An Efficient Approach for Approximating Multi-dimensional Range Queries and Nearest Neighbor Classification in Large Datasets

Carlotta Domeniconi CARLOTTA@CS.UCR.EDU
Dimitrios Gunopulos DG@CS.UCR.EDU
Computer Science Department, University of California Riverside, Riverside, CA 92521 USA

Abstract

We propose a locally adaptive technique to address the problem of setting the bandwidth parameters optimally for kernel density estimation. Our technique is efficient and can be performed in only two dataset passes. We also show how to apply our technique to efficiently solve range query approximation, classification and clustering problems for very large datasets. We validate the efficiency and accuracy of our technique by presenting experimental results on a variety of both synthetic and real datasets.

1. Introduction

Classification and clustering (Bradley, Fayyad & Reina, 1998) are key steps for many tasks in data mining, whose aim is to dig out unknown relationships and/or patterns from large set of data. A variety of methods has been proposed to address such problems. However, the inherent complexity of both problems is high, and the application of known techniques on large datasets can be time and resource consuming.

A simple and appealing approach to classification is the K-nearest neighbor method (McLachlan, 1992): it finds the K-nearest neighbors of the query point $\mathbf{x}_0$ in the dataset, and then predicts the class label of $\mathbf{x}_0$ as the most frequent one occuring in the K neighbors. It produces continuous and overlapping neighborhoods, and uses a different neighborhood for each individual query. Therefore, it results in a highly stable procedure with respect to perturbations of the data. However, when applied on large datasets, the time required to compute the neighborhoods (i.e., the distances of the query from the points in the dataset) becomes prohibitive, making exact answers intractable.

Another relevant problem for data mining applications is the approximation of multi-dimensional range queries. 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). As such, the problem of finding the K-nearest neighbors of a given query point reduces to this problem. However, when the number of dimensions increases, recent results (Webber, Schek & Blott, 1998) show that the query time is linear to the size of the dataset. Thus the problem of efficiently approximating the selectivity of range queries arises naturally.

In general, only efficient approximation algorithms can make data exploration tasks in large datasets interactive. Our method relies on the observation that the density of the dataset contains useful information for both the classification and clustering tasks. For classification, the main point is that, given a query, the values of the class density functions over the space around it quantify the contribution of the correspondent class within the neighborhood of the query point. The larger the contribution of a given class is, the larger is the likelihood for the query to belong to that class. As for clustering, local maxima of the density function of the dataset could represent cluster centers. A hill-climbing procedure applied at a given point, would identify its density attractor. Furthermore, the value of the integral of the estimated density, computed over the volume defined by the range query, will give an approximation of its selectivity.

For a compact representation of the density of a dataset, we use kernel density estimation methods. Kernel methods pose the problem of setting the bandwidth parameters. Current work on this problem in statistics has addressed only the one dimensional case satisfactorily (Scott, 1992). 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 (Scott, 1992). For example, one solution to the problem of computing the bandwidths is given by Scott's rule (Scott, 1992), which estimates one bandwidth parameter per attribute, by setting its value to a quantity proportional to the standard deviation of the sample on that attribute. The rule assumes attribute independence.

To achieve more accurate results we propose an adaptive bandwidth estimation technique that adapts the bandwidth of the kernels using local information, and does not assume independence between the attributes. The setting of the bandwidth for each kernel is based on the extention of the points in the neighborhood, and each kernel uses the same bandwidth for all dimensions. Our technique is efficient and can be performed in only two dataset passes.

Using the range query approximation problem as a benchmark, we show the performance improvement we achieve with our method over Scott's rule by using a variety of both synthetic and real datasets. We then show how to apply our technique to efficiently solve classification and clustering problems. For space limitation, we focus on classification and show the results we obtained on synthetic datasets.

