Using kernels to approximate multi-dimensional aggregate range queries over real attributes

Carlotta Domeniconi *Dimitrios Gunopulos †George Kollios ‡Vassilis J. Tsotras §

Technical Report UCR-CS-00-04

November 22, 2000

Abstract

Finding approximate answers to multi-dimensional range queries over real valued attributes has significant applications in data exploration and database query optimization. In this paper we consider the following problem: given a dataset of n points in the d-dimensional real space ($\Re^d$), and a query that specifies a range in each dimension, find a good approximation of the number of points in the dataset that satisfy the query, i.e. that fall within the specified d ranges.

The simplest approach to tackle this problem is to assume that the attributes are independent. More accurate estimators try to capture the joint data distribution of the attributes. In databases, such estimators include the construction of multi-dimensional histograms, random sampling, or the wavelet transform. In statistics, kernel estimation techniques are being used. However, the problem of setting the bandwidth parameters optimally has been addressed satisfactorily only for the one dimensional case. Approximately optimal bandwidth parameters in the multi-dimensional case have been obtained only for the special case in which the attributes are independent, the distribution along each dimension is Gaussian, and all bandwidths, for all kernels and dimensions, are to be set to the same value.

In [9], we have shown how to use multi-dimensional kernel density estimators to solve the multi-dimensional range query selectivity problem. Our technique generalizes in multiple dimensions the technique given in [3]. Kernel estimation is a generalization of sampling. Like sampling, finding a kernel estimator is efficient, and can be performed in one pass. In addition, kernel estimators produce a smoother approximation of the underlying density function, therefore achieving better results, as shown experimentally.

To achieve more accurate results, in this paper we address the problem of setting the bandwidth parameters optimally. Our technique locally adapts the bandwidth values of kernels along each dimension, by using information regarding the extension of points in the neighborhood.

Finally, we compare the accuracy of the proposed technique with existing methods using real and synthetic datasets. The experimental results show that the proposed technique has a robust and competitive bahavior across all the datasets we have considered.

*CS Dept., Univ. of California Riverside, Riverside, CA 92521, carlotta@cs.ucr.edu.

†CS Dept., Univ. of California Riverside, Riverside, CA 92521, dg@cs.ucr.edu.

‡CS Dept., Boston Univ., Boston, MA 02215, gkollios@cs.bu.edu.

§CS Dept., Univ. of California Riverside, Riverside, CA 92521, tsotras@cs.ucr.edu.

1 Introduction

Computing approximate answers to multi-dimensional range queries is a problem that arises in query optimization, data mining and data warehousing. The query optimizer requires accurate estimations of the sizes of intermediate query results in the evaluation of different execution plans. Recent work also shows that top-k queries can be mapped to multi-dimensional queries [4, 7], so selectivity estimation techniques can be used to optimize top-k queries.

The problem of approximating multi-dimensional range queries is also relevant for data mining applications. Answering range queries, in fact, is one of the simpler data exploration tasks. In this context, the user defines a specific region of the dataset that is worth exploring, and asks queries to find the characteristics of this region (like the number of points in the interior of the region, the average value or the sum of the values of attributes in the region). Consider, for example, a dataset that records readings of different environmental variables, such as types of pollution, at various locations. In exploring this dataset the user may be interested in answering range queries similar to: find how many locations exist for which the values of the given pollution variables are within a specified range. The user may want to restrict the answers to a given geographical range too. The size of such datasets makes exact answers intractable, and only efficient approximation algorithms can make this data exploration task interactive.

In data warehousing, datasets can be very large. Answering aggregate queries exactly can be computationally expensive. It is therefore very important to find approximate answers to aggregate queries quickly in order to allow the user to explore the data.

In this paper we address the problem of estimating the selectivity of multi-dimensional range queries when the datasets have numerical attributes with real values. The range queries we consider are intersections of ranges, each range being defined on a single attribute. In the multi-dimensional attribute space, the queries are then hyper-rectangles with faces parallel to the axes. Solving such a range query exactly involves counting how many points fall in the interior of the query. When the number of dimensions increases, recent results show [24] that the query time is linear to the size of the dataset.

The main approach to solve the selectivity estimation problem has been to compute a non-parametric density estimator to the data distribution function. The methods suggested in the literature employ different techniques to find the density estimator for attributes with finite discrete domains. They include computing multi-dimensional histograms [18] [1] [12] [2], using the wavelet transformation [22] [16], SVD [18] or the discrete cosine transform [14] on the data, using kernel estimators [3], and sampling [17] [13] [10].

