

CORE : Nonparametric Clustering of Large Numeric Databases

Andrej Taliun*

Michael H. Böhlen†

Arturas Mazeika‡

Abstract

Current clustering techniques are able to identify arbitrarily shaped clusters in the presence of noise, but depend on carefully chosen model parameters. The choice of model parameters is difficult: it depends on the data and the clustering technique at hand, and finding good model parameters often requires time consuming human interaction. In this paper we propose CORE, a new nonparametric clustering technique that explicitly computes the local maxima of the density and represents them with cores. CORE proposes an adaptive grid and gradients to define and compute the cores of clusters. The incrementally constructed adaptive grid and the gradients make the identification of cores robust, scalable, and independent of small density fluctuations. Our experimental studies show that CORE without any carefully chosen model parameters produces better quality clustering than related techniques and is efficient for large datasets.

1 Introduction.

This paper presents CORE, a novel nonparametric clustering technique that explicitly computes and represents local density maxima. Neither the shape nor the dimensionality of these maxima is restricted. Modeling clusters in terms of explicitly computed local maxima permits clusters with two- or higher-dimensional maxima that are close to each other or overlap. Related clustering techniques do not explicitly represent local maxima. Instead they model clusters by, e.g., sets of unconnected one-dimensional maxima points [12], dense areas [9, 2], and statistical properties such as mean and variance (e.g., centers and medoids, representative points [11], and CF vectors [23]).

The explicit computation of the local maxima of the density faces two challenges. First, the density maxima of numeric datasets are affected by small density fluctuations. Density fluctuations produce false peaks and a robust identification of local maxima is non-trivial. Second, the explicit computation of maxima of the density quickly becomes prohibitively expensive for large multi-dimensional datasets. CORE successfully solves these challenges with the help of a multi-dimensional sequential organization of a minimal number of grid points.

Our method does not assume that clusters follow an a-priori model and does not require model parameters that guide the clustering process. CORE is a nonparametric method since the clustering only depends on the precision of the density estimation of the AD-Tree. The precision of the AD-Tree is controlled by ε , which is not a model parameter and exhibits a monotonic behavior for values larger than the kernel bandwidth: a decrease of ε yields an increase of the estimation precision of the AD-Tree and, hence, a higher quality clustering (at the cost of a higher runtime). Related clustering techniques depend on various model parameters like the number of clusters, number of cluster representatives, shrinking parameter, and size of the neighborhood. Model parameters allow to adjust the method to different datasets and application scenarios. This can be useful for specialized applications, but in most cases choosing model parameters significantly complicates the analysis process and there is no guarantee that appropriate model parameters exist to get a specific clustering behavior. In general, the values of model parameters must be computed based on heuristics, statistical properties of the data, domain knowledge, or results of previous clusterings. Choosing model parameters becomes particularly challenging for large multi-dimensional data. In addition, the model parameters exhibit a non-monotonic behavior: an increase or decrease of the values of the parameters does not guarantee an increase of the quality of the clustering.

Problem Definition: Let $D \subset R^d$ be a d -dimensional dataset and let $\hat{f}(x)$ be a continuous estimate of the density function of D . We propose a nonparametric approximation and explicit representation of the maximal areas of $\hat{f}(x)$ that is robust to small density fluctuations.

CORE clusters the data in three steps: (i) computation of the AD-Tree density summary of the data with the help of the kernel density estimation method, (ii) computation of cores in the AD-Tree and assigning grid points to cores, and (iii) labeling the points from the dataset. Figure 1 illustrates the steps for a sample two dimensional dataset with two clusters. First, we compute the AD-Tree from a random sample of the dataset. The AD-Tree is a hierarchy of d -dimensional grids (cf. Figure 1(a)) that store density values at grid points (cf. Figure 1(g)). During the second step we compute the cores and assign grid points to cores. Figure 1(c) illustrates cores X_1 and X_2 , which are grid points denoted by triangles: X_1 consists of five grid points and X_2

*taliun@inf.unibz.it, Free University of Bozen-Bolzano.

†boehlen@inf.unibz.it, Free University of Bozen-Bolzano.

‡amazeika@mpi-inf.mpg.de, Max-Planck-Institut für Informatik.

consists of one grid point. Cores X_1 and X_2 approximate two peaks in the density function shown in Figure 1(g). The third step labels each tuple with the core of the closest grid point.

The main contributions of our work are the following:

- We develop **CORE**, a novel clustering technique that explicitly computes density maxima and represents them with cores. **CORE** does not require any model parameters and ensures a high quality clustering.
- We introduce the rectangular neighborhood of grid points in the **AD-Tree**, which, together with gradients ensures a robust computation of cores of any dimensionality that is resilient to density fluctuations in the dataset.
- We extensively evaluate **CORE**. Our experimental results show that the clustering quality of **CORE** is superior to state of the art clustering techniques if clusters are close to each other or overlap. **CORE** efficiently scales up with the dimensionality and the size of the sample.

The paper is organized as follows. Section 2 discusses related work. Section 3 describes the **AD-Tree**. Sections 4 and Section 5 define rectangular neighborhoods and cores, respectively. Section 6 explains the clustering of grid and data points. Section 6 analytically investigates **CORE**. Section 8 presents the clustering algorithms and Section 9 evaluates **CORE** experimentally. Section 10 summarizes and concludes the paper.

2 Related Work.

DBScan [9], OPTICS [2], and DENCLUE 2.0 [12] are density-based techniques that compute local maxima implicitly. Clusters identified by DBScan satisfy the following properties: *i*) inside a cluster there are at least $minPts$ points within radius ϵ and *ii*) border points of clusters are density reachable from points located inside the clusters. OPTICS observes that the identification of clusters at all density levels is not possible with $minPts$ and ϵ only. OPTICS orders data points according to their reachability distances and computes a reachability plot that summarizes DBScan for different values of ϵ , but introduces a parameter for reachability plots. Clusters in reachability plots correspond to valleys between steep areas and are easily determined visually. Similarly to DBScan, DENCLUE identifies clusters only at one density level and is sensitive to local fluctuation of the density. It employs two model parameters: σ controls the radius of the neighborhood and ξ defines the lowest density level. DENCLUE assigns data points to peaks of the density by moving them along gradients in automatically computed steps. DENCLUE efficiently captures spherical clusters but does not provide an algorithm for the compu-

tation of arbitrarily shaped clusters: according to the definition in Hinneburg et al. [13] grouping density peaks into arbitrarily shaped clusters is an NP complete problem. All techniques model clusters with dense areas and do not model the local maxima explicitly. A careful selection of the input parameters (if possible at all) is required to achieve a good clustering.

Data Bubbles [5] compresses a large dataset into a small number of data bubbles that improve the performance of OPTICS by an order of magnitude. Similar to clustering features in BIRCH, data bubbles store statistical information of a subset of the data. The detailed statistical information maintained by Data Bubbles allows a good clustering quality for high compression rates. Data Bubbles introduces the size of the control sample as a parameter.

Similar to DBScan and OPTICS, CLIQUE [1], WaveCluster [18] and Shrinking [19] are grid based clustering techniques and model clusters by dense areas (union of dense cells). CLIQUE finds all low-dimensional subspaces that contain clusters. WaveCluster uses wavelets to transform the data into the frequency domain where it computes k -connected dense cells. Shrinking moves data points of dense cells towards centroids and efficiently finds condensed and well-separated clusters. The techniques depend on parameter τ , which defines dense cells and parameter ξ , which defines the width of a cell. If the data within the cells are distributed uniformly then the dense cells yield the same results as our technique. If the cells are not distributed uniformly the ξ parameter must be selected very carefully to avoid that a cluster is split. The negative effects of an incorrectly chosen ξ can be partially eliminated with WaveCluster with two other parameters: the radius ϵ and the number of cells k in the k - ϵ neighborhood of a cell. Shrinking addresses the parameter selection by finding clusters on several differently sized grids and selecting the best quality clusters. The techniques work well for datasets with clusters that can be separated by a single density level. Still, they depend on a good choice of ξ , particularly, if clusters differ in size, shape or distribution. In contrast, **CORE** adaptively allocates the width ξ of cells and models the clusters with explicit local maxima.

