First publ. in: Proceedings of the 4th International Conference on Knowledge Discovery and Datamining (KDD'98), New York, NY, September, 1998, pp. 58-65

An Efficient Approach to Clustering in Large Multimedia Databases with Noise

Alexander Hinneburg, Daniel A. Keim

Institute of Computer Science, University of Halle, Germany
{hinneburg, keim}@informatik.uni-halle.de

Abstract

Several clustering algorithms can be applied to clustering in large multimedia databases. The effectiveness and efficiency of the existing algorithms, however, is somewhat limited, since clustering in multimedia databases requires clustering high-dimensional feature vectors and since multimedia databases often contain large amounts of noise. In this paper, we therefore introduce a new algorithm to clustering in large multimedia databases called DENCLUE (DENsity-based CLUstEring). The basic idea of our new approach is to model the overall point density analytically as the sum of 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 based on the overall density function. The advantages of our new 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 DBSCAN shows the superiority of our new approach.

Keywords: Clustering Algorithms, Density-based Clustering, Clustering of High-dimensional Data, Clustering in Multimedia Databases, Clustering in the Presence of Noise

1 Introduction

Because of the fast technological progress, the amount of data which is stored in databases increases very fast. The types of data which are stored in the computer become increasingly complex. In addition to numerical data, complex 2D and 3D multimedia data such as image, CAD, geographic, and molecular biology data are stored in databases. For an efficient retrieval, the complex data is usually transformed into high-dimensional feature vectors. Examples of feature vectors are color histograms [SH94], shape descriptors [Jag91, MG95], Fourier vectors [WW80], text descriptors [Kuk92], etc. In many of the mentioned applications, 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 algorithms do not work efficiently on high-dimensional data.The methods which are applicable to databases of high-dimensional feature vectors are the methods which are known from the area of spatial data mining. The most prominent representatives are partitioning algorithms such as CLARANS [NH94], hierarchical clustering algorithms,and locality-based clustering algorithms such as (G)DBSCAN [EKSX96, EKSX97] and DBCLASD [XEKS98]. The basic idea of *partitioning algorithms* is to partition the database into k clusters which are represented by the gravity of the cluster (k-means) or by one representative object of the cluster (k-medoid). Each object is assigned to the closest cluster. A well-known partitioning algorithm is CLARANS which uses a randomized and bounded search strategy to improve the performance. *Hierarchical clustering algorithms* decompose the database into several levels of partitionings 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 quatratic in the number of data objects. More efficient are locality-based 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 randomly distributed, allowing DBCLASD to work without any input parameters. A performance comparison [XEKS98] shows that DBSCAN is slightly faster than DBCLASD and both, DBSCAN and DBCLASD are much faster than hierarchical clustering algorithms and partitioning algorithms such as CLARANS. To improve the efficiency, optimized clustering techniques have been proposed. Examples include R*-Tree-based Sampling [EKX95], Gridfile-based clustering [Sch96], BIRCH [ZRL96] which is based on the Cluster-Feature-tree, and STING which uses a quadtree-like structure containing additional statistical information [WYM97].

A problem of the existing approaches in the context of clustering multimedia data is that most algorithms are not designed for clustering high-dimensional feature vectors and therefore, the performance of existing algorithms degenerates rapidly with increasing dimension. In addition, few algorithms can deal with databases containing large amounts of noise, which is quite common in multimedia databases where usually only a small portion of the database forms the interesting subset which accounts for the clustering. Our new approach solves these problems. It works efficiently for high-dimensional data sets and allows arbitrary noise levels while still guaranteeing to find the clustering. In addition, our approach can be seen as a generalization of different previous clustering approaches – partioning-based, hierarchical, and locality-based clustering. Depending on the parameter settings, we may initalize our approach to provide similar (or even the same) results as different other clustering algorithms, and due to its generality, in same cases our approach even provides a better effectiveness. In addition, our algorithm works efficiently on very large amounts of high-dimensional data, outperforming one of the fastest existing methods (DBSCAN) by a factor of up to 45.

The rest of the paper is organized as follows. In section 2, we introduce the basic idea of our approach and formally define influence functions, density functions, density-attractors, clusters, and outliners. In section 3, we discuss the properties of our approach, namely its generality and its invariance with respect to noise. In section 4, 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 5, we provide an experimental evaluation comparing our approach to previous approaches such as DBSCAN. 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 6 summarizes our results and discusses their impact as well as important issues for future work.

