DOI 10.1007/s10115-003-0086-9

Knowledge and Information Systems (2003) 5: 387–415

Knowledge and Information Systems

A General Approach to Clustering in Large Databases with Noise

Alexander Hinneburg[1] and Daniel A. Keim[2]

[1]University of Halle, Institute of Computer Science, Halle (Saale), Germany
[2]AT&T Research Labs and University of Constance, Konstanz, Germany

Abstract. Several clustering algorithms can be applied to clustering in large multimedia databases. The effectiveness and efficiency of the existing algorithms, however, are somewhat limited, since clustering in multimedia databases requires clustering of high-dimensional feature vectors and because multimedia databases often contain large amounts of noise. In this paper, we therefore introduce a new Kernel Density Estimation-based algorithm for clustering in large multimedia databases called DENCLUE (DENsity-based CLUstEring). Kernel Density Estimation (KDE) models the overall point density analytically as the sum of kernel (or influence) functions of the data points. Clusters can then be identified by determining density attractors and clusters of arbitrary shape can be easily described by a simple equation of the overall density function. The advantages of our KDE-based DENCLUE approach are: (1) it has a firm mathematical basis; (2) it has good clustering properties in data sets with large amounts of noise; (3) it allows a compact mathematical description of arbitrarily shaped clusters in high-dimensional data sets; and (4) it is significantly faster than existing algorithms. To demonstrate the effectiveness and efficiency of DENCLUE, we perform a series of experiments on a number of different data sets from CAD and molecular biology. A comparison with k-Means, DBSCAN, and BIRCH shows the superiority of our new algorithm.

Keywords: Clustering algorithms; Clustering of high-dimensional data; Clustering in multimedia databases; Clustering in the presence of noise; Density-based clustering; Kernel Density Estimation

1. Introduction

Owing to rapid technological progress, the amount of data which is stored in databases is increasing very quickly. The types of data which are stored in the

Received 21 March 2001
Revised 12 October 2001
Accepted 24 January 2002

computer are becoming increasingly complex. In addition to numerical data, complex 2D and 3D multimedia data such as image, CAD, geographic, and molecular biological data are stored in databases. For efficient retrieval, the complex data is usually transformed into high-dimensional feature vectors. Examples of feature vectors are color histograms (Hafner et al., 1995), shape descriptors (Jagadish, 1991; Mehrotra and Gary, 1995), Fourier vectors (Wallace and Wintz, 1980) and text descriptors (Kukich, 1992). In many of the applications mentioned, the databases are very large and consist of millions of data objects with several tens to a few hundreds of dimensions.

Automated knowledge discovery in large multimedia databases is an increasingly important research issue. Clustering and trend detection in such databases, however, is difficult since the databases often contain large amounts of noise and sometimes only a small portion of the large databases accounts for the clustering. In addition, most of the known general clustering algorithms do not work efficiently on high-dimensional data. The methods which are applicable to large databases of high-dimensional feature vectors are partitioning algorithms (vector quantization) such as CLARANS (Ng and Han, 1994), hierarchical clustering algorithms, and density-based clustering algorithms such as (G)DBSCAN (Ester et al., 1996, 1997) and DBCLASD (Xu et al., 1998).

The basic idea of *partitioning algorithms* is to partition the database into k clusters which are represented by the gravity of the clusters (k-means) or by one representative object of the cluster (k-medoid). Each object is assigned to the closest cluster. Usually a partitioning algorithm searches for a configuration of the representatives which minimizes the quantization error, which is the sum of the distances of a data point to the closest representative object. A well-known partitioning algorithm is CLARANS, which uses a randomized and bounded search strategy to improve performance. Other algorithms are k-means or LBG (Linde et al., 1980), LBG-U (Fritzke, 1997) or k-harmonic means (Zhang et al., 1999a). Since the BIRCH algorithm (Zhang et al., 1996) produces representatives, namely the leaves of the CF tree, it can be seen as a vector quantization algorithm as well. The main purpose of vector quantization algorithms is data reduction or compression.

Hierarchical clustering algorithms decompose the database into several levels of partitions which are usually represented by a dendrogram – a tree which splits the database recursively into smaller subsets. The dendrogram can be created top-down (divisive) or bottom-up (agglomerative). Although hierarchical clustering algorithms can be very effective in knowledge discovery, the costs of creating the dendrograms is prohibitively expensive for large data sets since the algorithms are usually at least quadratic in the number of data objects. The advantage of hierarchical clustering algorithms are that they allow exploration of the structure of the data, instead only compressing it.

More efficient are partitioning variants of hierarchical clustering algorithms since they usually group neighboring data elements into clusters based on local conditions and therefore allow the clustering to be performed in one scan of the database. DBSCAN, for example, uses a density-based notion of clusters and allows the discovery of arbitrarily shaped clusters. The basic idea is that for each point of a cluster the density of data points in the neighborhood has to exceed some threshold. DBCLASD also works locality-based but, in contrast to DBSCAN, assumes that the points inside of the clusters are uniformly distributed. This assumption allows DBCLASD to work without any input parameters. A performance comparison (Xu et al., 1998) shows that DBSCAN is slightly faster

than DBCLASD, and both DBSCAN and DBCLASD are much faster than hierarchical clustering algorithms.

To improve the efficiency, special techniques for certain application areas such as text retrieval (Cutting et al., 1992; Sahami et al., 1998) and optimized general clustering techniques have been proposed. Examples of optimized general clustering techniques include R*-Tree-based Sampling (Ester et al., 1995), Gridfile-based clustering (Schikuta and Erhart, 1997), BIRCH (Zhang et al., 1996, 1997) which is based on the Cluster-Feature-tree, and STING, which uses a quadtree-like structure containing additional statistical information (Wang et al., 1997).

For many of those algorithms, in the context of clustering multimedia data problems with respect to their efficiency and/or effectiveness arise. A few examples of these problems are provided in Section 6. The performance of DBSCAN, for example, degenerates rapidly with increasing size and dimension of the database, and efficient algorithms such as k-means or BIRCH have serious effectiveness problems for noisy data.

Our DENCLUE approach has been designed to handle these problems. The DENCLUE algorithm is based on the well-known theory of Kernel Density Estimation (KDE), which can be seen as a general basis for clustering. In KDE, the influence of each data point is modeled using an influence (or kernel) function, and the overall density of the data is calculated as the sum of the kernel functions of all data points. Clusters can be derived from a density function according to various cluster paradigms, namely data compression, density-based single linkage or hierarchical clustering. We propose a general cluster definition which is directly related to KDE and can be used for different paradigms. In KDE, the overall density function based on Gaussian kernels requires summing up of the kernel functions of all data points. The Gaussian kernel shows the best estimation properties (Scott, 1992). Since a clustering algorithm evaluates the density at all data points, the use of the original density function leads to algorithms with quadratic runtime behavior. So an efficient algorithm has to reduce the complexity of the underlying density function, which may cause a reduced estimation accuracy depending on the actual data set.

The DENCLUE algorithm is a highly optimized implementation of KDE. It reduces the complexity of the density function in two steps. In the first step a multidimensional histogram allows filtering of noise from the data. For the cluster detection DENCLUE uses a local density function which is based on the histogram and considers only the data points which actually contribute to the overall density function. We show that the DENCLUE approach is efficient for large data sets and easy to use with databases.

The rest of the paper is organized as follows. In Section 2, we recall the basic idea of KDE and formally define kernel functions, density functions, and density attractors. We then introduce two different cluster definitions, one of which is similar to the traditional KDE cluster definition, while the second one extends this definition to allow arbitrary shape clusters. In Sections 3 and 4, we discuss the properties of our KDE-based approach, namely its generality and its invariance with respect to noise. In Section 5, we then introduce our algorithm, including its theoretical foundations such as the locality-based density function and its error bounds. In addition, we also discuss the complexity of our approach. In Section 6, we provide an experimental evaluation comparing our approach to previous approaches such as DBSCAN and BIRCH. For the experiments, we use real data from CAD and molecular biology. To show the ability of our approach to deal with noisy data, we also use synthetic data sets with a variable amount of noise. Section 7 summarizes our results and discusses their impact as well as important issues for future work.

2. Kernel Density Estimation

In this section, we briefly recall the basic ideas of KDE and formally introduce the necessary notations (for a more detailed discussion of KDE see Silverman (1986); Scott (1992) and Wand and Jones (1995).

2.1. Overview

KDE is based on the idea that the influence of each data point can be modeled formally using a mathematical function, called the kernel. The kernel function can be seen as a function which describes the influence of a data point within its neighborhood. Examples of kernels are parabolic functions, square wave functions, and Gaussian functions. The kernel function is applied to each data point. An estimate of the overall density of the data space can be calculated as the sum of the influences of all data points. Clusters can then be determined mathematically by identifying density attractors. Density attractors are local maxima of the overall density function. Determining the density attractors can be done efficiently by a hill-climbing procedure. If the overall density function is continuous and differentiable at every point, the hill-climbing procedure may be guided by the gradient of the overall density function. The mathematical form of the overall density function also allows clusters of arbitrary shape to be described in a very compact mathematical form, namely by a simple equation of the overall density function.

2.2. Basic Definitions

We first have to introduce the general notion of kernel and density functions (Silverman, 1986). Informally, the kernel functions are a mathematical description of the influence a data object has within its neighborhood. We denote the d-dimensional feature space by F^d. The neighborhood of a data object is given by an appropriate metric $\text{dist} : F^d \times F^d \longrightarrow \mathbb{R}$ in the space F^d. For simplicity, in the following we assume a squared Euclidian distance function denoted as $\text{dist}_{\text{euclid}}(x, y) = (x-y)^T(x-y)$. A smoothness parameter h controls how much the influence of a data point depends on the distance to neighboring points. The density function at a point $x \in F^d$ is defined as the sum of the kernel functions (the influences) of all data objects at that point.

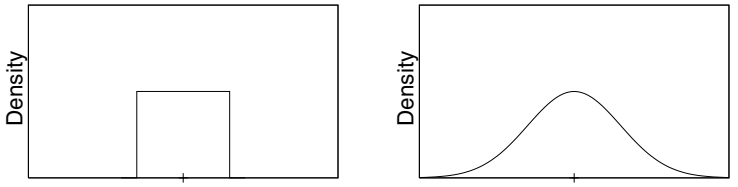
Definition 1 (Kernel and Density Function). A kernel function is a function $K : \mathbb{R}^d \longrightarrow \mathbb{R}$, $K(x) \geq 0$, which has

$$\int_{\mathbb{R}^d} K(x)dx = 1$$

The density function is defined as the sum of the kernels of all data points. Given N data objects described by a set of d-dimensional feature vectors $D = \{x_1, \ldots, x_N\} \subset F^d$ the density function is defined as

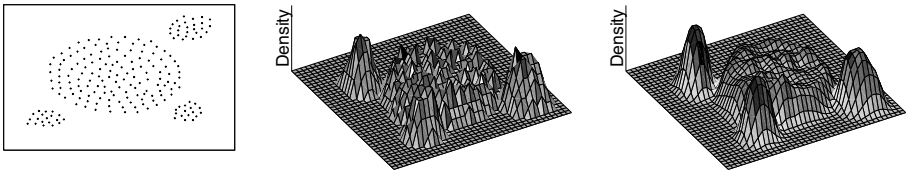
$$f^D(x) = \frac{1}{Nh^d} \sum_{i=1}^{N} K\left(\frac{1}{h}(x - x_i)\right)$$

In principle, the kernel function can be an arbitrary function, which in addition can vary for different data points. In general, however, it is desirable that the kernel function is a



(a) Square wave function (b) Gaussian function

Fig. 1. Example of kernel functions.



