DOI 10.1007/s10115-003-0131-8
Springer-Verlag London Ltd. © 2004
Knowledge and Information Systems (2004) 6: 750–770

**Knowledge and Information Systems**

# An Efficient Density-based Approach for Data Mining Tasks

Carlotta Domeniconi[1], Dimitrios Gunopulos[2]

[1]Information and Software Engineering Department, George Mason University, Fairfax VA, USA
[2]Computer Science Department, University of California, Riverside CA, USA

**Abstract.** We propose a locally adaptive technique to address the problem of setting the bandwidth parameters 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.

**Keywords:** Bandwidth setting; Classification; Clustering; Kernel density estimation; Range query approximation

## 1. Introduction

Classification and clustering (Bradley et al. 1998) are key steps for many tasks in data mining, whose aim is to discover unknown relationships and/or patterns from large sets 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 neighbours 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 occurring in the $K$ neighbours. It produces continuous and overlapping neighbourhoods, and uses a different neighbourhood for each individual query. Therefore, it results in a highly stable procedure with respect to perturbations of the data. However, when

*Received 26 January 2002*
*Revised 20 December 2002*
*Accepted 23 April 2003*
*Published online 6 February 2004*

applied on large datasets, the time required to compute the neighbourhoods (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 he/she is interested in exploring, and asks queries to find the characteristics of this region (e.g., the number of points in the interior of the region, the average value or the sum of the values of attributes in the region). When the number of dimensions increases, recent results (Weber et al. 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 neighbourhood 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, guided by the gradient of the density function, and applied at a given point, would identify a local maximum (or density attractor), corresponding to the center of the cluster the given point belongs to. 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. We observe that the use of the density function makes our technique suited for data mining tasks in search of sizes of data within regions, and not for tasks in search of the data points themselves.

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 among the attributes. The setting of the bandwidth for each kernel is based on the extension of the points in the neighbourhood, 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. 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 attributes. The domain of each attribute is scaled to lie between 0 and 1. We consider range queries of the form $(a_1 \leq R.A_1 \leq b_1) \wedge \cdots \wedge (a_d \leq R.A_d \leq b_d)$. 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 and 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 $Q$: $sel(f, Q) = \int_{[a_1,b_1]\times\cdots\times[a_d,b_d]} f(x_1, \ldots x_d)dx_1 \ldots dx_d$.

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.

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 a 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 and 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

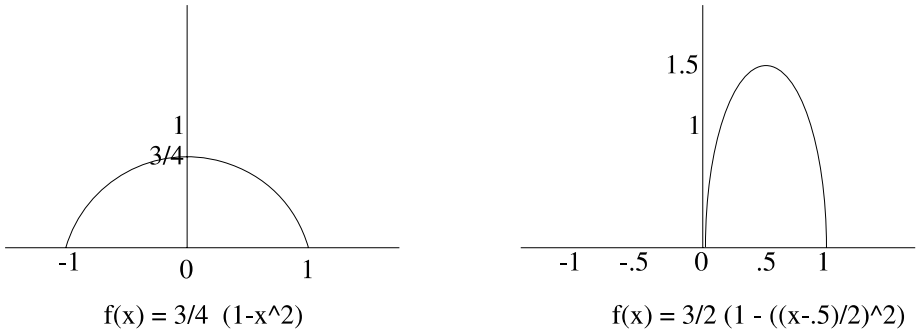


**Fig. 1.** The one-dimensional Epanechnikov kernel, with $B = 1$, centered around the origin, and, with $B = 1/2$, centered at 0.5.

$$sel(f, Q) = \int_{[0,1]^d \cap Q} f(x_1, \ldots, x_d) dx_1 \ldots dx_d =$$
$$\frac{1}{n} \sum_{t_i \in S} \int_{[0,1]^d \cap Q} k(x_1 - t_{i_1}, \ldots, x_d - t_{i_d}) dx_1 \ldots dx_d.$$

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 is easy to integrate, i.e., the $d$-dimensional Epanechnikov kernel function (Cressie 1993), whose equation centered at 0 is (Fig. 1)

$$k(x_1, \ldots, x_d) = \begin{cases} \left(\frac{3}{4}\right)^d \frac{1}{B_1 B_2 \ldots B_d} \prod_{1 \le i \le d} \left(1 - \left(\frac{x_i}{B_i}\right)^2\right) & \left|\frac{x_i}{B_i}\right| < 1 \\ 0 & otherwise. \end{cases}$$

The $d$ parameters $B_1, \ldots, B_d$ are the bandwidths of the kernel function along each of the $d$ dimensions. The magnitude of the bandwidth controls how far from the sample point the weight of the point is distributed. 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 and 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 and Swami 1992), multi-dimensional histograms (Poosala and 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 the dataset is sampled and the standard deviation along each attribute is computed.

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:

$$
\begin{aligned}
&sel(f, [a_1, b_1] \times \cdots \times [a_d, b_d]) = \\
&\frac{1}{n} \int_{[a_1,b_1]\times\cdots\times[a_d,b_d]} \Big( \sum_{1 \le i \le |S|} k_i(x_1, \ldots, x_d) dx_1 \ldots dx_d \Big) = \\
&\frac{1}{n} \int_{[a_1,b_1]\times\cdots\times[a_d,b_d]} \sum_{1 \le i \le |S|} \left(\frac{3}{4}\right)^d \frac{1}{B_1 B_2 \ldots B_d} \\
&\quad \times \prod_{1 \le j \le d} \left(1 - \left(\frac{x_j - X_{ij}}{B_j}\right)^2\right) dx_1 \ldots dx_d = \\
&\frac{1}{n} \left(\frac{3}{4}\right)^d \frac{1}{B_1 B_2 \ldots B_d} \sum_{1 \le i \le |S|} \int_{[a_1,b_1]} \left(1 - \left(\frac{x_1 - X_{i1}}{B_1}\right)^2\right) dx_1 \ldots \\
&\int_{[a_d,b_d]} \left(1 - \left(\frac{x_d - X_{id}}{B_d}\right)^2\right) dx_d.
\end{aligned}
$$

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-neighbour-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 neighbours. It is therefore reasonable to adapt the bandwidths of a kernel, centered at a specific point, according to the extension of the neighbourhood of its center. The kernel will mainly contribute to the density estimation of points within that same local neighbourhood. 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 neighbourhood. 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.

