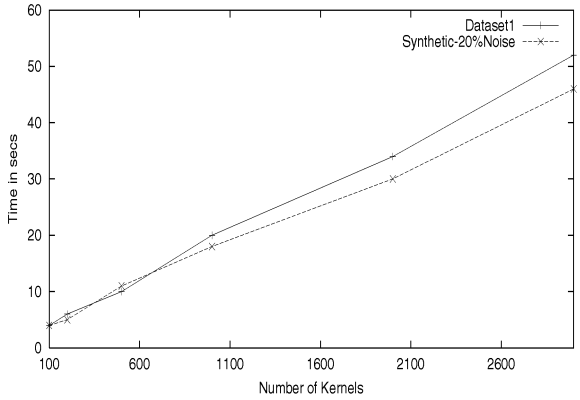


Fig. 5. Running time of the sampling phase of the algorithm with different number of kernels.

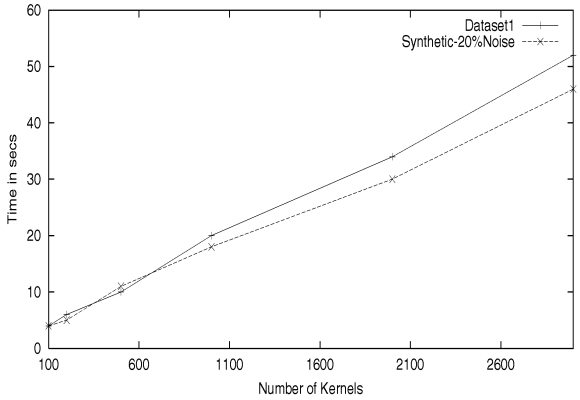


Fig. 6. Running time of the sampling phase of the algorithm for data sets with different sizes.

doing another pass for the actual sampling is more than offset by the time we save because we can run a quadratic algorithm on a smaller sample. For example, if we first run the algorithm on a 1 percent uniform random sample, the running time is $O(n + (n/100)^2)$. If we run the clustering algorithm on a 0.5 percent biased sample, the running time is $O(2n + (n/200)^2)$, which is faster than the uniform sample case for n larger than about 14,000. This illustrates the advantage of the biased sampling technique although, in practice, the constants would of course be different. For 1 million points, our method with a 0.4 percent density-biased sample is faster than running the hierarchical algorithm on a 0.8 percent random sample. As the sample size increases, the difference proportionally decreases so that, for the same 1 million point data set, our technique with a 0.8 percent density-biased sample is faster than the hierarchical algorithm with 1.1 percent random sample.

8.3.2 Biased Sampling versus Uniform Random Sampling

In this experiment, we explore the effects of using density-biased sampling as opposed to uniform random sampling in order to identify clusters. We used a data set (data set 1) used in [17]. This data set has five clusters with different shapes and densities. We draw a uniform random sample as well as a biased sample with $\alpha = -0.5$, both of size 1,000, and we run the hierarchical clustering method on both samples.

In Fig. 8a, we present the data set and, in Fig. 8b, we plot 10 representatives of each cluster found using the biased sample. The centers of all five clusters have been discovered. Fig. 8c plots 10 representatives of each cluster found using the uniform random sample. Note that the large cluster is split into three smaller ones and two pairs of neighboring clusters are merged into one. Clearly, the uniform random sample fails to distinguish the clusters.

It is worth noticing that, for this data set, there exists a sample size (well above 2,000 points in size) on which the hierarchical technique can now correctly identify all clusters. This is consistent with our analytical observation in Theorem 1. A much larger sample (in this case, twice the size of the biased sample) is required by uniform random sampling to achieve the same effects.

8.3.3 The Effects Of Noise In Accuracy

In this set of experiments, we show the advantage of density-biased sampling with $a > 0$ over uniform sampling in the presence of noise in the data set.

We employ our sampling technique and uniform random sampling on noisy data sets and we compare the results of the two sampling algorithms. We use synthetic data sets in two and three dimensions and we vary the noise by varying f_n from 5 percent to 80 percent points. The data sets (without noise added) contain 100K points distributed into 10 clusters of equal size, but with different areas (and therefore different densities). Since the objective is to find clusters in the presence of strong noise, we oversample the dense regions in biased sampling. In the experiments, we use two different values for a, 0.5, and 1 (note that, if $a > 0$, we oversample the dense areas).

We run experiments using sample sizes of 2 percent and 4 percent in two dimensions and a sample size of 2 percent in three dimensions. Figs. 9a, 9b, and 10 summarize the results for $a = 1$. To evaluate the results of the hierarchical algorithm, a cluster is found if at least 90 percent of its representative points are in the interior of the same cluster in the synthetic data set. Since BIRCH reports cluster centers and radiuses if it reports a cluster center that lies in the interior of a cluster in the synthetic data set, we assume that this cluster is found by BIRCH.