2. The Range Query Approximation Problem

Let R be a dataset of n points, each with d real (normalized) attributes. We consider range queries of the form $(a_1 \leq R.A_1 \leq b_1) \wedge \ldots \wedge (a_d \leq R.A_d \leq b_d)$. The *selectivity* of a range query Q, $sel(R,Q)$, is the number of points 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. Approaches proposed to address this problem include multidimensional histograms (Ioannidis & Poosala, 1999; Gunopulos et al., 2000), kernels (Shanmugasundaram et al., 1999; Gunopulos et al., 2000), and wavelets (Vitter et al. 1998; Chakrabarti et al., 2000).

To formalize the notion of approximating the selectivity of range queries, let $f(x_1,\ldots x_d)$ be a d-dimensional, non-negative function, defined in $[0,1]^d$ and with the property $\int_{[0,1]^d} f(x_1,\ldots x_d)dx_1 \ldots dx_d = 1$. f is a *probability density function*. The value of f at a specific point $\mathbf{x} = (x_1,\ldots x_d)$ 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 $\mathbf{x}$. Then, for a given such f, to find the selectivity of a query, we compute the integral of f in the interior of the given query Q: $sel(f,Q) = \int_{[a_1,b_1]\times\ldots\times[a_d,b_d]} f(x_1,\ldots x_d)dx_1 \ldots dx_d$.

Following Vitter et al. (1998), we define the *absolute error* of a given query 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 Vitter et al. (1998) in defining the relative error as the ratio of the absolute error over the maximum of the selectivity of Q and 1: $\epsilon_{rel}(Q,R,f) = \frac{|sel(R,Q) - n\ sel(f,Q)|}{\max(1, sel(R,Q))}$.

3. Multi-dimensional kernel density estimators

All the proposed techniques for approximating the query selectivity 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 (Scott, 1992; Wand & Jones, 1995), such as kernel estimators, are applicable to address the query estimation problem.

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 $\int_{[0,1]^d} k(x_1,\ldots,x_d)dx_1 \ldots dx_d = 1$. 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 (Cressie, 1993). The key feature is the standard deviation of the function, or, its bandwidth. Therefore, we choose a kernel function that it is easy to integrate, i.e. the d-dimensional Epanechnikov kernel function (Cressie, 1993), whose equation 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 $|\frac{x_i}{B_i}| < 1$ for all i, and 0 otherwise.

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, also the non-zero diameter of the kernel becomes smaller.

To estimate the bandwidths, typically Scott's rule (Scott, 1992) is used: $B_i = \sqrt{5}\ s_i\ |S|^{-\frac{1}{d+4}}$, 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. The rule assumes attribute independence. Other approaches for setting the bandwidths, such as one-dimensional least squares cross-validation, also assume attribute independence (Park & Turlach, 1992).

3.1 Computing the selectivity

In Gunopulos et al. (2000) we have shown how to use multi-dimensional kernel density estimators to efficiently address the multi-dimensional range query selectivity problem. We used Scott's rule for setting the bandwidths. We presented an experimental study that shows performance improvements over traditional techniques for density estimation, including sampling (Haas & Swami, 1992), multi-dimensional histograms (Poosala & Ioannidis, 1997), and wavelets (Vitter et al., 1998). The main advantage of kernel density estimators is that the estimator can be computed very efficiently in one dataset pass, during which we both sample the dataset and approximate the standard deviation along each attribute.

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: $sel(f, [a_1, b_1] \times \ldots \times [a_d, b_d]) = \frac{1}{n} \int_{[a_1,b_1]\times\ldots\times[a_d,b_d]} (\sum_{1\leq i\leq |S|} k_i(x_1, \ldots, x_d) dx_1 \ldots dx_d) = \frac{1}{n} \int_{[a_1,b_1]\times\ldots\times[a_d,b_d]} \sum_{1\leq i\leq |S|} (\frac{3}{4})^d \frac{1}{B_1 B_2 \ldots B_d} \prod_{1\leq j\leq d} (1 - (\frac{x_j - X_{ij}}{B_j})^2)) \, dx_1 \ldots dx_d = \frac{1}{n} (\frac{3}{4})^d \frac{1}{B_1 B_2 \ldots B_d} \sum_{1\leq i\leq |S|} \int_{[a_1,b_1]} (1 - (\frac{x_1 - X_{i1}}{B_1})^2) \ldots \int_{[a_d,b_d]} (1 - (\frac{x_d - X_{id}}{B_d})^2)\, dx_d \ldots dx_1$. It follows that, for a sample of $|S|$ tuples, $sel(f, Q)$ can be computed in $O(d|S|)$ time.