Density estimator techniques attempt to define a function that approximates the data distribution. Since we must be able to derive the approximate solution to a query quickly, the description of the function must be kept in memory. Further, we may have to answer queries on many datasets, so the description cannot be large. The success of the different methods depends on the simplicity of the function, the time it takes to find the function parameters, and the number of parameters we store for a given approximation quality.

Multi-dimensional density estimation techniques are typically generalizations of very successful one-dimensional density estimators. In general, in one dimension, estimators of small size (histograms, kernels, sampling) can be used to effectively approximate the data distribution. Indeed, one of the techniques used to solve the multi-dimensional problem is to assume that attributes are independent, and therefore an estimator for multiple dimensions can be obtained by multiplying one-dimensional estimators.

Unfortunately, if the independence assumption does not hold, as it frequently happens, the problem

of estimating the result of range queries in multiple dimensions becomes tougher as the dimensionality increases. One of the reasons is that the volume of the space increases exponentially with the dimensionality, and datasets of any size become sparse and do not allow accurate density estimation in any local area. This problem is referred to as the curse of dimensionality [24]. Finding density estimators for combinations of attributes also allows us to find out whether the independence assumption holds for a set of attributes or not. This is of independent importance for the query optimizer: many query optimizers [20] compute query execution costs under the attribute independent assumption. Knowing where this does not hold allows one to keep better statistics for these sets of attributes.

1.1 Our Contribution

In [9] we have shown how to use multi-dimensional kernel density estimators to solve the multi-dimensional range query selectivity problem. Our technique generalizes in multiple dimensions the technique given in [3]. Kernel estimation is a generalization of sampling. Like sampling, finding a kernel estimator is efficient, and can be performed in one pass. In addition, kernel estimators produce a smoother approximation of the underlying density function, therefore achieving better results, as shown experimentally.

To achieve more accurate results, here we address the problem of setting the bandwidth parameters optimally. Our technique locally adapts the bandwidth values of kernels along each dimension, by using information regarding the extension of points in the neighborhood.

We compare the performance of our method with other techniques by using both synthetic and real-life datasets with real valued attributes.

The paper is organized as follows. The following section formally defines the problem. In section 3 we briefly describe statistical estimators. We review the kernel estimator technique for multi-dimensional data, presented in [9], in section 4. In section 5 we describe our new heuristic to locally adapt the bandwidths of kernels. Section 6 discusses the experimental results, and we conclude in section 7.

2 Problem Description

Let R be a relation with d attributes and n tuples. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_d\}$ be the set of these 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 defined by the attributes. Let V_i be the set of values of A_i that are present in R. Since the values are real, each could be distinct and therefore $|V_i|$ can be as large as n.

The range queries we consider are of the form $(a_1 \leq R.A_1 \leq b_1) \wedge \ldots \wedge (a_d \leq R.A_d \leq b_d)$. All a_i, b_i are assumed to be in $[0, 1]$. Such a query is a hyper-rectangle with faces parallel to the axes. The *selectivity* of a range query Q, $sel(R, Q)$, is the number of tuples in the interior of the hyper-rectangle it represents.

Since n can be very large, the problem of approximating the selectivity of a given range query Q arises naturally. The main problem is how to preprocess R so that accurate estimations can be derived from a smaller representation of R without scanning the entire relation, or counting exactly the number of points in the interior of Q.

Let $f(x_1, \ldots x_d)$ be a d-dimensional, non-negative function that is defined in $[0, 1]^d$ and has the property

that

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

We call f a *probability density function.* The value of f at a specific point $\mathbf{x} = (x_1, \ldots x_d)$ of the d-dimensional space is the limit of the probability that a tuple exists in area U around $\mathbf{x}$ over the volume of U, when U shrinks to $\mathbf{x}$.

For a given f with these properties, to find the selectivity of the query $(a_1 \leq R.A_1 \leq b_1) \wedge \ldots \wedge (a_d \leq R.A_d \leq b_d)$, we compute the integral of f in the interior of the given query Q:

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

For a given R and f, f is a good *estimator* of R with respect to range queries if for any range query Q, the selectivity of Q on R and the selectivity of Q on f multiplied by n are similar. To formalize this notion, we define the following error metrics (also used by [22]).