2 A General Approach to Clustering

Before introducing the formal definitions required for describing our new approach, in the following we first try to give a basic understanding of our approach.

2.1 Basic Idea

Our new approach is based on the idea that the influence of each data point can be modeled formally using a mathematical function which we call influence function. The influence function can be seen as a function which describes the impact of a data point within its neighborhood. Examples for influence functions are parabolic functions, square wave function, or the Gaussian function. The influence function is applied to each data point. The overall density of the data space can be calculated as the sum of the influence function of all data points. Clusters can then be determined mathematically by identifying density-attractors. Density-attractors are local maxima of the overall density function. If the overall density function is continuous and differentiable at any point, determining the density-attractors can be done efficiently by a hill-climbing procedure which is guided by the gradient of the overall density function. In addition, the mathematical form of the overall density function allows clusters of arbitrary shape to be described in a very compact mathematical form, namely by a simple equation of the overall density function. Theoretical results show that our approach is very general and allows different clusterings of the data (partition-based, hierarchical, and locality-based) to be found. We also show (theoretically and experimentally) that our algorithm is invariant against large amounts of noise and works well for high-dimensional data sets.

The algorithm DENCLUE is an efficient implementation of our idea. The overall density function requires to sum up the influence functions of all data points. Most of the data points, however, do not actually contribute to the overall density function. Therefore, DENCLUE uses a local density function which considers only the data points which actually contribute to the overall density function. This can be done while still guaranteeing tight error bounds. An intelligent cell-based organization of the data allows our algorithm to work efficiently on very large amounts of high-dimensional data.

2.2 Definitions

We first have to introduce the general notion of influence and density functions. Informally, the influence 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 density function at a point $x \in F^d$ is defined as the sum of the influence functions of all data objects at that point (cf. [Schn64] or [FH 75] for a similar notion of density functions).

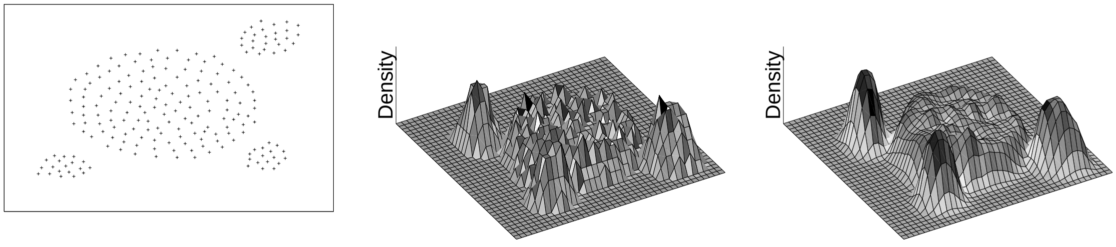
Def. 1 (Influence & Density Function)
The *influence function* of a data object $y \in F^d$ is a function $f_B^y : F^d \longrightarrow R_0^+$ which is defined in terms of a basic influence function f_B

$$f_B^y(x) = f_B(x, y).$$

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

$$f_B^D(x) = \sum_{i=1}^{N} f_B^{x_i}(x).$$

In principle, the influence function can be an arbitrary function. For the defintion of specific influence functions, we need a distance function $d : F^d \times F^d \longrightarrow R_0^+$ which determines the distance of two d-dimensional feature vectors. The distance function has to be reflexive and symmetric. For simplicity, in the following we assume a Euclidean distance function. Note, however, that the definitions are independent from the choice of the distance function. Examples of basic influence functions are:



(a) Data Set (b) Square Wave (c) Gaussian

Figure 1: Example for Density Functions

1. Square Wave Influence Function
$$f_{Square}(x,y) = \begin{cases} 0 & \text{if } d(x,y) > \sigma \\ 1 & \text{otherwise} \end{cases}$$
2. Gaussian Influence Function
$$f_{Gauss}(x,y) = e^{-\frac{d(x,y)^2}{2\sigma^2}}$$

The density function which results from a Gaussian influence function is
$$f^D_{Gauss}(x) = \sum_{i=1}^{N} e^{-\frac{d(x,x_i)^2}{2\sigma^2}}.$$

Figure 1 shows an example of a set of data points in 2D space (cf. figure 1a) together with the corresponding overall density functions for a square wave (cf. figure 1b) and a Gaussian influence function (cf. figure 1c). In the following, we define two different notions of clusters – center-defined clusters (similar to k-means clusters) and arbitrary-shape clusters. For the definitions, we need the notion of density-attractors. Informally, density attrators are local maxima of the overall denstiy function and therefore, we also need to define the gradient of the density function.