(a) Data set (b) Square wave (c) Gaussian

Fig. 2. Example of density functions.

symmetric, continuous, and differentiable function, and for simplicity in the following we use the same kernel function for all data points. Examples of basic kernel functions are:

1. Square wave kernel function:

$$K_{\text{Square}}(x) = \begin{cases} 0 & \text{if } x^T x > 1 \\ c_{\text{dist}} & \text{otherwise} \end{cases}$$

   The constant c_{dist} depends on the dimensionality d and is chosen such that $\int_{\mathbb{R}^d} K_{\text{Square}}(x)\,\mathrm{d}x = 1$.

2. Gaussian kernel function:

$$K_{\text{Gauss}}(x) = (2\pi)^{-d/2} \cdot \exp\left(-\frac{1}{2}x^T x\right)$$

The density function which results from a Gaussian kernel function with a constant smoothness h is (Silverman, 1986)

$$f^D_{\text{Gauss}}(x) = \frac{1}{Nh^d(2\pi)^{d/2}} \sum_{i=1}^{N} \exp\left(-\frac{1}{2h}(x - x_i)^T(x - x_i)\right)$$

Figure 1 shows an example of two kernel functions in a one-dimensional space and Fig. 2 shows an example of a set of data points in 2D space (Fig. 2a) together with the corresponding overall density functions for a square wave (Fig. 2b) and a Gaussian kernel function (Fig. 2c).

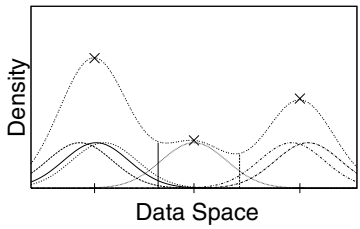


Fig. 3. Example of density attractors: the upper curve shows a univariate density function with the underlying kernel functions. The marked points are the density attractors (local maxima) of the density function.

2.3. Cluster Definitions

In the following, we define two different notions of clusters: center-defined clusters (similar to k-means clusters) and multicenter-defined clusters (similar to arbitrarily shaped clusters). The first definition is similar to the cluster definition which is implicitly used in most KDE-based clustering approaches (Schnell, 1964; Fukunaga and Hostler, 1975), while the second definition is an extention to allow arbitrarily shaped clusters with multiple centers. For the definitions, we need the notion of density attractors. Informally, density attractors are local maxima of the overall density function.

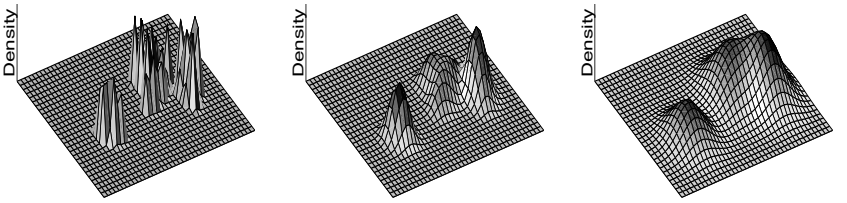
Definition 2 (Density Attractor). A point $x^* \in F^d$ is called a density attractor of a density function f^D, iff x^* is a local maximum of f^D. A point $x \in F^d$ is density-attracted to a density attractor x^*, iff a hill-climbing procedure started at x converges to x^*.

Behind the term *converges* stands a heuristic which is application dependent and – for differentiable density functions – is usually guided by the gradient. Some heuristics are given in Bock (1974, p 274 ff).

Figure 3 shows an example of density attractors in a one-dimensional space. For a continuous and differentiable kernel function, a simple hill-climbing procedure guided by the gradient of f^D can be used to determine the density attractor for a data point $x \in D$. See also Schnell (1964) or Fukunaga and Hostler (1975) for related clustering approaches based on density estimation. Note, however, that our notion of density attractors does not depend on a differentiable density function but only on a hill-climbing procedure which determines the local maxima. If the density function is not differentiable, as for example in the case of a square wave kernel function, the density attractors can still be determined by a step-wise hill-climbing procedure.

Now we are able to introduce our first definition of clusters and outliers. The definition is similar to the implicitly used cluster definition of other KDE-based clustering approaches (Schnell, 1964; Fukunaga and Hostler, 1975). These methods mainly determine density attractors and assign the points to them in order to form clusters. Our definition extends the previous definitions since it also characterizes noise and outliers and we allow clusters with multiple centers. Outliers or noise are points which are not influenced by 'many' other data points. In contrast to outliers, noise points are uniformly distributed in the data space. In both cases the points do not belong to any cluster. We need a bound called significance level ξ to formalize the 'many'.

Definition 3 (Center-Defined Cluster). A center-defined cluster (wrt to h, ξ) for a density attractor x^* ($f^D(x^*) \geq \xi$) is a subset $C \subseteq D$ with $x \in C$ being density-attracted



(a) $h = 0.2$ (b) $h = 0.6$ (c) $h = 1.5$

Fig. 4. Example of center-defined clusters for different h.

by x^*. Points $x \in D$ are called outliers or noise if they are density-attracted by a local maximum x_o^* with $f^D(x_o^*) < \xi$.

This cluster definition is rather restricted and does not account for arbitrary clusters. We therefore extend the definition to allow clusters which are defined by multiple density attractors and are spread over larger regions of the data space.

Definition 4 (Multicenter-Defined Cluster). A multicenter-defined cluster (wrt h, ξ) for the set of density attractors X is a subset $C \subseteq D$, where

1. $\forall x \in C\ \exists x^* \in X : f^D(x^*) \geq \xi$ and x is density-attracted to x^* and
2. $\forall x_1^*, x_2^* \in X : \exists$ a path $P \subset F^d$ from x_1^* to x_2^* with significance above ξ.

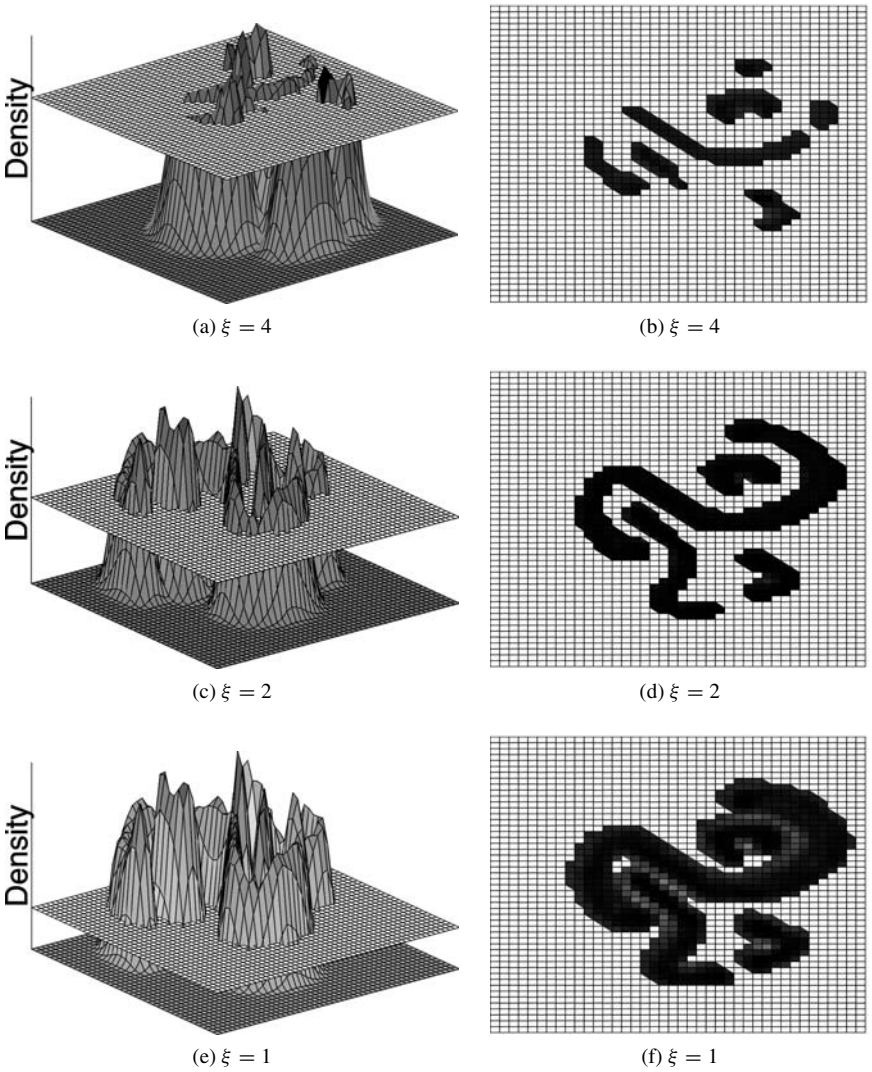
Figure 4 shows examples of center-defined clusters for different smoothness parameter values h. Note that the number of clusters found by our approach varies depending on h. Figures 4a–c show the density for different smoothness levels h. In Figures 5a, c, and e we provide examples of the density function together with the plane for different ξ. Figures 5b, d, and f show the resulting multicenter-defined clusters. The parameter h describes the influence of a data point in the neighborhood and ξ describes when a density attractor is significant. In Section 4.2, we discuss the effects of the parameters for the smoothness h and for the significance level ξ and give some advice for the parameter settings.

3. Generality

As already mentioned, the kernel density approach introduced so far allows the use of different cluster paradigms, namely partition-based data compression, density-based single linkage and hierarchical clustering. Due to the wide range of different clustering methods and the given space limitations, we cannot discuss the generality in full detail. Instead, we provide the basic idea of how our KDE-based approach is related to some specific well-known clustering methods.

3.1. Single Linkage Approach

One of the most popular approaches to clustering is the single-linkage algorithm (Sibson, 1973). In general, a single-linkage clustering with a given linkage distance k is defined as follows.



(a) $\xi = 4$ (b) $\xi = 4$

(c) $\xi = 2$ (d) $\xi = 2$

(e) $\xi = 1$ (f) $\xi = 1$

Fig. 5. Example of multicenter-defined clusters for different ξ.