*Input*: a $d$-dimensional dataset $R$ with $n$ points, and 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 $\mathbf{s}$.

**Fig. 2.** The AdaBand algorithm.

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 that may contain a fraction of the $\lceil \frac{|R|}{|S|} \rceil$ points), 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 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.

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 $\frac{EstSize}{2d}$, where EstSize is the size of the estimator. 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 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 et al. 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

$$\hat{f}_c(\mathbf{x}) = \frac{1}{n_c} \sum_{i \in S_c} \left(\frac{3}{4}\right)^d \frac{1}{{B_i}^d} \prod_{1 \le j \le d} \left(1 - \left(\frac{x_j - s_{ij}}{B_i}\right)^2\right). \tag{1}$$

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 $\int_V \hat{f}_c(\mathbf{x}_0) d\mathbf{x}_0$, computed over a volume $V$ centered at $\mathbf{x}_0$, is our prediction for the class of $\mathbf{x}_0$, i.e.,

$$c^* = \arg \max_{1 \le c \le C} \int_V \hat{f}_c(\mathbf{x}_0) d\mathbf{x}_0.$$

We observe that the integrals can be computed efficiently in $O(d|S_c|)$ time, as described in Sect. 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 (as formalized in the following lemma), where $\hat{P}(c|\mathbf{x}_0) = \int_V \hat{f}_c(\mathbf{x}_0) d\mathbf{x}_0$ 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 extent of the volume $V$ is an input parameter of the DenClass algorithm, which we optimize by cross-validation in our experiments. Figure 3 gives the outline of the DenClass algorithm.

We formalize the motivation for the DenClass algorithm in the following lemma:

**Lemma.** *The classification outcome of the DenClass algorithm corresponds to the outcome provided by the K-nearest neighbour method, under the assumption of equal prior probabilities.*

*Proof.* Equation 1 defines the within class density function estimates $\hat{f}_c$, for $c = 1, \ldots, C$. By definition of a distribution function of a continuous random variable, we have:

$$\int_V \hat{f}_c(\mathbf{x}_0) d\mathbf{x}_0 = P(\mathbf{x}_i \in V | c), \tag{2}$$

**Classifier construction**.

*Input*: a $d$-dimensional dataset $R$ with $n$ points and $C$ classes, the 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**.

*Input*: a query point $\mathbf{x}_0$.

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

**Fig. 3.** The DenClass algorithm.

which represents the probability that a point of class $c$ is in $V$. Applying Bayes theorem, we obtain:

$$P(c|\mathbf{x}_i \in V) = \frac{\int_V \hat{f}_c(\mathbf{x}_0) d\mathbf{x}_0 P(c)}{P(\mathbf{x}_i \in V)}, \tag{3}$$

where $P(c)$ is the prior probability of class $c$.

According to the K-nearest neighbour method

$$P_{approx}(c|\mathbf{x}_0) = \frac{\sum_{i=1}^{n} 1(\mathbf{x}_i \in N(\mathbf{x}_0)) 1(y_i = c)}{\sum_{i=1}^{n} 1(\mathbf{x}_i \in N(\mathbf{x}_0))}, \tag{4}$$

that is $P_{approx}(c|\mathbf{x}_0) = P(c|\mathbf{x}_i \in V)$, where $V = N(\mathbf{x}_0)$ is a neighborhood of $\mathbf{x}_0$, and $1()$ is an indicator function such that it returns 1 when its argument is true, and 0 otherwise. Then it follows

$$P_{approx}(c|\mathbf{x}_0) = P(c|\mathbf{x}_i \in V) = \frac{\int_V \hat{f}_c(\mathbf{x}_0) d\mathbf{x}_0 P(c)}{P(\mathbf{x}_i \in V)}. \tag{5}$$

Under the assumption of equal prior probabilities, we can ignore the factor $P(c)/P(\mathbf{x}_i \in V)$ for classification purposes, and obtain

$$\arg \max_{1 \le c \le C} \int_V \hat{f}_c(\mathbf{x}_0) d\mathbf{x}_0 = \arg \max_{1 \le c \le C} P_{approx}(c|\mathbf{x}_0). \tag{6}$$

This concludes the proof of the lemma. □

## 6. Clustering

The method we present here is an extension of the approach presented in (Hinneburg and Keim 1998). The technique in (Hinneburg and Keim 1998) employs all data points; a grid approximation is proposed to cope with large datasets, and the resulting complexity depends on the size of the grid.

The DenClass algorithm can be extended to also 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 identifies

**Density Estimation**.

*Input*: 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**.

*Input*: 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 $\mathbf{x}^{*'}$.