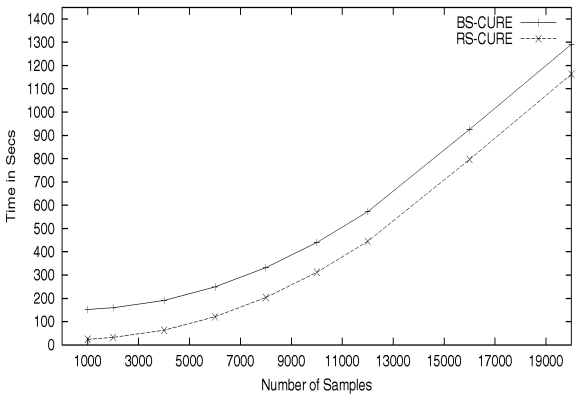


Fig. 7. Running time of the clustering algorithm with random (RS-CURE) and biased sampling (BS-CURE).

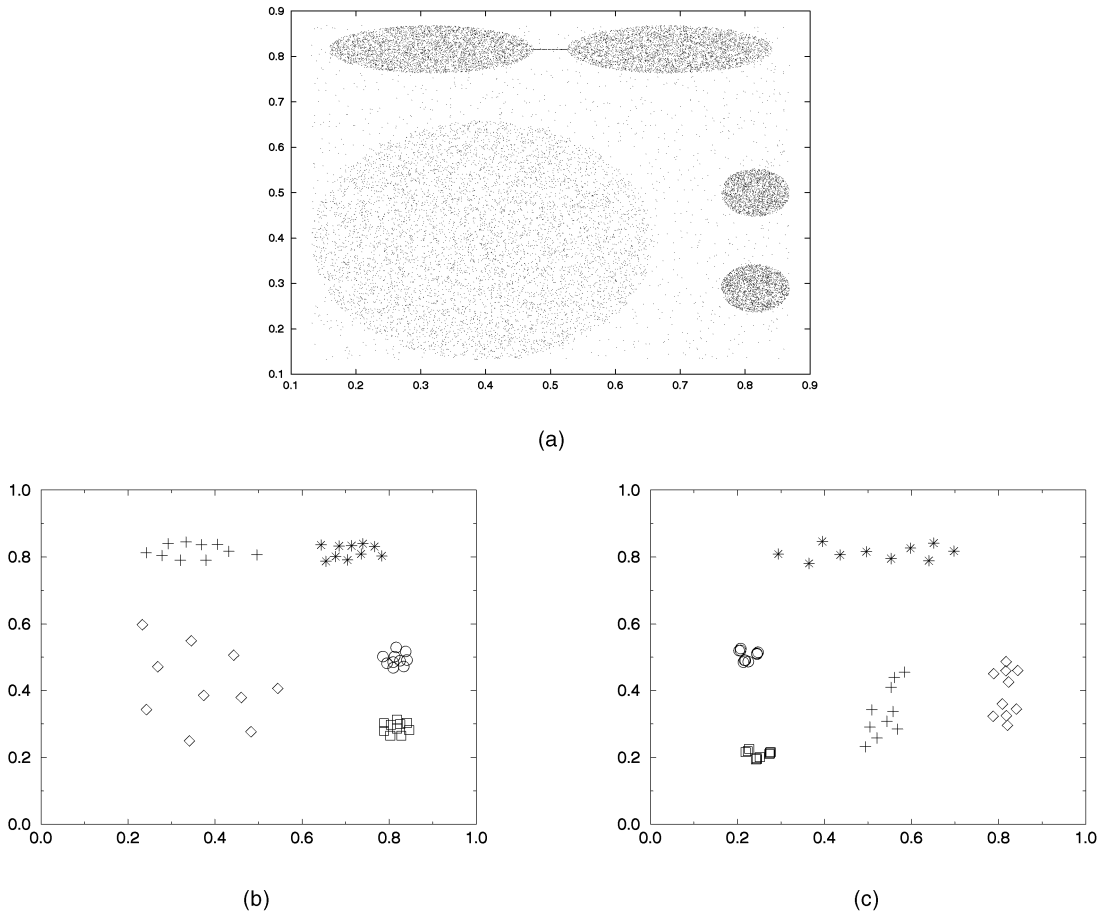


Fig. 8. Clustering using the hierarchical algorithm, sample size 1,000 points. (a) Data set 1 [17], (b) biased sample, and (c) uniform sample.