Definition 5 (Single-Linkage Clustering). Given a set of objects O, the partition $C = \{C_1, \ldots, C_l\}$ of O is a single-linkage clustering for a distance function $\text{dist}(\cdot, \cdot)$ and a given linkage distance k, iff the following properties are satisfied for all $C_i, i = 1, \ldots, l$.

1. $C_i \neq \emptyset$.
2. If the object $o \in C_i$, then all objects o' in the k-environment of $o\{o' \in O : \text{dist}(o, o') \leq k\}$ belong to C_i.
3. There are no non-empty subsets C', C'' of C_i, which satisfy 1 and 2.

There are many algorithms to determine such a clustering. Some are designed to deal with the distance matrix of the data objects in cases when a transformation to feature vectors is not available. In our case, we assume that the objects are described by a set D of feature vectors in a d-dimensional metric space with a distance function $\text{dist}(\cdot, \cdot)$.

To obtain a single-linkage clustering according to Definition 5, in our KDE-based approach we use a square wave function as kernel function and the clusters have to be defined as multicenter-defined clusters according to Definition 4. If we choose $h = k/2$ and $\xi = \frac{c_{\text{dist}}}{2 \cdot N h^d} = \frac{c_{\text{dist}} \cdot 2^{d-1}}{N k^d}$, we can prove the following lemma.

Lemma 6 (Equivalence to Single-Linkage). Let D be a set of data points and $C = \{C_1, \ldots, C_l\}$, $C_i \subseteq D$ be the multicenter-defined clustering obtained with square wave kernel function, $h = k/2$, $\text{dist}(x, y) = (x - y)^T (x - y)$ and a significance level $\xi = \frac{c_{\text{dist}} \cdot 2^{d-1}}{N k^d}$. Then C corresponds to the single-linkage clustering for a linkage distance $k > 0$.

Proof. Let C_i be any cluster from the C. C_i has at least one density attractor and therefore C_i is not empty since every density attractor attracts at least one data point.

Let $x \in C_i$ and $y \in D$, $\text{dist}(x, y) \leq k$. If x and y are attracted by the same density attractor y also belongs to C_i. Otherwise y is attracted by a different density attractor. Because of $\text{dist}(x, y) \leq k$ between x and y, there exists a path of significance ξ. Since the density between a data point and its density attractor is monotonously increasing, the density attractors of x and y are also connected by a path of significance ξ. With significance level $\xi = \frac{c_{\text{dist}} \cdot 2^{d-1}}{N k^d}$, the support of one data point is enough to establish a significant density attractor, and so the density attractor of y is significant. Therefore, y belongs to C_i.

Now we prove the last condition of Definition 5. Assume C_i contains two different subsets C', C'', which satisfy 1 and 2 of the definition. From the second property follows $C' \cap C'' = \emptyset$ and for all $x_1 \in C'$ and $x_2 \in C''$ holds $\text{dist}(x_1, x_2) > k$. This means that there exists no path of significance ξ between x_1 and x_2, and therefore there is also no path between the density attractors of the points of C' and C''. This contradicts that C_i is a multicenter-defined cluster. □

3.2. Density-Based Approaches

The single-linkage method sometimes produces clusters which are not homogeneous, especially if clusters of similar objects are linked by a chain. If the data contains noise, this is likely to occur, especially if the data set is large. Figure 6 illustrates such a case. A method to reduce the chaining effect has been proposed by Wishart (1969; see also Bock, 1974). Wishart suggests a preprocessing step to remove all object o from the data set for which

$$\#\{o' \in D : \text{dist}(o, o') \leq k\} < \text{den}, \quad \text{den} \in \mathbb{N} \tag{1}$$

and then applies the single linkage method to the reduced data set. The preprocessing removes the points which are in regions of low density in the data space and in that way reduces the chaining effect.

The DBSCAN approach (Ester et al., 1996) uses a definition of density-connected sets which is similar to the method of Wishart. Setting the parameters of DBSCAN $EPS = k$ and $MinPts = den$, the preprocessing in the method of Wishart preserves

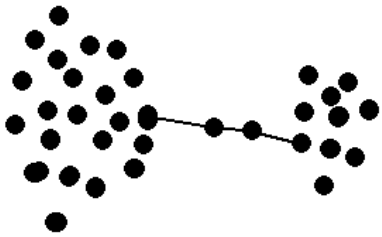
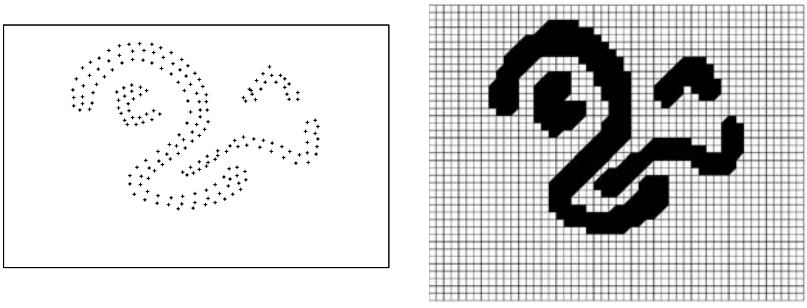


Fig. 6. Two-dimensional example of the chaining effect.



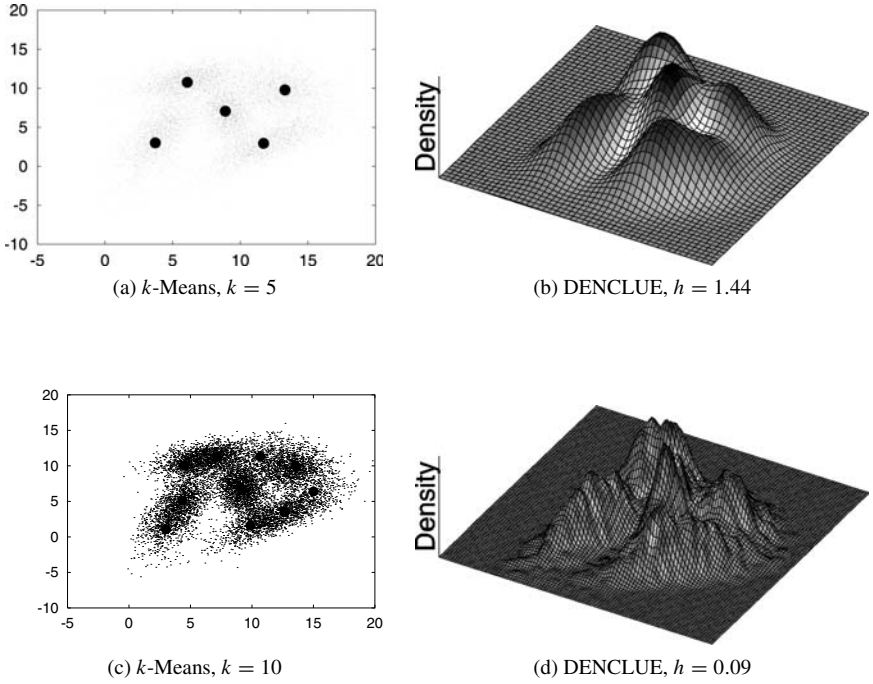
(a) DBSCAN (b) DENCLUE

Fig. 7. Clustering results by DBSCAN and DENCLUE.

exactly the points which satisfy the core point condition in the DBSCAN approach (Ester et al., 1996). Core points in DBSCAN are points which have at least $MinPts$ data points in their EPS neighborhood. The core point condition is basically equivalent to equation (1) with $MinPts = den$ and $EPS = k$. Since each cluster from DBSCAN contains at least one core point, the number of clusters of the method of Wishart and DBSCAN are the same. The difference is that a cluster C_i according to the cluster definition of Ester et al. (1996) additionally contains all the non-core points y, which are *directly density reachable* from a core point, i.e., there exists a core point x in C_i with $\text{dist}(x, y) < k$. Note that the clustering obtained from DBSCAN is not unique but depends on the processing order of the non-core points. There may be two core points in different clusters with a distance smaller than k to a non-core point y and it is not clear by definition to which cluster y belongs.

In our KDE-based approach we can obtain the same clusterings as DBSCAN. Using a square wave kernel function with $h = EPS/2$ and an outlier-bound $\xi = MinPts \cdot \frac{c_{\text{dist}} \cdot 2^{d-1}}{N \cdot EPS^d}$, the multiter-defined clusters defined by our method (see Definition 4) are the same as the clusters found by DBSCAN. The reason is that in case of the square wave kernel function the points $x \in D : f^D(x) > \xi$ satisfy the core point condition of DBSCAN and each non-core point $x \in D$ which is directly density reachable from a core point x_c is attracted by the density attractor of x_c. An example showing the identical clustering of DBSCAN and our KDE-based approach is provided in Fig. 7. Note that the results are only identical for the square wave kernel function.

The usefulness of the core point condition is limited in cases of very noisy data (see Section 4) or for detecting subclusters, because the condition describes only the



(a) k-Means, $k = 5$ (b) DENCLUE, $h = 1.44$

(c) k-Means, $k = 10$ (d) DENCLUE, $h = 0.09$

Fig. 8. Using our KDE-based approach like k-means.

minimum density at the border of a cluster. Since the border density of clusters may vary, in general good values for EPS and $MinPts$ are difficult to determine. For example, assume five clusters with normal distributions and some overlap (see Fig. 8(a)). In that case, it is not possible to distinguish all clusters in one run without identifying many border points as outliers. The same situation occurs in the case of noisy data.

3.3. Prototype-Based Approaches

Clustering algorithms based on prototypes mainly have the goal to compress (quantize) given vector data. Such approaches have been developed in many research areas, namely statistics, artificial intelligence, machine learning, databases, and data mining. Popular examples are k-means, LBG (Linde et al., 1980), neural gas (Martinetz and Schulten, 1993), growing neural cell structures (Fritzke, 1995), LBG-U (Fritzke, 1997), k-harmonic means (Zhang et al., 1999a), CLARANS (Ng and Han, 1994) and BIRCH (Zhang et al., 1996) (until phase 2). The effectiveness of a vector quantizer can be measured by the distortion error. For a set of prototypes $P = \{p_1, \ldots, p_k\}$ and a data set D the distortion error is defined as:

$$E(P, D) = \frac{1}{N} \sum_{x \in D} \text{dist}(p_{I(x)}, x)^2 \text{ with}$$

$$I(x) = \min\{i \mid \text{dist}(p_i, x) \leq \text{dist}(p_j, x)\ \forall j \in \{1, \ldots, n\}\}$$

A small average distortion error indicates a good approximation of the data. The compression rate can be controlled via the size of the set of the prototypes. There are two opposite issues regarding clustering which are related to prototype-based approaches. The first is that center-defined clusters based on density estimation can be used like a vector quantizer to reduce a data set. The opposite issue is to speed up a density function using a compressed representation from a vector quantizer.