**Fig. 4.** The DenClust algorithm.

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 modelled 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 4 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, Random Sampling and GenHist (Gunopulos et al. 2000) on synthetic and real life datasets with real valued attributes. For both AdaBand and Scott's rule we use the formulas described in Sect. 3.1 to compute the selectivity of range queries. We examine the behaviour 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. We also compare the performance of DenClass with an algorithm (that we call DenScott) that proceeds as DenClass, but sets the bandwidth values according to Scott's rule (one bandwidth value for each dimension and for each class).

Procedural parameters ($|I|$ for DenClass and $K$ for K-NN) are determined empirically through cross-validation.

## 7.1. Synthetic Datasets

We have designed the following synthetic datasets for the range query selectivity problem. In Figs. 5–7 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.

**MultiGaussian.** 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 for all dimensions set to 0.25. Each Gaussian generates the same number of data points.

**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, 6,000 for Ex3, 2,000 for Ex4, and 3,000 for Ex5) 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, $n = 500{,}000$ data points, and $C = 2$ classes (250,000 points per class). The data for both classes are generated from a multivariate normal distribution with standard deviation 8 along each dimension, 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 four methods.

**Example 2.** This dataset has $d = 2$ attributes, $n = 500{,}000$ data points, 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

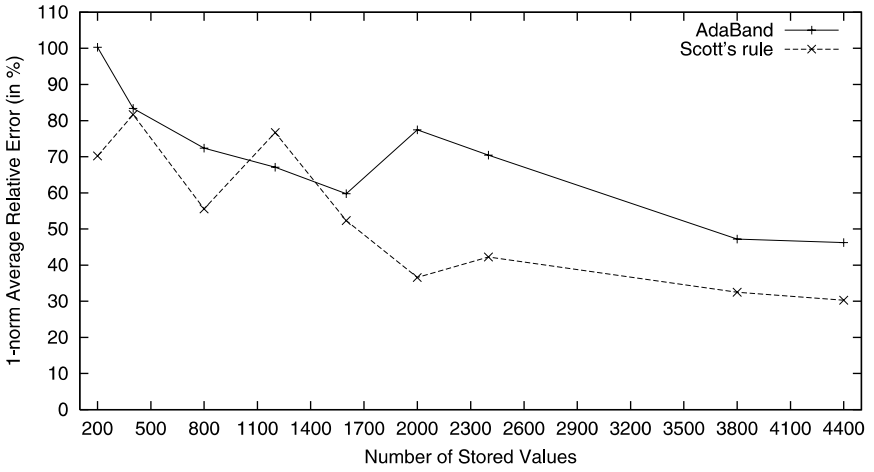


**Fig. 5.** OneGaussian dataset, Query workload 3, 5-dim.

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.

**Example 4.** This dataset has $d = 10$, $n = 400{,}000$, and $C = 2$ classes. 20% of the $n$ points is uniformly distributed random noise. The data for both classes are generated from a multivariate normal distribution with standard deviation 9 along each dimension, and mean vector $(40, \ldots, 40)$ in one case, and $(50, \ldots, 50)$ in the other.

**Example 5.** This dataset has $d = 10$, $n = 600{,}000$, and $C = 3$ classes (200,000 points per class). 20% of the $n$ points is uniformly distributed random noise. The data for all three classes are generated from a multivariate normal distribution with standard deviation 10 along each dimension. The mean vectors are: $(40, \ldots, 40)$, $(50, \ldots, 50)$, and $(60, \ldots, 60)$.

## 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 Figs. 8–9 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 three types of queries.

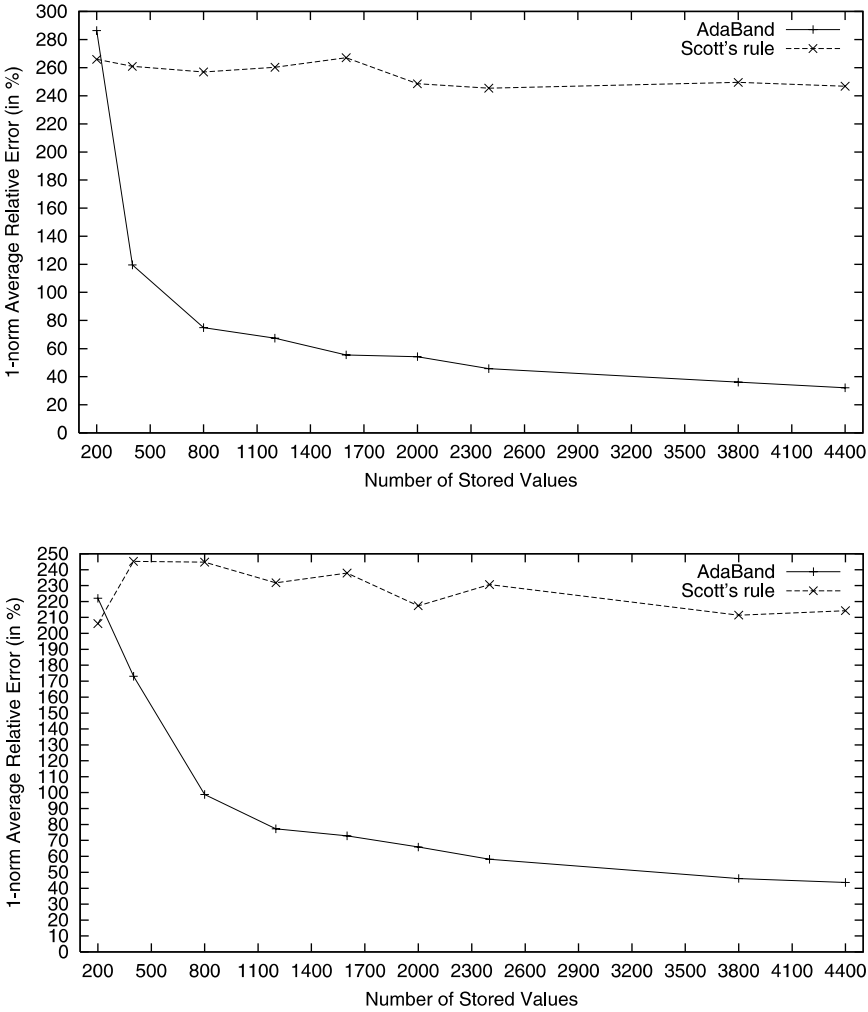


**Fig. 6.** (Upper panel) MultiGaussian dataset, Query workload 2, 10-dim. (Lower panel) DiffGaussian dataset, Query workload 2, 10-dim.