4. Locally adaptive bandwidths

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. This allows us to take into account local attribute correlations: kernels with more points close to them (according to the L_2 distance metric) will have smaller bandwidths than those with fewer points close to them. Real life data often present correlations among attributes, and therefore performance benefits from this approach (Scott, 1992).

As a consequence, we develop a heuristic (ADAptive BANDwidth) that locally adapts the bandwidths of kernels, according to the extension of points within the kernel neighborhood. AdaBand uses the same bandwidth for all the dimensions of a given kernel, but changes the bandwidth from kernel to kernel. 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 (and double it to avoid possible uncovered areas), 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 1 gives the outline of the algorithm.

For comparison purposes, we have also performed experiments in which we estimate a bandwidth value for each dimension and each kernel by using a localized standard deviation at the kernel's center along each dimension. In this case we store $2d$ numbers per kernel, and therefore the sample size is reduced to $\frac{EstSize}{2d}$. We have observed that the loss due to the reduced sample size overcomes the gain achieved by storing a distinct bandwidth value for each dimension. AdaBand, instead, seems to capture sufficient local information by storing one bandwidth per kernel, without over penalizing the sample size.

4.1 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

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 1. The AdaBand algorithm

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 Manku, Rajagopalan and Lindsay (1998), which guarantees arbitrarily tight error bounds and, for a given desirable accuracy, allows the estimation of the optimal space complexity.

5. Classification

Here we show how we can apply the AdaBand algorithm to address classification problems. In a classification problem we are given C classes and n training observations. The training observations consist of d attribute measurements $\mathbf{x} = (x_1, \ldots, x_d) \in \Re^d$ and the known class labels: $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $y_i \in \{1, \ldots, C\}$. The objective is to predict the class label of a given query $\mathbf{x}_0$. The given training data are used to obtain the estimates.

We assume, again, that the given dataset is large, and we want to be able to perform our prediction efficiently. The AdaBand algorithm, applied within the data of each class, allows to efficiently compute an estimation of the class density distributions of the given dataset. Formally, by denoting with S_c the sample extracted from the n_c training data of class c, the within class density function estimate at a given point $\mathbf{x}$ is $\hat{f}_c(\mathbf{x}) = \frac{1}{n_c} \sum_{i \in S_c} (\frac{3}{4})^d \frac{1}{B_i{}^d} \prod_{1 \le j \le d} (1 - (\frac{x_j - s_{ij}}{B_i})^2))$.
For a given query point $\mathbf{x}_0$, we have then C within class density function estimates: $\hat{f}_1(\mathbf{x}_0), \ldots, \hat{f}_C(\mathbf{x}_0)$. The class c^* that gives the largest value $\int_I \hat{f}_c(\mathbf{x}_0) d\mathbf{x}_0$, computed over an interval I centered at $\mathbf{x}_0$, is our prediction for the class of $\mathbf{x}_0$, i.e. $c^* = \arg\max_{1 \le c \le C} \int_I \hat{f}_c(\mathbf{x}_0) d\mathbf{x}_0$.