BIRCH [23] and CURE [11] are hierarchical clustering techniques that model clusters with the help of average and standard deviation of clusters (clustering features). BIRCH organizes clustering features, each computed for a subset of the data, into the CF tree. The clustering features of each leaf node represent non-overlapping d -dimensional disks that partition the entire dataset. CURE models clusters with c representative points and the α shrinking factor. Representative points naturally extend the modeling of clusters from one (average) point per cluster to multiple representative points per cluster. In each iteration CURE chooses the most scattered points in the cluster as representative points

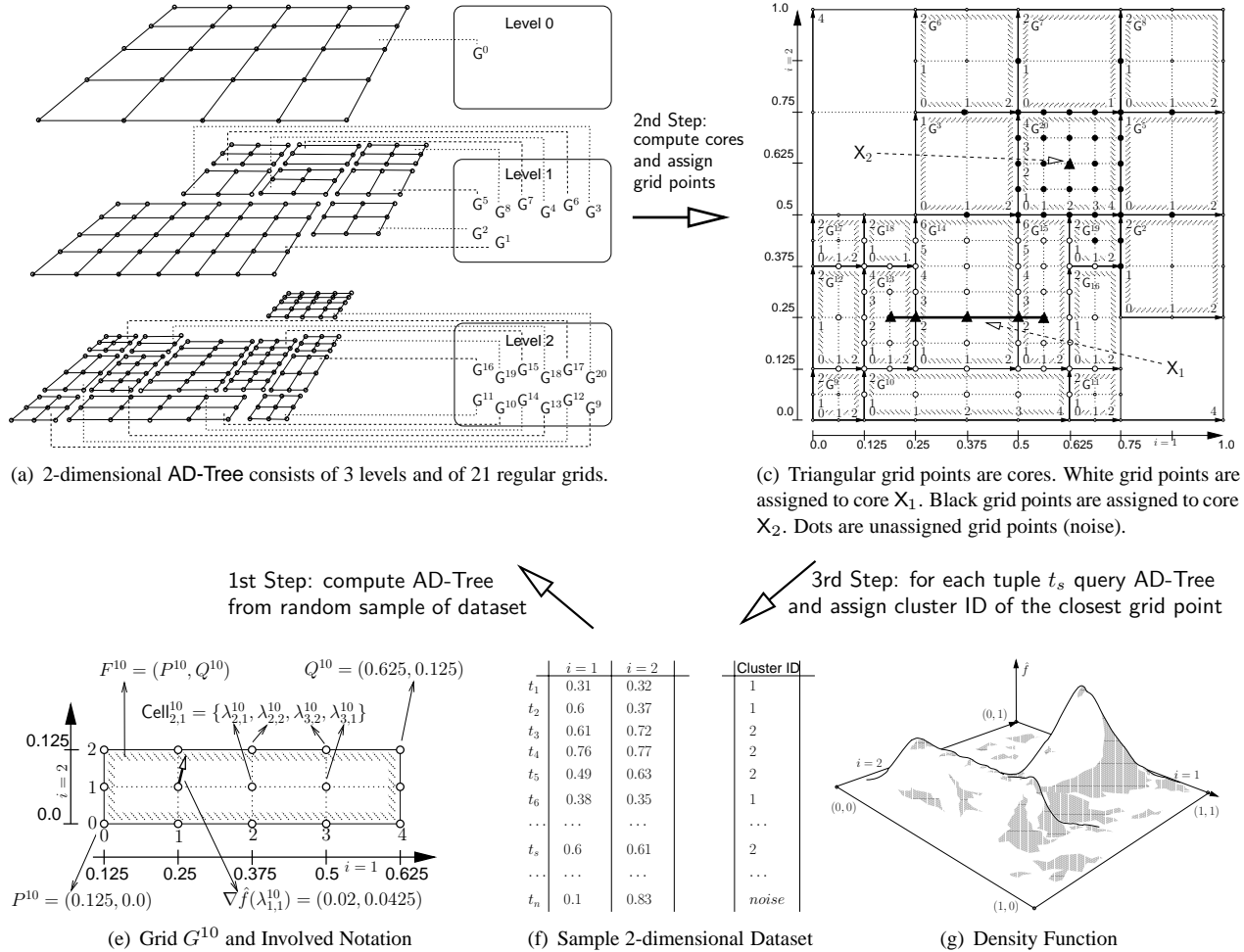


Figure 1: Three steps of CORE: computation of AD-Tree, clustering of grid points and querying.

and shrinks the representative points towards the centroids by factor α to avoid anomalies. For symmetric clusters the average of the cluster coincides with the local maxima. For asymmetric clusters, with multiple maxima points, the quality of the BIRCH and CURE clustering deteriorates. The method could place the average and the representative points even outside the actual clusters. Adjusting the input parameters can help to reduce the negative impact for spherical clusters but remains difficult or impossible for clusters with local maxima forming curves, surfaces, or other non-single point sets.

RIC [3] and ClusteringAggregation [10] are the latest general purpose techniques that aim to improve the quality of existing parametric clustering methods. RIC is an automatic framework to refine clusters by assigning to each of them a probability density function and, then, merging them based on the Volume After Compression (VAC) criterion. ClusteringAggregation proposes five algorithms that aggregate results of several clustering techniques according

to the measure of disagreement. Both techniques improve the clustering quality, but cannot split merged or identify undetected clusters. Moreover, our experiments show that RIC improves the quality only if clusters are clearly separated in space. The complexity of the techniques is quadratic in the number of data points and do not scale to large datasets.

Recently, a number of parametric clustering techniques were proposed to deal with specific data. Chen et al. [8] and Cao et al. [6] are new methods for clustering data streams without assumption about the number of clusters. Chen et al. [8] is density based and uses a decay factor to deal with changing clusters in data streams. Cao et al. [6] discover arbitrarily shaped clusters with the help of a core-micro-cluster synopsis. Both, works depend on four input parameters. Kriegel et al. [14, 15] adopt OPTICS and DBScan for clustering uncertain data by representing uncertainty with a distance density function. Zhao et al. [24] propose a graph-based clustering technique which, based on seven input parameters, finds coherent clusters in three-

dimensional gene expressions. Moise et al. [17] propose a parametric subspace clustering technique that is based on statistically significant regions. Network clustering is investigated in Xu et al. [22]. Evolutionary clustering [7] considers the problem of clustering data over time. Binary clustering is investigated in Li [16]. Wu et al. [21] improve K-Means for sparse data by replacing the objective function with the Shannon entropy. 4C [4] and CURLER [20] incorporate linear correlation information into clustering with the help of a λ -dimensional linear set. None of these methods models clusters with explicit local maximas and the methods are based on parametric models with different model parameters.

3 Preliminaries.

CORE clustering uses the AD-Tree to incrementally compute the density estimate \hat{f} and compress it into a hierarchy of d -dimensional grids. This section defines and illustrates the AD-Tree and its key elements: *frame*, *grid*, and *cell*.

Throughout we use the following notation. We denote a *grid* by G^k , a *grid point* by λ_j^k , and a *cell* by C_j^k . $J = (j_1, \dots, j_d)$ is d -dimensional index to identify cells and grid points within a grid. We denote the gradient at point x by $\nabla \hat{f}(x)$. We write $[x]_i$ to refer to the coordinate of a d -dimensional point x in the i -th dimension. Grid, frame, cell and grid point belong to a node in the AD-Tree. We use superscript k for the index of the node. The *origin* is the d -dimensional point with coordinate $(0, \dots, 0)$.

DEFINITION 3.1. (FRAME) A frame $F^k = (P^k, Q^k)$ is a pair of d -dimensional points. P^k and Q^k define a d -dimensional hyper-rectangle with edges parallel to the coordinate axes. $P^k = (P_1^k, \dots, P_d^k)$ is closest to the origin and $Q^k = (Q_1^k, \dots, Q_d^k)$ is farthest from the origin.

DEFINITION 3.2. (GRANULARITY) A granularity $S^k = (S_1^k, \dots, S_d^k)$ is a d -dimensional vector of positive integers.

DEFINITION 3.3. (GRID) Let $F^k = (P^k, Q^k)$ be a frame and S^k be a granularity. A grid $G^k(F^k, S^k) = \{\lambda_{J=(j_1, \dots, j_d)}^k : j_i = 0, \dots, S_i^k; i = 1, \dots, d\}$ is a set of grid points where $[\lambda_j^k]_i = P_i^k + j_i(Q_i^k - P_i^k)/S_i^k$.