Workloads 1 and 2 contain $10^4$ random queries with selectivity approximately 10% and 1%, respectively. Workload 3 consists of 20,000 queries of the form $(R.A_1 < a_1) \wedge \cdots \wedge (R.A_d < a_d)$, 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 $\| e_{mod} \|_1$.

## 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 5 shows the results for query workload 3. As expected, Scott's rule shows the best performance, but AdaBand is not too far from it. This means

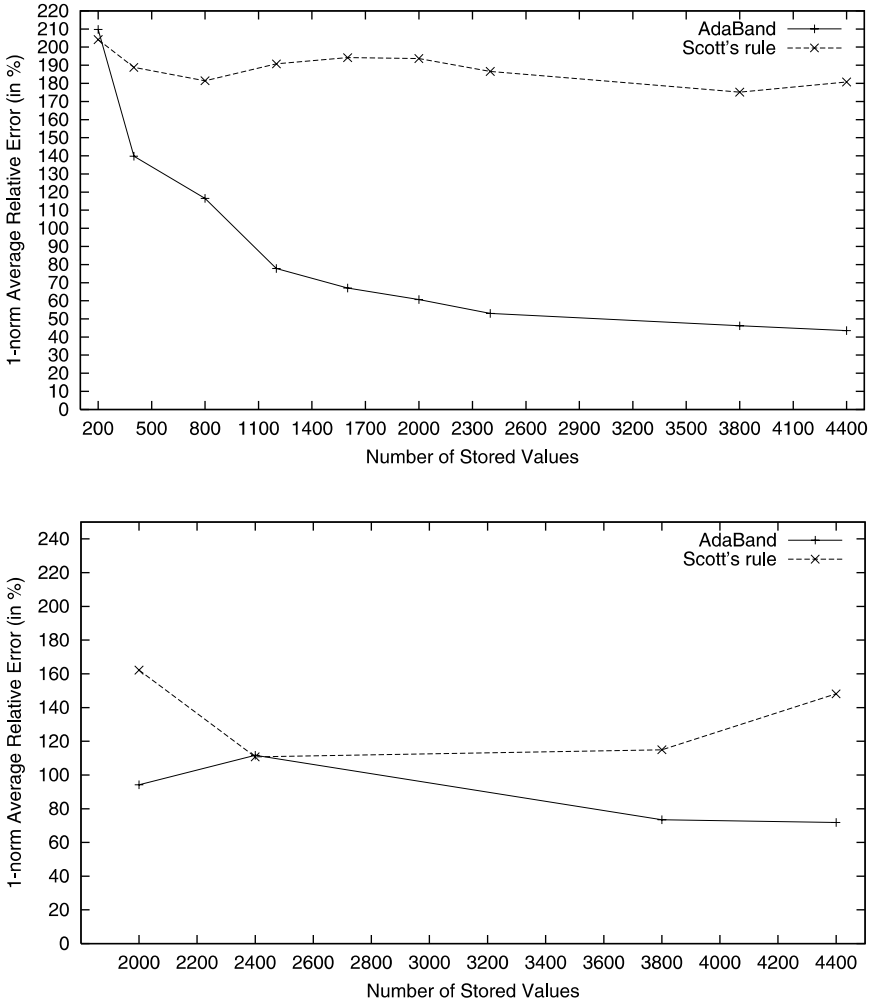


**Fig. 7.** NoisyGaussian dataset, 10-dim. (Upper panel) Query workload 2. (Lower panel) Query workload 3.

that we don't lose too much in performance with our adaptive technique in the ideal case for Scott's rule.

The variance (spikes) observed in Fig. 5 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 behaviour. Furthermore, workload 3 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.

Figures 6–7 show the results for the MultiGaussian, DiffGaussian and NoisyGaussian datasets on query workloads 2 and 3. We obtained similar results for the MultiGaussian and DiffGaussian datasets. 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

