Neural Comput & Applic
DOI 10.1007/s00521-015-2145-z

ORIGINAL ARTICLE

A density invariant approach to clustering

Manish Kashyap[1] · **Mahua Bhattacharya**[1]

Received: 20 May 2015 / Accepted: 21 December 2015

Abstract Organizing data into sensible groups is called as 'data clustering.' It is an open research problem in various scientific fields. Neither a universal solution nor an absolute strategy for its evaluation exists in the literature. In this context, through this paper, we make following three contributions: (1) A new method for finding 'natural groupings' or clusters in the data set is presented. For this, a new term 'vicinity' is coined. Vicinity captures the idea of density together with spatial distribution of data points in feature space. This new notion has a potential to separate various type of clusters. In summary, the approach presented here is non-convex admissive (i.e., convex hulls of the clusters found can intersect which is desirable for non-convex clusters), cluster proportion and omission admissive (i.e., duplicating a cluster arbitrary number of times or deleting a cluster does not alter other cluster's boundaries), scale covariant, consistent (shrinking within cluster distances and enlarging inter-cluster distances does not affect the clustering results) but not rich (does not generates exhaustive partitions of the data) and density invariant. (2) Strategy for automatic detection of various tunable parameters in the proposed 'Vicinity Based Cluster Detection' (VBCD) algorithm is presented. (3) New internal evaluation index called 'Space-Density Index' (SDI) for the clustered results (by any method) is also presented. Experimental results reveal that VBCD captures the idea of 'natural groupings' better than the existing approaches. Also, SDI evaluation scheme provides a better judgment as compared to earlier internal cluster validity indices.

Keywords Data clustering · Classification · Density invariant clustering · Pattern recognition · Cluster validity indices · Internal cluster evaluation · Cluster validity indices

1 Introduction

Finding natural groupings present in a given data set is one of the most fundamental tasks in various sciences. This process is referred to as data clustering. It is trivial for human brain to find natural groupings in the data set. Due to the fact that our understanding of how human brain works is limited, imitating the human brain is difficult if not impossible.

Clustering algorithms present in the literature can be broadly classified into two categories—*Hierarchical* and *Partitional*. Hierarchical algorithms try to find clusters by starting with the individual points as clusters in the data set and then grouping them with other nearby and similar points. However, partitional algorithms do not impose any such structure on the data and try to find the partitions at once. Popular algorithms used in each of these categories are 'complete-link' and 'K-means,' respectively.

To be able to do clustering successfully, one needs to be very clear about what their definition of cluster is. What features will they be using for measuring similarity between two data points? Does the data that they are using have any clustering tendency? How many clusters are expected? Such fundamental challenges were first highlighted in [1]. Answering these questions is a matter of choice w.r.t. the philosophy used to construct the clustering

✉ Manish Kashyap
Manishkashyap.iiit@gmail.com

Mahua Bhattacharya
mb@iiitm.ac.in

[1] Department of ICT, ABV-Indian Institute of Information Technology and Management, Gwalior, India

Published online: 04 January 2016

algorithm. For example, FCM recognizes every data point as belonging to one or the other cluster while DBSCAN treats data points in extremely low-density regions as noise. Hence, they differ in their definition. Similarly the distance measure between two data points may be any of Pearson's correlation coefficient, minimum ratio, Euclidean distance or otherwise as given in [2]. FCM forcibly clusters the data. Methods like DBSCAN may declare the complete data as noise or a single cluster if density does not vary. Further FCM requires no. of clusters to be supplied by the user while DBSCAN and other density-based methods do not. There are numerous algorithms for clustering in the literature. Despite the above fact, there is no 'universally accepted' or 'correct in all aspects' algorithm for clustering. A good survey of clustering algorithms can be found in [3]. Nevertheless, we make a mention of two most popular algorithms that are used even today (may be their different variants). One of them is among the category of algorithms that try to find clusters based on density and called as *DBSCAN* [4]. This algorithm searches for connected and dense regions in the data. The density is calculated over a small neighborhood around the point to be processed. If the density is above certain threshold, the point being processed is declared to be in cluster and marked appropriately. To address the issue of nested clusters and variable density, a variant of the said algorithm was proposed in [5]. However, when the data size is large, density-based methods usually fail to distinguish between high- and low-density regions. The second algorithm called *K-means* falls under the category of objective function-based methods. These methods try to find out the clusters iteratively till an objective function is maximized or minimized. K-means has a shortcoming that it fails to detect non-convex and nested clusters. Despite this fact, its fuzzy variant called *Fuzzy C-means* is widely used in engineering applications.

Evaluation of clustering algorithms is a subjective issue. It is often measured by using cluster validity indices. They were defined in [1] as *internal, relative and external*. Evaluation based on internal indices is done by using the data alone and by measuring the goodness of fit in terms of inter- and intra-cluster variance, etc. Some popular examples of internal indices are—Dunn Indices [6], DB Index [7] and Silhouette Index [8]. Relative indices compare the results of various clustering algorithms on the same data. External indices are used when truth maps or class labels are available. Recently, to address the issue of overlapping clusters, new cluster validity index is proposed in [9]. Proposed index is non-distance based and uses the relationship of membership for measuring intra-cluster compactness and inter-cluster distance. A survey of cluster validity indices is given in [10].