For a given query Q, we define the *absolute error* of Q to be simply the difference between the real value and the estimated value:

$$\epsilon_{abs}(Q, R, f) = |sel(R, Q) - n\ sel(f, Q)|.$$

The *relative error* of a query Q is generally defined as the ratio of the absolute error over the selectivity of the query. Since in our case a query can be empty, we follow [22] in defining the relative error as the ratio of the absolute error over the maximum of the selectivity of Q and 1:

$$\epsilon_{rel}(Q, R, f) = \frac{|sel(R, Q) - n\ sel(f, Q)|}{\max(1, sel(R, Q))}.$$

Finally the *modified relative error* of a query Q is the absolute error over the minimum of the actual selectivity and the estimated selectivity of Q (since this can be zero, we again have to take the maximum with 1):

$$\epsilon_{rel\_mod}(Q, R, f) = \frac{|sel(R, Q) - n\ sel(f, Q)|}{\max(1, \min(sel(R, Q), n\ sel(f, Q)))}.$$

To represent the error of a set of queries, we define the *p-norm average error.* Given a query workload $\{Q_1, \ldots, Q_k\}$ comprising of k queries, R, f, and an error metric ϵ that can be any of the above, the p-norm average error for this workload is:

$$\| \epsilon \|_p = (\frac{1}{k} \sum_{1 \leq i \leq k} \epsilon(Q_i, R, f)^p)^{\frac{1}{p}}.$$

We can also define different aggregate queries such as the sum on one attribute:

$$sum(R, Q, i) = \sum_{(x_1,\ldots,x_d) \in R \cap Q} x_i,$$

or the average

$$ave(R, Q, i) = \frac{\sum_{(x_1,\ldots,x_d) \in R \cap Q} x_i}{sel(R, Q)}.$$

Following [21], we can approximate such a query using the density estimator f:

$$sum(f, Q, x_i) = \int_Q x_i\ f(x_1, \ldots, x_d) dx_1 \ldots dx_d.$$

3 Statistical Estimators

The simplest statistical method for selectivity estimation is sampling. One finds a random subset S of size b of the tuples in R. Given a query Q, the selectivity $sel(S,Q)$ is computed. The value $\frac{n}{b}sel(S,Q)$ is used to estimate $sel(R,Q)$. Sampling is simple and efficient, and so it is widely used for estimating the selectivity [20] [5] [8] or for on-line aggregation [11]. Sampling can be used to estimate the selectivity of a query regardless of the dimensionality of the space, and can directly be applied to real domains.

More sophisticated kernel estimation statistical techniques [23] [6] have rarely been applied in database problems. One of the reasons is that in statistics a dataset is considered an instance of a probability distribution function, and the goal is to approximate the probability distribution function itself. On the other hand, in databases the goal is simply to approximate the dataset. Recently [3] used kernels to estimate the selectivity of one-dimensional range queries on metric attributes.

One similar statistical technique is clustering the dataset and using a Gaussian function to model each cluster [21]. This technique can be quite accurate if the clusters themselves can be accurately modeled by multi-dimensional Gaussian functions. However, even assuming that this is the case, the technique requires clustering the dataset, and therefore it is much less efficient than simple sampling.

4 Multi-dimensional kernel density estimators

In this section we review the method presented in [9] for using multi-dimensional kernel density estimators to solve the multi-dimensional range query selectivity problem.

For the problem of computing the query selectivity, all the proposed techniques compute a density estimation function. Such function can be thought as an approximation of the probability distribution function, of which the dataset at hand is an instance. It follows that statistical techniques which approximate a probability distribution, such as kernel estimators, are applicable to address the query estimation problem. The problem of estimating an underlying data distribution has been studied extensively in statistics [19] [23].

Kernel estimation is a generalized form of sampling. The basic step is to produce a uniform random sample from the dataset. As in random sample, each sample point has weight one. In kernel estimation however, each point distributes its weight over the space around it.

A *kernel function* describes the form of the weight distribution. Generally, a kernel function distributes most of the weight of the point in the area very close the point, and tapers off smoothly to zero as the distance from the point increases. If two kernel centers are close together, there may be a considerable region where the non-zero areas of the kernel functions overlap, and both distribute their weight in this area. Therefore, a given location in the data space gets contributions from each kernel point that is close enough to this location so that its respective kernel function has a non zero value. Summing up all the kernel functions we obtain a density function for the dataset.

Let us consider the one dimensional case first. Assume that R contains tuples with one attribute A whose domain is $[0,1]$. Let S be a random subset of R (our sample). Also assume that there is a function $k_i(x)$ for each tuple t_i in S, with the property that $\int_{[0,1]} k(x)dx = 1$. Then the function

$$f(x) = \frac{1}{n}\sum_{t_i \in S} k_i(x - t_i)$$

is an approximation of the underlying probability distribution according to which R was drawn.