We observe that the integrals can be computed efficiently in $O(d|S_c|)$ time, as described in section 3.1. The integral operation allows to smooth away the estimated density functions, thereby achieving more accurate results than with a pointwise estimation. This method, which we call DenClass, can be seen as an attempt to approximate the optimal bayesian classification error rate, where $\hat{P}(c|\mathbf{x}_0) = \int_I \hat{f}_c(\mathbf{x}_0) d\mathbf{x}_0$ is our density based approximation of class posterior probabilities at query points. We note that, for a correct class assignment, the classifier $\hat{f}_c(\mathbf{x})$ needs only to preserve the order relation among the estimated quantities. This means that we can afford biased estimates, as long as all are affected roughly in the same proportion.

The experimental results indeed suggest that the integration operation conveys robustness to our method. The length of the interval I is an input parameter of the DenClass algorithm, which we optimize by cross-validation in our experiments. Figure 2 gives the outline of the DenClass algorithm.

Classifier construction: Given a d-dimensional dataset R with n points and C classes, and the input parameter $|S_c|$ for each class $c \in C$:

1. Run AdaBand on R_c and $EstSize = |S_c|(d+1)$ (R_c is the set of points in R with label c).

Output: $\hat{f}_c(\mathbf{x})$, for $c = 1, \ldots, C$. **Testing phase**: Given a query point $\mathbf{x}_0$:

1. Classify $\mathbf{x}_0$ to class c^* s.t.
   $c^* = \arg\max_{1 \le c \le C} \int_I \hat{f}_c(\mathbf{x}_0) d\mathbf{x}_0$.

Figure 2. The DenClass algorithm

6. Clustering

The method we present here is an extension of Hinneburg's and Keim's (1998) approach. Their technique employs all data points and therefore is impractical for large datasets.

The DenClass algorithm can be extended to address clustering problems. In clustering, data are unlabelled, and the density estimation is conducted for the whole dataset. Local maxima of $\hat{f}(\mathbf{x})$, that are above a certain threshold t, can be considered cluster centers. A hill-climbing procedure applied at a given point iden-

Density Estimation: Given a d-dimensional dataset R with n points, and the input parameter $|S|$:

1. Run AdaBand on R and $EstSize = |S|(d+1)$.

Output: $\hat{f}(\mathbf{x})$. **Clustering phase**: Given a query point $\mathbf{x}_0$, and the input parameter t:

1. Compute the gradient $\nabla\hat{f}(\mathbf{x}_0)$;
2. Perform hill-climbing. Let $\mathbf{x}^*$ be the resulting density attractor;
3. If $\hat{f}(\mathbf{x}^*) > t$, assign $\mathbf{x}_0$ to the cluster identified by $\mathbf{x}^*$;
4. If $\hat{f}(\mathbf{x}^*) \leq t$,
   (a) Use a grid to merge $\mathbf{x}_0$ with a connected cluster center $\mathbf{x}^{*'}$;
   (b) Assign $\mathbf{x}_0$ to the cluster identified by $\mathbf{x}^{*'}$.

Figure 3. The DenClust algorithm

Table 1. Average classification error rates and standard deviation values.

METHOD	EX1	EX2	EX3
DENCLASS	**0.3**± 0.1	15.4 ± 0.1	**0.3** ± 0.6
K-NN	0.4 ± 0.1	**15.3**± 0.1	**0.3** ± 0.6
C4.5	2.5 ± 0.5	17.0 ± 0.4	0.6 ± 0.7
K-MEANS	0.4 ± 0.1	15.6 ± 0.2	26.7 ± 8.7

tifies its density attractor. Points that converge to the same attractor belong to the same cluster. For density attractors below t, we can use connectivity information (using a grid in input space, for example) to merge them with connected cluster centers.

What is a good choice for t? If we assume that the dataset R is noise-free, all density attractors $\mathbf{x}^*$ for S are significant and t should be chosen in $0 \leq t \leq \min_{\mathbf{x}^*}\{\hat{f}(\mathbf{x}^*)\}$. In most cases the dataset will contain noise. If the noise level can be modeled by taking into account knowledge specific to the problem at hand, then t should be chosen above such level. As an alternative, the value of t could be set above the average value of the density function evaluated at the attractors: $\frac{1}{|\mathbf{x}^*|}\sum_{\mathbf{x}^*}\hat{f}(\mathbf{x}^*)$. In general, the smaller the value of t is, the more sensitive the clustering algorithm will be to outliers; the larger t is, the less details will be captured by the algorithm. Figure 3 gives the outline of the method, which we call DenClust.