EXAMPLE 3.1. [Grid] Consider grid G^{10} in Figures 1(e) and 1(a). $F^{10} = ((0.125, 0.0), (0.625, 0.125))$ is the frame and $S^{10} = (4, 2)$ is the granularity of G^{10} . There are $(S_1^{10} + 1) \cdot (S_2^{10} + 1) = 15$ grid points in G^{10} . Along each dimension grid points are equally spaced, and in our example the position of grid point $\lambda_{3,1}^{10}$ in dimension $i = 1$ is $[\lambda_{3,1}^{10}]_1 = 0.125 + 3 \cdot (0.625 - 0.125)/4 = 0.5$ and in dimension $i = 2$ is $[\lambda_{3,1}^{10}]_2 = 0.0 + 1 \cdot (0.125 - 0.0)/2 = 0.0625$.

DEFINITION 3.4. (CELL) Let $G^k(F^k, S^k)$ be a grid and $J = (j_1, \dots, j_d) < (S_1^k, \dots, S_d^k)$ be a d -dimensional index.

A cell C_j^k consists of all vertices on a minimal frame and is identified by the index of the grid point that is closest to the origin: $C_j^k = \{\lambda_{l_1, \dots, l_d}^k : l_i \in \{j_i, j_i + 1\}, i = 1, \dots, d\}$.

EXAMPLE 3.2. [Cell] Consider grid G^{10} in Figures 1(e) and 1(c). Then, $C_{2,1}^{10} = \{\lambda_{2,1}^{10}, \lambda_{2,2}^{10}, \lambda_{3,2}^{10}, \lambda_{3,1}^{10}\}$.

The AD-Tree for the running example is illustrated in Figure 1(a). It consists of twenty one grids organized into a three-level hierarchy, such that grids of the same level do not overlap and a higher level grid splits cells of one lower level grid. The computation of the AD-Tree is an iterative partitioning of the initial coarse grid. In each iteration the construction procedure of the AD-Tree automatically detects cells within which the density exhibits a non-linear behavior and splits these cells along dimensions of non-linearity. After the splitting, new cells of the same shape are grouped into grids (cf. grids G^1, \dots, G^8 and G^9, \dots, G^{20} on levels 1 and 2). The splitting and grouping is repeated until the density is linear within all cells.

The grouping of cells into grids and a sequential encoding of nodes reduces the number of pointers to two per node. This is crucial for the performance and scalability. The AD-Tree is applicable for multi-dimensional datasets because it does not store coordinates but only density values at grid points and because it uses a sequential organization of nodes to minimize the number of pointers.

DEFINITION 3.5. (AD-TREE) The AD-Tree is a sequence of nodes (N^k) . Each node N^k stores the density values at the grid points of a rectangular grid $G^k = G^k(F^k, S^k)$ and is defined as a five-tuple $N^k = (F^k, S^k, V^k, L^k, H^k)$ where:

- + F^k is the frame of grid G^k . F^k defines the region of the parent node that is partitioned by N^k .
- + S^k is the granularity of G^k .
- + V^k is a d -dimensional array of size $|G^k|$, where $\forall v_j^k \in V^k, \lambda_j^k \in G^k : v_j^k = \hat{f}(\lambda_j^k)$.
- + L^k and H^k point to the first and last child of N^k . L^k points to the child with the lowest and H^k to the child with the highest index in the sequence of child nodes.

DEFINITION 3.6. (GRADIENT) The gradient $\nabla \hat{f}(x)$ at point x is a d -dimensional vector of derivatives of \hat{f} at x :

$$\nabla \hat{f}(x) = \left(\frac{\partial \hat{f}}{\partial x^1}(x), \dots, \frac{\partial \hat{f}}{\partial x^d}(x) \right)$$

EXAMPLE 3.3. [Gradient] Consider grid G^{10} in Figures 1(e). The arrow at grid point $\lambda_{1,1}^{10} = (0.25, 0.0625)$ is gradient $\nabla \hat{f}(\lambda_{1,1}^{10}) = (0.02, 0.0425)$.

4 Rectangular Neighborhood.

A rectangular neighborhood localizes stationary points of \hat{f} in the AD-Tree and is the key concept for the exact computation of core points. A core point is a stationary point that correspond to local maxima. Rectangular neighborhood may span multiple grids and are generalizations of frame and cell, respectively.

Intuitively, a rectangular neighborhood consists of grid points that lie on the sides of an *embedding frame*. The embedding frame $E(\lambda_j^k)$ encloses a grid point λ_j^k into a minimal frame that satisfies the following requirements: *i*) only one grid point λ_j^k lies inside $F(\lambda_j^k)$ and *ii*) each edge of $F(\lambda_j^k)$ contains at least two points of cells that include λ_j^k .

4.1 Embedding Frame. Figure 2 illustrates the embedding frame $E(\lambda_j^k) = ((0.5625, 0.0625), (0.6875, 0.1875))$ of grid point $\lambda_{0,0}^{16}$. There is only grid point $\lambda_{0,0}^{16}$ inside $E(\lambda_j^k)$ and each edge of $E(\lambda_j^k)$ includes at least two grid points of a cell that include $\lambda_{0,0}^{16}$.

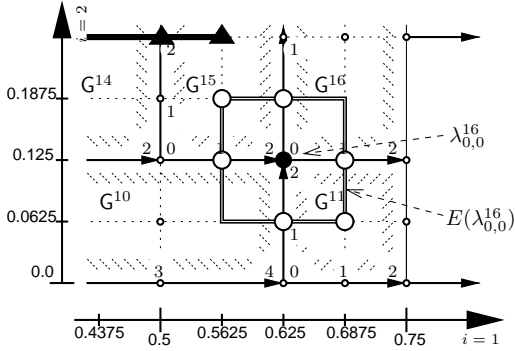


Figure 2: Incomplete Rectangular Neighborhood

DEFINITION 4.1. (EMBEDDING FRAME) Let $\Phi(\lambda_j^k)$ be all grid points from cells that contain a grid point with the coordinates of λ_j^k :

$$\Phi(\lambda_j^k) = \bigcup_{C_M^l: \lambda_j^k \in C_M^l} C_M^l$$

Let $\Phi_i^-(\lambda_j^k)$ be the set of *i*th coordinates smaller than the *i*th coordinate of λ_j^k :

$$(4.1) \quad \Phi_i^-(\lambda_j^k) = \{[\lambda]_i : [\lambda]_i < [\lambda_j^k]_i, \lambda \in \Phi(\lambda_j^k)\}$$

$$(4.2) \quad \Phi_i^+(\lambda_j^k) = \{[\lambda]_i : [\lambda]_i > [\lambda_j^k]_i, \lambda \in \Phi(\lambda_j^k)\}$$

The embedding frame $E(\lambda_j^k) = (P(\lambda_j^k), Q(\lambda_j^k))$ for λ_j^k is

defined for each coordinate as follows:

$$(4.3) \quad [P(\lambda_j^k)]_i = \begin{cases} \max\{\Phi_i^-(\lambda_j^k)\} & \text{if } \Phi_i^-(\lambda_j^k) \neq \emptyset \\ [\lambda_j^k]_i & \text{otherwise} \end{cases}$$

$$(4.4) \quad [Q(\lambda_j^k)]_i = \begin{cases} \min\{\Phi_i^+(\lambda_j^k)\} & \text{if } \Phi_i^+(\lambda_j^k) \neq \emptyset \\ [\lambda_j^k]_i & \text{otherwise.} \end{cases}$$

EXAMPLE 4.1. [Embedding Frame] Consider $\lambda_{0,0}^{16} = (0.625, 0.125)$ in Figure 2. The set $\Phi(\lambda_{0,0}^{16})$ of grid points of cells that contain grid point $\lambda_{0,0}^{16}$ is

$$\begin{aligned} \Phi(\lambda_{0,0}^{16}) = & \{(0.5, 0.0625), (0.625, 0.0625), (0.5, 0.125), \\ & (0.5625, 0.125), (0.5625, 0.1875), (0.625, 0.1875), \\ & (0.6875, 0.0625), (0.6875, 0.125), (0.625, 0.25), \\ & (0.6875, 0.25)\} \end{aligned}$$

Set $\Phi_1^-(\lambda_{0,0}^{16})$ contains the first coordinates of $\Phi(\lambda_{0,0}^{16})$ that are smaller than the first coordinate of $\lambda_{0,0}^{16}$. Similarly, $\Phi_2^-(\lambda_{0,0}^{16})$ contains the second coordinates of $\Phi(\lambda_{0,0}^{16})$ that are smaller than the second coordinate of $\lambda_{0,0}^{16}$:

$$\Phi_1^-(\lambda_{0,0}^{16}) = \{0.5, 0.5625\}$$

$$\Phi_2^-(\lambda_{0,0}^{16}) = \{0.0625\}$$

From Equation 4.3:

$$\begin{aligned} P(\lambda_{0,0}^{16}) &= (\max\{\Phi_1^-(\lambda_{0,0}^{16})\}, \max\{\Phi_2^-(\lambda_{0,0}^{16})\}) \\ &= (0.5625, 0.0625). \end{aligned}$$

The computation of $Q(\lambda_{0,0}^{16})$ is equivalent. Thus, the embedding frame of $\lambda_{0,0}^{16}$ is $E(\lambda_{0,0}^{16}) = (P(\lambda_{0,0}^{16}), Q(\lambda_{0,0}^{16})) = ((0.5625, 0.0625), (0.6875, 0.1875))$.