The first issue contributes to the generality of our clustering framework. The density attractors of the center-defined clusters can be used as prototypes. An advantage is that the compression rate can be controlled by the smoothing parameter h, which may vary continuously. The value of h determines the number of density attractors. The smoothness parameter can be naturally adapted to a data set (see Section 4.2). This way shows how clusters for data compression can be derived from a density function.

For an example we assume that the data follows a mixture of k Gaussian distributions, which is the best case for the k-means algorithm. Figure 8(a) shows an example scenario of normally distributed data with five clusters. The big dots in Fig. 8(a) represent the result of a k-means clustering with five prototypes, which approximate the center points of the normally distributed clusters. Figure 8(b) displays the density function for a smoothness $h = 1.44$ and also provides the five clusters with local maxima at the center points. If more prototypes are used for the k-means clustering, the algorithm splits clusters, which means that it represents a single cluster by more than one prototype (see Fig. 8(c)). A similar clustering (Fig. 8(d)) can be observed if the smoothness h is set to lower values. An advantage of our KDE-based method is that using the multicenter-defined cluster definition the correct clustering can still be observed. Density attractors which are close to each other are connected using the density function. In the k-means result, the information for connecting the prototypes is lost.

The second issue is accelerating the density function using the result of a vector quantizer. There are some publications in the literature describing how to generate a density function from binned and reduced data (Härdle and Scott, 1992; Wand and Jones, 1995; Zhang et al., 1999b). The complexity of such an approximation depends on the number of prototypes used, instead of the number of data points. Note, that due to the data reduction used the estimation error of the resulting density function is enlarged (see Härdle and Scott, 1992, for more details). We discuss the impact of this issue in more detail in Section 5.

3.4. Hierarchical Approaches

Another class of clustering methods are hierarchical methods. By using different smoothness levels we are able to generate a hierarchy of clusters. Since, in general, the different hierarchical clustering methods provide different results, it is difficult to show that the hierarchical clustering obtained by our KDE-based approach exactly corresponds to the result of one specific method. Instead, we therefore show that our KDE-based method can be used to obtain a hierarchical clustering which we believe to be a rather natural one (in the case of Gaussian kernels).

Methodology to generate a Cluster Hierarchy. Let D be a set of data points. The center-defined or multi-center-defined clusters based on a Gaussian kernel function according to Definition 3 or 4 can be used to generate a hierarchy of clusters. If we want to generate a hierarchy of clusters we simply have to determine the center-defined clusters (i.e., the density attractors) for different smoothness levels $h_{\min} = h_1 < h_2 < \ldots < h_l = h_{\max}$. In the first step (i.e., for $h_{\min}$), we obtain N^* clusters (N^* is the number

of non-identical data points), each containing only identical points. With increasing h, at a certain point density attractors start to merge and we obtain the next level of the hierarchy. If we further increase h, more and more density attractors merge, and for each merge we get a new level of the hierarchy. Note that for efficiency reasons h cannot be varied continuously but has to be varied in discrete steps. This, however, does not restrict the practicability of the approach.

4. Noise Invariance and Parameter Discussion

In this section, we discuss other important aspects of our approach, namely noise invariance and parameter settings.

4.1. Noise Invariance

Let $D \subset F^d$ be a given data set and DS_D (data space) the relevant portion of F^d. Since D consists of clusters and noise, it can be partitioned $D = D_C \cup D_N$, where D_C contains the clusters and D_N contains the noise (e.g., points that are uniformly distributed in DS_D). Let $X = \{x_1^*, \ldots, x_k^*\}$ be the canonically ordered set of density attractors of D (wrt h, ξ) and $X_C = \{\hat{x}_1^*, \ldots, \hat{x}_{\hat{k}}^*\}$ the density attractors for D_C (wrt $h, 0$), and let $\#(S)$ denote the cardinality of S. Then, we are able to show the following lemma.

Lemma 7 (Noise Invariance). The number of density attractors of D and D_C is the same and the probability that the density attractors remain identical goes against 1 for $\#(D_N) \to \infty$. More formally,

$$\#(X) = \#(X_C) \text{ and}$$

$$\lim_{\#(D_N)\to\infty} \left(P \left(\sum_{i=1}^{\#(X)} \text{dist}(x_i^*, \hat{x}_i^*) = 0 \right) \right) = 1$$

Proof. The proof is based on the fact that the normalized density $\tilde{f}^{D_N}$ of a uniformly distributed data set is nearly constant with $\tilde{f}^{D_N} \approx c, (0 < c \leq 1)$. According to Schuster (1970) and Nadaraya (1965),

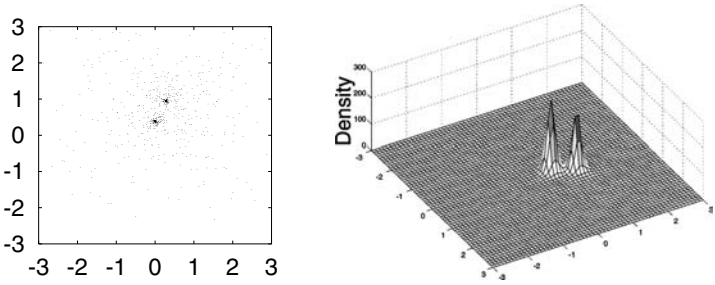
$$\lim_{\#(D_N)\to\infty} \sup_{y \in DS_D} |c - f^{D_N}(y)| = 0$$

for any smoothness $h > 0$. So the density distribution of D can approximated by

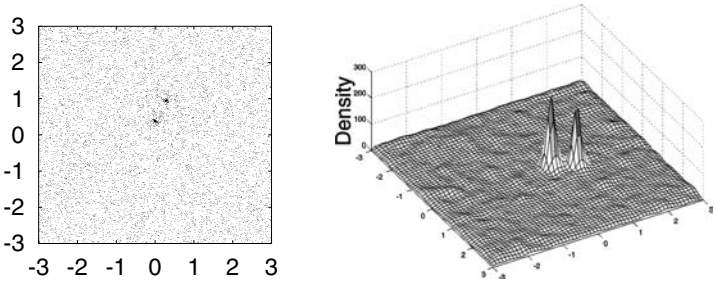
$$f^D(y) = f^{D_C}(y) + f^{D_N}(y) \approx f^{D_C}(y) + c$$

for any $y \in DS_D$ and large $\#(D_N) \in \mathbb{N}$. The density of the noise portion approximates a constant for a given D, and therefore the probability that the density attractors defined by f^{D_C} do not change goes to one. □

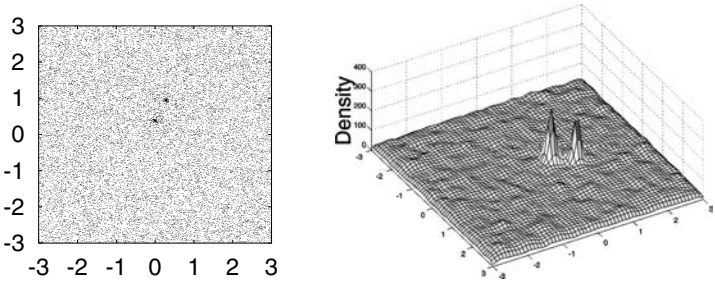
To demonstrate the impact of the lemma, in the following we show the change of the density function of a two-dimensional data set in the presence of noise. The noiseless example data set D_C contains 1000 data points and two clusters. The points in each cluster approximate a two-dimensional normal distribution. Figure 9(a) shows the data set and its density function. Now we add uniformly distributed noise to D_C (10.000, 25.000, 50.000 data points). Figures 9(b), (c), and (d) show the data sets with noise. The two clusters remain clearly visible despite the noise. The applicability of the noise



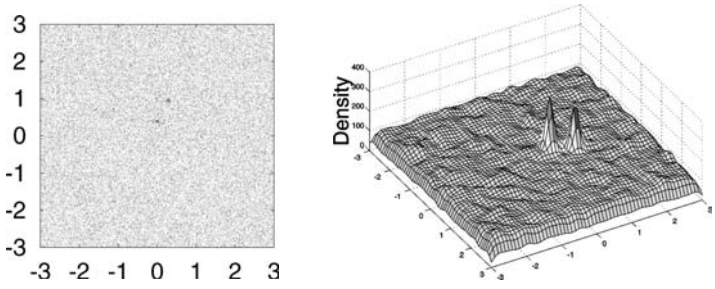
(a) Data: 1000, Noise=0



(b) Data: 1000, Noise=10000



(c) Data: 1000, Noise=25000



(d) Data: 1000, Noise=50000

Fig. 9. Noise invariance.

lemma, however, is limited to cases where the noise is uniformly distributed, because only uniform distributions result in a constant density for the noise portion. In addition, in high-dimensional spaces a reasonable approximation of a uniform distribution requires a very large sample which is usually not available. In such spaces, however, uniformly distributed points have a high probability of being far away from the rest, which means that the influence on the clustered data is very low and does not change the density

function significantly. The term 'noise' describes semantically non-relevant data objects. Note that it is desirable that the feature transformation from the multimedia data objects to the vector representation guarantees that the vectors of the non-relevant objects are uniformly distributed or at least far away from relevant clusters.

4.2. Parameter Discussion

As in most other approaches, the quality of the resulting clustering depends on an appropriate choice of the parameters. In our approach, we have two important parameters, namely the smoothness h and the noise level ξ. The parameter h determines the influence of a point in its neighborhood and ξ describes whether a density attractor is significant, allowing a reduction of the number of density attractors and helping to improve the performance. In the following, we describe how the parameters should be chosen to obtain good results.