The results show that, by using biased sampling, clusters can be identified efficiently even in the presence of heavy noise. It is evident that biased sampling can identify all clusters present in the data set, even with $f_n = 70\%$ and misses one cluster at $f_n = 80\%$ noise. As expected, uniform random sampling works well when the level of noise is low, but its accuracy drops as more noise is added. Using a larger sized sample (Fig. 9b), biased sampling is essentially unaffected; uniform random sampling starts missing clusters for $f_n = 40\%$, but, eventually, its effectiveness degrades to work worse than BIRCH. The advantage of biased sampling is even more pronounced in three dimensions as shown in Fig. 10. The effectiveness of biased sampling to identify clusters is robust to the increase of dimensionality from two to three. The effectiveness of uniform sampling and BIRCH degrades quickly. For

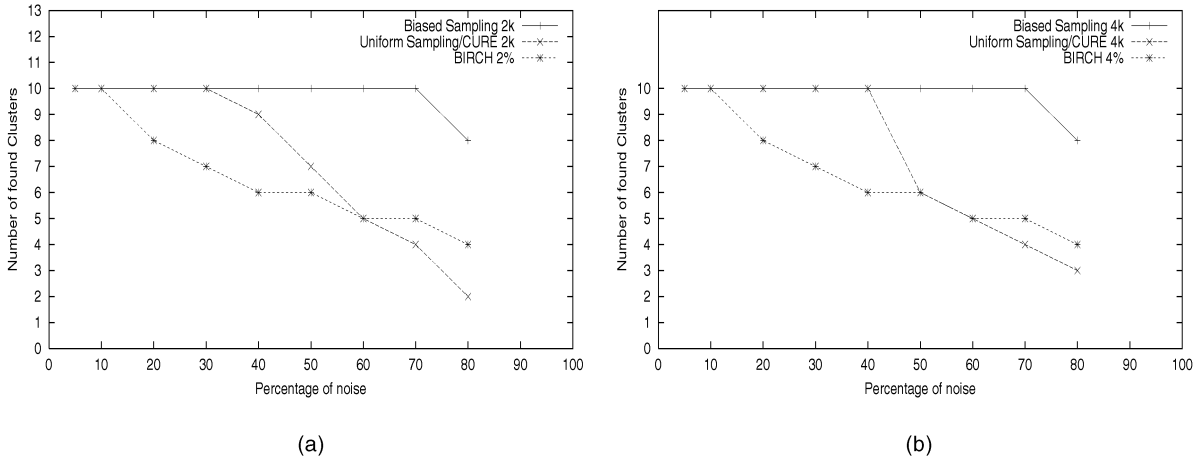


Fig. 9. Varying noise in two-dimensions (ranging f_n from 5 percent to 80 percent). (a) Sample size 2 percent and (b) sample size 4 percent.

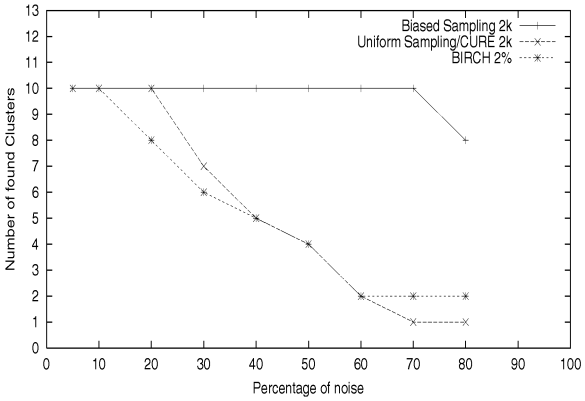


Fig. 10. Varying Noise in 3-dimensions, (ranging f_n from 5 percent to 80 percent), sample size 2 percent.

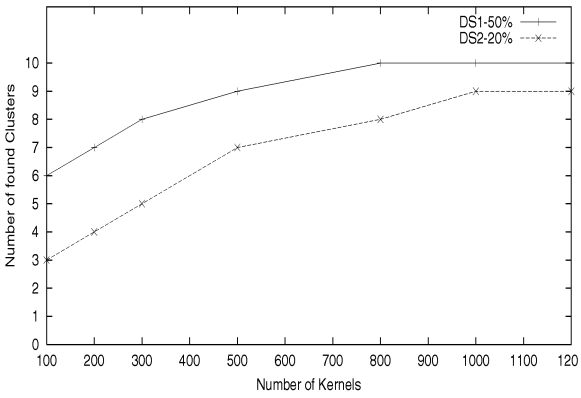


Fig. 11. Varying the number of kernels.

example, at 60 percent noise and three dimensions, both uniform sampling and BIRCH identify less than half the number of clusters identified in the two-dimensional case.