4.2 Rectangular Neighborhood. The white circles in Figures 2 and 3 illustrate the rectangular neighborhoods for grid points $\lambda_{0,0}^{16}$ and $\lambda_{1,1}^{11}$. Intuitively, the rectangular neighborhood of a grid point λ_j^k consists of grid points that are on the embedding frame of λ_j^k and belong to the set $\Phi(\lambda_j^k)$.

DEFINITION 4.2. (RECTANGULAR NEIGHBORHOOD) Let $E(\lambda_j^k) = (P(\lambda_j^k), Q(\lambda_j^k))$ be the embedding frame of grid point λ_j^k . The rectangular neighborhood of grid point λ_j^k consists of all grid points in $\Phi(\lambda_j^k)$ that are on the embedding frame:

$$\begin{aligned} R(\lambda_j^k) = & \{ \lambda \in \Phi(\lambda_j^k) : \\ & [P(\lambda_j^k)]_i \leq [\lambda]_i \leq [Q(\lambda_j^k)]_i, i = 1, \dots, d \} \setminus \lambda_j^k \end{aligned}$$

where $\Phi(\lambda_j^k)$ is defined according to Definition 4.1.

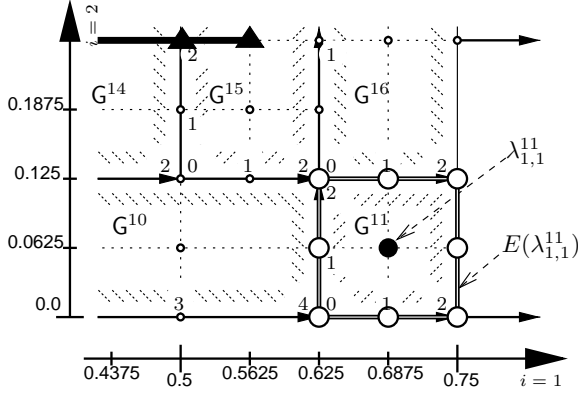


Figure 3: Complete Rectangular Neighborhood

EXAMPLE 4.2. [Rectangular Neighborhood]. Rectangular neighborhood of $\lambda_{0,0}^{16}$ is the following set

$$\begin{aligned} R(\lambda_{0,0}^{16}) = \{ & (0.5625, 0.1875), (0.625, 0.1875), \\ & (0.5625, 0.125), (0.625, 0.125), (0.6875, 0.125), \\ & (0.625, 0.0625), (0.6875, 0.0625) \} \end{aligned}$$

The rectangular neighborhood $R(\lambda_j^k)$ is maximal if all grid points of $\Phi(\lambda_j^k) \setminus \{\lambda_j^k\}$ are on the embedding frame. We call such a rectangular neighborhood complete. In Figure 3, all grid points of cells that contain $\lambda_{1,1}^{11}$, except $\lambda_{1,1}^{11}$ are on the embedding frame and therefore $R(\lambda_{1,1}^{11})$ is complete.

DEFINITION 4.3. (COMPLETE AND INCOMPLETE RECTANGULAR NEIGHBORHOOD) Rectangular neighborhood $R(\lambda_j^k)$ is complete iff $|R(\lambda_j^k)| = 3^d - 1$ and incomplete otherwise.

In Figure 4 black squares are grid points with a complete rectangular neighborhood and white polygons are grid points with an incomplete rectangular neighborhood. White polygons occurs only in places where f is monotonically increasing or decreasing, i.e., in places where f has no stationary points.

THEOREM 4.1. A grid point with an incomplete rectangular neighborhood is not a stationary point of f .

5 Cores of Clusters.

A core approximates a local maxima of the density function of the data from the AD-Tree. A rectangular neighborhood localizes stationary points, i.e., points where the derivative of the density function is zero, including local minima, local maxima, saddle and inflection points. The next definition specifies two conditions that are satisfied for local maxima and are not satisfied for all other types of stationary points.

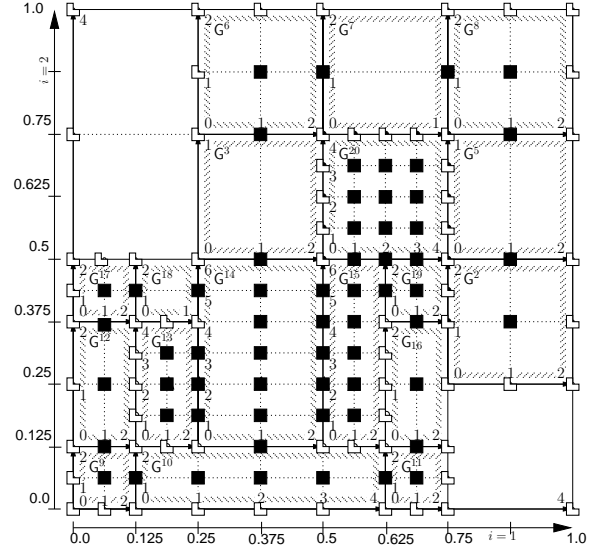


Figure 4: Grid points with complete and incomplete R.

	1st Condition	2nd Condition
Core Point	Satisfied	Satisfied
Neighborhood Point	Satisfied	Unsatisfied
Saddle Point	Unsatisfied	Satisfied
Inflection Point	Unsatisfied	Unsatisfied
Minima	Unsatisfied	Unsatisfied
Not Stationary Point	Unsatisfied	Unsatisfied

Table 1: Definition 5.1 for Different Points.

DEFINITION 5.1. (CORE OF A CLUSTER) . Let $X = \{\lambda_{J_1}^{k_1}, \dots, \lambda_{J_t}^{k_t}, \dots\}$ be a set of grid points in a d -dimensional AD-Tree, such that for each $\lambda_{J_t}^{k_t} \in X$ its rectangular neighborhood $R(\lambda_{J_t}^{k_t})$ is complete. X is a core of a cluster iff it is a maximal and connected set with the following conditions satisfied:

1. At least $2 \cdot 3^{d-1}$ gradients of points from the rectangular neighborhood point towards the core point:

$$(5.5) \quad \left| \left\{ \lambda \in R(\lambda_{J_t}^{k_t}) : \angle(\lambda_{J_t}^{k_t} - \lambda, \nabla f(\lambda)) < 90^\circ \right\} \right| \geq 2 \cdot 3^{d-1}$$

2. There are two gradients pointing to $\lambda_{J_t}^{k_t}$ from opposite directions, i.e., there exist $\lambda, \mu \in R(\lambda_{J_t}^{k_t})$ such that the following conditions are satisfied:

- (a) For each coordinate $i : 1 \leq i \leq d$ either Equation 5.6 or Equation 5.7 is satisfied:

$$(5.6) \quad [\lambda]_i = [\mu]_i = [\lambda_{J_t}^{k_t}]_i$$

$$(5.7) \quad [\lambda]_i = [P(\lambda_{J_t}^{k_t})]_i \text{ and } [\mu]_i = [Q(\lambda_{J_t}^{k_t})]_i$$

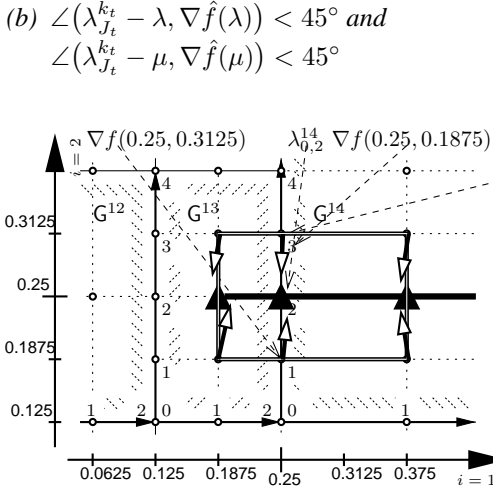


Figure 5: All Conditions are Satisfied for Core Point.