Def. 2 (Gradient)
The gradient of a function $f^D_B(x)$ is defined as
$$\nabla f^D_B(x) = \sum_{i=1}^{N} (x_i - x) \cdot f^{x_i}_B(x).$$

In case of the Gaussian influence function, the gradient is defined as:
$$\nabla f^D_{Gauss}(x) = \sum_{i=1}^{N} (x_i - x) \cdot e^{-\frac{d(x,x_i)^2}{2\sigma^2}}.$$

In general, it is desirable that the influence function is a symmetric, continuous, and differentiable function. Note, however, that the definition of the gradient is independent of these properties. Now, we are able to define the notion of density-attractors.

Def. 3 (Density-Attractor)
A point $x^* \in F^d$ is called a *density-attractor* for a given influence function, iff x^* is a local maxium of the density-function f^D_B.
A point $x \in F^d$ is *density-attracted* to a density-attractor x^*, iff $\exists k \in N : \ d(x^k, x^*) \leq \epsilon$ with
$$x^0 = x, \ x^i = x^{i-1} + \delta \cdot \frac{\nabla f^D_B(x^{i-1})}{\|\nabla f^D_B(x^{i-1})\|}.$$

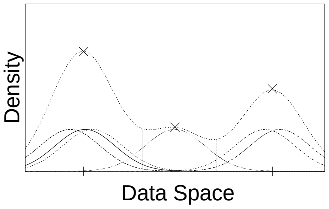


Figure 2: Example of Density-Attractors

Figure 2 shows an example of density-attrators in a one-dimensional space. For a continuous and differentiable influence function, a simple hill-climbing algorithm can be used to determine the density-attractor for a data point $x \in D$. The hill-climbing procedure is guided by the gradient of f^D_B.
We are now are able to introduce our definitions of clusters and outliers. Outliers are points, which are not influenced by "many" other data points. We need a bound ξ to formalize the "many".

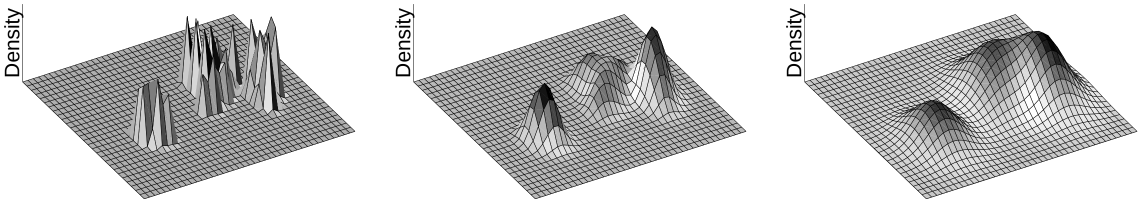
Def. 4 (Center-Defined Cluster)
A *center-defined cluster* (wrt to σ,ξ) for a density-attractor x^* is a subset $C \subseteq D$, with $x \in C$ being density-attracted by x^* and $f^D_B(x^*) \geq \xi$. Points $x \in D$ are called *outliers* if they are density-attraced by a local maximum x^*_o with $f^D_B(x^*_o) < \xi$.

This definition can be extended to define clusters of arbitrary shape.

Def. 5 (Arbitrary-Shape Cluster)
An *arbitrary-shape cluster* (wrt σ,ξ) for the set of density-attractors X is a subset $C \subseteq D$, where

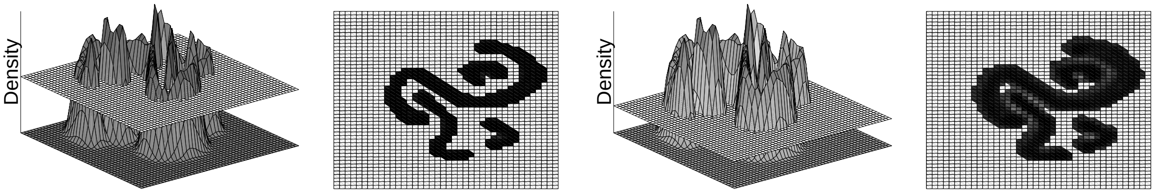
1. $\forall x \in C \ \exists x^* \in X : \ f^D_B(x^*) \geq \xi$, x is density-attracted to x^* and
2. $\forall x^*_1, x^*_2 \in X : \exists$ a path $P \subset F^d$ from x^*_1 to x^*_2 with $\forall p \in P : \ f^D_B(p) \geq \xi$.