8.3.4 Clusters With Various Densities

In this set of experiments, we show the advantage of biased sampling with $0 > a$ in the case that the size and density of some clusters is very small in relation to other clusters. The difficulty in sampling data sets with this distribution lies in the fact that, if the clusters are small, there may be too few points from a cluster in the sample and, therefore, the algorithm may dismiss these points. We evaluate the effectiveness of biased sampling in discovering all the clusters in the data set when the density of the clusters varies a lot. We use two-dimensional synthetic data sets containing 100,000 and 10 clusters; the largest five of which are 10 times larger than the smaller cluster. Figs. 12a, 12b, and 12c shows the results when the data set has 10 percent, 20 percent, and 30 percent additional noise, respectively.

We run biased sampling with $a = -0.25$ and $a = -0.5$ and, therefore, we sample dense areas proportionally less than sparser areas. However, since we run for values of a that are between -1 and 0, we still expect the larger, denser clusters to be found and, in addition, we make the smaller clusters proportionally larger in the sample. The lower the overall level of noise, the smaller the value of a should be to obtain high accuracy. For example, for 10 percent noise, the best results are achieved for $a = -0.5$, for 20 percent noise, $a = -0.25$, and, for 30 percent noise, uniform sampling (that is $a = 0$) becomes competitive. This result is intuitive because, when a is small, as the noise level increases, proportionally more noise points are added to the sample than when a is larger. Interestingly, BIRCH is not affected as much by the noise as it is affected by the relative size of the clusters. In all cases, BIRCH discovers between six and eight out of the 10 clusters and misses most small clusters altogether.

We also run experiments using the biased sampling technique of [30] (using the code that is available on the Web). This technique has been designed for this situation and samples sparse regions at a higher rate in order to identify small clusters. It uses a grid to partition the data space. Since the number of cells can be very large, grid cells are hashed into a hash table. In the experiments, we allow up to 5MB of memory to store the hash table. To make a fair comparison, we use this technique to obtain a sample and then we use the hierarchical algorithm to find the clusters in the sample. This grid-based technique works well in lower dimensions and no noise, but it is not very accurate at higher dimensions and when there is noise in the data set (Fig. 12d), even when we allow 5MB of memory to be used. In five dimensions and 10 percent noise, its performance is better than uniform random sampling, but worse than our technique which estimates the density more accurately.

8.3.5 Varying The Sample Size

In both previous experiments, we also varied the size of the sample. We observe that both density-biased and uniform sampling do not improve much when the size of the sample is increased beyond a certain value. An interesting observation is that this occurs much earlier when using density-biased sampling than uniform sampling, an observation that is consistent with our analytical expectations. In our experiments, this value was generally 1K points for density-biased sampling and 2K points for uniform sampling.

8.3.6 Varying The Number of Kernels

In another set of experiments, we varied the number of kernels and, therefore, the quality of the density approximation. In Fig. 11, we present results for two data sets. The data set $DS1 - 50$ percent contains 100k points with 10 clusters of the same size and an additional 50 percent noise. The second data set, $DS2 - 20$ percent, consists of 100k points of 10 clusters with very different sizes plus 20 percent noise. We used biased sampling with a equal to 1.0 and -0.25, respectively, and 500 sample points. Notice that, in the second case, the accuracy of the density estimation is more important because there is a very large variation in the densities of the clusters. In both cases, when increasing the number of kernels from 100 to 1,200, the quality of the clustering results initially improves considerably but the rate of the improvement is reduced continuously.

8.3.7 Real Data Sets

We report on clusterings obtained for the northeast data set and the California data set, using biased sampling and uniform random sampling.