7. 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. For both AdaBand and Scott's rule we use the formulas described in section 3.1 to compute the selectivity of range queries. 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.

To test the accuracy of the DenClass algorithm we use synthetic datasets and compare its performance with well known methods in the literature: K-NN, C4.5 decision tree (Quinlan, 1993), and K-means. We include K-means since it allows a compact representation of the dataset, specifically the within class mean vectors. Note that since we are applying K-means to classification problems the value of K is equal to the number of classes. Procedural parameters ($|I|$ for DenClass and K for K-NN) are detemined empirically through cross-validation.

7.1 Synthetic datasets

We have designed the following synthetic datasets for the range query selectivity problem. In Figures 4-5 the 1-norm average relative errors computed over five runs are reported. **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 values. **DiffGaussian**: This dataset contains 10^6 10-dimensional points equally 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 $\{0.1, 0.2, 0.3, \ldots, 1.0\}$. **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 equally 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.

The following datasets are used for the classification problem. For each of them, five independent training data were generated. For each of these, an additional test set (of size 2,000 for Ex1, 1,000 for Ex2, and 6,000 for Ex3) was generated. Error rates and standard deviation values are computed over all such classifications and reported in Table 1. **Example 1**: This dataset has $d = 5$ attributes, $n = 500,000$ datapoints, and $C = 2$ classes (250,000 points per class). The data for

both classes are generated from a bivariate normal distribution with standard deviation 8, and mean vector $(40, \ldots, 40)$ in one case, and $(60, \ldots, 60)$ in the other. The sample size used for the DenClass algorithm is $|S_c| = 500$ for both classes. By taking into account both the sample points and the bandwidth values, the resulting classifier $\hat{f}_c(\mathbf{x})$, $c = 1, 2$, requires the storage of 6,000 numbers. We therefore allow a sample of 600 points for both classes for the other three methods. **Example 2**: This dataset has $d = 2$ attributes, $n = 500,000$ datapoints, and $C = 2$ classes (250,000 points per class). The data for this problem are generated as in the previous example, with the addition of 25% uniformly distributed random noise. We use $|S_c| = 500$ for DenClass, and accordingly a sample size of 750 points for the other methods. **Example 3**: This dataset has $d = 2$, $n = 240,000$, and $C = 2$ classes. Each class contains six spherical bivariate normal subclasses, having standard deviation one. The means of the 12 subclasses are chosen at random without replacement from the integers $[25+2k]_{k=0}^{24} \times [25+2k]_{k=0}^{24}$. For each class, data are evenly drawn from each of the six normal subclasses. We use $|S_c| = 1,000$ for DenClass for this example, and accordingly a sample size of 1,500 data points per class for the other methods.

7.2 Real datasets

We use three real datasets. The USCities and the NorthEastern datasets contain, respectively, 1,300,000 postal addresses of cities in the US, and 130,000 postal addresses of the North Eastern states. Each point has two attributes. We also use the Forest Cover Dataset from the UCI KDD archive. 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 our experiments we use the entire set of 10 numerical attributes. In this dataset the distribution of the attributes is non-uniform, and there are correlations between pairs of attributes. In Figures 6-7 the 1-norm average relative errors computed over five runs are reported.

7.3 Query workloads

To evaluate the techniques on the range query approximation problem we generated workloads of two types of queries. Workload 1 contains 10^4 random queries with selectivity approximately 1%. Workload 2 consists 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 error $|| e_{mod} ||_1$.