Among the recent developments, in [11] the problem of clustering large sized data is addressed where the authors have first found cluster centers by sampling the data using large sampling size and then estimated the actual cluster centers within that sample. While doing so, linear space and time complexity is incurred. In [12], cellular automata-based clustering algorithm is presented. In [13], issue of rounding (How many eigenvectors of data similarity matrix are to be used) in spectral clustering is addressed using latent tree models. In [14], Bee colony optimization was used to automatically detect the number of clusters. Dimension selection, dimension weighting and data assignment are three circular dependent essential tasks for high dimensional data clustering. Integrated constraint-based clustering (ICBC) algorithm is proposed in [15] to address all the three issues simultaneously. An empirical evaluation of cluster quality measures is presented in [16]. Results from various clustering algorithms to same data sets are combined using long-term behavior of a dynamical system in [17]. Meta data learning (what type of algorithms solves what type of problems) in context of selecting appropriate algorithms for clustering is done in [18] for making recommendations to select or reject one algorithm over another. Ad hoc work in case of handwriting recognition is done in [19]. A recent variant of FCM is presented in [20]. Hybrid clustering methods like in [21] utilize desirable properties of two or more algorithms. In [21], global search capability of 'Bacterial foraging algorithm' and local search capability of FCM is utilized. Similarly in [22], for brain MRI images, a variant of FCM is proposed which combines FCM and probabilistic C-means using Gaussian weights to indicate spatial extent of neighborhood. For big data, clustering using 'Local Sparse Representation' is proposed in [23]. In [24], an improved version of 'Incremental FCM' is proposed to handle the same problem. Divide and conquer approach to FCM by using seeds at different locations for clustering for parallel computing and speedup is proposed in [25]. To achieve the same, membrane computing is used in [26]. The problem of seed initialization in efficient way is addressed in [27]. When the data comes from several interconnected manifolds, subspace clustering techniques are useful. One such technique is given in [28] which used curvature connectivity for various initial landmarks. The issue of noise for subspace density is addressed in [29]. Sensitivity of spectral clustering methods have been taken care of in [30] using newly proposed 'Density Adaptive Neighborhood' technique. Similarly neighborhood-based clustering is done in [29].

Motivated by the problems of density invariance, automatic detection of number of clusters, computational complexity and nesting of clusters and a better evaluation strategy, in this work we propose a method to identify natural groupings in data (unsupervised) based on mechanisms like DBSCAN and its variants. A new term called

vicinity is defined which takes into consideration density as well as the spatial distribution of feature points around the point being processed. The power of this notion in context above-mentioned problems is revealed in Sect. 4. The method developed here is applicable to *n*-dimensional clustering problem. However, results in this work are illustrated on two-dimensional data sets for visualization of feature space, and therefore intuitively viewing the expected clustering results in advance. Further, a new internal evaluation index called 'Space-Density Index' (SDI) for the clustered results (by any method) is also presented.

So, the main contributions of this paper are—New algorithm for clustering the data is proposed (Vicinity Based Cluster Detection) together with automation of their tunable parameters and new evaluation index for clustering is proposed (SDI). Rest of the paper is organized as follows—In Sect. 2, related work is presented. Fuzzy C-means, DBSCAN and EnDBSCAN algorithms are discussed in detail as some of the definitions will be used in the proposed work too. In Sect. 3, we present the proposed work together with the automation of tunable parameters and algorithm speed up. Section 4 presents the experimental results and also proposed SDI for evaluating the clustering results for any arbitrary clustering algorithm. Section 5 is for discussion and conclusion. It also provides necessary future directions.

2 Related work in the field of clustering

In this section, the two most popular techniques for clustering 'Fuzzy C-means' and 'DBSCAN,' in their fundamental forms, are discussed. However, there are numerous variants of these techniques addressing various issues in them or making their generalizations. Along with DBSCAN, its variant called EnDBSCAN is also discussed as the framework it uses is also utilized in the proposed work. EnDBSCAN addresses the issue of nested variable density clusters in DBSCAN.

2.1 Fuzzy C-means

The objective of fuzzy C-means algorithm, given the number of clusters to be detected a priori in a data set, is to assign each data point in the data set a number between zero and one corresponding to every cluster such that the number represents the membership of that data point in a particular cluster. This is followed by hardening (for a data point, setting the highest membership value to 1 and rest all to 0) for crisp classification.

A notation for this technique (by no means unique) is given as follows—Data set X with n elements as $X = \{x_1, x_2, x_3, \ldots, x_n\}$. Each data sample x_i defined by m features, i.e., $x_i = \{x_{i1}, x_{i2}, x_{i3} \ldots x_{im}\}$ (normalized to uniform scale). Number of clusters $= c$.

Now since the number of clusters 'c' is predefined, cluster centers are randomly initialized. Then following an iterative procedure, distances from every data point to the cluster centers are calculated and hence their membership values. Then, the energy function or the value of objective function for that particular solution is determined. If it is above some threshold, new cluster centers are calculated and this iteration is continued till the solution converges. This is followed by hardening of the fuzzy result by appropriate threshold. The objective function, formula for distance, formula for updating cluster centers and membership values at next iteration are given below

1. The Objective Function to be minimized is: $J_m(U, v) = \sum_{k=1}^{n} \sum_{i=1}^{c} (\mu_{ik})^{m'} d_{ik}^2$
2. Distance calculation formula: $d_{ik} = d(x_i - v_i) = \left[\sum_{j=1}^{m} (x_{kj} - v_{ij})^2\right]^{1/2}$
3. Updating of cluster centers: $v_{ij} = \frac{\sum_{k=1}^{n} \mu_{ik}^{m'} \cdot x_{ki}}{\sum_{k=1}^{n} \mu_{ik}^{m'}}$
4. Optimum value selection: $J_m^*(U^*, v^*) = \min_{M_{fc}} J_m(U, v)$

With J_m as the objective function U as the partitioning matrix (size $c \times n$), μ_{ik} is the membership value of kth data point in the ith cluster, d_{ik} is the distance of kth data point to the ith cluster, v_{ij} is the jth coordinate of the ith cluster center, x_{ik} is the kth coordinate of ith data point. '*' on any script indicates optimum value.