To approximate the selectivity of a query Q of the form $a \leq R.A \leq b$, one has to compute the integral of the probability function f in the interval $[a, b]$:

$$\sigma(f, Q) = \int_{[a,b]} f(x) = \frac{1}{n} \sum_{t_i \in S} \int_{[a,b]} k_i(x - t_i).$$

As defined, kernel estimation is a very general technique. [19] shows that any non-parametric technique for estimating the probability distribution, including histograms, can be recast as a kernel estimation technique for appropriately chosen kernel functions.

In practice the functions $k_i(x)$ are all identical. The approximation can be simplified to

$$f(x) = \frac{1}{n} \sum_{t_i \in S} k(x - t_i).$$

To use kernels in d-dimensions we have to provide a d-dimensional kernel function.

For a dataset R, let S be a set of tuples drawn from R at random. 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 R is

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

and the estimation of the selectivity of a d-dimensional range query Q is

$$sel(f, Q) = \int_{[a,b]^d \cap Q} f(x_1, \ldots, x_d) = \frac{1}{n} \sum_{t_i \in S} \int_{[a,b]^d \cap Q} k(x_1 - t_{i_1}, \ldots, x_d - t_{i_d}) dx_1 \ldots dx_d.$$

It has been shown that the shape of the kernel function does not affect the approximation substantially [6]. What is important is the standard deviation of the function, or, its bandwidth. Therefore, we choose a kernel function that it is easy to integrate. The Epanechnikov kernel function has this property [6]. The d-dimensional Epanechnikov kernel function centered at 0 is

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

if, for all i, $|\frac{x_i}{B_i}| < 1$, and 0 otherwise (Figure 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 non-zero diameter of the kernel becomes smaller.

There are two problems that we have to solve before we can use the multi-dimensional kernel estimation method. The first is setting the bandwidth parameters. The second problem is that in high dimensions many tuples, and therefore many samples, are likely to be close to one of the faces of the $[0, 1]^d$ cube. If a kernel is close to the space boundary and its bandwidth goes beyond it, then part of the volume that is covered by the kernel function lies outside the data (and the query) space. The result is that these points distribute less

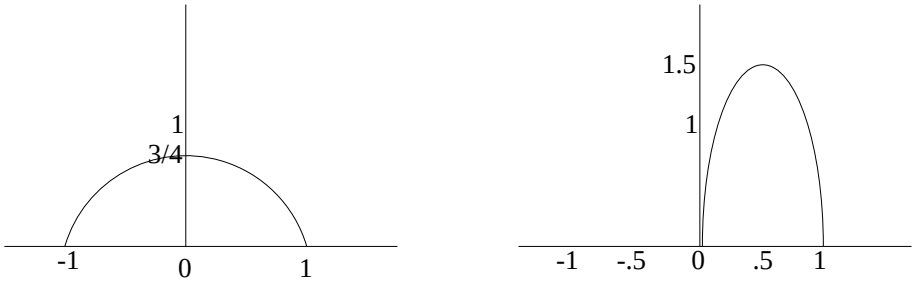

Figure 1: The one-dimensional Epanechnikov kernel, with $B = 1$, centered around the origin, and, with $B = 1/2$, centered at 0.5.

than weight 1 to the area around them, and so $\int_{[0,1]^d} f(x_1, \ldots, x_d)dx_1 \ldots dx_d < 1$. This problem is referred to as the boundary problem.

Both problems have been addressed before in statistics [23]. No efficient solution exists for finding the optimal bandwidths. To get an initial estimate for the bandwidths, in [9] we used Scott's rule [19] in d-dimensional space: $B_i = \sqrt{5}\ s_i\ |S|^{-\frac{1}{d+4}}$, where s_i is the standard deviation of the sample on the i-th attribute. This rule is derived using the assumption of gaussian data distribution, therefore in general it oversmoothes the actual underlying function. We discuss our new technique for estimating bandwidths in section 5. To solve the second problem, we project the parts of the kernel function that lie outside $[0,1]^d$ back into the data space. The complexity of this projection increases with the dimensionality, because each d-dimensional corner of $[0,1]^d$ partitions $\mathcal{R}^d$ into 2^d quadrangles, and we have to find the intersection of the kernel function with each quadrangle.