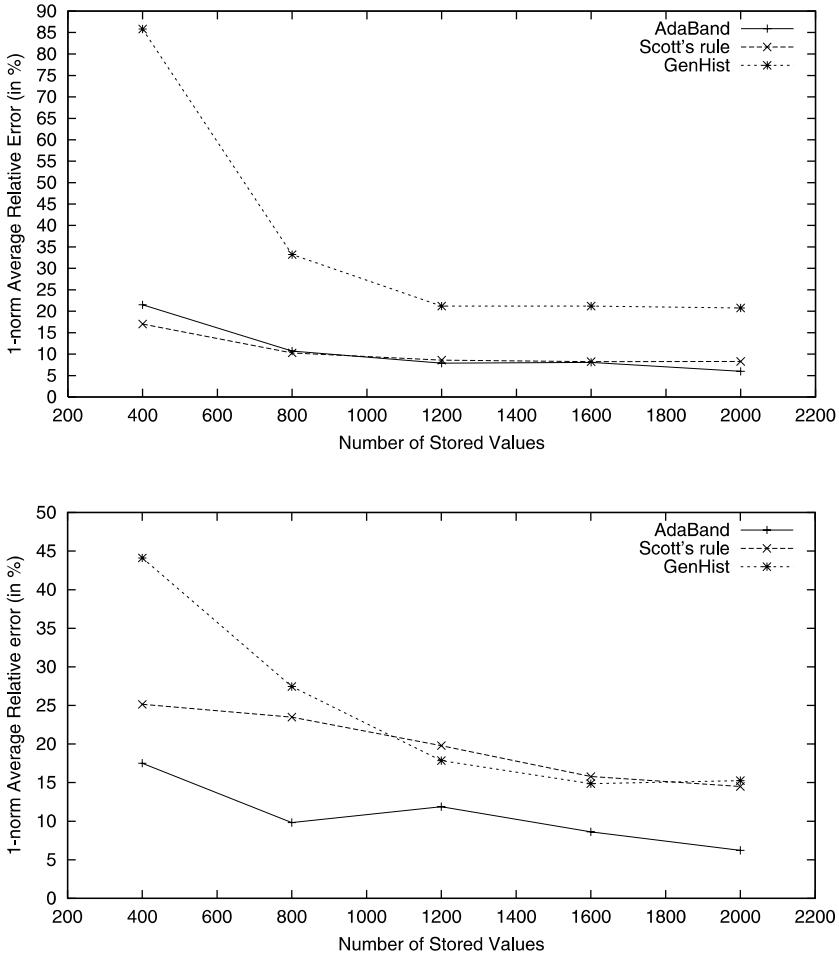


**Fig. 8.** (Upper panel) NorthEastern dataset, Query workload 1, 2-dim. (Lower panel) NorthEastern dataset, Query workload 3, 2-dim.

according to the number of kernels that become available. A similar behaviour is observed for the NoisyGaussian dataset on query workload 2 (Fig. 7, upper panel). On query workload 3, the increase in error of Scott's rule is likely to be due to the fact that the NoisyGaussian dataset is 10-dimensional, and workload 3 includes queries of arbitrary sizes. Indeed, errors on different runs showed a large variance for the tested estimator sizes. The plot for AdaBand is rather flat, but does show improvement for larger estimator sizes.

Figures 8–9 show the results for the real datasets. AdaBand shows large improvements in performance over Scott's rule with both the NorthEastern and USCities datasets for query workload 3. For query workload 1 AdaBand performs slightly better than Scott's rule on the NorthEastern and Forest Cover datasets. Figures 8–9 also show the results obtained applying Random Sampling, and GenHist. The results confirm the superiority of kernel estimators over histograms, particularly with data in higher dimensions.

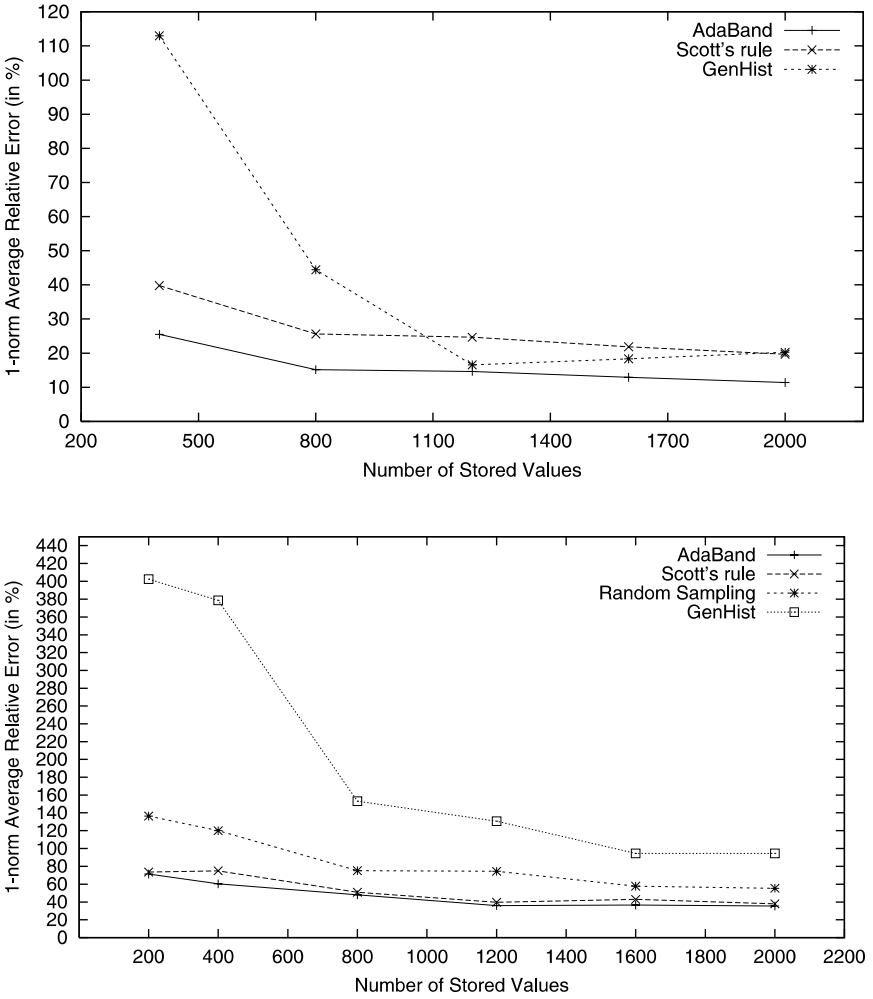


**Fig. 9.** (Upper panel) USCities dataset, Query workload 3, 2-dim. (Lower panel) Forest Cover dataset, Query workload 2, 10-dim.