FCM fails to detect non-convex nested or partially nested clusters however detecting dense regions is its forte. Since it is iterative in nature, it is computationally complex. It forcibly detects clusters in the input data irrespective of the fact that the input data can or cannot be clustered. It requires number of clusters to be detected as an input which violates the idea of 'natural groupings.' However, it acts as foundation to its various variants which address some of the above issues.

2.2 Density-based spatial clustering of applications with noise (DBSCAN and EnDBSCAN)

The objective of DBSCAN algorithm is to discover natural groupings in a given data set. Not every point needs to be classified as a data point belonging to cluster. There will be some data points belonging to cluster (which appear in relatively dense regions) and some points not belonging to any (called as noise points).

Authors have defined *Eps*—neighborhood of a point p in database, *MinPts* is the minimum number of points a data point must be surrounded by to be included in that cluster.

A point p is directly density-reachable from a point q w.r.t. *Eps* and *MinPts* if $p \in N(Eps(q))$ and $|N(Eps(q))| \geq MinPts$ (core point condition). A point p is density-reachable from a point q w.r.t. *Eps* and *MinPts* if there is a chain of points $p_1, p_2, p_3, \ldots, p_n$ with $p_1 = q$, $p_n = p$ such that $p_i + 1$ is directly density-reachable from p_i. A point p is density connected to a point q w.r.t. *Eps* and *MinPts* if there is a point o such that both, p and q are density-reachable from o w.r.t. *Eps* and *MinPts*.

To find a cluster, DBSCAN starts with an arbitrary point p and retrieves all points density-reachable from p with respect to *Eps* and *MinPts*. If p is a core point, this procedure yields a cluster with respect to *Eps* and *MinPts*. If p is a border point, no points are density-reachable from p and DBSCAN visits the next point of the database.

The complexity of a neighborhood query processing is $O(n)$. The use of a spatial index such as R^*-tree, reduces it to $O(\log_m n)$ where m is the number of entries in a page of R^*-tree and n is the size of the data set. The complexity of the DBSCAN becomes $O(n \log_m n)$ on use of spatial index, otherwise it is $O(n^2)$. The algorithm can handle large amounts of data. DBSCAN can identify all shapes of clusters; however, it cannot identify complex cluster structures over variable density space.

In EnDBSCAN algorithm, to address the problem of Global settings of *Eps* (Tunable) and nested clusters, authors in [5] proposed to extend DBSCAN in the following way. They used a term called '*core distance*' from a popular algorithm Ordering Points to Identify the Clustering Structure (OPTICS) which says 'The core distance of an object p is the smallest ε' value that makes p a core object (a core object is an object with at least *MinPts* in its neighborhood of predefined size). If p is not a core object, the core distance of p is undefined'. Then they defined *core neighbor* in the same way as neighbor in DBSCAN and other definitions similar to DBSCAN. Intuitively, it simply means that despite of the neighborhood size considered, core distance is independent of it and core neighbors agree on certain range of density. The advantage of using such a modification was detection of nested clusters. However, it leads to introduction of additional tunable parameter called ε'. The complexity is same as DBSCAN, i.e., $O(n \log_m n)$.

3 Proposed work on *vicinity* based notion of clusters

Intuitively, in context of density-based algorithms, the idea behind inclusion of a data point in a cluster is presence of at least a pre-specified minimum number of other data points (*MinPts*) in a neighborhood (defined by *Eps*) centered on it. On similar grounds in context of vicinity-based notion of clusters, the idea behind inclusion of a data point in a cluster is same accept the fact that we bring into picture how the points in the said neighborhood are distributed. To better understand this, consider the following example (refer to Fig. 1).

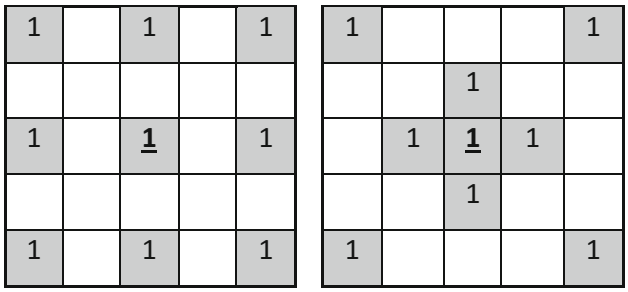


Fig. 1 Two 5 × 5 patches from a two-dimensional feature space of certain data set with cells marked *1* corresponding to data points. The feature point in the center of both, i.e., the ones with location (3, 3) are the feature points under consideration

Let's say we define the *MinPts* (as per the EnDBSCAN algorithm) to be 9. Then, the core distance (the minimum size of neighborhood around a feature point which contains *MinPts*) of the feature point under processing (the center one in both the patches) will be calculated for a 5 × 5 window. No other smaller square window will contain 9 points. The core distance for both the center data points is then as $2\sqrt{2}$ (Considering the distance between two neighboring diagonal points as $\sqrt{2}$ and between two neighboring non-diagonal points as 1, and consecutive distances are measured from center of cell to center of another). However, the two patches are structurally different. The first one is uniformly distributed but the second is not. Therefore, to distinguish the two center points, we propose a new term called as *vicinity* in Definition 2. But prior to that lets define *Eps*-Neighborhood in Definition 1 below. We will return to the calculation of vicinity for the above patches shortly.

Definition 1 (*Eps-neighborhood of a feature point p*) The *Eps*-neighborhood of a feature point p is denoted by $N_{Eps}(p)$ and is defined by $N_{Eps}(p) = \{q \in F | distance(p, q) < Eps\}$. Note that this definition is over the feature space F and not over data points themselves.

The essence of the proposed algorithm lies in the fact that here calculations are made for every data point, but in the feature space F while most of the other algorithms operate only on data points not in feature space. Also note that feature space is infinite in extent. We make calculations in a limited region of it such that all the data points are accommodated.