The parameter ξ is the minimum density level for a density attractor to be significant. If ξ is set to zero, all density attractors together with their density-attracted data points are reported as clusters. This, however, is often not desirable since – especially in regions with a low density – each point may become a cluster of its own. A good choice for ξ helps the algorithm to focus on the densely populated regions and to save computational time. Note that only the density attractors have to have a point density $\geq \xi$. The points belonging to the resulting cluster, however, may have a lower point density since the algorithm assigns all density-attracted points to the cluster (see Definition 3). But what is a good choice for ξ? If we assume the database D to be noise-free, all density attractors of D are significant and ξ should be chosen in $0 \leq \xi \leq \min_{x^* \in X}\{f^D(x^*)\}$. In most cases the database will contain noise ($D = D_C \cup D_N$; see Section 4.1). If the assumption is true that the data consists uniformly of noise and clustered data, then according to Lemma 7 the density of the noise data approximates a constant $f^{D_N}(y) \approx c \in \mathbb{R}$ and therefore ξ should be chosen in

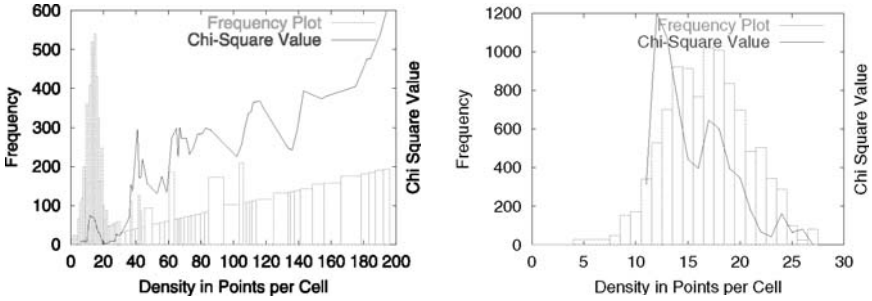
$$c \leq \xi \leq \min_{x^* \in X}\{f^{D_C}(x^*) + c\}$$

We developed a heuristic to estimate ξ for the case of uniformly distributed noise. The heuristic discretizes the interval $[\min\{\hat{f}(x)\}, \max\{\hat{f}(x)\}], x \in D$ into $k \in \mathbb{N}, k > 2$ bins. In a second step it calculates the frequencies for the bins using the data set D. To estimate ξ, we use the observation that the frequencies are binomially distributed for uniform data. For finding a good ξ we try to find the largest subinterval of the form $[\min\{\hat{f}(x)\}, t], t \leq \max\{\hat{f}(x)\}$ which contains a binomial distribution. The estimation of ξ can be done visually or with the help of the χ^2-test. For an example see Fig. 10.

Choosing a good value for the smoothing parameter h can be done by considering different h and determining the largest interval between the minimum smoothness ($h_{\max}$) and the maximum smoothness ($h_{\min}$) where the number of density attractors $m(h)$ remains constant. The clustering which results from this approach can be seen as naturally adapted to the data set. In Fig. 11, we provide an example for the number of density attractors depending on h.

5. The Algorithm

In this section, we describe the DENCLUE algorithm which implements our ideas described in Sections 2 to 4. For an efficient determination of the clusters, we have to find a way to efficiently calculate the density function and the density attractors.



(a) Uniform noise and 1 cluster (b) Uniform noise

Fig. 10. Experiment for noise level estimation for 5D data sets with 5000 data points. A low χ^2-value corresponds to a good fit of the theoretical and the observed binomial distribution. For $\xi = 21$ in (a), 94% of the uniformly distributed noise points can be separated.

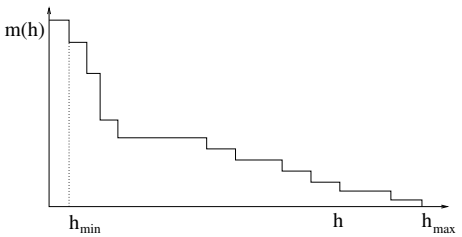


Fig. 11. Number of density attractors depending on h.

There have been various scaling methods for density estimation proposed in the literature (Wand and Jones, 1995; Zhang et al., 1996; Fritzke, 1997; Wang et al., 1997; Hinneburg and Keim, 1998). Table 1 presents an overview with the storage complexity and the run-time complexity needed to estimate the density at a given point $x \in F$. Note that there may exist different methods to build a density estimator, which may have different run-time complexities. For instance, the output of BIRCH (Zhang et al., 1996) and LBG (Linde et al., 1980) are k prototypes approximating the data set. However, both algorithm show a different run-time behavior. The histogram and k prototype methods can be integrated into the KDE concept by using binned kernel estimators (Silverman, 1982; Scott and Sheather, 1985; Härdle and Scott, 1992; Wand and Jones, 1995). The scaling methods based on prototypes are efficient for $k \ll N$, but these methods lead to raw and smooth estimations. The use of local kernels allows an efficient estimation without strong smoothing.

Table 1. Overview of the storage size and run-time complexity of different methods for density estimation (N, number of data points, d, number of dimensions). The run-time complexity is the time needed to estimate the density at a single point $x \in F$

Method	Size	Time
KDE	$O(N)$	$O(N)$
Local kernel + mult. dim. index	$O(N)$	O(range query time)
Histogram (heap)	$O(N)$	$O(\log(N))$
Histogram (array)	$O(2^d)$	$O(1)$
k Prototypes, $k \ll N$	$O(k)$	$O(k)$

An important observation for local kernels is that to calculate the density at a point $x \in F^d$, only points of the data set which are close to x actually contribute to the density. For a square-wave kernel function, only points which are closer to x than the extension of the square wave function contribute to the density, and if the 'closeness' is defined in a way that it includes all those points, no error of density is induced. Even for a Gaussian kernel function, points which are far from x may be neglected without making a substantial error. In the following, we restrict our considerations to the more complex case of a Gaussian kernel function. Before describing the details of our algorithm, we first introduce a local variant of the density function together with its error bound.

5.1. Local Density Function

The local density function is an approximation of the overall density function and it is therefore not necessarily normalized. The idea is to consider the influence of nearby points whereas the influence of far points is neglected. This introduces an error which can, however, be guaranteed to be in tight bounds. To define the local density function, we need the function near(x) with $x_1 \in \text{near}(x)$ if $\text{dist}(x_1, x) \leq \delta_{\text{near}}$. The local density is then defined as follows.

Definition 8 (Local Density Function). The local density $\hat{f}^D(x)$ is

$$\hat{f}^D(x) = \frac{1}{Nh^d} \sum_{x' \in \text{near}(x)} K\left(\frac{1}{h}(x' - x)\right)$$

Using the local density function with Gaussian kernel instead of the global density function is equivalent to using a modified Gaussian kernel:

$$K_{\text{Gauss}-0}(x) = \begin{cases} K_{\text{Gauss}}(x) & \text{if } (x-y)^T(x-y) \leq \delta_{\text{near}} \\ 0 & \text{otherwise} \end{cases}$$

Kernels which satisfy $K(x) = 0$, for $\text{dist}(x, x') > c, c \in \mathbb{R}$ are suggested by Silverman (1986) to be used in combination with an index structure for an efficient retrieval of the contributing data points.

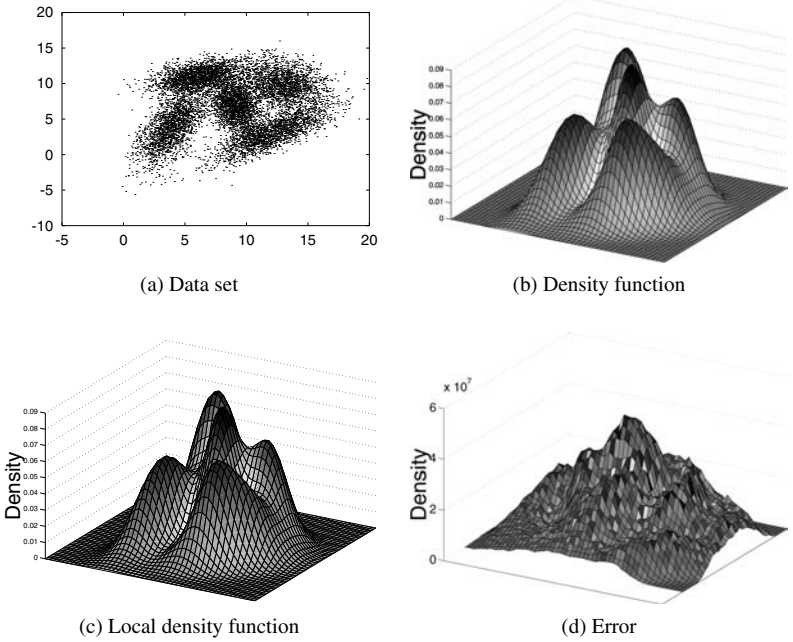
In the following, we assume that $\delta_{\text{near}} = k \cdot h$. A discussion of how to choose $k \cdot h$ will be provided at the end of Section 5 after introducing the CubeMap data structure. An upper bound for the error made by using the local density function $\hat{f}^D(x)$ instead of the global density function $f^D(x)$ with Gaussian kernel is given by the following lemma.

Lemma 9 (Error Bound for a Gaussian Kernel Function). Let $D' = D - \text{near}(x)$, $n' = \#D'$ and e be a unit vector. If the points $x_i \in D : \text{dist}(x_i, x) > kh$ are neglected, the error is bound by $e^{-\frac{k^2 h}{2}} / h^d$:

$$\text{Error}(x) = |\hat{f}^D_{\text{local}}(x) - \hat{f}^D_{\text{global}}(x)| = \frac{1}{Nh^d} \sum_{x_i \in D'} K\left(\frac{1}{h}(x_i - x)\right)$$

$$\leq \frac{1}{Nh^d} \cdot n' K\left(\frac{1}{h}(\delta_{\text{near}} \cdot e)\right) = \frac{1}{Nh^d} \cdot n' \cdot \exp\left(-\frac{k^2 h}{2}\right) \leq \frac{e^{-\frac{k^2 h}{2}}}{h^d}$$

Proof. The error bound assumes that all points are on a hypersphere of kh around x. □



(a) Data set (b) Density function

(c) Local density function (d) Error

Fig. 12. Impact of a local density function.

The error bound given by Lemma 9 is only an upper bound for the error made by the local density function. In real data sets, the error will be much smaller since many points are much further away from x than kh. An example for the impact of the local density function is shown in Fig. 12. Figure 12(a) shows a two-dimensional data set and Fig. 12(b) a corresponding density function $f^D(x)$ using $h = 1.44$ and a Gaussian kernel. Figure 12(c) shows the corresponding local density function $\hat{f}^D(x)$ for $h = 1.44$, $k = 4$ and Fig. 12(d) the error $f^D(x) - \hat{f}^D(x)$. The theoretical error bound for $\sigma = 1$, $k = 4$ and $h = 1.44$ is error $\leq 9.9295 \cdot 10^{-6}$. The observed maximum error in the two-dimensional example is about $6 \cdot 10^{-7}$, which is a factor 10 lower.

5.2. The DENCLUE Algorithm

The DENCLUE algorithm works in two steps: First, the density function is approximated by a multidimensional histogram (*approximation of the density function*), and then the dense areas of the histogram are identified and connected with the surrounding bins to form the clusters (*clustering step*). For the approximation of the density function, the DENCLUE algorithm can be coupled with different scaling methods for density estimation. Efficient methods are prototype-based methods (Linde et al., 1980; Zhang et al., 1999b) or histogram-based (cell-based) approaches (Wang et al., 1997; Hinneburg and Keim, 1998). Local kernel methods relying on range queries supported by multidimensional indexes (Ester et al., 1996) have an acceptable run-time for multidimensional spaces with dimensionality $d \leq 16$. For spaces with higher dimensionality the query time converges to the time needed by a linear scan, which causes a quadratic run-time behavior of the cluster algorithm. In Hinneburg and Keim (1998) we proposed to identify the clusters as the highly populated bins of a multidimensional histogram and connected them with the surrounding populated bins using a CubeMap data structure.