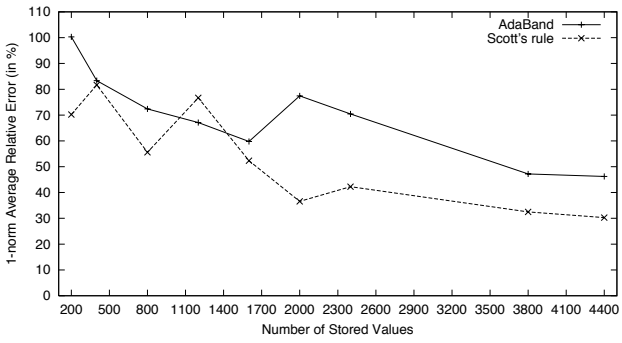


Figure 4. OneGaussian dataset, query workload 2, 5-dim.

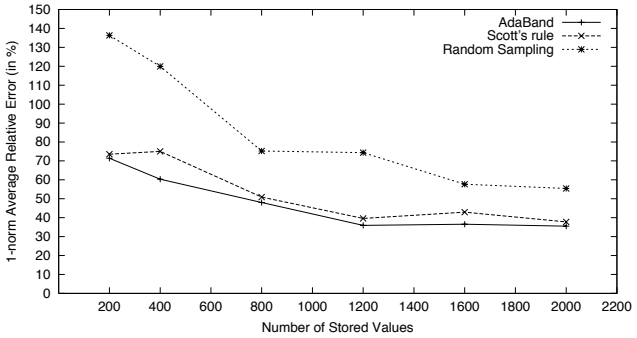


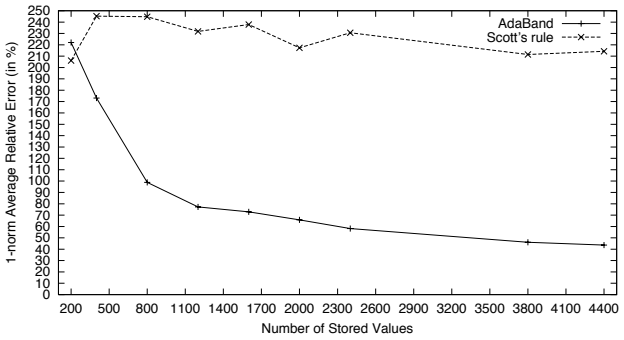
Figure 7. Forest Cover dataset, Query workload 1, 10-dim.

7.4 Experimental Results for Query Approximation

The OneGaussian dataset has been designed to test Adaband performance under optimal conditions for Scott's rule. Scott's rule finds optimal bandwidth values for this dataset. Figure 4 shows the results for query workload 2. As expected, Scott's rule shows the best performance, but AdaBand is 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 Scott's rule.

The variance (spikes) observed in Figure 4 for smaller estimator sizes may be due to the fact that the dataset is 5-dimensional, and therefore larger sample sizes are required to attain a smoother performance behavior. Furthermore, workload 2 presents a higher degree of difficulty since queries in this case have arbitrary sizes. This characteristic may also have contributed to the variance of the performance.

Figure 5 shows the results for both the DiffGaussian and the NoisyGaussian datasets on query workload 1.



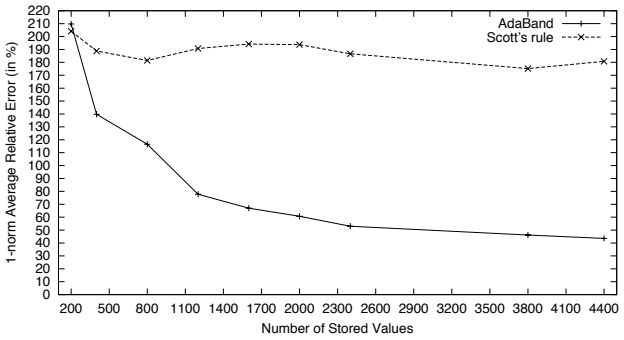
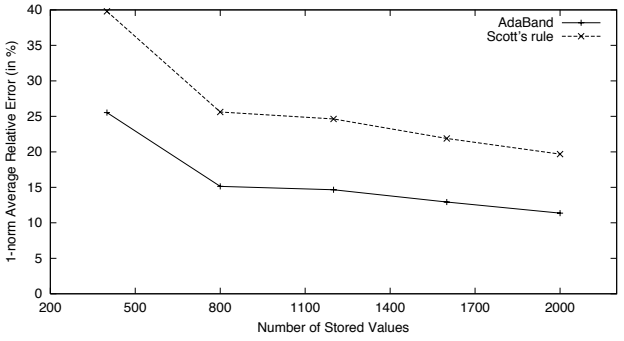