4.1 Computing the selectivity

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:

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

$$\frac{1}{n} \int_{[a_1,b_1]\times\ldots\times[a_d,b_d]} \sum_{1\le i\le b} (\frac{3}{4})^d \frac{1}{B_1 B_2 \ldots B_d} \prod_{1\le j\le d} (1 - (\frac{x_j - X_{ij}}{B_j})^2))\ dx_1 \ldots dx_d =$$

$$\frac{1}{n}(\frac{3}{4})^d \frac{1}{B_1 B_2 \ldots B_d} \sum_{1\le i\le b} \int_{[a_1,b_1]} (1 - (\frac{x_1 - X_{i1}}{B_1})^2) \ldots \int_{[a_d,b_d]} (1 - (\frac{x_d - X_{id}}{B_d})^2)\ dx_d \ldots dx_1$$

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

5 Locally adaptive estimation of bandwidths

Current work in statistics on the problem of setting the bandwidth parameters optimally has addressed only the one dimensional problem satisfactorily [19]. Approximately optimal bandwidth parameters in the multi-dimensional case have been obtained only for the special case in which the following conditions are all true: (i) the attributes are independent, (ii) the distribution along each dimension is Gaussian and (iii) all bandwidths, for all kernels and dimensions, are to be set to the same value [19].

Kernel density estimators place a kernel at each data point. If the kernels have the appropriate widths, the sum of the kernel contributions at any point can be shown to converge to the value of the true density function, as the number of data points tends to infinity. Of course, we can afford only a finite number of data points, and therefore an approximation of the underlying density function. To achieve accuracy, the challenge consists in finding a mechanism to properly set the bandwidth parameters.

Kernel-based methods are nearest-neighbor-type algorithms: to obtain the density estimate at a given point, assuming far-off points have negligible contribution to the sum, one has to consider only the kernel contributions of the nearest neighbors. It is therefore reasonable to adapt the bandwidths of a kernel centered at a specific point according to the extension of the neighborhood of its center, so that the kernel will mainly contribute to the density estimation of points within that same local neighborhood.

Therefore, we develop a heuristic (ADAptive BANDwidth) to locally adapt the bandwidths of kernels along each dimension, according to the extension of points within the neighborhood. In details, the heuristic works as follows.

A uniform random sample S of a given size, say $|S|$, is first produced. Let R be the original dataset, and $|R|$ its size. Each point in S distributes its weight over the space around it. We want each kernel to distribute its weight over an equal number of points in the dataset, i.e. as many points as $\frac{|R|}{|S|}$. For each point $\mathbf{s} \in S$, we compute its distance from the data points in R. Among these $|R|$ distances, we identify the one that corresponds to the $\frac{1}{|S|}$-quantile, i.e. the distance at position $\lceil \frac{|R|}{|S|} \rceil$ in the sorted sequence of distances. Let D be such quantile. D can be seen as the distance of $\mathbf{s}$ from the vertex of the hypercube centered at $\mathbf{s}$ that includes a neighborhood of $\lceil \frac{|R|}{|S|} \rceil$ points. To set the bandwidth B of a kernel centered at $\mathbf{s}$, we compute the projection of D along each dimension, resulting in $B = \frac{2 \times D}{\sqrt{d}}$. Each kernel has one bandwidth value B associated with it, valid for all the d dimensions. The algorithm, therefore, stores $(d+1)$ numbers per kernel: d values for the coordinates of the center, and one value for the bandwidth. Figure 2 gives the outline of the algorithm.

5.1 Running Time

Computing a kernel density estimator with $|S|$ kernels, as described above, can be done in two dataset passes. During the first pass, a random sample of size $|S|$ is taken. During the second pass, an approximation of the $\frac{1}{|S|}$-quantiles for the points in S is computed.

In the implementation of AdaBand, to efficiently estimate the quantiles we use the technique described in [15], which guarantees arbitrarily tight error bounds and, for a given desirable accuracy, it allows to estimate the optimal space complexity.

Given a d-dimensional dataset R with n points and input parameter $EstSize$ (Estimator Size):

1. Set $KCount = \frac{EstSize}{d+1}$;
2. Take a random sample S of size $KCount$;
3. For each point $\mathbf{s} \in S$:
   (a) Compute the n distances $d_1, \ldots, d_n$ of $\mathbf{s}$ from the points in R;
   (b) Compute the $\frac{1}{KCount}$-quantile D of the distances $d_1, \ldots, d_n$;
   (c) Set $B_i = \frac{2 \times D}{\sqrt{d}}$, for $i = 1, \ldots, d$, where B_i is the bandwidth along dimension i of the kernel centered at $\mathbf{s}$.

Figure 2: The AdaBand algorithm

6 Experimental Evaluation

In our experiments we compare the performance of AdaBand, Scott's rule and Random Sampling on synthetic and real-life datasets with real valued atributes. We examine the behavior of the methods as additional space for storing the estimator becomes available. We also evaluate the accuracy of the methods as the dimensionality of data increases.