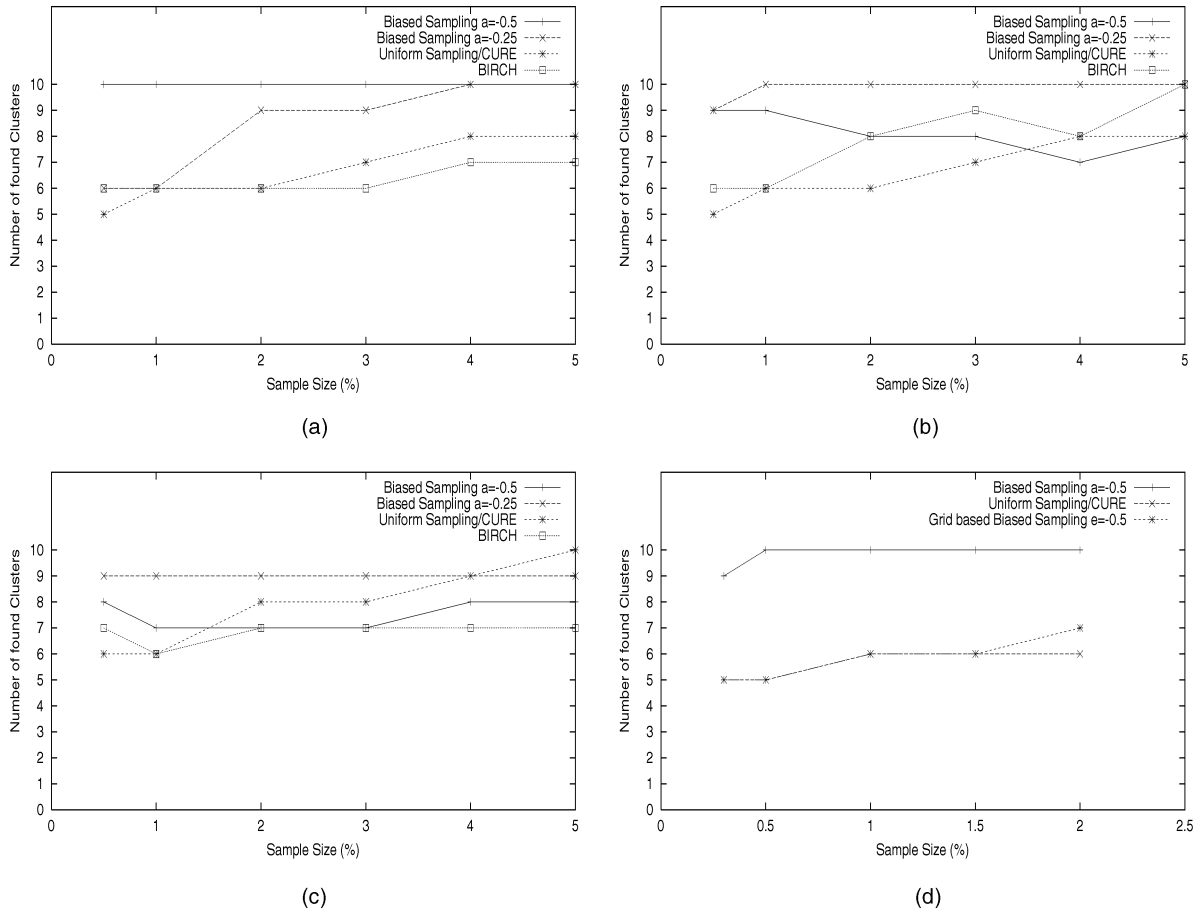


Fig. 12. Finding clusters of low density, in the presence of noise. (a) 2D noise 10 percent, (b) 2D noise 20 percent, (c) 2D noise 30 percent, and (d) 5D noise 10 percent.

- *northeast data set* (Fig. 1a). The results of the clustering algorithm using 10 representatives shows that density-biased sampling (Fig. 1e) outperforms random sampling and accurately clusters the points around the cities of Philadelphia, New York, and Boston. Random sampling fails to identify the high density areas that correspond to the major metropolitan areas of the region, even when the sample size increases from 1K (Fig. 1d) to 4k or 6K (Fig. 13a and Fig. 13b). When clustering this data set with BIRCH we find one spurious cluster, while the three largest concentrations are partitioned into two clusters.
- *California data set*. Similarly, a density-biased sample is more effective in identifying large clusters in the California data set as well. The graphs are omitted for brevity.

8.4 Practitioner's Guide

We summarize the results of our experimental evaluation, and we offer some suggestions on how to use density-biased sampling for cluster detection in practice.

- For noisy data sets, setting $a = 1$ allows reliable detection of dense clusters.
- If the data set is not expected to include noise, setting $a = -0.5$ allows detection of very small or very sparse clusters.
- Setting the size of the density-biased sample at 1 percent of the data set size offers very accurate results and is a good balance between accuracy and performance.
- Setting the number of kernels in the density estimation $\beta = 1,000$ allows accurate estimation of the data set density. As the performance results of Sections 8.3.1 and 8.3.6 indicate, there is a linear dependency between the number of kernels and the performance of our approach. Although the suggested value of β is sufficient, increasing it is not going to harm the overall performance of our approach.

8.5 Outlier Detection Experiments

We implemented the algorithm described in Section 6.2 and we report on experimental results using synthetic data sets and real data sets. In the synthetic data sets, we generate dense clusters and add a number of uniformly distributed outliers. The real data sets include the geospatial data sets and a two-dimensional data set containing telecommunications data from AT&T. This data set contains one large cluster (so, we did not use it in the clustering experiments) and a set of outliers. Table 1 summarizes the experiments on the real data sets. In each experiment, we show how many outliers are in the data set (to find them, we check for each point if it is an outlier), how many likely outliers we