Figure 3 shows examples of center-defined clusters for different σ. Note that the number of clusters found by our approach varies depending on σ. In figures 4a and 4c, we provide examples of the density function together with the plane for different ξ. Figures 4b and 4d show the resulting arbitrary-shape clusters. The parameter σ describes the influence of a data point in the data space and ξ describes when a density-attractor is significant. In subsection 3.3, we discuss the effects of the parameters σ and ξ and provide a formal procedure of how to determine the parameters.



(a) $\sigma = 0.2$ (b) $\sigma = 0.6$ (d) $\sigma = 1.5$

Figure 3: Example of Center-Defined Clusters for different σ



(a) $\xi = 2$ (b) $\xi = 2$ (c) $\xi = 1$ (d) $\xi = 1$

Figure 4: Example of Arbitray-Shape Clusters for different ξ

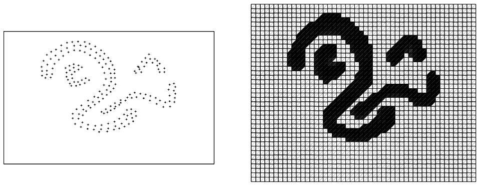
3 Properties of Our Approach

In the following subsections we discuss some important properties of our approach including its generality, noise-invariance, and parameter settings.

3.1 Generality

As already mentioned, our approach generalizes other clustering methods, namely partition-based, hierarchical, and locality-based clustering methods. Due the wide range of different clustering methods and the given space limitations, we can not discuss the generality of our approach in detail. Instead, we provide the basic ideas of how our approach generalizes some other well-known clustering algorithms.

Let us first consider the locality-based clustering algorithm DBSCAN. Using a square wave influence function with σ =EPS and an outlier-bound ξ =MinPts, the abitary-shape clusters defined by our method (c.f. definition 5) are the same as the clusters found by DBSCAN. The reason is that in case of the square wave influence 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 approach is provided in figure 5. Note that the results are only identical for a very simple influence function, namely the square wave function.



(a) DBSCAN (b) DENCLUE

Figure 5: Generality of our Approach

To show the generality of our approach with respect to partition-based clustering methods such as the k-means clustering, we have to use a Gaussian influence function. If ϵ of definition 3 equals to $\sigma/2$, then there exists a σ such that our approach will find k density-attractors corresponding to the k mean values of the k-means clustering, and the center-defined clusters (cf. definition 4) correspond to the k-means clusters. The idea of the proof of this property is based on the observation that by varying the σ we are able to obtain between 1 and N^* clusters (N^* is the number of non-identical data points) and we therefore only have to find the appropriate σ to obtain a corresponding k-means clustering. Note that the results of our approach represent a globally optimal clustering while most k-means clustering methods only provide a local optimum of partitioning the data set D into k clusters. The results k-means clustering and our approach are therefore only the same if the k-means clustering provides a global optimum.

The third class of clustering methods are the hierarchical methods. By using different values for σ and the notion of center-defined clusters according to definition 5, we are able to generate a hierarchy of clusters. If we start with a very small value for σ, we obtain N^* clusters. By increasing the σ, density-attractors start to merge and we get the next level of the hierarchy. If we further increase σ, more and more density-attractors merge and finally, we only have one density-atractor representing the root of our hierarchy.

3.2 Noise-Invariance

In this subsection, we show the ability of our approach to handle large amounts of noise. 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 belonging to D (wrt σ, ξ) and $X_C = \{\hat{x}_1^*, \ldots, \hat{x}_{\hat{k}}^*\}$ the density attractors for D_C (wrt σ, ξ_c) with ξ and ξ_c being adequately chosen, and let $\|S\|$ denote the cardinality of S. Then, we are able to show the following lemma.

Lemma 1 (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,

$$\|X\| = \|X_C\| \text{ and}$$

$$\lim_{\|D_N\|\to\infty} \left(P\Big(\sum_{i=1}^{\|X\|} d(x_i^*, \hat{x}_i^*) = 0\Big) \right) = 1.$$

Idea of the 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 $\tilde{f}^{D_N} = c$ $(0 < c \leq 1)$. According to [Schu70] and [Nad65],

$$\lim_{\|D_N\|\to\infty} \sup_{y \in DS_D} \left| c - \frac{1}{\|D_N\| \sqrt{2\pi\sigma^2}^d} f^{D_N}(y) \right| = 0$$

for any $\sigma > 0$. So the density distribution of D can approximated by

$$f^D(y) = f^{D_C}(y) + \|D_N\| \sqrt{2\pi\sigma^2}^d \cdot c$$

for any $y \in DS_D$. The second portion of this formula is constant for a given D and therefore, the density-attractors definded by f_{D_C} do not change. □

3.3 Parameter Discussion

As in most other approaches, the quality of the resulting clustering depends on an adequate choice of the parameters. In our approach, we have two important parameters, namely σ and ξ. The parameter σ 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.