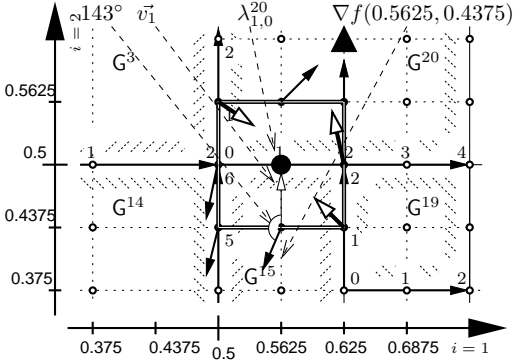


Figure 6: 1st Condition is Unsatisfied for Saddle Point.

Figures 5, 6 and 7 illustrate how Definition 5.1 distinguishes core points from other types of stationary points. Grid point $\lambda_{0,2}^{14}$ in Figure 5 is a core point since it satisfies both conditions: 6 gradients (cf. Equation 5.5, $2 \cdot 3^{2-1} = 6$) are pointing towards the core point and there are 2 gradients at $\lambda = \lambda_{0,3}^{14}$ and $\lambda = \lambda_{0,1}^{14}$ pointing to the core point from opposite directions. Points $\lambda_{1,0}^{20}$ in Figure 6 and $\lambda_{3,3}^{20}$ in Figure 7 do not satisfy both conditions in Definition 5.1 and are not core points. $\lambda_{1,0}^{20}$ does not satisfy the first condition, since there are only 3 gradients pointing towards the grid point. In fact, $\lambda_{1,0}^{20}$ is a saddle point (maxima point towards one diagonal and minima point towards the other diagonal). $\lambda_{3,3}^{20}$ does not satisfy the second condition, since there are not 2 gradients that point to $\lambda_{3,3}^{20}$ from opposite directions. This situation happens in the neighborhood of core points.

The rationale for at least $2 \cdot 3^{d-1}$ gradients pointing towards core points is the following. If λ_j^k is a core point then all gradients in the rectangular neighborhood of λ_j^k should point towards λ_j^k , except the gradients at grid points of the rectangular neighborhood which also are core

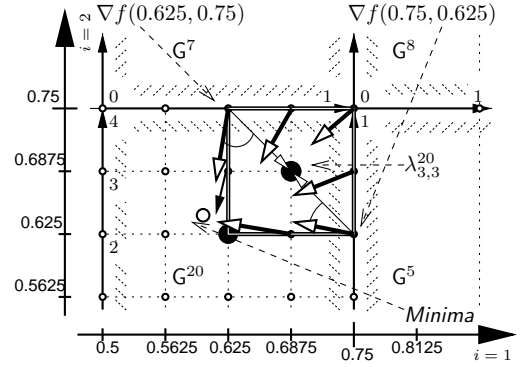


Figure 7: 2nd Condition is Unsatisfied for Neighbor.

points. In general, in d -dimensional datasets with up to $(d-1)$ -dimensional cores (0 - d are points, 1 - d are curves, ...), there are 3^d number of gradients in the complete rectangular neighborhood, and of these 3^{d-1} do not necessarily point towards the core point. Therefore, λ_j^k is a core point iff there are $3^d - 3^{d-1} = 2 \cdot 3^{d-1}$ gradients in its rectangular neighborhood that point towards λ_j^k .

EXAMPLE 5.1. [Core, first condition, Equation 5.5]. We show that gradient $\nabla \hat{f}(0.5625, 0.4375)$ in Figure 6 is not pointing towards grid point $\lambda_{1,0}^{20} = (0.5625, 0.5)$.

We compute the angle between the gradient $\nabla \hat{f}(0.5625, 0.4375)$ and direction connecting points $(0.5625, 0.4375)$ and $(0.5625, 0.5)$. The angle between any two vectors \vec{v}_1 and \vec{v}_2 is computed as follows:

$$\angle(\vec{v}_1, \vec{v}_2) = \arccos\left(\frac{\vec{v}_1 \cdot \vec{v}_2}{\|\vec{v}_1\| \cdot \|\vec{v}_2\|}\right)$$

In our case

$$\begin{aligned} \vec{v}_1 &= \nabla \hat{f}(0.5625, 0.4375) = (-0.03, -0.004) \\ \vec{v}_2 &= (0.5625, 0.5) - (0.5625, 0.4375) = (0.0, 0.0625) \end{aligned}$$

Therefore,

$$\arccos\left(\frac{(0.0, 0.0625) \cdot (-0.03, -0.004)}{\|(0.0, 0.0625)\| \cdot \|(-0.03, -0.004)\|}\right) \approx 143^\circ.$$

$143^\circ > 90^\circ$ and, hence gradient $\nabla \hat{f}(0.5625, 0.4375)$ is not pointing towards $\lambda_{1,0}^{20}$.

Close to a local maxima many gradients are pointing towards one point and, hence, the first condition of Definition 5.1 is satisfied for several grid points (cf. Figure 7). The second condition ensures that only the grid point that is the closest to the local maxima is declared as a core point. Grid points $\lambda_{3,3}^{20}$ and $\lambda_{2,2}^{20}$ are approximating the same local maxima and both satisfy the first condition. The second condition

of Definition 5.1 selects $\lambda_{3,3}^{20}$ as core point, since it is the best approximation of the local maxima.

Technically, the second condition consists of two sub-conditions (a) and (b): (a) defines opposite points of $R(\lambda_j^k)$ and (b) defines gradients that are pointing to the core point from opposite points of $R(\lambda_j^k)$.

EXAMPLE 5.2. [Core, second condition]. Grid points $\lambda = (0.625, 0.75)$ and $\mu = (0.75, 0.625)$ from $R(\lambda_{3,3}^{20})$ in Figure 7 are opposite. Indeed, the embedding frame of $\lambda_{3,3}^{20}$ is $E(\lambda_{3,3}^{20}) = ((0.625, 0.625), (0.75, 0.75))$. λ and μ are opposite since for each coordinate i they satisfy Equation 5.7:

$$\begin{aligned} [\lambda]_1 &= 0.625 = [P(\lambda_{3,3}^{20})]_1 \\ [\mu]_1 &= 0.755 = [Q(\lambda_{3,3}^{20})]_1 \\ [\lambda]_2 &= 0.750 = [P(\lambda_{3,3}^{20})]_2 \\ [\mu]_2 &= 0.625 = [Q(\lambda_{3,3}^{20})]_2 \end{aligned}$$

Grid points $\lambda = (0.6875, 0.7500)$ and $\mu = (0.7500, 0.6250)$ of the same $R(\lambda_{3,3}^{20})$ are not opposite since Equation 5.6 and 5.7 are not satisfied for coordinate $i = 1$:

$$[\lambda]_1 \neq [\mu]_1; [\lambda]_1 \neq [P(\lambda_{3,3}^{20})]_1; [\lambda]_1 \neq [Q(\lambda_{3,3}^{20})]_1.$$

The angles between the gradients at the opposite points and the direction to the core point should be less than 45° . Any two opposite grid points $\lambda, \mu \in R(\lambda_{3,3}^{20})$ do not satisfy this condition (cf. Figure 7) and, hence $\lambda_{3,3}^{20}$ is not a core point. Core point $\lambda_{0,2}^{14}$ (cf. Figure 5) satisfies this condition with $\lambda = (0.25, 0.1875)$ and $\mu = (0.25, 0.3125)$.

6 Labeling Grid and Data Points.

We label each data point t_s with the cluster number of its closest grid point. The cluster number of grid point λ_j^k is the index of a core to which the gradient path of λ_j^k leads. The gradient path of λ_j^k is a sequence of grid points that starts at λ_j^k and the gradient at each grid point except the last one is pointing towards the next grid point of the path.

DEFINITION 6.1. (GRADIENT PATH) . The gradient path of λ_j^k is a sequence of grid points $P(\lambda_j^k) = (\lambda_1, \dots, \lambda_s)$, such that the first element is $\lambda_1 = \lambda_j^k$, last element is a core point and each λ_t is a neighbor of λ_{t-1} that satisfies:

$$(6.8) \quad \angle(\nabla \hat{f}(\lambda_{t-1}), \lambda_t - \lambda_{t-1}) = \min_{x_u \in R(\lambda_{t-1})} \angle(\nabla \hat{f}(\lambda_{t-1}), x_u - \lambda_{t-1})$$

7 Analytical Investigation.

This section analytically investigates Definition 5.1.