6.1 Synthetic datasets

We have designed the following synthetic datasets.

OneGaussian. This dataset contains 10^6 5-dimensional points drawn according to a Gaussian distribution with standard deviation set to 5 along each dimension. In this situation Scott's rule finds the optimal bandwidth's values.

DiffGaussian. This dataset contains 10^6 10-dimensional points drawn according to 25 Gaussian distributions with mean values randomly chosen within the range $[25, 74]$, and standard deviation values randomly chosen within the set $\{0.1, 0.2, 0.3, \ldots, 1.0\}$. Each Gaussian generates the same number of data points.

NoisyGaussian. This dataset contains 10^6 10-dimensional points. 25% of the data ($250,000$ points) is uniformly distributed random noise. The remaining $750,000$ points are generated according to 25 Gaussian distributions with mean values randomly chosen again within the range $[25, 74]$, and standard deviation values for all dimensions set to 0.25. Each Gaussian generates the same number of data points.

6.2 Real datasets

We use the Forest Cover Dataset from the UCI KDD archive[1]. This dataset was obtained from the US Forest Service (USFS). It includes $590,000$ points, and each point has 54 attributes, 10 of which are numerical. In

[1] available from kdd.ics.uci.edu/summary.data.type.html

our experiments we use subsets of 5 numerical attibutes. In this dataset the distribution of the attributes is non-uniform, and there are correlations between pairs of attributes.

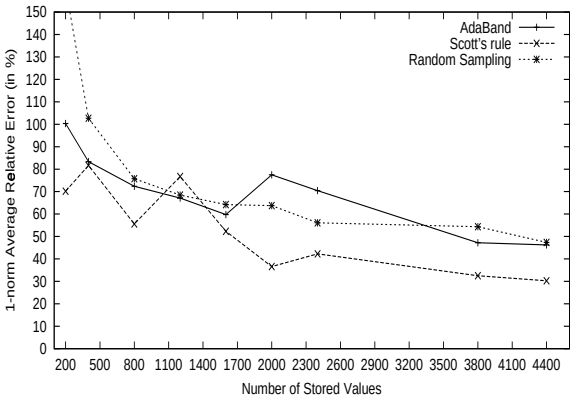

Figure 3: OneGaussian dataset, query workload 3, 10-dim.

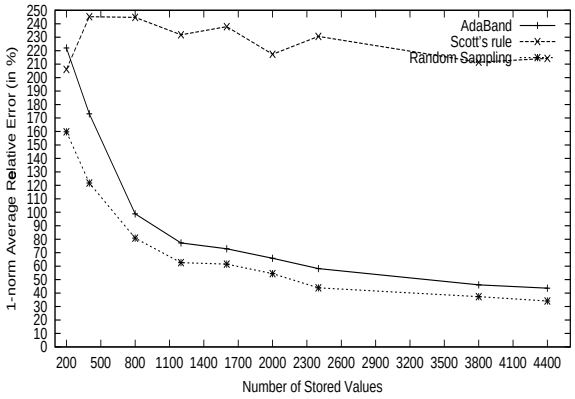

Figure 4: DiffGaussian dataset, query workload 2, 10-dim.

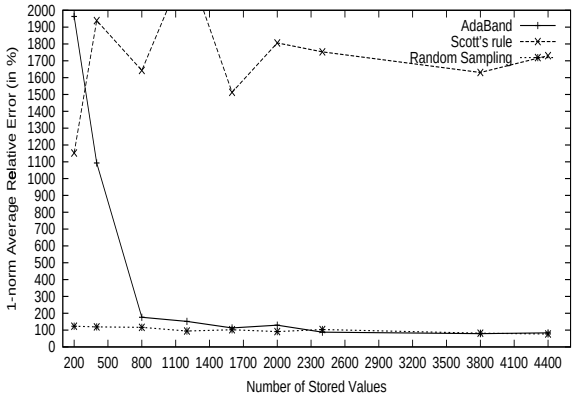

Figure 5: DiffGaussian dataset, query workload 3, 10-dim.

6.3 Query workloads

To evaluate the techniques we created workloads of 3 types of queries. We create a workload 1 of random queries with selectivity approximately 10%, and a workload 2, of random queries with selectivity approximately 1%. These workloads comprise 10^4 queries each. We also create a workload 3 of $20,000$ queries of the form $(R.A_1 < a_1) \wedge \ldots \wedge (R.A_d < a_d)$ for a randomly chosen point $(a_1, \ldots, a_d) \in [0,1]^d$.