Definition 2 (*Vicinity*) Vicinity $V(i)$ of a feature point at location i (i being an n-dimensional vector) in

n-dimensional feature space F, labeled by data points in such a way that the label at a point in feature space represents frequency of that data point in the data set D concerned, is defined as

$$V(i) = \frac{F(i)}{a} + \sum_{j \in W} \frac{F(j)}{d(i,j)} \quad (1)$$

$d(i,j)$ is Euclidian distance. Any other radial, strictly decreasing function will also do.

Where $V(i)$ is the vicinity of ith point in the feature space (note that whether a point belongs to data space or not, vicinity is calculated for every point in feature space as it will help in finding some connected objects as explained later), $F(i)$ is the frequency (label) of the ith point in feature space (calculated from the given data set), $d(i,j)$ is the distance (any suitable distance measure could also be used) between ith and jth feature point in feature space. Consider the distance of a point from itself as an arbitrary number 'a'. Also, $0 < a \leq 1$. The first term on RHS of equation may become infinite if we choose $a = 0$, therefore we choose $a = 1$. 'a' in Eq. 1 affects the data points only as for other points in the feature space, $F(i) = 0$. So, if we choose a near 0, the vicinity value for neighboring points will be discontinuous (very high) owing to high value of $1/a$ and this is undesirable. To maintain continuity, any value near 1 will do. So we choose $a = 1$ which signifies that immediate neighbors are equally important as pixel under processing). For an illustration of vicinity on one-dimensional data see Fig. 2a, b.

Apparently the more close the feature points are to the point being processed, higher is the vicinity. It is to be noted that while calculating vicinity values for a particular point, it is not necessary to consider the contribution of all the other feature points [due to inverse of distance term in the vicinity equation and due to this window size W, centered on $F(i)$, comes into play]. So a suitable neighborhood will suffice. If we threshold the 'vicinity image' (an n-dimensional array for n-dimensional feature space) containing the vicinity values of all points in feature space) by a suitable threshold such that points whose vicinity is below the 'vicinity threshold,' are set to zero (as vicinity is a positive quantity we don't have negative values) we get a binary image (array for n-dimensional case) containing some components. These components are called as clusters (for the case when the density of feature points is nearly uniform). However, the issue of non-uniform density (to achieve density invariance) is addressed in the coming sections. Also coming sections provide a faster way of calculating vicinity values and addresses the problem of automating the tunable parameters like vicinity threshold and window size for neighborhood.

Returning to the calculation of vicinity for the two patches in Fig. 1 (for center feature points), we get the vicinity values of 0.0518 and 0.0653 for the two patches, respectively. This provides some sort of discrimination which is desired. Let's now define some more useful terms and definitions that will lead to theoretical foundations of the proposed method.

Definition 3 (*Direct vicinity connected*) A point 'p' is directly vicinity connected to a point 'q' with respect to *Eps*, *MinPts* and vicinity threshold 'λ' if it satisfies

1. $p \in N_8(q)$—(8 neighborhood of point q)
2. $V(p) > \lambda$ and $V(q) > \lambda$

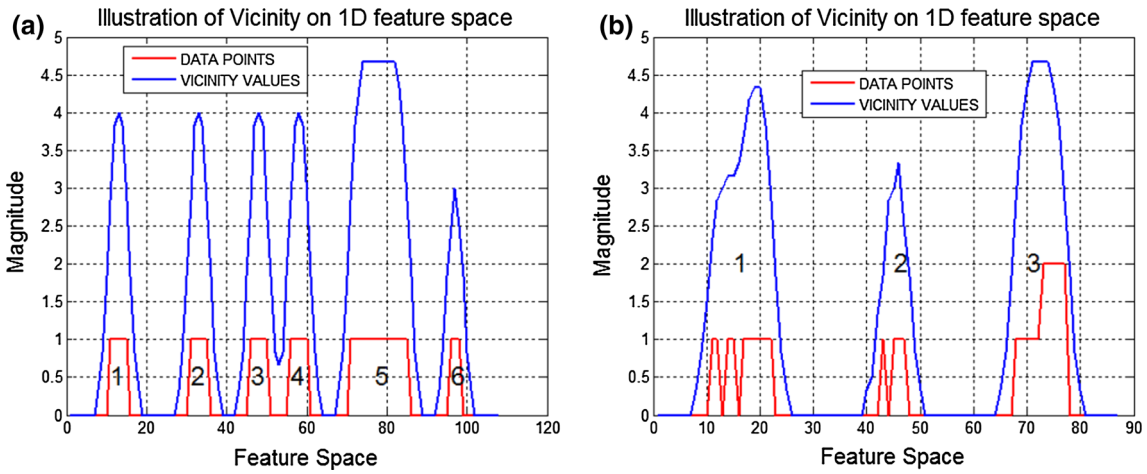


Fig. 2 Illustration of vicinity values for one-dimensional data. **a** *Red colored curve* shows the grouped data points and *blue curve* shows the vicinity values. Clusters 1 and 2 are well separated. Hence there is no interaction in vicinity values. Different is the case with clusters 3 and 4. Cluster 5 is relatively thick w.r.t. 6 that is why vicinity value magnitude differs. **b** Less dense clusters are 1 and 2. 3 is a cluster with $F(i) = 2$ for some points. Corresponding difference is noted in vicinity values (color figure online)

Definition 4 (*Vicinity Connected*) A point 'p' is vicinity connected to a point 'q' with respect to *Eps*, *MinPts* and λ if there is a chain of points $p_1, p_2, p_3, \ldots, p_n$ such that p_{i+1} is directly vicinity connected to p_i, $(p, q \in F)$.

We are now in a position to define vicinity based cluster in a feature space. Intuitively, a cluster in feature space is defined as a set of vicinity connected points in feature space which is maximal with respect to vicinity connectedness. *Ether* is similarly defined as set of points in feature space not belonging to any cluster.