## 7.5. Experimental Results for Classification

Table 1 shows the error rates obtained for classification. We observe that DenClass outperforms DenScott, C4.5 and K-means in all three cases. DenScott shows a high sensitivity to uniformly distributed noise (Example 2). This is likely due to the global nature of the settings of bandwidth values. In general, the error rates for the DenScott procedure suffer from large variance. This result demonstrates the lack of robustness of techniques based on Scott's rule for classification purposes, and shows the superiority of our local DenClass method.

DenClass and K-NN show similar performances in each problem (Table 1). Figures 10–11 plot CPU times and Error Rates versus the Number of Stored Values for both DenClass and K-NN, and for Examples 4 and 5 respectively. In both cases we observe that as the number of Stored Values increases, DenClass is capable of

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

| Method | Ex1 | Ex2 | Ex3 |
|---|---|---|---|
| DenClass | **0.3** $\pm$ 0.1 | 15.4 $\pm$ 0.1 | **0.3** $\pm$ 0.6 |
| K-NN | 0.4 $\pm$ 0.1 | **15.3** $\pm$ 0.1 | **0.3** $\pm$ 0.6 |
| DenScott | 10.4 $\pm$ 12.4 | 46.4 $\pm$ 1.4 | 0.6 $\pm$ 0.9 |
| C4.5 | 2.5 $\pm$ 0.5 | 17.0 $\pm$ 0.4 | 0.6 $\pm$ 0.7 |
| K-means | 0.4 $\pm$ 0.1 | 15.6 $\pm$ 0.2 | 26.7 $\pm$ 8.7 |

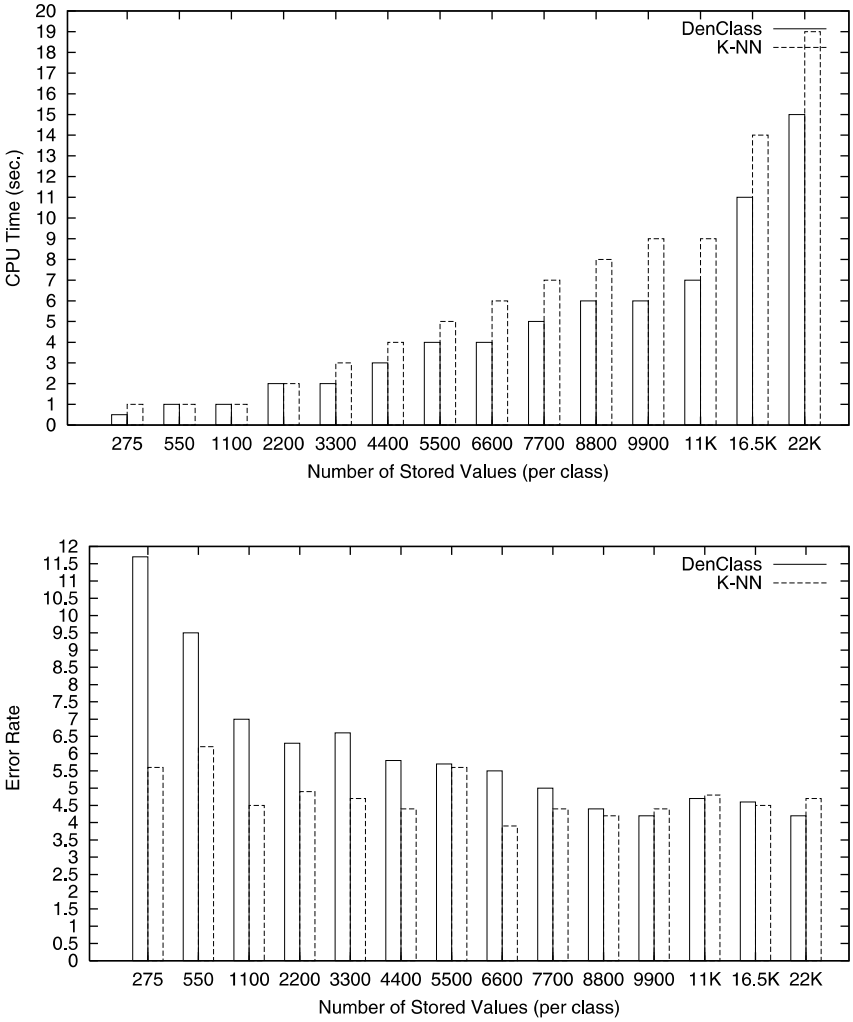


**Fig. 10.** Example 4, 10-dim, two classes. (Upper panel) CPU-time versus Number of Stored Values. (Lower panel) Error Rate versus Number of Stored Values.