Choosing a good σ can be done by considering different σ and determining the largest interval between $\sigma_{\max}$ and $\sigma_{\min}$ where the number of density-attractors $m(\sigma)$ remains constant. The clustering which results from this approach can be seen as naturally adapted to the data-set. In figure 6, we provide an example for the number of density-attractors depending on σ.

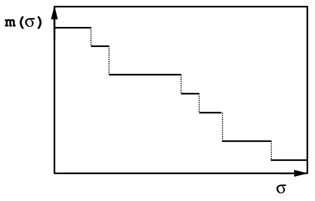


Figure 6:
Number of Density-Attractors depending on σ

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 (cf. definition 4). 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 choosen in $0 \leq \xi \leq \min_{x^* \in X}\{f^D(x^*)\}$. In most cases the database will contain noise ($D = D_C \cup D_N$, cf. subsection 3.2). According to lemma 1, the noise level can be described by the constant $\|D_N\| \cdot \sqrt{2\pi\sigma^2}^d$ and therefore, ξ should be chosen in

$$\|D_N\| \cdot \sqrt{2\pi\sigma^2}^d \leq \xi \leq \min_{x^* \in X}\{f^{D_C}(x^*)\}.$$

Note that the noise level also has an influence on the choice of σ. If the difference of the noise level and the density of small density-attractors is large, then there is a large intervall for choosing σ.

4 The Algorithm

In this section, we describe an algorithm which implements our ideas described in sections 2 and 3. For an efficient determination of the clusters, we have to find a possibility to efficiently calculate the density function and the density-attractors. An important observation is that to calculate the density for a point $x \in F^d$, only points of the data set which are close to x actually contribute to the density. All other points may be neglected without making a substantial error. Before describing the details of our algorithm, in the following we first introduce a local variant of the density function together with its error bound.

4.1 Local Density Function

The local density function is an approximation of the overall density function. The idea is to consider the influence of nearby points exactly 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 $x_1 \in near(x) : d(x_1, x) \leq \sigma_{near}$. Then, the local density is defined as follows.

Def. 6 (Local Density Function)
The local density $\hat{f}^D(x)$ is

$$\hat{f}^D(x) = \sum_{x_1 \in near(x)} f_B^{x_1}(x) .$$

The gradient of the local density function is defined similar to definition 2. In the following, we assume that

$\sigma_{near} = k\sigma$. An upper bound for the error made by using the local density function $\hat{f}^D(x)$ instead of the real density function $f^D(x)$ is given by the following lemma.

Lemma 2 (Error-Bound)
If the points $x_i \in D : d(x_i, x) > k\sigma$ are neglected, the error is bound by

$$\text{Error} = \sum_{x_i \in D,\ d(x_i,x) > k\sigma} e^{-\frac{d(x_i,x)^2}{2\sigma^2}}$$
$$\leq \|\{x_i \in D | d(x_i, x) > k\sigma\}\| \cdot e^{-k^2/2}.$$

Idea of the Proof: The error-bound assumes that all points are on a hypersphere of $k\sigma$ around x. □
The error-bound given by lemma 5 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 $k\sigma$.

4.2 The DENCLUE Algorithm

The DENCLUE algorithm works in two steps. Step one is a preclustering step, in which a map of the relevant portion of the data space is constructed. The map is used to speed up the calculation of the density function which requires to efficiently access neighboring portions of the data space. The second step is the actual clustering step, in which the algorithm identifies the density-attractors and the corresponding density-attracted points.

Step 1: The map is constructed as follows. The minimal bounding (hyper-)rectangle of the data set is divided into d-dimensional hypercubes, with an edge-length of 2σ. Only hypercubes which actually contain data points are determined. The number of populated cubes ($\|C_p\|$) can range from 1 to N^* depending on the choosen σ, but does not depend on the dimensionality of the data space. The hypercubes are numbered depending on their relative position from a given origin (cf. figure 7 for a two-dimensional example). In this way, the populated hypercubes (containing d-dimensional data points) can be mapped to one-dimensional keys. The keys of the populated cubes can be efficiently stored in a randomized search-tree or a B^+-tree.