Definition 5 (*Cluster*) Let F be the feature space of feature points labeled with frequency of each point in the data set D and has a vicinity value as calculated by Definition 2. A cluster C in feature space F is a non-empty subset of F satisfying:

1. $\forall p, q$ if $p \in C$ and q is vicinity connected to p w.r.t. *Eps*, *MinPts* and λ then $q \in C$ (maximality)
2. $\forall p, q \in C$, p is vicinity connected to q w.r.t. *Eps*, *MinPts* and λ (connectedness)

Definition 6 (*Ether*) Let $C, C_2, C_3, \ldots C_n$ be the clusters in feature space F w.r.t. parameters *Eps*, *MinPts* and λ, points in feature space F which do not belong to any cluster are called as ether points.

Definition 7 (*Data Cluster and Noise*) For a particular cluster C_i $(i \in \{1:k\})$ in F, data cluster is defined by all those points in $C_i \in D$. Rest of the points in C_i are possible locations of another observations in the same cluster. Similarly data points in 'ether' are called as noise points.

To validate the correctness of the proposed work, we have the following two lemmas.

Lemma 1 *Let 'p' be a point in F satisfying $v(p) > \lambda$. Then the set $O = \{o | o \in F$ and o is vicinity connected to p w.r.t Eps, MinPts and $\lambda\}$ is a cluster w.r.t. Eps and λ.*

Lemma 2 *Let C be the cluster w.r.t. Eps, MinPts and λ, then C equals to the set $O = \{o | o \in F$ and o is vicinity connected to p w.r.t Eps, MinPts and $\lambda\}$.*

3.1 Algorithm for proposed work on Vicinity Based Cluster Detection

Keeping above definitions in mind, a new algorithm called *Vicinity Based Cluster Detection*, for detecting natural groupings in data, is now proposed. We present the pseudo code followed by explanation of every step.

```
Input : D_MxN
Output : R_Mx1
Intermediate Data Structures F, V, Labelled_array
BEGIN :
/* Step 1 : Calculating the feature space dimensions */
        L(D(:,j)) = |max(D(:,j)) − min(D(:,j))|;
/* Step 2 : Initialize Feature space F with all zeros */
        F_{L(D(1,j)),L(D(2,j)),L(D(3,j)),... L(D(N,j))} = 0;
/* Step 3 : Labelling the feature space with data points */
        For i = 1: M
                F(D((i,1), D(i,2), ... D(i,N))) = F(D((i,1), D(i,2), ... D(i,N))) + 1;
        End for
/* Step 4 : Calculation of Vicinity values for every point of feature space*/
        For i = 1: No_of_elements(V)
                V(i) = F(i)/a + Σ_{j∈W} F(j)/d(i,j)
        End for
/* Step 5 : Thresholding */
        For i = 1: No_of_elements(V)
                If V(i) < λ
                        V(i) = 0
                End if
        End for
/* Step 6 : Labelling */
        Labelled_array = Labelling_routine(V(i))

/* Step 7 : Back mapping */
        For i = 1: M
                R_Mx1(i,1) = Labelled_array((i,1), D(i,2), ... D(i,N))
        End for
END
```

- *Step 1* (*Calculating the feature space dimensions*) Data set $\mathbf{D_{MXN}}$ is a two-dimensional array of size $M \times N$ with rows corresponding to data points and columns corresponding to features. Feature space will have same number of dimensions as number of columns in $\boldsymbol{D}$ i.e. $\boldsymbol{N}$. $L(\cdot)$ denotes the length of any set inside it. $D(:, j)$ is the set of all jth dimension data point from data vector D. Length is calculated using the following formulation $L(x) = |X_{\max} - X_{\min}|$ where $|\cdot|$ is used to calculate absolute value.
- *Step 2* (*Create the feature space*) Initialize an array '$\boldsymbol{F}$,' with all zeros, having size $= L(D(1,j)), L(D(2,j)), L(D(3,j)), \ldots, L(D(N,j))$.
- *Step 3* (*Label the feature space by mapping data points onto it*) To map data points from data set to feature space, we use $F(D(i,1), D(i,2), \ldots, D(i,N)) = F(D(i,1), D(i,2), \ldots, D(i,N)) + 1$. With this step, all the data points are mapped to feature space such that value at ith point in the feature space $F(i)$ represents the frequency of the data point in the data set.
- *Step 4* (*Calculate the Vicinity values for every point in the feature space*) Initialize an array '$\boldsymbol{V}$,' called vicinity array, with all zeros, having same size as $\boldsymbol{F}$. Make use of Eq. (1) i.e. $V(i) = \frac{F(i)}{a} + \sum_{j \in W} \frac{F(j)}{d(i,j)}$ to calculate the vicinity value corresponding to each point in feature space. This step seems to be computationally complex. However, to speed up the algorithm, alternative mechanism is suggested in coming sections. Also, automatic detection of window size 'W' is a topic of interest of Sect. 3.3.