For each workload we compute the average absolute error $\| e_{abs} \|_1$ and the average relative $\| e_{mod} \|_1$ error.

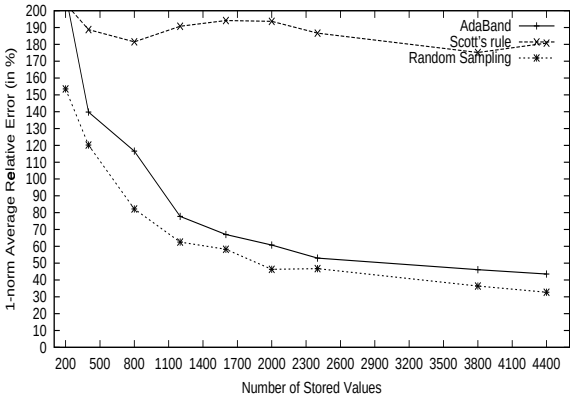

Figure 6: NoisyGaussian dataset, query workload 2, 10-dim.

6.4 Experimental Results

The OneGaussian dataset has been designed to test how close Adaband performs to the Scott's rule, which finds the optimal bandwidth's values for this dataset. Figure 3 shows the results for query workload 3. As expected, the Scott's rule shows the best performance, but both AdaBand's and Random Sampling's performances are not too far from it. This means that we don't loose too much in performance with our adaptive technique in the ideal case for the Scott's rule.

Figures 4 and 5 show the results for the DiffGaussian dataset on query workloads 2 and 3. AdaBand outperforms by far the Scott's rule in both cases. The Scott's rule is not able to scale its performance as the size of the estimator increases, whereas our technique is capable of adapting the bandwidths according to the number of kernels that become available. Random sampling shows a similar performance to AdaBand. We observe the same behavioral pattern for the NoisyGaussian dataset on query workload 2. The results for this case are shown in Figure 6.

Figures 7 and 8 show the results for the Forest Cover dataset in 5 dimensions on query workloads 1 and 2. The three methods show similar performances, with AdaBand and Scott's rule being slightly better on query workload 2.

7 Conclusions

We observe that our technique manifests a robust and competitive behavior across all the datasets we have considered in our experiments. Moreover, it has the advantage of being rather simple and can be implemented efficiently.

In the future, we plan to conduct more extensive experiments with real data in higher dimensions.

References

[1] A. Aboulnaga, S. Chaudhuri. Self-tuning Histograms: Building Histograms Without Looking at Data. in *Proc. of the 1999 ACM SIGMOD Intern. Conf. on Management of Data*, June 1999.

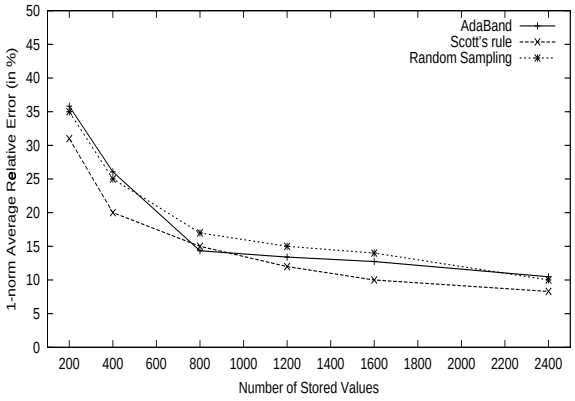

Figure 7: Forest Cover dataset, query workload 1, 5-dim.

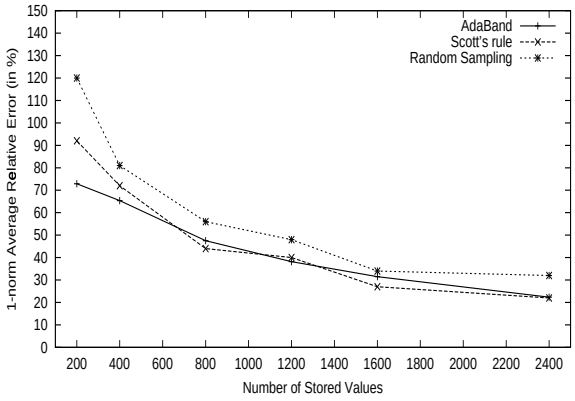

Figure 8: Forest Cover dataset, query workload 2, 5-dim.