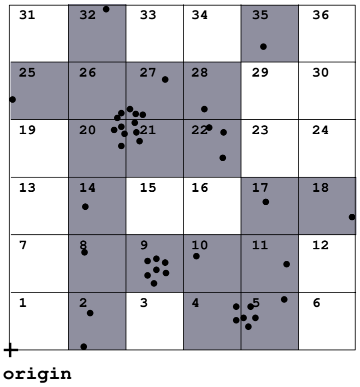


Figure 7: Map in a 2-dim. data space

For each populated cube $c \in C_p$, in addition to the key the number of points (N_c) which belong to c, pointers to those points, and the linear sum $\sum_{x \in c} x$ are stored. This information is used in the clustering step for a fast calculation of the mean of a cube ($mean(c)$). Since clusters can spread over more than one cube, neighboring cubes which are also populated have to be accessed. To speed up this access, we connect neighboring populated cubes. More formally, two cubes $c_1, c_2 \in C_p$ are connected if $d(mean(c_1), mean(c_2)) \leq 4\sigma$. Doing that for all cubes would normally take $O(C_p^2)$ time. We can, however, use a second outlier-bound ξ_c to reduce the time needed for connecting the cubes. Let $C_{sp} = \{c \in C_p | N_c \geq \xi_c\}$ be the set of highly populated cubes. In general, the number of highly populated cubes C_{sp} is much smaller than C_p, especially in high-dimensional space. The time needed for connecting the highly populated cubes with their neighbors is then $O(\|C_{sp}\| \cdot \|C_p\|)$ with $\|C_{sp}\| << \|C_p\|$. The cardinality of C_{sp} depends on ξ_c. A good choice for ξ_c is $\xi_c = \xi/2d$, since in high-dimensional spaces the clusters are usually located on lower-dimensional hyperplanes.
The data structure generated in step 1 of our algorithm has the following properties:

- The time to access the cubes for an arbitrary point is $O(log(C_p))$.
- The time to access the relevant portion around a given cube (the connected neighboring cubes) is $O(1)$.

Step 2: The next step of our algorithm is the clustering step. Only the highly populated cubes and cubes which are connected to a highly populated cube are considered in determining clusters. This subset of C_p is noted as $C_r = C_{sp} \cup \{c \in C_p | \exists c_s \in C_{sp} \text{ and } \exists \text{connection}(c_s, c)\}$. Using the data structure constructed in step 1 of the algorithm, we are able to efficiently calculate the local density function $\hat{f}^D(x)$. For $x \in c$ and $c, c_1 \in C_r$, we set $near(x) = \{x_1 \in c_1 | d(mean(c_1), x) \leq k\sigma \text{ and } \exists \text{connection}(c_1, c)\}$ The limit $k\sigma$ is chosen such that only marginal influences are neglegted. A value of $k = 4$ is sufficient for pratical purposes (compare to the 3σ rule for the Gaussian distribution [Sch70]). The resulting local density-function is

$$\hat{f}^D_{Gauss}(x) = \sum_{x_1 \in near(x)} e^{-\frac{d(x,x_1)^2}{2\sigma^2}}.$$

Similarly, the local gradient $\nabla \hat{f}^D_{Gauss}(x)$ can be computed. With these settings and according to lemma 5, the *Error* at a point x can be approximated as

$$Error \leq e^{-\frac{k^2}{2}} \cdot \|D - near(x)\|.$$

where $k = 4$ according to the definition of $near(x)$.
To determine the density-attractors for each point in the cubes of C_r, a hill-climbing procedure based on the local density function $\hat{f}^D$ and its gradient $\nabla \hat{f}^D$ is used. The density-attractor for a point x is computed as

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

The calculation 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 $\hat{f}^D_{Gauss}(x^*) \geq \xi$, the point x is classified and attached to the cluster belonging to x^*. For efficiency reasons, the algorithm stores all points x' with $d(x^i, x') \leq \sigma/2$ 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 appling the hill-climbing procedure to them.