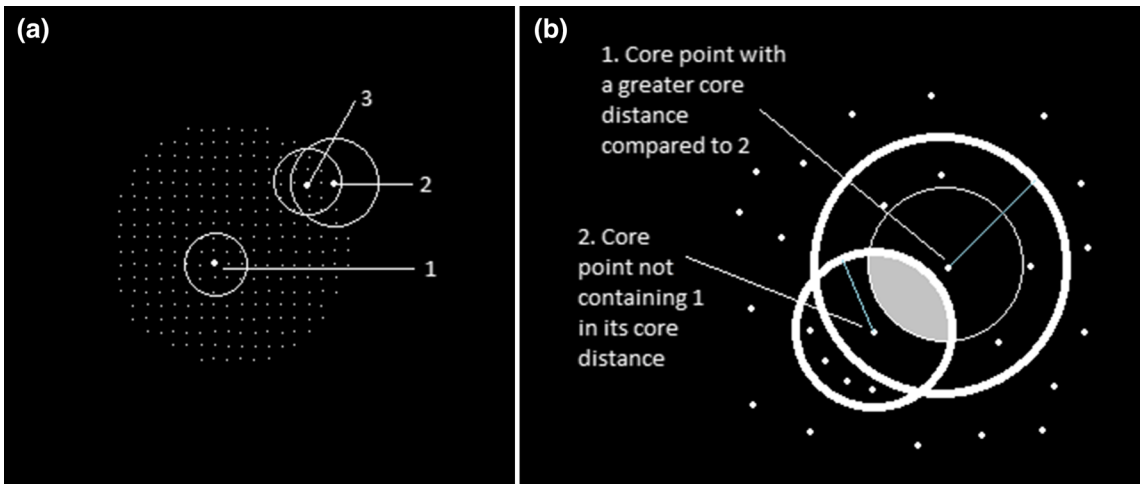


Fig. 3 **a** Various type of feature points w.r.t. core distance are shown, **b** illustration for effective core distance

- *Step 5* (*Thresholding*) For a particular vicinity threshold λ, one may obtain various partitions of the vicinity array. Every connected component of thresholded vicinity array is then, a cluster. This automatically raises a question—what should be a good value of vicinity threshold? The answer lies in the Sect. 3.2.2 where threshold curve and its interpretation is given.
- *Step 6* (*Labelling*) Label the connected points of the binary array (Thresholded vicinity array) to get connected components (These are the desired clusters).
- *Step 7* (*Back Mapping*) Back-map the labels to their corresponding data points in a separate result array '*R*' having size $M \times 1$.

3.2 Selecting the tunable parameters in density invariant way

This section addresses the problem of automatic selection/detection of the tunable parameters of VBCD algorithm.

3.2.1 Estimating the neighborhood (window) size

The density of points in feature space is variable. So, it is hard to select a single window size arbitrarily. By window size, we mean circular neighborhood. Even if rectangular windows are used, due to the definition of vicinity, their weights will form a radial pattern. So, the terms window and neighborhood size are used interchangeably. In this section, we present a method to get an estimate of window size automatically and independent of the fact that the average density of various clusters present may vary. The estimated window size is—'Median of effective core distance'. To understand the meaning of this, following definition of '*core objects*' and '*core distance*' from [31] will be used.

Core objects Core objects are those points in feature space which are surrounded by at least a pre-specified number of other feature points (called *MinPts*).

Core distance For a particular core point, the minimum distance within which those *MinPts* are contained is called as the core distance of that core point.

A general way to select the window size is then to take the average (or any measure of central tendency) of all the core distances for all the core points. However, while using this procedure to estimate the window size, the following problem may be encountered.

In Fig. 3a, three types of feature points (core points) are marked. (1) Point which is well within the cluster. (2) Point which is on the boundary of the cluster. (3) Point which is near the boundary of the cluster (but still inside the cluster such that the core distance is same as point of type 1). Core distance is represented by drawing a circular window with core points as centers and core distance as radius. It is expected that core distance of points well inside the cluster is less than the points on the boundary. Hence, if we take average or any measure of central tendency, it will be affected by core distance of boundary points also. But, an important observation is, point 2 (boundary point) lies in the core distance of point 3 (point inside the boundary). Hence, it is not necessary to include the window size of points at boundary for calculating the average. Because it is connected to the cluster through a smaller window size of point 3. Therefore, alternatively, we may define *effective core distance* of any core points as follows:

Effective core distance Effective core distance of a core point under consideration is defined as the minimum of all the core distances of the core points contained within the core distance of the feature point under consideration.

Referring to Fig. 3b, one may argue that as in case of point 1, the point with minimum core distance in the core neighborhood of it (which happens to be point 2) may not contain the point under consideration (point 1). This should imply that effective core distance should not be the minimum. But we still take the effective core distance as the minimum. The reason for this is high probability of intersection between the selected minimum window size as shown by another thin boundary window centered around 1 (but with size equal to that of 2) with window at point 2. This guarantees a positive vicinity in the region of intersection (shown shaded) and hence connectivity within the cluster. Of course when this does not happen, the points are disconnected. But in Sect. 4, as observed, the possibility of such an event is rare.

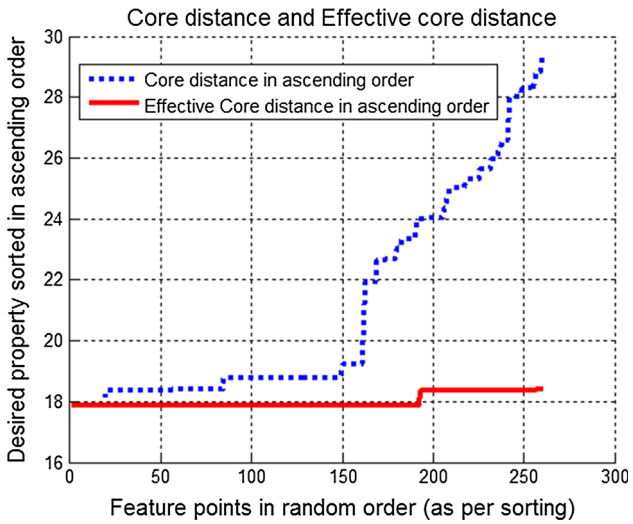


Fig. 4 Core distance and effective core distance curve for feature space of Fig. 3a

For Feature space of Fig. 3a, we calculate core distance and effective core distance for every point, sort them in ascending order and then plot them as shown in Fig. 4.