Figure 5. (Left) DiffGaussian dataset, Query workload 1, 10-dim. (Right) NoisyGaussian dataset, Query workload 1, 10-dim.



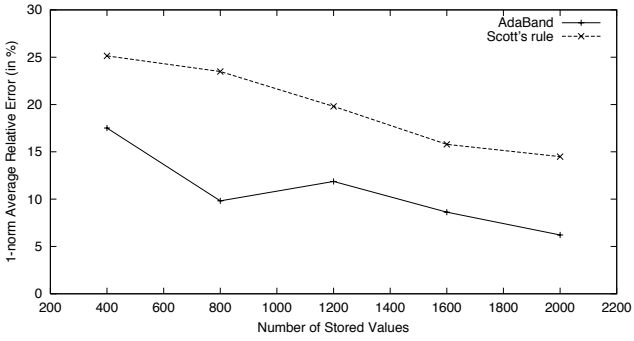


Figure 6. (Left) USCities dataset, Query workload 2, 2-dim. (Right) NorthEastern dataset, Query workload 2, 2-dim.

AdaBand outperforms by far Scott's rule in both cases. 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. Figure 6 shows the results for the USCities and the NorthEastern datasets on query workload 2. AdaBand shows large improvements in performance over Scott's rule in both cases. Figure 7 shows the results for the Forest Cover dataset on query workload 1. Here, we also use Random Sampling for comparison; it gives the worst performance. AdaBand performs slightly better than Scott's rule in this example.

7.5 Experimental Results for Classification

Table 1 shows the error rates obtained for classification. We observe that DenClass outperforms both C4.5 and K-means in all three cases. DenClass and K-NN show similar performances in each problem. These results provide evidence that we have successfully designed an efficient approximation scheme for nearest neighbor approaches to classification. Such approximation makes K-NN techniques applicable in very large datasets. Given that nearest neighbor methods in many benchmark studies turn out to be competitive, and often are among the best performers, an efficient approximation that allows its usage for large datasets is indeed highly desirable.

8. Related Work

Multi-dimensional histograms are particularly suited as density estimators when each attribute has a finite discrete domain. Efficient construction of accurate histograms becomes a problem in high dimensional spaces and when the attributes are real valued. In such cases, in fact, histogram constructions become inefficient (Gunopulos et al., 2000). In contrast, our locally adaptive kernel approach allows an efficient estimator construction that requires only two dataset passes. Efficient query approximation can be performed in time linear to the size of the estimator and to the dimensionality. Furthermore, kernel density estimators have sufficient expressive power, since any distribution can be represented as the sum of a sufficient number of kernel contributions. As a consequence, they are able to provide accurate estimators.

In Bennett et al. (1999) the density function of the data is estimated in order to build a clustered index for efficient retrieval of approximate nearest neighbor queries. Both our density estimation approach and the clustering process in Bennett et al. (1999) work on all dimensions simultaneously. The data density modeling is performed in the two cases for different purposes. In Bennett et al. (1999), the model of the density is used to reorganize the data on the disk, with the objective of minimizing the number of cluster scans at query time. In our case it synthesizes the relevant information about the data to directly address the tasks.