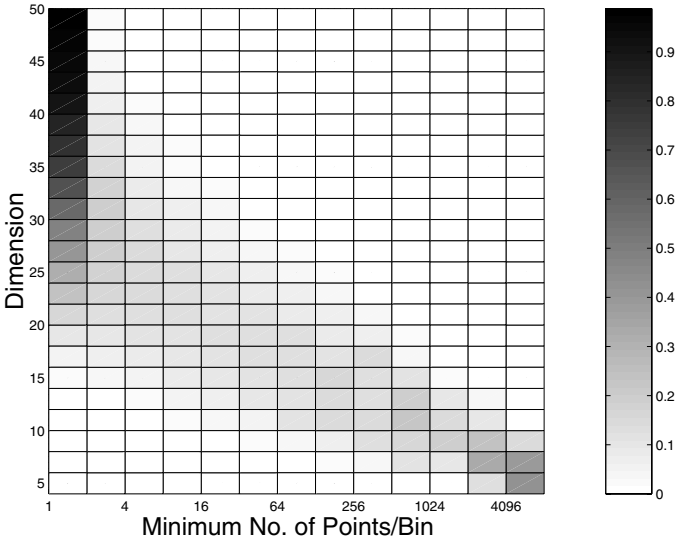


Fig. 13. Bin population in higher dimensions: due to the inherent sparsity of high-dimensional spaces the number of highly populated bins decreases with increasing dimensionality. For the experiments we used data with 10 clusters, each having a normal distribution. The data set contains 100,000 data points and has a maximum dimensionality $d_{\max} = 50$. In high-dimensional spaces, however, nearly all data points are in bins which contain only a single data point. This effect makes the selection of highly populated bins infeasible.

A general problem of multidimensional histograms, however, is that highly populated bins become unlikely in high-dimensional spaces. An example is shown in Fig. 13. A better method to approximate the density function is to use average shifted histograms (ASH) (Scott, 1985, 1992). The idea of ASHs is to construct several simple histograms with the same bin width h but a shifted origin $o = \delta h \cdot e$ (e is the unit vector). Note that histograms are very sensitive to the choice of origin. The density at a single point $x \in F$ is estimated by averaging the densities (transformed bin counts) from the multiple histograms. When the steps between the shifts become infinitesimally small the resulting density function approaches the density of the local kernel. For our experiments we used the following shifts (for d-dimensional data): $o_1 = 1/2h \cdot e_1, \ldots, o_d = 1/2h \cdot e_d$, $o_{\text{diag}} = 1/2h \cdot [1, \ldots, 1]$ and $o_0 = [0, \ldots, 0]$, where e_i is the unit vector with a 1 in dimension i and 0 else. This means that we have to construct $(d + 2)$ d-dimensional histograms.

In Hinneburg and Keim (1998) we proposed the key idea of our efficient implementation of high-dimensional histograms. Given the minimal bounding rectangle MBR = [Min, Max], (Min, Max $\in F^d$) of the data set, the histogram may contain

$$M = \prod_{i=1}^{d} \left\lceil \frac{|\text{Min}_i - \text{Max}_i|}{h} \right\rceil$$

bins. The number of possible bins M grows exponentially. However, since a populated bin has to contain at least one data point, the number of populated bins is at most N. The populated bins can be efficiently stored in a heap data structure, if for each bin a unique and sortable key can be determined. If the MBR is known such a key can be efficiently determined by linearization. In the case that the MBR is not known, such a key can be

determined by discretization of the axes and converting the discrete vector into a string which then can be sorted lexicographically. Possible heap data structures are randomized search trees for main memory (Mehlhorn and Näher, 1999) or the B^+ tree for storage on hard disk. In the latter case, the histogram can be determined with standard SQL92 queries and can be stored in a table. The following example SQL statement generates an equi-distant 15-dimensional histogram with bin width 0.2 from the numerical attributes col1, ..., col15 of table 'DataTable' by producing the key of a bin and the corresponding bin count:

```
select  concat(
    round(col1/0.2)," ",round(col2/0.2)," ",round(col3/0.2)," ",
    round(col4/0.2)," ",round(col5/0.2)," ",round(col6/0.2)," ",
    round(col7/0.2)," ",round(col8/0.2)," ",round(col9/0.2)," ",
    round(col10/0.2)," ",round(col11/0.2)," ",round(col12/0.2)," ",
  round(col13/0.2)," ",round(col14/0.2)," ",round(col15/0.2)
    ) as key_attr,
    count(*) as density
from DataTable
group by key_attr
```

In addition to the bin count, sufficient statistics ($LS = \sum x_i$, $SS = \sum x_i^2$, N) can be generated for each bin by using the sum aggregation function. The local density function can then be approximated using average shifted histograms and the sufficient statistics of the bins (Scott, 1985, 1992). Note that the algorithm for approximating the density estimation function can be replaced by other methods, like kernel function based on prototypes from LBG (Linde et al., 1980) or BIRCH (Zhang et al., 1999b).

The next step of our algorithm is the clustering step. In the clustering step, first the density of the noise is estimated using the heuristic described in Section 4.2. The heuristic requires the calculation of the density at each data point. After determining ξ for all data points x with $\hat{f}(x) > \xi$, the density attractors are determined. The hill-climbing procedure is guided by the gradient $\nabla \hat{f}^D_{\text{Gauss}}(x)$ of the local density function, which can be approximated in a way similar to the density function. The density attractor for a point x is computed as

$$x = x^0, \quad x^{i+1} = x^i + \delta \frac{\nabla \hat{f}^D_{\text{Gauss}}(x^i)}{\|\nabla \hat{f}^D_{\text{Gauss}}(x^i)\|}$$

The δ is a parameter of the hill-climbing procedure which controls the speed of convergence. Dynamic setting of δ can be done similarly to other hill-climbing procedures as, for example, reported in Bock (1974) and Rojas (1996). The calculation of the hill-climbing procedure stops at $k \in N$ if $\hat{f}^D(x^{k+1}) < \hat{f}^D(x^k)$ and takes $x^* = x^k$ as a new density attractor. After determining the density attractor x^* for a point x and $\hat{f}^D_{\text{Gauss}}(x^*) \geq \xi$, the point x is classified and attached to the cluster belonging to x^*. For efficiency reasons, the algorithm stores all points x' with $d(x^i, x') \leq h/2$ for any step $0 \leq i \leq k$ during the hill-climbing procedure and attaches these points to the cluster of x^* as well. Using this heuristics, all points which are located close to the path from x to its density attractor can be classified without applying the hill-climbing procedure.

5.3. Time Complexity

First, let us recall the main steps of our algorithm (see Fig. 14). The construction of the ASH causes the construction of $d + 2$ simple histograms. The construction of a simple

DENCLUE (D, h):
(1) ASH $\leftarrow$ DetAvrShiftedHist(D, h)
(2) $D_{\text{cluster}}, \xi \leftarrow$ DetHighDensityAreas(ASH, D)
(3) clusters $\leftarrow$ DetDensAttractors(ASH, D_{cluster}, ξ)

Fig. 14. Sketch of the DENCLUE algorithm.

histogram takes time in $O(d \cdot N \cdot \log N)$ and space in $O(d \cdot N)$. So the construction of the ASH takes time in $O(d^2 \cdot N \cdot \log(N))$ and space in $O(d^2 \cdot N)$.

Step (2) requires N queries of the ASH and takes time in $O(d^2 \cdot N \cdot \log(N))$. Since the resulting clustered data D_{cluster} may include all data points in the worst case, step (3) may also need time $O(d^2 \cdot N \cdot \log(N))$. This means that the DENCLUE algorithm has a worst case time complexity of $O(d^2 \cdot N \cdot \log(N))$ and space complexity of $O(d^2 \cdot N)$.

6. Experimental Evaluation

To show the practical relevance of our method, we performed an experimental evaluation of the DENCLUE algorithm and compared it to other efficient and effective clustering algorithms: DBSCAN (Ester et al., 1996), BIRCH (Zhang et al., 1996), and k-means (Linde et al., 1980; Gersho and Gray, 1992). The test data sets used for the experiments are real multimedia data sets from a CAD and a molecular biology application as well as synthetic data sets.

6.1. Comparison with Index-Based Methods

Index-based clustering methods perform a nearest neighbor query (DBCLASD; Xu et al., 1998) or a range query (DBSCAN; Ester et al., 1996) for each data point to estimate the density at the data point. The queries are supported by a multidimensional index structure, such as R*-tree, X-tree, VA file, or IQ-tree. In Weber et al. (1998) it is shown that the query time of a multidimensional index structure approaches the performance of the linear scan with increasing dimensionality. For each data point at least one nearest neighbor or range query has to be performed, and the worst case time complexity of index-based clustering algorithms approaches a quadratic run-time behavior. We support these theoretical considerations by comparing the run-time of the R*-tree-based implementation of DBSCAN with DENCLUE.

The data used for the efficiency tests is polygonal CAD data which are transformed into 11-dimensional feature vectors using a Fourier transformation. We extended the original data set of 40,000 data points by large amounts of uniformly distributed noise and varied the number of data points between 20,000 and 100,000. We set the parameters of our algorithm and DBSCAN such that both algorithms produce the same clustering of the data set, i.e., the effectiveness of both approaches is the same for $\sigma = EPS$ and $\xi = MinPts \cdot \frac{c_{\text{dist}} \cdot 2^{d-1}}{N \cdot EPS^d}$ as discussed in Section 3.2.

The results of our experiments are presented in Fig. 15. The figure shows that DBSCAN has a slightly super-linear performance, and in Fig. 16(a) we show the speed-up of our approach (DENCLUE is up to a factor of 45 faster than DBSCAN). Note that the performance of DBSCAN strongly depends on the index used and the run-time may vary for other index structures. Since in Fig. 15 the performance of our approach is difficult to discern, Fig. 16(b) shows the performance of DENCLUE separately. The good performance of our approach results from the fact that, in contrast to a multidimensional index

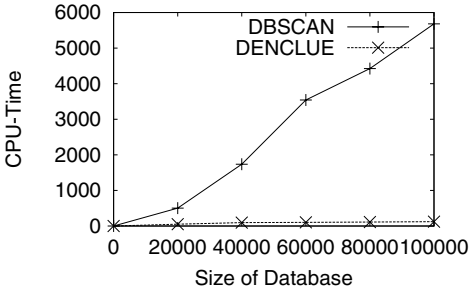
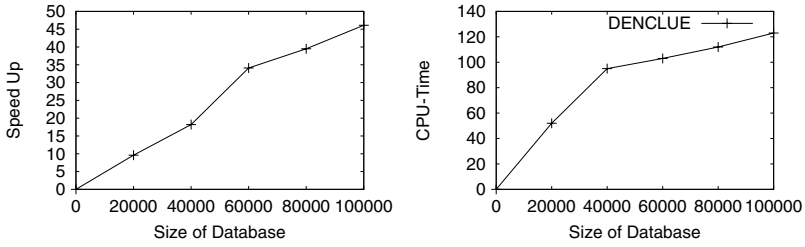


Fig. 15. Comparison of DENCLUE and DBSCAN.