Same general observations can be made from plots of Fig. 4 (in Fig. 3a). *Y*-axis corresponds to the window size required. We can easily see that the swing on Y scale for effective core distance curve is much smaller than core distance curve. So there is less ambiguity in selecting the window size. We propose to select median of effective core distance as a suitable window size for obvious advantages associated with median. The median is of central importance in robust statistics, as it is the most resistant statistic, having a breakdown point of 50 %, i.e., even if half the data are changed, the median will not give an arbitrarily large result.

For feature space of Fig. 10b, the plots are shown in Fig. 5a. Notice the variation of curves. Although the density of both spirals in the feature space happens to be almost same, but due to presence of huge boundaries, the core distance on *Y*-axis has a very big swing. But intuitively, it is evident that a single window size is suitable for the feature space. Same fact is reflected in effective core distance curve. It is almost a straight line (two major line segments with little difference, shown in red). Similarly in Fig. 11, corresponding plot is shown in Fig. 5b. Here, four different stable regions can be seen. However, it is experimentally seen that median of effective core distance still works for high value of *MinPts* and due to the reasons explained above.

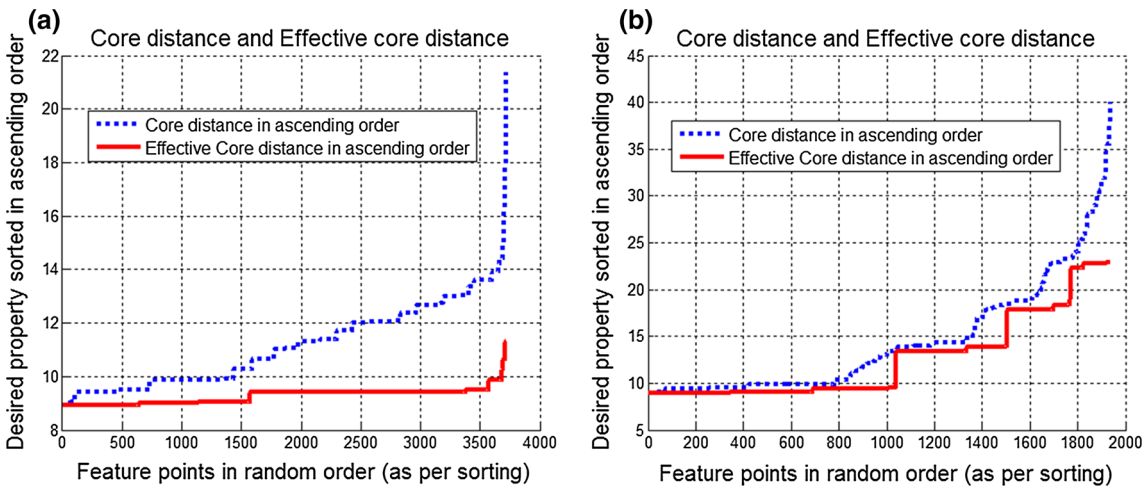


Fig. 5 Effective core distance and core distance curves for feature space of **a** Fig. 10b, **b** Fig. 10c

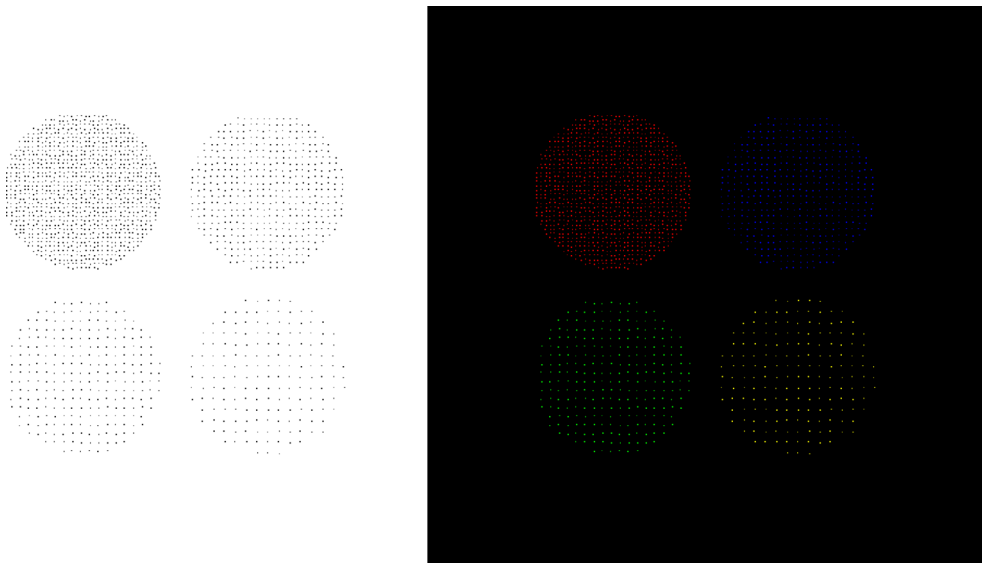


Fig. 6 A two-dimensional feature space having four variable density clusters and the corresponding clustering result by VBCD

3.2.2 *Selection of vicinity threshold*

Vicinity threshold(s) for a particular feature space is (are) those value(s) of vicinity (vicinities) for which the number of big objects remains constant in number of big object versus vicinity threshold curve. To explain the process, we consider a two-dimensional feature space as shown in Fig. 5 with the expected result. Note that there are four clusters of different density. The first step is to calculate the vicinity array/image by using the window size calculated in the previous section. Then, we threshold it step by step starting from a high threshold. The process is shown in Fig. 7 for feature space of Fig. 6.

Some general observations from Fig. 7

Let's first look at the densest cluster out of the four present. A study of its formulation (by decreasing the thresholds on vicinity image) revels the following.