4.3 Time Complexity

First, let us recall the main steps of our algorithm.

```
DENCLUE $(D, \sigma, \xi, \xi_c)$
1. $MBR \leftarrow DetermineMBR(D)$
2. $C_p \leftarrow DetPopulatedCubes(D, MBR, \sigma)$
   $C_{sp} \leftarrow DetHighlyPopulatedCubes(C_p, \xi_c)$
3. $map, C_r \leftarrow ConnectMap(C_p, C_{sp}, \sigma)$
4. $clusters \leftarrow DetDensAttractors(map, C_r, \sigma, \xi)$
```

Step one is a linear scan of the data set and takes $O(\|D\|)$. Step two takes $O(\|D\| + \|C_p\| \cdot log(\|C_p\|))$ time because a tree-based access structure (such as a randomized search tree or B^+-tree) is used for storing the keys of the populated cubes. Note that $\|C_p\| \leq \|D\|$ and therefore in the worst case, we have $O(\|D\| \cdot log(\|D\|)$. The complexity of step 3 depends on σ and ξ, which determine the cardinatilty of C_{sp} and C_p. Formally, the time complexity of step 3 is $O(\|C_{sp}\| \cdot \|C_p\|)$ with $\|C_{sp}\| << \|C_p\| \leq \|D\|$. In step 4, only points which belong to a cube $c \in C_r$ are examined. All other points are treated as outliers and are not used. If the portion of D without outliers is denoted by D_r, then the time complexity of this step is $O(\|D_r\| \cdot log\|C_r\|)$, since the density-attractor of a point can be determined locally. Note that all mentioned time complexities are worst case time complexities.

As we will see in the experimental evaluation (cf. section 5), in the average case the total time complexity of our approach is much better (in the order $O(log(\|D\|))$). If there is a high level of noise in the data, the average time complexity is even better (in the order $O(\|\{x \in c | c \in C_r\}\|)$). Note that for a new σ the data structure does not need to be built up from scratch, especially if $\sigma_{new} = k\sigma_{old}$. Note further that the algorithm is designed to work on very large data sets and therefore, all steps work secondary memory based which is already reflected in the time complexities above.

5 Experimental Evaluation

To show the practical relevance of our new method, we performed an experimental evaluation of the DENCLUE algorithm and compared it to one of the most efficient clustering algorithms – the DBSCAN algorithm [EKSX96]. All experimental results presented in this section are computed on an HP C160 workstation with 512 MBytes of main memory and several GBytes of secondary storage. The DENCLUE algorithm has been implemented in C++ using LEDA [MN89]. The test data used for the experiments are real multimedia data sets from a CAD and a molecular biology application. To show the invariance of our approach with respect to noise, we extended the CAD data set by large amounts of uniformly distributed noise.

5.1 Efficiency

The data used for the efficiency test is polygonal CAD data which are transformed into 11-dimensional feature vectors using a Fourier transformation. We varied the number of data points between 20000 and 100000. We chose the parameters of our algorithm and DBSCAN such that both algorithms produced the same clustering of the data set.

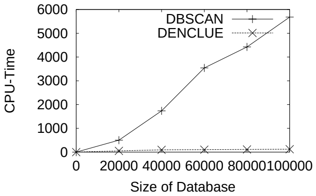
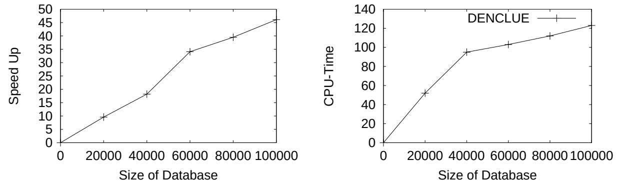


Figure 8: Comparison of DENCLUE and DBSCAN



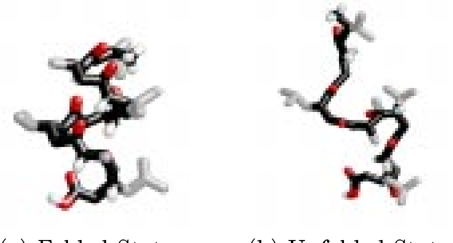
(a) Speed Up (b) DENCLUE

Figure 9: Performance of DENCLUE

The results of our experiments are presented in figure 8. The figure shows that DBSCAN has a slightly superlinear performance and in figure 9a, we show the speed up of our approach (DENCLUE is up to a factor of 45 faster than DBSCAN). Since in figure 8 the performance of our approach is difficult to discern, in figure 9b we show the performance of DENCLUE separately. Figure 9b shows that the performance of DENCLUE is logarithmic in the number of data items. The logarithmic performance of our approach results from the fact that only data points in highly populated regions are actually considered in the clustering step of our algorithm. All other data points are only considered in the construction of the data structure.

5.2 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. (The time for 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 [DJSGM97]. The simulation was done for a period of 50 nanoseconds with two snapshots taken every picosecond, resulting in about 100000 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 molecule. 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 causes the data to contain large amounts of noise (more than 50 percent).