(a) Speed-up (b) DENCLUE

Fig. 16. Performance of DENCLUE.

structure, heap-based histograms guarantee a query time of $O(\log(N))$. An additional performance gain comes from the fact that only data points in highly populated regions are actually considered in the clustering step. All other data points are only considered in the construction of the heap-based histograms.

6.2. Comparison with Prototype-Based Approaches

In this subsection we compare our approach with two other methods, namely BIRCH and k-means. Both methods reduce the data set (N data points) to a set of prototypes (k prototypes with $k < N$). There are two ways to use the prototypes: The first possibility is to directly use the prototypes as cluster centers and to build a clustering by assigning each data point to the nearest prototype. The second possibility is to use the prototypes for density estimation by reducing the time complexity of the estimation function for an estimate at a single point from $O(N)$ to $O(k)$.

6.2.1. Comparison with Prototype-Based Clustering (k-Means)

The k-means algorithm (also called LBG (Linde et al., 1980)) is very efficient and therefore applicable to large high-dimensional data sets. The complexity of k-means is $O(d \cdot k \cdot \#(\text{iterations}) \cdot N)$ when the Euclidean metric is used. In this subsection we evaluate the effectiveness of the clusterings produced by k-means.

Table 2. Effectiveness problems of k-means. The table shows the confusion matrices of the clustering for $2 \leq k \leq 11$. A cell (data, i)/(noise, i) of the confusion matrix specifies how many data/noise points belong to cluster i. The experiments show that k-means needs 11 prototypes to discern the two clusters in the data

k		p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11
2	data	36,704	0									
	noise	26,779	36,516									
3	data	36,704	0	0								
	noise	19,238	22,970	21,087								
4	data	36,581	123	0	0							
	noise	13,540	15,138	18,485	16,132							
5	data	36,703	1	0	0	0						
	noise	11,316	13,014	13,609	13,396	11,960						
6	data	36,440	248	16	0	0	0					
	noise	9216	10,861	10,559	11,224	11,040	10,395					
7	data	36,507	54	46	44	23	16	14				
	noise	7992	8793	9856	9117	9192	9229	9116				
8	data	36,475	108	77	38	5	1	0	0			
	noise	7155	8944	8621	8553	7640	7271	7643	7468			
9	data	33,030	3512	76	62	14	10	0	0	0		
	noise	5706	6299	7684	7257	7472	7271	7532	7130	6944		
10	data	36,580	51	32	19	14	8	0	0	0	0	
	noise	6035	6653	6402	6087	6890	6463	6809	6292	5842	5822	
11	data	23,144	13,514	18	14	6	6	1	1	0	0	0
	noise	4012	4764	6441	6307	6420	5853	6226	5760	6102	6039	5371

The main problem of using k-means directly as a clustering algorithm is that it only provides useful results if the data set is a mixture of Gaussian distributions. For a high-dimensional data set, however, it is difficult to decide whether it follows a Gaussian distribution. To make the difficulty clearer we performed a series of experiments using the extended CAD data set (36,706 clustered points and 63,295 noise points). We used random starting points for the k-means algorithm. Since averaging the results is not meaningful for a confusion matrix, we exemplify the results by a typical confusion matrix from our experiments, which is shown in Table 2. From the results of DENCLUE and DBSCAN we knew that the data set contains two major clusters. We marked the added noise points to determine how k-means deals with noise. Table 2 shows the results of k-means for different values of k ($2 \leq k \leq 11$). The results show that until we use 11 prototypes the two clusters are not discovered by k-means. With fewer than 11 prototypes both clusters are represented by one prototype and the remaining prototypes are used to approximate the noise. With 11 prototypes the two clusters are found and with more than 11 prototypes the two clusters are already split. Note that we were only able to discover this behavior because we marked the noise points. In a normal situation, it would not be possible to discover this effect. The reason for the effectiveness problems of k-means is that the quantization error function which is optimized by k-means does not contain sufficient information to decide whether the algorithm discovered a good clustering.

6.2.2. Comparison with Prototype-Based Density Estimation

The results from k-means as well as BIRCH can be used to construct a density estimation function, which has a lower run-time complexity but a higher estimation error than the density function based on all data points. Especially BIRCH has been reported to provide a very efficient pre-clustering step (Zhang et al., 1996, 1997). Since the result of the pre-clustering step is a set of prototypes, it can also be used for density estimation (Zhang

et al., 1999b). In the previous subsection we showed some drawbacks of prototype-based clustering (k-means). Now we examine the quality of prototype-based density estimation and compare this to our approach.

Prototype-based density estimation as well as ASHs are a special case of the framework for efficient density estimation WARP (Härdle and Scott, 1992). We use the multidimensional kernel density function $\hat{f}_p(x)$ proposed in Zhang et al. (1999b) for the prototype sets resulting from BIRCH and k-means. We compare these functions with our approximation of the local density function based on ASHs. We use the comparison method from Zhang et al. (1999b) to evaluate the quality of the density functions. Zhang et al. (1999b) proposed to compare two density functions by averaging the relative density difference at several breakpoints. Using a standard kernel density function with Gaussian kernel as reference function the relative density difference is defined as

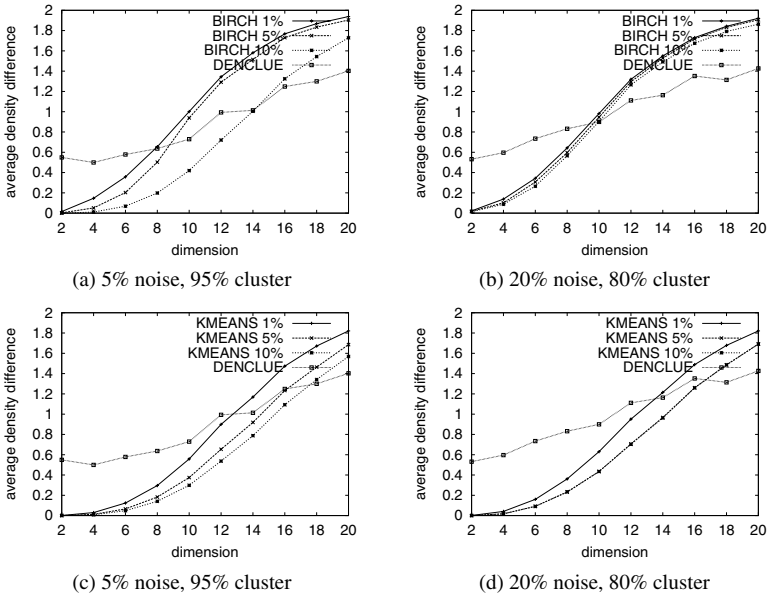
$$D(\hat{f}, x) = \frac{2 \cdot |\hat{f}_{\mathrm{KDE}}(x) - \hat{f}(x)|}{\hat{f}_{\mathrm{KDE}}(x) + \hat{f}(x)}$$

For our first experiments, we use synthetic data and 100 randomly chosen break points. The data consists of 5% and 20% noise (uniformly distributed) and 95% and 80% clustered points (normally distributed) respectively. In the experiments, we examine the dependency between the relative density difference and the dimensionality. We configured BIRCH and LBG to use prototype sets with a size of 1%, 5% and 10% of the original data set. The smoothing parameter h is chosen to be the same for all density functions. The average density difference is the average of the relative density difference of the 100 breakpoints. The plots show the average results of 10 experiments with randomly generated data distributions with the parameters described above.

Figure 17 shows that for a higher dimensionality our local density approach is better than prototype-based density estimation with BIRCH or k-means. The diagrams also show that prototype-based density estimation improves when more prototypes are used (1%, 5%, 10% of the size of the data set). Note that the best results of prototype-based density estimation require a prototype set with a size of 10% of the original data set, which means that the clustering algorithm would approach a quadratic run-time behavior. In this case, an evaluation of the prototype-based density function at a single point would become an expensive operation. The local density function based on heap-based histograms guarantees a run-time complexity for a single point evaluation of $O(\log N)$. Since heap-based histograms can be determined with SQL92 and can be stored in a database table, the worst case space complexity of $O(N)$ may be acceptable for high-dimensional data. The optimal trade-off between prototype-based density estimation and the local density function based on ASHs depends on the given computer system and the software used. As a consequence, we propose that DENCLUE uses a prototype-based density estimation function for low-dimensional data ($d \leq 12$) and a local density function based on ASHs for higher-dimensional data.

6.3. Application to Molecular Biology

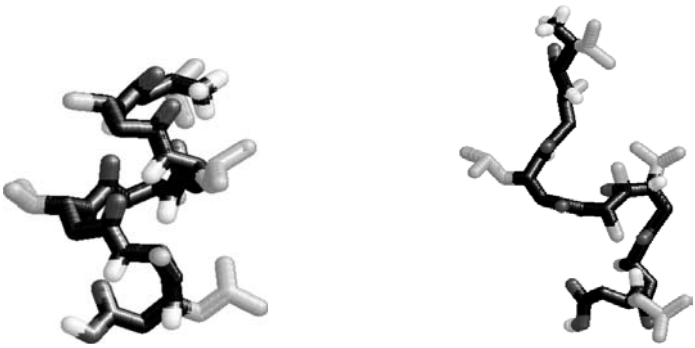
To evaluate the effectiveness of DENCLUE, we applied our method to an application in the area of molecular biology. The data used for our tests comes from a complex simulation of a very small but flexible peptide. Performing the simulation took several weeks of CPU time. The data generated by the simulation describes the conformation of the peptide as a 19-dimensional point in the dihedral angle space (Daura et al., 1998). The simulation was done for a period of 50 nanoseconds with two snapshots



(a) 5% noise, 95% cluster (b) 20% noise, 80% cluster

(c) 5% noise, 95% cluster (d) 20% noise, 80% cluster

Fig. 17. Comparison of the local density function based on average shifted histograms (DENCLUE) and prototype-based density estimation (BIRCH, *k*-means).



(a) Folded state (b) Unfolded state

Fig. 18. Folded conformation of the peptide.

taken every picosecond, resulting in about 100,000 data points. The purpose of the simulation was to determine the behavior of the molecule in the conformation space. Due to the large amount of high-dimensional data, it is difficult to find, for example, the preferred conformations of the peptide. This, however, is important for applications in the pharmaceutical industry since small and flexible peptides are the basis for many medicaments. The flexibility of the peptides, however, is also responsible for the fact that the peptide has many intermediate non-stable conformations which cause the data to contain large amounts of noise (more than 50%).