THEOREM 7.1. Let L be a d -dimensional line segment satisfying the following two properties: (i) L is a local maxima

set of the density function of the data and (ii) L is parallel to one of the axis of the coordinate system. Let $\lambda \in L$, $(P(\lambda), Q(\lambda))$ be the embedding frame such that $P(\lambda)$ and $Q(\lambda)$ are close enough to λ . Then there are at least $2 \cdot 3^{d-1}$ gradients at points from $\Psi(\lambda)$ pointing towards λ .

Proof. λ is the local maxima point of the density function. Therefore, the function is increasing towards λ close around λ except line L . Let us assume that frame $(P(\lambda), Q(\lambda))$ are selected close enough so the density function is increasing towards the local maxima in the frame. Then all gradients from $\Psi(\lambda) \setminus L$ are pointing towards λ . $|\Psi(\lambda) \setminus L| \geq 2 \cdot 3^{d-1}$ finishes the proof.

COROLLARY 7.1. Let L be a hyper rectangle satisfying the following three properties: (i) L is a local maxima set of the density function of the data (ii) the edges of L are parallel to one of the axis of the coordinate system, and (iii) at least one edge of L is zero. Let $\lambda \in L$, $(P(\lambda), Q(\lambda))$ be the embedding frame such that $P(\lambda)$ and $Q(\lambda)$ are close enough to λ . Then there are at least $2 \cdot 3^{d-1}$ gradients at points from $\Psi(\lambda)$ pointing towards λ .

The condition that the edges of the core must be parallel to the axis of the coordinate system in Corollary 7.1 can be dropped without loss of generality. In this case frame $(P(\lambda), Q(\lambda))$ must be selected closer to λ so the density function is increasing towards the local maxima in the frame.

THEOREM 7.2. Assume an AD-Tree such that the density function in each unsplit cell can be approximated linearly with ε precision. Let grid point λ be a local maxima of the density function with a full rectangular neighborhood $R(\lambda)$. Then there are at least $2 \cdot 3^{d-1}$ gradients at grid points from $R(\lambda)$ pointing towards λ .

Proof. The proof follows from Corollary 7.1 and the approximation properties of the AD-Tree .

8 Algorithms and Complexity.

Algorithms 1 and 2 perform CORE clustering on a sample of input dataset D and have a linear time complexity wrt to the number of grid points and the size of the sample. computeCORE first computes the AD-Tree from a sample of the dataset. The computation of the AD-Tree requires one scan through the sample per each level. Line 3 iterates through all grid points and computes for each grid point the last grid point of the gradient path. Lines 4-6 query the AD-Tree for each data point and find its closest grid point. A data point t_s is labeled with the core to which the gradient path starting at the grid point closest to t_s leads.

lastOfP recursively computes and returns the last grid point of the gradient path that starts at a given grid point λ_j^k . First, at line 3, lastOfP computes the rectangular

neighborhood of λ_j^k . Next, if gradients of $R(\lambda_j^k)$ satisfy Definition 5.1, then λ_j^k is a core point and, hence, the last element of the gradient path. Otherwise, lines 8-11 compute the next grid point of the path and repeat lastOfP. The first line of lastOfP ensures that each grid point is considered only once. Lines 4-7 add a new core point to a core and refine all computed cores so that they are maximal and connected.

Algorithm 1: computeCORE(D)

```

1 compute AD-Tree from sample of  $D$ ;
2 for  $\lambda_j^k$  of AD-Tree do  $\lambda_j^k$ .processed = false;
3 CORES =  $\emptyset$ ;
4 for  $\lambda_j^k$  of AD-Tree do  $\lambda_j^k$ .last = lastOfP( $\lambda_j^k$ );
5 for  $t_s \in D$  do
6    $t_s$ .ClusterId =  $i : \lambda_j^k$ .last  $\in X_i$ , where  $\lambda_j^k$  is
   closest to  $t_s$ .
7 end

```

Algorithm 2: lastOfP(λ_j^k)

```

1 if  $\lambda_j^k$ .processed then return  $\lambda_j^k$ .last;
2  $\lambda_j^k$ .processed = true;
3 compute  $R(\lambda_j^k)$  by querying AD-Tree;
4 if  $R(\lambda_j^k)$  satisfies Definition 5.1 then
5   CORES = CORES  $\cup \{\lambda_j^k\}$ ;
6   merge all  $X_1, X_2 \in$  CORES that satisfy
    $X_1 \cap (R(\lambda_j^k) \cup \lambda_j^k) \neq \emptyset$  and
    $X_2 \cap (R(\lambda_j^k) \cup \lambda_j^k) \neq \emptyset$ ;
7    $\lambda_j^k$ .last =  $\lambda_j^k$ ;
8 else
9   find  $\lambda \in R(\lambda_j^k)$  satisfying Equation 6.8;
10   $\lambda_j^k$ .last = lastOfP( $\lambda$ );
11 end
12 return  $\lambda_j^k$ .last;

```

The computation of the rectangular neighborhood of λ_j^k is done with a single query on the AD-Tree. We start with the root grid and initialize $\Phi(\lambda_j^k)$ (cf. Definition 4.1) with all cells of the root that contain λ_j^k . Next, we refine $\Phi(\lambda_j^k)$ by descending the AD-Tree. We scan child grids and check if a child partitions a cell in $\Phi(\lambda_j^k)$. If so, then we substitute partitioned cell with cells from the child grid that contain λ_j^k . When the leaf level is reached, we compute the embedding frame and the rectangular neighborhood from $\Phi(\lambda_j^k)$ following Definition 4.1 and Definition 4.2. In the worst case, querying the AD-Tree checks all grids, in the best case one grid is checked. Typically, the number of processed grids is close to the number of levels.

9 Experiments.

We compare CORE with CURE, K-Means, DBScan, OPTICS, DataBubbles, and Robust Information-Theoretic

Clustering (RIC) which are efficient for a broad range of datasets. We run these techniques for many different input parameters and present only their best results. We implement THE automatic computation of clusters in OPTICS based on the hierarchy of steep down and steep up regions of reachability plots [2]. Independent of the dataset we build the AD-Tree with a fixed $\varepsilon = 0.02$.

We measure the clustering quality in terms of *precision*, *recall* and *F-Score*. Let $W \subseteq D$ be a cluster in the dataset and let $V \subseteq D$ be a computed cluster. Then precision, recall and F-Score of V wrt to W are $p(V, W) = |V \cap W|/|V|$, $r(V, W) = |V \cap W|/|W|$, and $F(V, W) = 2 \cdot p(V, W) \cdot r(V, W)/(p(V, W) + r(V, W))$. Let \mathcal{W} be the set of all clusters in a dataset and let \mathcal{V} be the set of computed clusters. We compute the matching from \mathcal{W} to \mathcal{V} that maximizes the average of the F-scores of each pair ($V \in \mathcal{V}, W \in \mathcal{W}$) in the matching.

9.1 Quality of Clustering. Table 2 compares CORE for four synthetic datasets. The datasets differ in the number, position and type of clusters. Up to 5% of the data points in each dataset are noise. The F-score shows that CORE performs significantly better than other techniques. CORE clearly outperforms the other methods for databases with complex shaped clusters (cf. TwoSpirals in first row) and hierarchical clusters (cf. HierarchicalClusters in second row). For overlapping clusters (cf. OverlappingSpheres in third row) CORE has a very high average F-score (95.7), but also the other solutions perform fairly well (76.5-83.2). For each dataset we used all its data points and, hence, quality of DataBubbles correspond to quality of OPTICS in Table 2.