(a) Folded State (b) Unfolded State

Figure 10: Folded Conformation of the Peptide

In figures 10a we show the 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.

6 Conclusions

In this paper, we propose a new approach to clustering in large multimedia databases. We formally introduce the notion of influence and density functions as well as different notions of clusters which are both 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 a map-oriented 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 hierachical clustering. We also formally show the noise-invariance and error-bounds of our approach. An experimental evaluation shows the superior efficiency and effectiveness of DENCLUE. 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 and an extensive comparison to other clustering methods (such as Birch).

References

[DJSGM97] X. Daura, B. Jaun, D. Seebach, W.F. van Gunsteren A.E. Mark:*'Reversible peptide folding in solution by molecular dynamics simulation'*, submitted to Science (1997)

[EKSX96] Ester M., Kriegel H.-P., Sander J., Xu X.:*'A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise'*, Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press, 1996.

[EKSX97] Ester M., Kriegel H.-P., Sander J., Xu X.:*'Density-Connected Sets and their Application for Trend Detection in Spatial Databases'*, Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining, AAAI Press, 1997.

[EKX95] Ester M., Kriegel H.-P., Xu X.:*'Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification'*, Proc. 4th Int. Symp. on Large Spatial Databases, 1995, in: Lecture Notes in Computer Science, Vol. 951, Springer, 1995, pp. 67-82.

[FH75] K. Fukunaga, L.D. Hostler:*'The estimation of the gradient of a density function, with application in pattern recognation'*, IEEE Trans. Info. Thy. 1975, IT-21, 32-40

[Jag91] Jagadish H. V.:*'A Retrieval Technique for Similar Shapes'*, Proc. ACM SIGMOD Int.Conf. on Management of Data, 1991, pp. 208-217.

[Kuk92] Kukich K.:*'Techniques for Automatically Correcting Word in Text'*, ACM Computing Surveys, Vol. 24, No. 4, 1992, pp. 377-440.

[MG95] Methrotra R., Gray J. E.:*'Feature-Index-Based Similar Shape retrieval'*, Proc. of the 3rd Working Conf. on Visual Database Systems, 1995.

[MN89] Mehlhorn K., Naher S.:*'LEDA, a libary of efficient data types and algorithms'*, University of Saarland, FB Informatik, TR A 04/89.

[Nad65] Nadaraya E. A.:*'On nonparametric estimates of density functions and regression curves'*, Theory Prob. Appl. 10, 1965, pp. 186-190.

[NH94] Ng R. T., Han J.:*'Efficient and Effective Clustering Methods for Spatial Data Mining'*, Proc. 20th Int. Conf. on Very Large Data Bases, Morgan Kaufmann, 1994, pp. 144-155.

[Sch96] Schikuta E.:*'Grid clustering: An efficient hierarchical clustering method for very large data sets'*, Proc. 13th Conf. on Pattern Recognition, Vol. 2, IEEE Computer Society Press, pp. 101-105.

[Schn64] Schnell P.:*'A method to find point-groups'*, Biometrika, 6, 1964, pp. 47-48.

[Schu70] Schuster E. F.:*'Note on the uniform convergence of density estimates'*, Ann. Math. Statist 41, 1970, pp. 1347-1348.

[SH94] Shawney H., Hafner J.:*'Efficient Color Histogram Indexing'*, Proc. Int. Conf.on Image Processing, 1994, pp. 66-70.

[WW80] Wallace T., Wintz P.:*'An Efficient Three-Dimensional Aircraft Recognition Algorithm Using Normalized Fourier Descriptors'*, Computer Graphics and Image Processing, Vol. 13, 1980.

[WYM97] Wang W., Yang J., Muntz R.:*'STING: A Statistical Information Grid Approach to Spatial Data Mining'*, Proc. 23rd Int. Conf. on Very Large Data Bases, Morgan Kaufmann, 1997.

[XEKS98] Xu X., Ester M., Kriegel H.-P., Sander J.:*'A Nonparametric Clustering Algorithm for Knowlege Discovery in Large Spatial Databases'*, will appear in Proc. IEEE Int. Conf. on Data Engineering, 1998, IEEE Computer Society Press.

[ZRL96] Zhang T., Ramakrishnan R., Linvy M.:*'BIRCH: An Efficient Data Clustering Method for very Large Databases'*, Proc. ACM SIGMOD Int. Conf. on Management of Data, ACM Press, 1996, pp. 103-114.