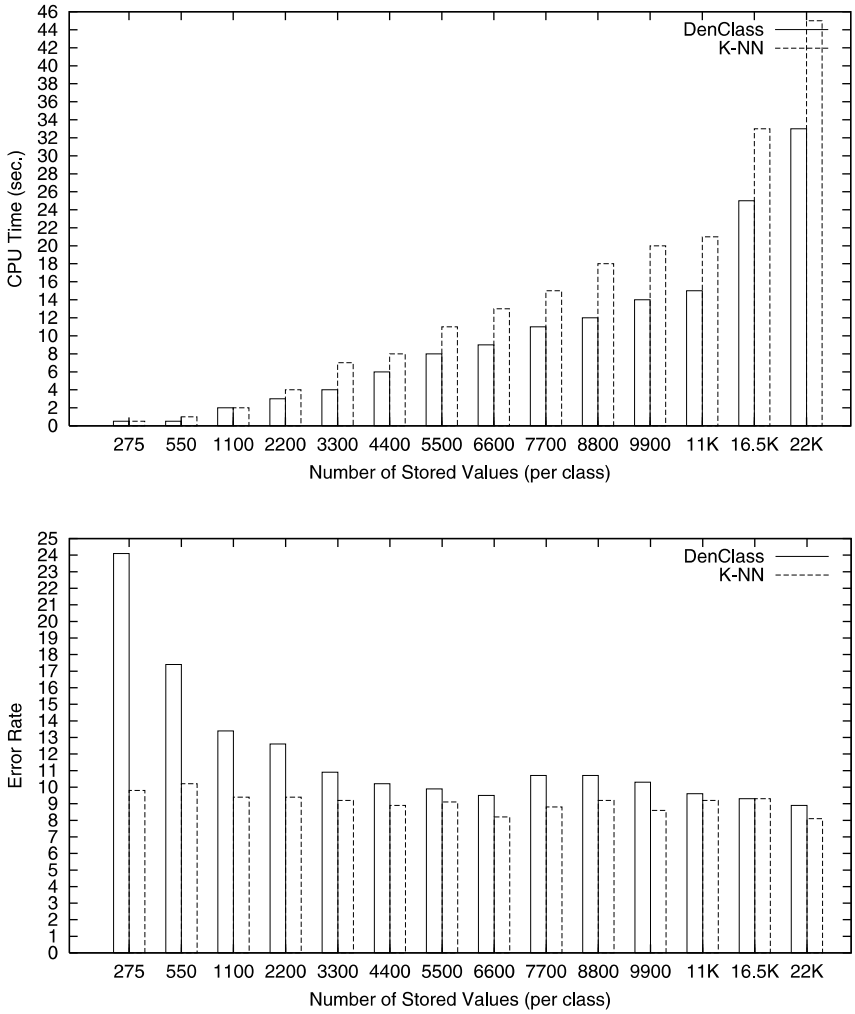


**Fig. 11.** Example 5, 10-dim, three classes. (Upper panel) CPU-time versus Number of Stored Values. (Lower panel) Error Rate versus Number of Stored Values.

approximating K-NN in accuracy, while significantly improving the execution time. These results provide evidence that we have successfully designed an efficient approximation scheme for nearest neighbour approaches to classification. Such approximation makes K-NN techniques applicable in very large datasets. Given that nearest neighbour 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 neighbour 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 determined 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.

Locally adaptive density estimators have been introduced in multivariate statistics. The *balloon* estimator (Terrell and Scott 1992; Sain 1999) varies the bandwidth at each estimation point. Given a point $\mathbf{x}$ at which the density is to be estimated, the bandwidth value is set to the distance $h_k(\mathbf{x})$ of $\mathbf{x}$ from the $K$th nearest data point. Then, kernels of the same size $h_k(\mathbf{x})$ are centered at each data point, and the density estimate is computed by taking the average of the heights of the kernels at the estimation point.

The balloon estimator requires all data points to be kept in memory, since the bandwidth value depends on the estimation point $\mathbf{x}$, and on its distance from the $K$th nearest data point (unless an approximation scheme is used). Furthermore, $K$ acts as a smoothing parameter, and its setting is critical. The computation of a proper value for $K$ is an open problem, and expensive least-squares cross-validation techniques are used to determine its value.

These limitations, along with the fact that a different density approximation function is computed for each estimation point, make this approach not suited for efficient solutions of data exploration tasks considered here.

Another locally adaptive technique is the *sample-point* estimator (Breiman et al. 1977; Sain 1999). It places a kernel at each data point $\mathbf{x}$. Each kernel has its own bandwidth, set to the distance of the center $\mathbf{x}$ from the $K$th nearest point. Again, as for the balloon estimator, the choice of $K$ is critical, and the problem of setting its value is open. We also observe that both the balloon and sample-point estimators fit a kernel at each data point. It is not clear how to couple these techniques with sampling, since the bandwidth values won't properly adjust to the sample size. In contrast, in our algorithm bandwidth values are function of both sample and original dataset sizes.

Our approach is related to the Variable-kernel Similarity Metric (VSM) technique, introduced in (Lowe 1995). Here, the $K$-nearest neighbour technique is combined with a variable kernel method to address classification problems. The bandwidth of

a Gaussian kernel centered at a point $\mathbf{x}$ is set proportionally to the average distance from $\mathbf{x}$ of the $k$ neighbours. The classification problem for a given query point is then solved by taking the weighted average of the known correct outputs of the $k$ nearest neighbours of the query point. The weight values are provided by the kernel, based on the distance of each neighbour from the query point. Distances are also weighted using global weights computed by mean of a cross-validation procedure, and the conjugate gradient optimization method.

The VSM technique requires, for each given query point, the computation of the $k$-nearest neighbours, making it an expensive procedure especially for high dimensional data. Furthermore, it has large memory requirements, since it needs to store the entire dataset. To reduce memory requirements, Lowe (1995) implements a process for thinning data in regions where class labels are uniform. Clearly, the effectiveness of this technique depends on the distribution of data, and, in general, the memory requirement will still be much larger than the space utilization of DenClass, which only retains the estimated density function.