For the TwoSpirals dataset, CORE correctly identifies both spirals and has the highest average F-score. OPTICS, CURE and DBScan accurately identify the inner spiral, even outperforming CORE a bit (cf. OPTICS with F-score of 99.6 vs. CORE with 97.1). However, all competitors are substantially worse (by 36.6-86.1) for the outer spiral. CURE exhibits a low F-score value, since it shifts the control points toward the inner spiral and the places where the spirals are close to each other. OPTICS and DBScan fail because valleys in the reachability plots are not separated by a steep area. The best clustering for OPTICS and CURE is achieved if the number of clusters is set to five. This leads to one cluster being assigned to the inner spiral and the other four clusters to fragments of the outer spiral. Higher k values splits the inner spiral, while lower k values merge the inner with the outer spiral. The best clustering of DBScan merges the inner spiral with the outer spiral. Adjusting the other parameters does not improve the quality: it either eliminates the outer spiral as noise or merges it with the inner spiral. The best output of K-Means is two clusters having approximately the same number of data points and dividing each spiral in two equal parts. For more than 15 clusters K-

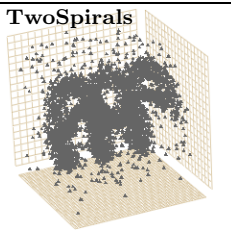
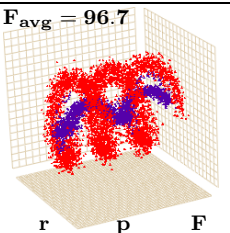
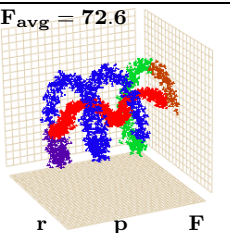
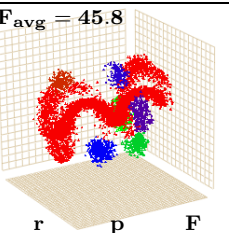
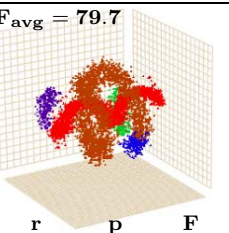

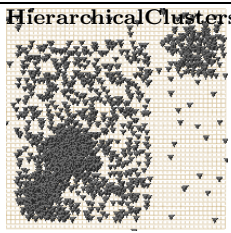
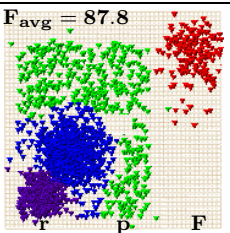
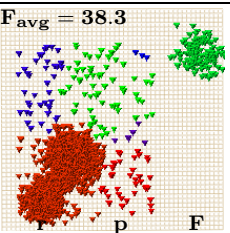
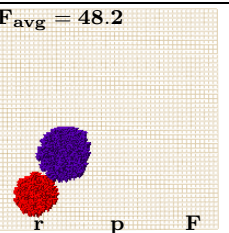
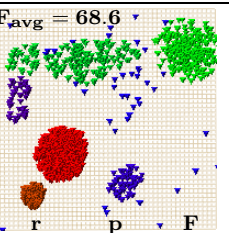

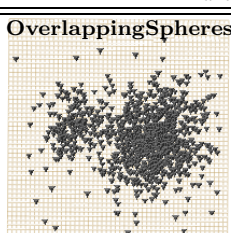
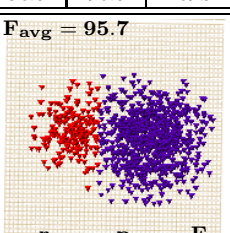
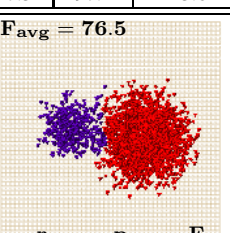
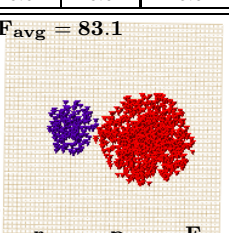
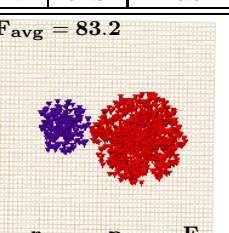

Dataset	CORE			CURE			DBScan			OPTICS and DataBubbles		
TwoSpirals	$F_{avg} = 96.7$			$F_{avg} = 72.6$			$F_{avg} = 45.8$			$F_{avg} = 79.7$		
												
Inner Spiral	100.0	94.27	97.1	91.8	99.9	95.7	100.0	68.5	81.3	99.7	99.5	99.6
Outer Spiral	92.88	100.0	96.3	32.9	100.0	49.5	5.4	100.0	10.2			
HierarchicalClusters	$F_{avg} = 87.8$			$F_{avg} = 38.3$			$F_{avg} = 48.2$			$F_{avg} = 68.6$		
												
Embedded Sphere 1	99.1	86.9	92.5	0.0	0.0	0.0	95.3	100.0	97.6	95.4	88.3	91.7
Embedded Sphere 2	98.8	80.8	88.9	95.0	53.7	68.6	90.6	100.0	95.0	53.4	100.0	69.6
Outside Sphere	99.9	99.7	99.8	55.0	100.0	70.9	0.0	0.0	0.0	86.9	100.0	93.0
Plane	56.8	90.8	69.9	7.3	97.1	13.6	0.0	0.0	0.0	11.2	94.5	20.0
OverlappingSpheres	$F_{avg} = 95.7$			$F_{avg} = 76.5$			$F_{avg} = 83.1$			$F_{avg} = 83.2$		
												
Right Sphere	97.9	98.6	98.2	72.3	99.6	83.8	80.3	98.7	88.6	80.3	98.7	88.6
Left Sphere	94.4	92.0	93.2	57.6	86.6	69.2	65.7	95.0	77.7	65.9	95.0	77.8
Thirty Spheres	99.0			98.5			97.1			96.7		

Table 2: Numerical and Visual Comparison of Clustering Quality for Different Distribution of Clusters

Means does not merge the spirals but represents each with a number of spherical clusters. For any output of K-Means, RIC merges all parts of spirals into one cluster that results on the lowest F-score. RIC fails to separate spirals because of two reasons: *i*) spirals cannot be modeled by one probability density function as done in RIC; *ii*) RIC considers only the global orientation of clusters which is similar for both spirals.

The HierarchicalClusters dataset is the most challenging for all clustering techniques. The dataset consists of a plane, two dense spheres embedded in the plane cluster, and a sparse sphere outside the plane. All competitors perform poorly. CURE merges half of the plane and the embedded spheres into one cluster. It splits the other half of the plane and removes many border points of the outside sphere. DBScan separates clusters according to the density level and is able to identify the embedded spheres only. In contrast to DBScan, OPTICS is able to analyze all density levels, but

due to the fluctuations in the reachability plot OPTICS splits the plane and removes many border points from the spheres. K-Means finds all spheres however fails with the plane: the upper and right parts of a the plane are identified with two clusters and other parts are assigned to the spheres. RIC separates dense spheres from the plane, however, merges sparse outside sphere with the plane due to the similarity in their density distribution.

The OverlappingSpheres dataset compares techniques for two overlapping spherical clusters. CORE is the most accurate. CURE incorrectly assigns some points of the left sphere to the right sphere and has the lowest average F-score. OPTICS and DBScan remove all points where the density level is lower than the density level in the overlap. K-Means performs better than OPTICS and DBScan but, still, incorrectly assigns points of the right sphere to the left sphere. RIC correctly assigns border points of spheres in non-overlap areas but, fails to do that for many border points

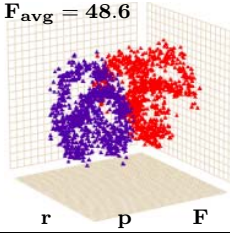
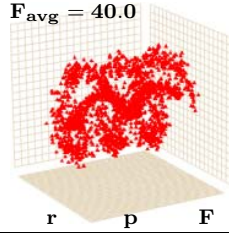
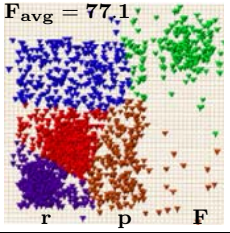
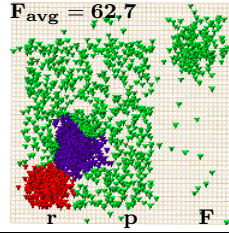
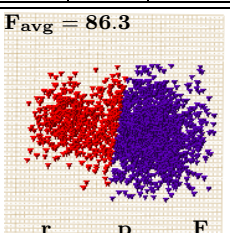
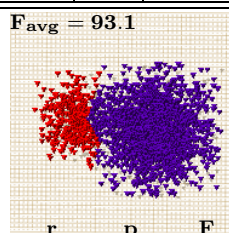
	K-Means			RIC		
Two Spirals	$F_{\text{avg}} = 48.6$			$F_{\text{avg}} = 40.0$		
						
	50.0	67.0	57.2	100.0	66.8	80.15
	50.3	33.2	40.0	-	-	-
Hierarchical Clusters	$F_{\text{avg}} = 77.1$			$F_{\text{avg}} = 62.7$		
						
	95.8	78.8	86.5	86.4	86.3	86.3
	100.0	73.8	84.9	98.4	83.8	90.5
	100.0	75.6	86.1	-	-	-
	35.29	92.1	51.0	98.9	59.6	74.3
Overlapping Spheres	$F_{\text{avg}} = 86.3$			$F_{\text{avg}} = 93.1$		
						
	87.2	100.0	93.2	99.5	95.5	97.5
	100.0	66.0	79.5	81.4	97.6	88.7
	90.7			22.4		

Table 2: Continued From Previous Page

in the overlap. CORE separates clusters along the center of the overlap and achieves the highest average F-score.