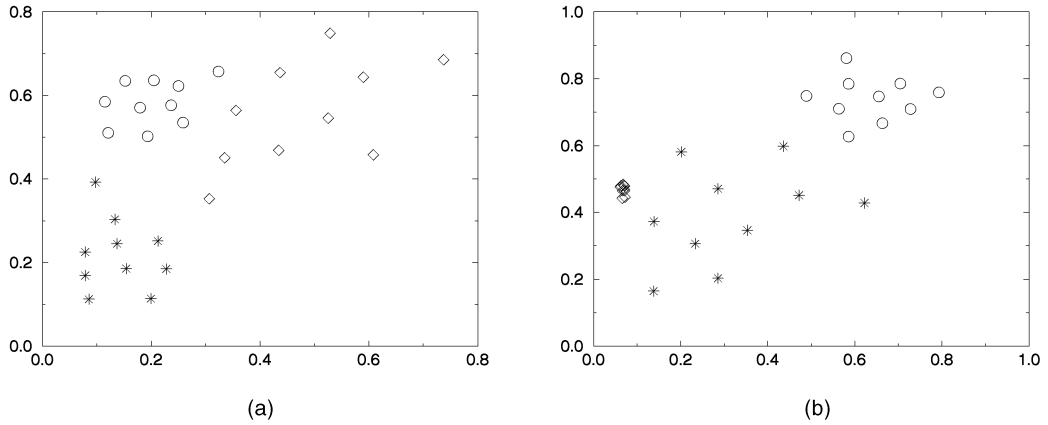


Fig. 13. Ten representative points for clusters found using uniform random sampling of different sizes in the northeast data set. (a) Uniform sample (4,000 points) and (b) uniform sample (6,000 points).

find in the first scan, and how many of them are found to be real outliers in the second scan.

The results show that, in almost all cases, the algorithm finds all the outliers with at most two data set passes plus the data set pass that is required to compute the density estimator.

9 Conclusions

We have presented a biased sampling technique based on density to expedite the process of cluster and outlier detection. We have analytically demonstrated the potential benefits of biased sampling approaches, we have presented an efficient algorithm to extract a biased sample from a large data set capable of factoring in user requirements and presented its important properties. Moreover, we have proposed efficient algorithms to couple our sampling approach with existing algorithms for cluster and outlier detection. Finally, we have experimentally evaluated our techniques using "off-the-self" clustering and outlier detection algorithms and we have experimentally demonstrated the benefits of our approach.

This work raises several issues for further study. Although we demonstrated that biased sampling can be an asset in two specific data mining tasks, it is interesting to explore the use of sampling in other data mining tasks as well. Several other important tasks, like classification, construction of decision trees, or finding association rules, can potentially benefit both in construction time and usability by the application of similar biased sampling

TABLE 1
Experiments for Outlier Detection

Dataset name	Dataset size	Dimensionality	distance k	number of points p	$D(k,p)$-outliers (real)	Outliers found in first pass	Total number found in 2 passes
NorthEast	130000	2	0.1	1000	58	558	57
NorthEast	130000	2	0.1	2000	865	4819	865
California	62000	2	0.1	200	4	8	3
California	62000	2	0.1	500	67	153	66
California	62000	2	0.1	1000	759	1750	759
California	62000	2	0.1	2000	2267	5332	2265
AT&T	25000	2	0.1	200	324	255	255
AT&T	25000	2	0.1	500	460	490	460
AT&T	25000	2	0.1	1000	494	547	494
AT&T	25000	2	0.2	500	226	229	226
AT&T	25000	2	0.2	1000	226	229	226
Forest Cover	59000	3	0.1	200	1833	2645	1816
Forest Cover	59000	3	0.1	500	3913	5312	3913
Forest Cover	59000	3	0.2	500	533	786	527
Forest Cover	59000	3	0.2	1000	1211	1725	1211

techniques suitably adjusted to take the specifics of these mining tasks into account.

The biased sampling technique we presented extends naturally in high dimensions. The quality of the sample depends on the accuracy of the density estimator chosen. Different density estimation techniques can also be employed in higher dimensions. We believe that extension of our techniques to data sets of very high dimensionality is possible by taking into account several properties of high-dimensional spaces. This topic is the focus of current work.

ACKNOWLEDGMENTS

The authors wish to thank Piotr Indyk, H.V. Jagadish, and S. Muthukrishnan for sharing their thoughts and time during the course of this work and Daniel Barbara and V. Ganti for providing the code for hierarchical clustering and BIRCH, respectively. Also, we would like to thank Chris Palmer for putting his biased sampling generator on the Web and the anonymous reviewers for their valuable comments that helped to improve the presentation of the paper. The work of Dimitrios Gunopulos was supported by US National Science Foundation (NSF) CAREER Award 9984729, US NSF IIS-9907477, the US Department of Defense, and AT&T.

REFERENCES

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications," *Proc. 1998 ACM SIGMOD Int'l Conf. Management of Data,* pp. 94-105, 1998.

[2] S. Acharya, P. Gibbons, V. Poosala, and S. Ramaswamy, "Join Synopses for Approximate Query Answering," *Proc. SIGMOD,* pp. 275-286, June 1999.