The DenClass algorithm is also related to the procedure introduced in (Friedman and Fisher 1999). The authors in (Friedman and Fisher 1999) discuss a *bump hunting* method that seeks sub-regions of the space of input values within which the average value of the target is much larger than its average over the entire input space. This approach can be used for function approximation, classification, and clustering. For classification, the goal is to identify those regions within which an observation is most likely to be from one specific class $j$. These are the regions where $P(j|\mathbf{x})$ is larger than that of any other class. Similarly, our method classifies the query point $\mathbf{x}_0$ with the label of the class whose density function gives the largest "bump" contribution within a region centered at $\mathbf{x}_0$.

## 9. Conclusions

We have proposed a locally adaptive technique to address the problem of setting the bandwidth parameters optimally for kernel density estimation. We have also shown how to apply our technique to efficiently solve range query approximation, classification and clustering problems for very large datasets.

Our technique manifests a robust and competitive behaviour across all the datasets we have considered in our experiments. Moreover, it has the advantage of being simple and can be implemented efficiently in only two dataset passes. In the future, we plan to extend the AdaBand algorithm for an on-line setting via efficient quantile updates.

**Acknowledgements.** The authors thank the anonymous reviewers for their useful comments and suggestions. We also thank George Kollios for developing part of the code, and Joe Hellerstein for providing the USCities and NorthEastern datasets. This research has been supported by NSF IIS-9984729, the US Dept. of Defense, and AT&T.

## References

Bennett KP, Fayyad U, Geiger D (1999) Density-Based Indexing for Approximate Nearest-Neighbor Queries. Proc of the Int Conf on Knowl Discovery and Data Mining

Bradley PS, Fayyad U, Reina C (1998) Scaling Clustering Algorithms to Large Datasets. Proc of the Int Conf on Knowl Discovery and Data Mining

Breiman L, Meisel W, Purcell E (1977) Variable Kernel Estimates of Multivariate Densities. Technometrics 13:135–144
Chakrabarti K, Garofalakis MN, Rastogi R, Shim K (2000) Approximate Query Processing Using Wavelets. Proc of the Int Conf on Very Large Data Bases
Cressie NAC (1993) Statistics For Spatial Data. Wiley, New York
Friedman JH, Fisher NI (1999) Bump Hunting in High-Dimensional Data. Stat Comput 9(2):123–143
Gunopulos D, Kollios G, Tsotras V, Domeniconi C (2000) Approximating multi-dimensional aggregate range queries over real attributes. Proc of the ACM SIGMOD Int Conf on Management of Data
Haas PJ, Swami AN (1992) Sequential Sampling Procedures for Query Size Estimation. Proc of the ACM SIGMOD Int Conf on Management of Data
Hinneburg A, Keim DA (1998) An Efficient Approach to Clustering in Large Multimedia Databases with Noise. Proc of the Int Conf on Knowledge Discovery and Data Mining
Ioannidis Y, Poosala V (1999) Histogram-Based Approximation of Set-Valued Query-Answers. Proc of the Int Conf on Very Large Data Bases
Lowe DG (1995) Similarity Metric Learning for a Variable-Kernel Classifier Neural Computation 7:72–95
Manku GS, Rajagopalan S, Lindsay BG (1998) Approximate Medians and other Quantiles in One Pass and with Limited Memory. Proc of the ACM SIGMOD Int Conf on Management of Data
McLachlan GJ (1992) Discriminant Analysis and Statistical Pattern Recognition. Wiley, New York
Park BV, Turlach BA (1992) Practical performance of several data driven bandwidth selectors. Comput Stat 7:251–270
Poosala V, Ioannidis YE (1997) Selectivity Estimation Without the Attribute Value Independence Assumption. Proc of the Int Conf on Very Large Data Bases
Quinlan JR (1993) C4.5: Programs for Machine Learning. Morgan-Kaufmann
Scott D (1992) Multivariate Density Estimation: Theory, Practice and Visualization. Wiley, New York
Sain SR (1999) Multivariate Locally Adaptive Density Estimation. Technical Report, Department of Statistical Science, Southern Methodist University
Shanmugasundaram J, Fayyad U, Bradley P (1999) Compressed Data Cubes for OLAP Aggregate Query Approximation on Continuous Dimensions. Proc of the Int Conf on Knowl Discovery and Data Mining
Terrell GR, Scott DW (1992) Variable Kernel Density Estimation. Ann Stat 20:1236–1265
Vitter JS, Wang M, Iyer BR (1998) Data Cube Approximation and Histograms via Wavelets. Proc of the ACM CIKM Int Conf on Information and Knowledge Management
Wand MP, Jones MC (1995) Kernel Smoothing. Monographs on Statistics and Applied Probability. Chapman & Hall
Weber R, Schek HJ, 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

## Author Biographies

**Carlotta Domeniconi** received the Laurea degree in Computer Science from the University of Milano, Italy, in 1992, the M.S. degree in Information and Communication Technologies from the International Institute for Advanced Scientific Studies, Italy, in 1997, and the Ph.D. in Computer Science from the University of California at Riverside in 2002. She is currently an Assistant Professor in the Information and Software Engineering Department at George Mason University. Her research interests include machine learning, pattern recognition, data mining, and bioinformatics.

**Dimitrios Gunopulos** is an Associate 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, local adaptive classification techniques. Dr. Gunopulos has held positions at the IBM Almaden Research Center (1996–1998) and at the Max-Planck-Institut for Informatics (1995–1996). He completed his undergraduate studies at the University of Patras, Greece (1990) and graduated with M.A. and PhD degrees from Princeton University (1992 and 1995 respectively). His research is supported by NSF (including a NSF CAREER award), DoD, and ATT.

*Correspondence and offprint requests to*: Carlotta Domeniconi, Information and Software Engineering Department, George Mason University, Fairfax, VA 22030, USA. Email: carlotta@ise.gmu.edu