[2] S. Acharya, V. Poosala, S. Ramaswamy. Selectivity Estimation in Spatial Databases. in *Proc. of the 1999 ACM SIGMOD Intern. Conf. on Management of Data*, June 1999.

[3] B. Blohsfeld, D. Korus, B. Seeger. A Comparison of Selectivity Estimators for Range Queries on Metric Attributes. In *Proc. of the 1999 ACM SIGMOD Intern. Conf. on Management of Data*, June 1999.

[4] S. Chaudhuri and L. Gravano Evaluating Top-K Selection Queries. In *Proc. 25th Intern. Conf. on Very Large Data Bases (VLDB-99)*, Edinburgh, Scotland, Sept. 1999.

[5] S. Chaudhuri, R. Motwani, V.R. Narasayya. Random Sampling for Histogram Construction: How much is enough? In *Proc. of the 1998 ACM SIGMOD Intern. Conf. on Management of Data*, June 1998.

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

[7] D. Donjerkovic and R. Ramakrishnan. Probabilistic Optimization of Top N Queries. In *Proc. 25th Intern. Conf. on Very Large Data Bases (VLDB-99)*, Edinburgh, Scotland, Sept. 1999.

[8] P.B. Gibbons, Y. Matias. New Sampling-Based Summary Statistics for Improving Approximate Query Answers. In *Proc. of the 1998 ACM SIGMOD Intern. Conf. on Management of Data*, June 1998.

[9] D. Gunopulos, G. Kollios, V. Tsotras, and C. Domeniconi. Approximating multi-dimensional aggregate range queries over real attributes. In *Proc. of the 19th ACM SIGMOD Intl. Conference on Management of Data*, Dallas 2000.

[10] P.J. Haas, A.N. Swami. Sequential Sampling Procedures for Query Size Estimation. In *Proc. of the 1992 ACM SIGMOD Intern. Conf. on Management of Data*, June 1992.

[11] J.M. Hellerstein, P.J. Haas, H. Wan. Online Aggregation. In *Proc. of the 1997 ACM SIGMOD Intern. Conf. on Management of Data*, May 1997.

[12] F. Korn, T. Johnson and H. Jagadish. Range Selectivity Estimation for Continuous Attributes. In *Proc. of 11th Int. Conf. on SSDBMs*, 1999.

[13] R.J. Lipton, J.F. Naughton. Practical Selectivity Estimation through Adaptive Sampling. In *Proc. of the 1990 ACM SIGMOD International Conference on Management of Data*, May 1990.

[14] J. Lee, D. Kim and C. Chung. Multi-dimensional Selectivity Estimation Using Compressed Histogram Information. in *Proc. of the 1999 ACM SIGMOD Intern. Conf. on Management of Data*, June 1999.

[15] G. S. Manku, S. Rajagopalan, B. G. Lindsay. Approximate Medians and other Quantiles in One Pass and with Limited Memory. In *Proc. of the 17th ACM SIGMOD Intl. Conference on Management of Data*, Seattle, WA, 1998.

[16] Y. Matias, J. Scott Vitter, M. Wang. Wavelet-Based Histograms for Selectivity Estimation. In *Proc. of the 1998 ACM SIGMOD Intern. Conf. on Management of Data*, June 1998.

[17] F. Olken, D. Rotem. Random Sampling from database Files:A Survey. In *Proc. of 5th International Conference on Statistical and Scientific Database Management*, Charlotte, N.C., 1990.

[18] V. Poosala and Y.E. Ioannidis. Selectivity Estimation Without the Attribute Value Independence Assumption. In *Proc. of 23rd Intern. Conf. on Very Large Data Bases*, August 1997.

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

[20] P.G. Selinger, M.M. Astrahan, D.D. Chamberlin, R.A. Lorie, T.G. Price. Access Path Selection in a Relational Database Management System. In *Proc. of the 1979 ACM SIGMOD Intern. Conf. on Management of Data*, June 1979.

[21] Jayavel Shanmugasundaram, Usama Fayyad, Paul Bradley. Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions In *Fifth ACM SIGKDD Inter. Conf. on Knowledge Discovery and Data Mining*, August 1988.

[22] J.S. Vitter, M. Wang, B. R. Iyer. Data Cube Approximation and Histograms via Wavelets. In *Proc.of the 1998 ACM CIKM Intern. Conf. on Information and Knowledge Management*, November 1998.

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

[24] R. Webber, H.-J. Schek and S. Blott. A Quantitative Analysis and Performance Study for Similarity Search Methods in High-Dimensional Spaces. In *Proc. of 24rd Intern. Conf. on Very Large Data Base*, August 1998.