[3] S. Acharya, V. Poosala, and S. Ramaswamy, "Selectivity Estimation in Spatial Databases," *Proc. SIGMOD,* June 1999.

[4] D. Barbara, C. Faloutsos, J. Hellerstein, Y. Ioannidis, H.V. Jagadish, T. Johnson, R. Ng, V. Poosala, K. Ross, and K.C. Sevcik, "The New Jersey Data Reduction Report," *Data Eng. Bull.,* Sept. 1996.

[5] P. Bradley, U. Fayyad, and C. Reina, "Scaling Em (Expectation-Maximization) Clustering to Large Databases," Microsoft Research Report, MSR-TR-98-35, Aug. 1998.

[6] M. Breunig, H.P. Kriegel, R. Ng, and J. Sander, "LOF: Identifying Density-Based Local Outliers," *Proc. SIGMOD,* May 2000.

[7] B. Blohsfeld, D. Korus, and B. Seeger, "A Comparison of Selectivity Estimators for Range Queries on Metric Attributes," *Proc. 1999 ACM SIGMOD Int'l Conf. Management of Data,* 1999.

[8] Y. Barnett and T. Lewis, *Outliers in Statistical Data.* John Wiley & Sons, 1994.

[9] A. Borodin, R. Ostrovsky, and Y. Rabani, "Subquadratic Approximation Algorithms for Clustering Problems in High Dimensional Spaces," *Proc. Ann. ACM Symp. Theory of Computing,* pp. 435-444, May 1999.

[10] S. Chaudhuri, R. Motwani, and V. Narasayya, "Random Sampling for Histogram Construction: How Much Is Enough?" *Proc. SIGMOD,* pp. 436-447, June 1998.

[11] S. Chaudhuri, R. Motwani, and V. Narasayya, "On Random Sampling over Joins," *Proc. SIGMOD,* pp. 263-274, June 1999.

[12] N.A.C. Cressie, *Statistics For Spatial Data.* Wiley & Sons, 1993.

[13] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data Via Em Algorithm," *J. Royal Statistical Soc., Series B (Methodological),* vol. 39, no. 1, pp. 1-38, 1977.

[14] M. Ester, H.P. Kriegel, J. Sander, and X. Xu, "A Density Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," *Proc. Int'l Conf. Knowledge Discovery and Databases,* Aug. 1996.

[15] D. Gunopulos, G. Kollios, V. Tsotras, and C. Domeniconi, "Approximating Multi-Dimensional Aggregate Range Queries over Real Attributes," *Proc. SIGMOD,* May 2000.

[16] P.B. Gibbons and Y. Matias, "New Sampling-Based Summary Statistics for Improving Approximate Query Answers," *Proc. 1998 ACM SIGMOD Int'l Conf. Management of Data,* 1998.

[17] S. Guha, R. Rastogi, and K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," *Proc. ACM SIGMOD,* pp. 73-84, June 1998.

[18] P. Haas and A. Swami, "Sequantial Sampling Procedures for Query Size Estimation" *Proc. ACM SIGMOD,* pp. 341-350, June 1992.

[19] P. Indyk, "Sublinear Time Algorithms for Metric Space Problems," *Proc. 31st Symp. Theory of Computing,* 1999.

[20] P. Indyk, "A Sublinear-Time Approximation Scheme for Clustering in Metric Spaces," *Proc. 40th Symp. Foundations of Computer Science,* 1999.

[21] H.V. Jagadish, N. Koudas, and S. Muthukrishnan, "Mining Deviants in a Time Series Database," *Proc. Very Large Data Bases Conf.,* 1999.

[22] F. Korn, T. Johnson, and H. Jagadish, "Range Selectivity Estimation for Continuous Attributes," *Proc. 11th Int'l Conf. SSDBMs,* 1999.

[23] E. Knorr and R. Ng, "Algorithms for Mining Distance Based Outliers in Large Databases," *Proc. Very Large Data Bases Conf.,* pp. 392-403, Aug. 1998.

[24] E. Knorr and R. Ng, "Finding Intensional Knowledge of Distance Based Outliers," *Proc. Very Large Data Bases,* pp. 211-222, Sept. 1999.

[25] J. Lee, D. Kim, and C. Chung, "Multi-Dimensional Selectivity Estimation Using Compressed Histogram Information," *Proc. 1999 ACM SIGMOD Int'l Conf. Management of Data,* 1999.

[26] R.J. Lipton, J.F. Naughton, and D.A. Schneider, "Practical Selectivity Estimation through Adaptive Sampling," *Proc. ACM SIGMOD,* pp. 1-11, May 1990.

[27] Y. Matias, J.S. Vitter, and M. Wang, "Wavelet-Based Histograms for Selectivity Estimation," *Proc. 1998 ACM SIGMOD Int'l Conf. Management of Data,* 1998.