9.2 Performance Evaluation. The thirty spheres dataset is five dimensional and consists of thirty non-overlapping, differently sized and randomly placed spheres. The last line in Table 2 shows that all methods perform well for this dataset. We use this dataset to empirically evaluate the performance and the dependence on the sample size for each method.

Figure 8 evaluates CORE for different sample sizes on the thirty spheres dataset which consists of 10^5 data points. The results present here are similar to results achieved with other datasets.

CORE outperforms other methods and produces better quality clusters for the same sample size (cf. Figure 8(a)). The quality of CORE monotonically decreases as the size of the sample becomes smaller. The quality of DataBubbles

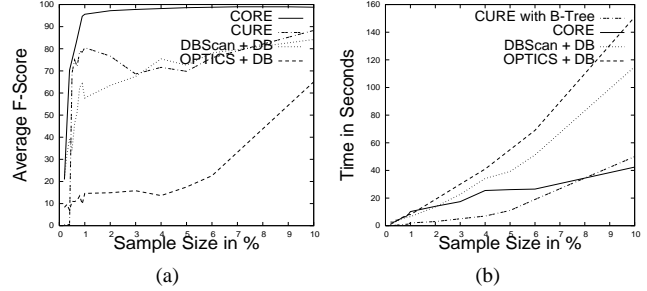


Figure 8: Comparison of Performance

using OPTICS is the lowest because identification of valleys in reachability plot computed using data bubbles is not effective with steepness parameter of OPTICS. The computation time of CORE (cf. Figure 8(b)) is similar to CURE which uses B-Tree for fast computation of the neighborhood, and less than OPTICS or DBScan which are implemented with DataBubbles but without any indexing structure. Supporting their implementation with an indexing structure would bring them to the level of CORE.

9.3 Real World Data. Figure 9 evaluates CORE for two real world datasets that contain time-varying data: a web log dataset and a financial dataset. Datasets of this type are usually very large and often require precise quality clustering which is a challenge for parametric techniques.

Web logs. The dataset contains information about the web page accesses handled by the `sunsite.dk` server, a mirror of open source projects. The server is widely used in Europe and generates a clickstream log of more than 100MB per day from more than 250 countries. Figure 9(a) shows the clicks received from `.com` domain for one day. The X coordinate shows the time of the click and the Y coordinate shows the URL of the retrieved document. URL's are ordered according to their time of first occurrence. This yields curves for the click of search robots who scan entire collections of web documents. Search engines aim to scan the entire collection of web documents resulting in a curve, while humans tend to visit a few selected documents only resulting in the horizontal lines.

CORE successfully identifies the accesses of a search robot (cf. cluster C_D in Figure 9(b)) and separates them from other accesses (cf. cluster C_A in Figure 9(b)). Other techniques produce less quality results: merge C_D with C_A , split C_A or remove many border points from both clusters.

Financial Data. We use a database with 150000 transactions by more than 4500 clients of a bank over a period of four years. Each point in Figure 9(c) indicates one transaction. The Y coordinate denotes the account identifier and the X coordinate corresponds to the day of the transaction. The data reveals two patterns in the behavior of clients: (i) the ac-

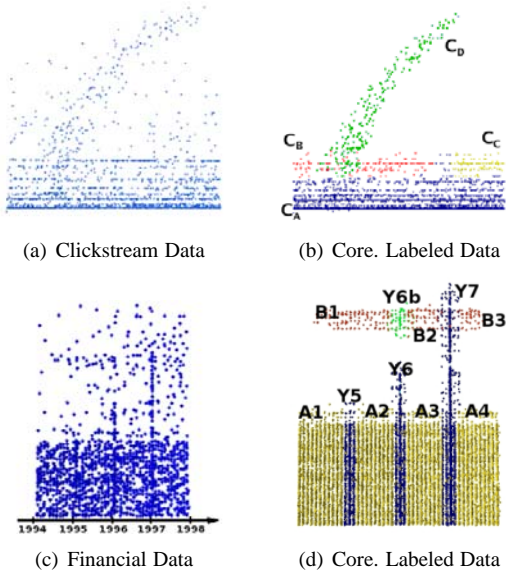


Figure 9: Real World Datasets

tivity and the number of clients increases over the years (Figure 9(c) has more points to the right than to the left) and (ii) there are clients with many and clients with few transactions (points are dense at the bottom of Figure 9(c)). Besides that, we can see intersection patterns at the beginning of each calendar year. At these times all customers show an increased activity. For a detailed analysis it is useful to separate the data and investigate the beginning-of-the-year activities independently from the rest of the data.

Comparing to other methods CORE identifies all intersection patterns (cf. Figure 9(d)). Other methods split cluster Y7 and its parts merge with clusters B2, B3, A3 and A4.

10 Conclusions and Future Work.

This paper presents CORE a new nonparametric clustering technique for large numeric databases. CORE explicitly computes maxima of the density and represents them with cores. The computation of cores is based on the AD-Tree, an incrementally constructed density estimation of the data. Key properties of the AD-Tree are a minimal adaptive grid that does not oversplit the space. CORE builds on this property and introduces rectangular neighborhoods, which, together with gradients, are used to reliably identify cores. The experimental results show that CORE outperforms other clustering methods—particularly for datasets with clusters that overlap or vary in density. In the future work we want to generalize the definition of cores to permit core with varying densities. It is also interesting to adopt CORE to separate intersecting clusters, which often occur in transactional data.

References

- [1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *SIGMOD Rec.*, 1998.
- [2] M. Ankerst, M. Breunig, H-P. Kriegel, and J. Sander. Optics: ordering points to identify the clustering structure. *SIGMOD Rec.*, 1999.
- [3] C. Böhm, C. Faloutsos, J. Pan, and C. Plant. Ric: Parameter-free noise-robust clustering. *TKDD*, 1(3), 2007.
- [4] C. Böhm, K. Kailing, P. Kröger, and A. Zimek. Computing clusters of correlation connected objects. In *SIGMOD*, 2004.
- [5] M. M. Breunig, H-P. Kriegel, P. Kröger, and J. Sander. Data bubbles: quality preserving performance boosting for hierarchical clustering. In *SIGMOD*, 2001.
- [6] F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, 2006.
- [7] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *KDD*, 2006.
- [8] Yixin Chen and Li Tu. Density-based clustering for real-time stream data. In *KDD*, 2007.
- [9] M. Ester, H-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996.
- [10] A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. In *KDD*, 2007.
- [11] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. In *SIGMOD*, 1998.
- [12] A. Hinneburg and H.-H. Gabriel. Denclue 2.0: Fast clustering based on kernel density estimation. In *IDA*, 2007.
- [13] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. 1998.
- [14] H-P. Kriegel and M. Pfeifle. Hierarchical density-based clustering of uncertain data. In *ICDM*, 2005.
- [15] Hans-Peter Kriegel and Martin Pfeifle. Density-based clustering of uncertain data. In *KDD*, 2005.
- [16] T. Li. A general model for clustering binary data. In *KDD*, 2005.
- [17] G. Moise and J. Sander. Finding non-redundant, statistically significant regions in high dimensional data: a novel approach to projected and subspace clustering. In *KDD*, 2008.
- [18] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: a wavelet-based clustering approach for spatial data in very large databases. *The VLDB Journal*, 2000.
- [19] Y. Shi, Y. Song, and A. Zhang. A shrinking-based approach for multi-dimensional data analysis. In *VLDB*, 2003.
- [20] A. K. H. Tung, X. Xu, and B. C. Ooi. Curler: finding and visualizing nonlinear correlation clusters. In *SIGMOD*, 2005.
- [21] J. Wu, H. Xiong, and J. Chen. Sail: summation-based incremental learning for information-theoretic clustering. In *KDD*, 2008.
- [22] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger. Scan: a structural clustering algorithm for networks. In *KDD*, 2007.
- [23] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Rec.*, 1996.
- [24] L. Zhao and M. J. Zaki. Tricluster: an effective algorithm for mining coherent clusters in 3d microarray data. In *SIGMOD*, 2005.