Furthermore, the density estimation process itself is different. In Bennett et al. (1999), the location in space for placing the Gaussian kernels is detemined by finding clusters in the data. We instead extract a uniform random sample from the data, and center the kernels at the sampled points. As a consequence, in our case the number of kernels used is driven by the estimator size we can afford. In Bennett et al. (1999), the number of clusters used affects the amount of data to be scanned at query time, and its "optimal" value needs to be estimated.

9. 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 simple and can be implemented efficiently.

In the future, we plan to extend the AdaBand algorithm for an on-line setting via efficient quantile updates, and to conduct more extensive experiments with real data in higher dimensions.

Acknowledgements

This research has been supported by NSF IIS-9984729, the US Dept. of Defense, and AT&T. The authors thank George Kollios for his contribution in developing part of the code, and Joe Hellerstein for providing the USCities and NorthEastern datasets.

References

Bennett, K. P., Fayyad, U., & Geiger, D. (1999). Density-Based Indexing for Approximate Nearest-Neighbor Queries. *Proc. of the Intern. Conf. on Knowledge Discovery and Data Mining.*

Bradley, P. S., Fayyad, U., Reina, C. (1998). Scaling Clustering Algorithms to Large Datasets. *Proc. of the Intern. Conf. on Knowledge Discovery and Data Mining.*

Chakrabarti, K., Garofalakis, M. N., Rastogi, R., & Shim, K. (2000). Approximate Query Processing Using Wavelets. *Proc. of the Intern. Conf. on Very Large Data Bases.*

Cressie, N. A. C. (1993). *Statistics For Spatial Data.* Wiley & Sons.

Gunopulos, D., Kollios, G., Tsotras, V., & Domeniconi, C. (2000). Approximating multi-dimensional aggregate range queries over real attributes. *Proc. of the ACM SIGMOD Intern. Conf. on Management of Data.*

Haas, P. J., & Swami, A. N. (1992). Sequential Sampling Procedures for Query Size Estimation. *Proc. of the ACM SIGMOD Intern. Conf. on Management of Data.*

Hinneburg, A., Keim, D. A. (1998). An Efficient Approach to Clustering in Large Multimedia Databases with Noise. *Proc. of the Intern. Conf. on Knowledge Discovery and Data Mining.*

Ioannidis, Y., & Poosala, V. (1999). Histogram-Based Approximation of Set-Valued Query-Answers. *Proc. of the Intern. Conf. on Very Large Data Bases.*

Manku, G. S., Rajagopalan, S., & Lindsay, B. G. (1998). Approximate Medians and other Quantiles in One Pass and with Limited Memory. *Proc. of the ACM SIGMOD Intern. Conf. on Management of Data.*

McLachlan, G. J. (1992). *Discriminant Analysis and Statistical Pattern Recognition.* New York: Wiley.

Park, B. V., & Turlach, B. A. (1992). Practical performance of several data driven bandwidth selectors. *Computational Statistics*, 7: 251-270.

Poosala, V., & Ioannidis, Y. E. (1997). Selectivity Estimation Without the Attribute Value Independence Assumption. *Proc. of the Intern. Conf. on Very Large Data Bases.*

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan-Kaufmann Publishers, Inc.

Scott, D. (1992). *Multivariate Density Estimation: Theory, Practice and Visualization.* Wiley & Sons.

Shanmugasundaram, J., Fayyad, U., & Bradley, P. (1999). Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions. *Proc. of the Intern. Conf. on Knowledge Discovery and Data Mining.*

Vitter, J. S., Wang, M., & Iyer, B. R. (1998). Data Cube Approximation and Histograms via Wavelets. *Proc. of the ACM CIKM Intern. Conf. on Information and Knowledge Management.*

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

Webber, R., Schek, H. J., & Blott, S. (1998). A Quantitative Analysis and Performance Study for Similarity Search Methods in High-Dimensional Spaces. *Proc. of the Intern. Conf. on Very Large Data Bases.*