[28] R.T. Ng and J. Han, "Efficient and Effective Clustering Methods for Spatial Data Mining," *Proc. Very Large Data Bases Conf.,* pp. 144-155, Sept. 1994.

[29] F. Olken and D. Rotem, "Random Sampling from Database Files: A Survey," *Proc. Fifth Int'l Conf. Statistical and Scientific Database Management,* 1990.

[30] C. Palmer and C. Faloutsos, "Density Biased Sampling: An Improved Method for Data Mining and Clustering," *Proc. SIGMOD,* May 2000.

[31] V. Poosala and Y. Ioannidis, "Selectivity Estimation without the Attribute Value Independence Assumption," *Proc. Very Large Data Bases Conf.,* pp. 486-495, Aug. 1997.

[32] P. Rousseeuw and A. Leory, *Robust Regression and Outlier Detection.* Wiley Series in Probability and Statistics, 1987.

[33] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient Algorithms for Mining Outliers from Large Data Sets," *Proc. SIGMOD,* May 2000.

[34] D. Scott, *Multivariate Density Estimation: Theory, Practice and Visualization.* Wiley and Sons, 1992.

[35] B.W. Silverman, "Density Estimation for Statistics and Data Analysis," *Monographs on Statistics and Applied Probability,* Chapman & Hall, 1986.

[36] G. Singh, S. Rajagopalan, and B. Lindsay, "Random Sampling Techniques for Space Efficient Computation Of Large Data Sets," *Proc. SIGMOD,* June 1999.

[37] S.K. Thompson, *Sampling.* New York: John Wiley & Sons, 1992.

[38] H. Toivonen, "Sampling Large Databases for Association Rules," *Proc. Very Large Data Bases Conf.,* pp. 134-145, Aug. 1996.

[39] S.K. Thompson and G.A.F. Seber, *Adaptive Sampling.* New York: John Wiley & Sons, 1996.

[40] J.S. Vitter, "Random Sampling with a Reservoir," *ACM Trans. Math. Software,* vol. 11, no. 1, 1985.

[41] J.S. Vitter, M. Wang, and B.R. Iyer, "Data Cube Approximation and Histograms via Wavelets," *Proc. 1998 ACM CIKM Int'l Conf. Information and Knowledge Management,* 1998.

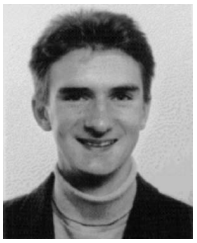
[42] M.P. Wand and M.C. Jones, "Kernel Smoothing," *Monographs on Statistics and Applied Probability,* Chapman and Hall, 1995.

[43] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Databases," *Proc. ACM SIGMOD,* pp. 103-114, June 1996.

George Kollios received the Diploma in electrical and computer engineering in 1995 from the National Technical University of Athens, Greece, and the MSc and PhD degrees in computer science from Polytechnic University, New York, in 1998 and 2000, respectively. He is currently an assistant professor in the Computer Science Department at Boston University. His research interests include temporal and spatiotemporal indexing, index benchmarking, and data mining. He is a member of the ACM and the IEEE Computer Society.

Dimitrios Gunopulos completed his undergraduate studies at the University of Patras, Greece (1990) and graduated with the MA and PhD degrees from Princeton University (1992 and 1995), respectively. He is an assistant professor in the Department of Computer Science and Engineering at the University of California, Riverside. His research is in the areas of data mining, databases, and algorithms. His research interests include efficient indexing techniques for moving points, approximating range queries, similarity searching in sequence databases, finding frequent sequences or sets, approximate set indexing, and local adaptive classification techniques. He has held positions at the IBM Almaden Research Center (1996-1998) and at the Max-Planck-Institut for Informatics (1995-1996). His research is supported by the US National Science Foundation (NSF) (including an NSF CAREER award), DoD, and ATT.

Nikos Koudas received the BSc degree from the University of Patras, Greece, the MSc degree from the University of Maryland at College Park, and the PhD degree from the University of Toronto, all in computer science. He is a member of the technical staff at AT&T Labs Research. His research interests include databases, information systems, and algorithms. He is an editor for the *Information Systems* journal and an associate information editor for *SIGMOD* online.



Stefan Berchtold received the MS degree from the Technical University of Munich in 1993. From 1993 to 1994, he worked in industry as a consultant. In 1995, he joined the University of Munich from which he received his PhD degree in 1997. From 1997 to 1998, he worked for AT&T Research Labs as a senior researcher. In 1999, he founded stb ag, a German software and consulting company for which he acts as the CEO. His research interests include the areas of data mining, data reduction, visualization, and high-dimensional indexing. He is a member of the IEEE and the IEEE Computer Society.