Table 3. Comparison of the quality and CPU costs of the density estimations of BIRCH and DENCLUE on the molecular biology data set. For the experiments, we use main memory-based algorithms. For the density comparison, we use standard KDE and 1000 breakpoints

	DENCLUE	BIRCH 100	BIRCH 500	BIRCH 1000
Avg. density diff.	1.45	1.97	1.66	1.45
CPU time (s)	1	0.5	1	3

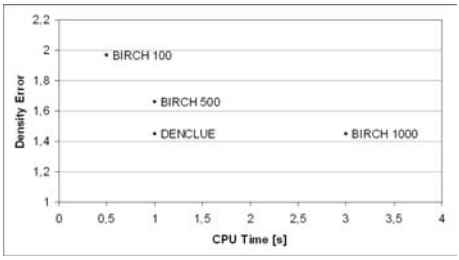


Fig. 19. Comparison of run-time and the averaged density estimation difference using standard kernel density estimation as reference.

In Figs 18(a) and 18(b), we show the two most important conformations (corresponding to the largest clusters of conformations in the 19-dimensional dihedral angle space) found by DENCLUE. The two conformations correspond to a folded and an unfolded state of the peptide (Daura et al., 1998).

In Table 3 we compare our approach to the density estimation approach of BIRCH (Zhang et al., 1999b). The results show that BIRCH may reach an equal quality but requires about 1000 prototypes for achieving that, which also increases the CPU costs. From Fig. 19 it becomes clear that there is a trade-off between the time efficiency and quality of the estimation. Figure 19 also shows clearly that DENCLUE provides a better compromise between time complexity and quality.

7. Conclusions

In this paper, we propose a new KDE-based algorithm to clustering in large multimedia databases with noise. We recall the basics of KDE and introduce different notions of clusters which are based on determining the density attractors of the density function. For an efficient implementation of our approach, we introduce a local density function which can be efficiently determined using an ASH-based representation of the data. We show the generality of our approach, i.e., that we are able to simulate a locality-based, partition-based, and hierarchical clustering. We also formally show the noise invariance and error bounds of our approach. An experimental evaluation shows the advantages of DENCLUE over BIRCH and k-means with respect to efficiency and effectiveness. Due to the promising results of the experimental evaluation, we believe that our method will have a noticeable impact on the way clustering will be done in the future. Our plans for future work include an application and fine tuning of our method for specific applications.

References

Bock HH (1974) Automatic classification. Vandenhoeck and Ruprecht, Göttingen

Cutting DR, Pedersen JO, Karger DR, Tukey JW (1992) A cluster-based approach to browsing large document collections. In Proceedings of the 15th annual international ACM SIGIR conference on research and development in information retrieval. ACM Press, New York, pp 318–329

Daura X, Jaun B, Seebach D, van Gunsteren WF, Mark AE (1998) Reversible peptide folding in solution by molecular dynamics simulation. Journal of Molecular Biology 280: 925–932

Ester M, Kriegel H, Xu X (1995) Knowledge discovery in large spatial databases: focusing techniques for efficient class identification. In SSD'95, fourth international symposium on large spatial databases. Lecture Notes in Computer Science 951, Springer, Berlin, pp 67–82

Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In KDD'96, Proceedings of the second international conference on knowledge discovery and data mining. AAAI Press, Menlo Park, CA, pp 226–231

Ester M, Kriegel H-P, Sander J, Xu X (1997) Density-connected sets and their application for trend detection in spatial databases. In Proceedings of the third international conference on knowledge discovery and data mining (KDD-97). AAAI Press, Menlo Park, CA, pp 10–15

Fritzke B (1995) A growing neural gas network learns topologies. Advances in Neural Information Processing Systems 7. MIT Press, Cambridge, MA, pp 625–632

Fritzke B (1997) The LBG-U method for vector quantization: an improvement over LBG inspired from neural networks. Neural Processing Letters 5(1). Kluwer Publishers

Fukunaga K, Hostler L (1975) The estimation of the gradient of a density function, with application in pattern recognition. IEEE Transactions on Information Theory 21: 32–40

Gersho A, Gray RM (1992) Vector quantization and signal compression. Kluwer, Dordrecht

Hafner JL, Sawhney HS, Equitz W, Flickner M, Niblack W (1995) Efficient color histogram indexing for quadratic form distance functions. IEEE Transactions on Pattern Analysis and Machine Intelligence 17(7): 729–736

Härdle W, Scott DW (1992) Smoothing in low and high dimensions by weighted averaging using rounded points. Computational Statistics 7: 97–128

Hinneburg A, Keim D (1998) An efficient approach to clustering in large multimedia databases with noise. In KDD'98, Proceedings of the fourth international conference on knowledge discovery and data mining. AAAI Press, Menlo Park, CA, pp 58–65

Jagadish HV (1991) A retrieval technique for similar shapes. In Proceedings of the 1991 ACM SIGMOD international conference on management of data, 1991. ACM Press, New York, pp 208–217

Kukich K (1992) Techniques for automatically correcting words in text. ACM Computing Surveys 24(4): 377–440

Linde Y, Buzo A, Gray R (1980) An algorithm for vector quantizer. IEEE Transactions on Communications COM-28: 84–95

Martinetz T, Schulten K (1993) A neural gas network learns topologies. Neural Networks 7:507–522

Mehlhorn K, Näher S (1999) The LEDA platform of combinatorial and geometric computing. Cambridge University Press, Cambridge, UK

Mehrotra R, Gary JE (1995) Feature-index-based similar shape retrieval. In Proceedings of the third IFIP 2.6 working conference on visual database systems. VDB'95, pp 46–65

Nadaraya EA (1965) On nonparametric estimates of density functions and regression curves. Theory of Probability and its Applications 10: 186–190

Ng RT, Han J (1994) Efficient and effective clustering methods for spatial data mining. In VLDB'94, Proceedings of 20th international conference on very large data bases, pp 144–155

Rojas R (1996) Neural networks: a systematic introduction. Springer, Berlin

Sahami M, Yusufali S, Baldonado MQW (1998) A service for organizing networked information autonomously. In Proceedings of the 3rd ACM international conference on digital libraries. ACM Press, New York, pp 200–209

Schikuta E, Erhart M (1997) The bang-clustering system: grid-based data analysis. In Lecture Notes in Computer Science 1280. Springer, Berlin, pp 513–524

Schnell P (1964) A method to find point-groups. Biometrika 6: 47–48

Schuster EF (1970) Note on the uniform convergence of density estimates. Annals of Mathemathical Statistics 41: 1347–1348

Scott DW (1985) Average shifted histograms: effective nonparametric density estimators in several dimensions. Annals of Statistics 13: 1024–1040

Scott DW (1992) Multivariate density estimation. Wiley, New York

Scott DW, Sheather SJ (1985) Kernel density estimation with binned data. Communications in Statistics – Theory and Methods 14: 1353–1359

Sibson R (1973) Slink: an optimally efficient algorithm for the single-linkage cluster method. Computer Journal 16(1): 30–34
Silverman BW (1982) Kernel density estimation using the fast fourier transformation. Applied Statistics 31: 93–99
Silverman BW (1986) Density estimation for statistics and data analysis. Chapman & Hall, London
Wallace T, Winz P (1980) An efficient three-dimensional aircraft recognition algorithm using normalized fourier descriptors. Computer Graphics and Image Processing 13
Wand MP, Jones MC (1995) Kernel smoothing. Chapman & Hall, London
Wang W, Yang J, Muntz RR (1997) Sting: a statistical information grid approach to spatial data mining. In VLDB'97, Proceedings of 23rd international conference on very large data bases. pp 186–195
Weber R, Schek H-J, Blott S (1998) A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In Gupta A, Shmueli O, Widom J (eds). VLDB'98, Proceedings of 24th international conference on very large data bases, 24–27 August, New York. Morgan Kaufmann, San Mates, CA, pp 194–205
Wishart D (1969) A numerical classification methods for deriving natural classes. Nature 221: 97–98
Xu X, Ester M, Kriegel H-P, Sander J (1998) A distribution-based clustering algorithm for mining in large spatial databases. In Proceedings of the 14th international conference on data engineering, USA. IEEE Computer Society, Los Alamitos, CA, pp 324–331
Zhang T, Ramakrishnan R, Livny M (1996) Birch: an efficient data clustering method for very large databases. In Proceedings of the 1996 ACM SIGMOD international conference on management of data. ACM Press, New York, pp 103–114
Zhang T, Ramakrishnan R, Livny M (1997) Birch: a new data clustering algorithm and its applications. Data Mining and Knowledge Discovery 1(2): 141–182
Zhang B, Hsu M, Dayal U (1999a) K-harmonic means: a data clustering algorithm. Technical Report HPL-1999-124, HP Research Labs
Zhang T, Ramakrishnan R, Livny M (1999b) Fast density estimation using cf-kernel for very large databases. In Proceedings of the fifth ACM SIGKDD international conference on knowledge discovery and data mining. ACM Press, New York, pp 312–316

Author Biography

Daniel A. Keim works in the area of data mining and information visualization, as well as similarity search and indexing in multimedia databases. In the field of visual data mining, he has developed several novel techniques which use visualization technology for the purpose of exploring large databases. He has published extensively on data mining and information visualization; he has given tutorials on related issues at several large conferences including Visualization, SIGMOD, VLDB, and KDD; he has been program co-chair of the IEEE Information Visualization Symposia in 1999 and 2000; he is program co-chair of KDD-2002; and he is assistant editor of *TVCG* and the *Information Visualization Journal*. Daniel Keim received his diploma (equivalent to an MS degree) in Computer Science from the University of Dortmund in 1990 and his PhD in Computer Science from the University of Munich in 1994. He has been assistant professor at the CS department of the University of Munich, associate professor at the CS department of the Martin-Luther-University Halle, and full professor at the CS department of the University of Constance, Germany. Currently, he is on leave of absence from the University of Constance and works as senior researcher at AT&T Shannon Labs, New Jersey, USA.

Alexander Hinneburg works in the areas of clustering and visualization. He has developed and published novel algorithms including *OptiGrid* and *HD-Eye*. Together with Daniel Keim he has given tutorials about clustering at the SIGMOD, SIGKDD, and PKDD conferences. His research interests also include bioinformatics, similarity search, and information visualization. Alexander Hinneburg received his diploma (equivalent to an MS degree) in Computer Science from the University of Halle in 1997 and is currently finishing writing his PhD thesis.

Correspondence and offprint requests to: A. Hinneburg, University of Halle, Institute of Computer Science, von Seckendorf Platz 1, 06120 Halle (Saale), Germany.
Email: hinneburg@informatik.uni-halle.de