1. At very high threshold, only feature points with very high vicinity appear in the resultant image [as is the case with 1st tile in Fig. 7 where vicinity threshold (λ) is kept = 231].
2. On decreasing the threshold further, more points which are spatially near and have vicinities greater than the threshold are added to form bigger object (as is the case with tile 2 and 3 with vicinity thresholds = 193 and 184, respectively).
3. From tile 4 onwards, since the connected component for the densest cluster is completely identified, there is no change in the boundary of the identified cluster; i.e., the cluster boundary is now stable w.r.t. variation in vicinity threshold.
4. Meanwhile the formation and stabilization of densest cluster, while lowering the vicinity threshold, clusters at low vicinity started appearing (refer to tile 5 of Fig. 7 where cluster 2 and 3 also started forming).
5. However, due to the formation or starting phase, there were initially very small connected regions (relatively small in size) and consequently they transformed to big cluster similar to the 1st cluster (refer to tiles 5 onwards).

In order to achieve density invariance, we plot number of 'Big Objects' detected versus vicinity threshold as shown in Fig. 8. Big objects mean those objects that have size greater than certain predefined threshold. This is done to avoid counting newly formed small clusters as matured or stable clusters.

Interpretation of 'number of objects' versus 'vicinity threshold' curve

For Figs. 6 and 7, the said plot is shown in Fig. 8 in blue color. Red 'stem' plots are the markers corresponding to the starting points of horizontal (or flat) regions in the blue curve. The magnitude of red markers gives us the relative range of thresholds for which the blue curve remains flat. Starting from a very high vicinity threshold (i.e., from the right of X-axis), and moving toward the low thresholds, several peaks and flat regions in blue curve are obtained. There is a correlation between peaks and formation of small objects in Fig. 7 and finally their merging into one object. Flat regions correspond to the range of vicinity thresholds over which cluster boundary become stable and/or new small patches of other clusters are in the process of formation. One can similarly correlate the other four flat regions and four peaks corresponding to four clusters of Fig. 6. In order to find the final clustering results, we need to combine the results of only those thresholded vicinity images where the flat

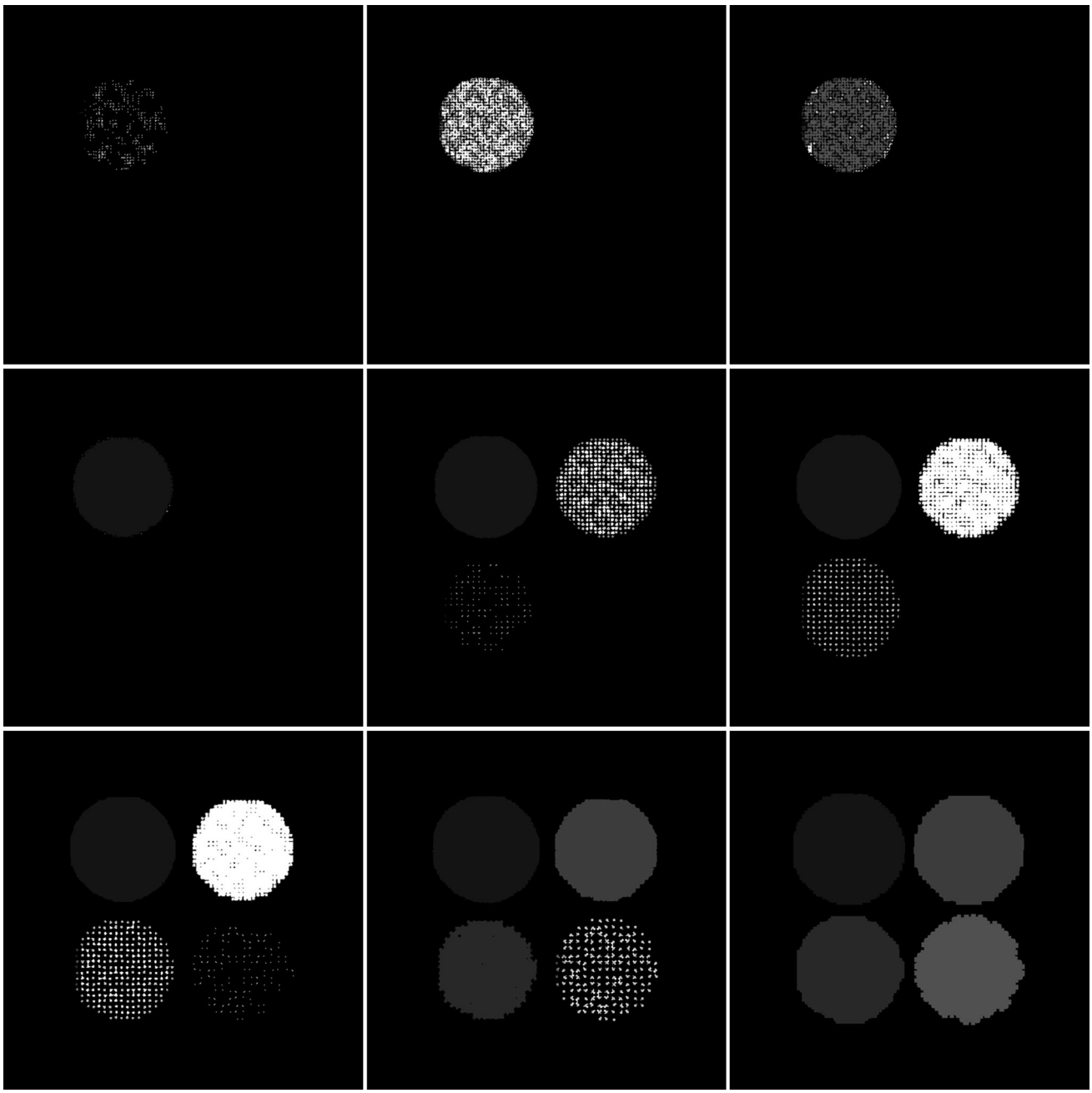


Fig. 7 9 × 9 tiles (to be read from *left* to *right* and *top* to *bottom*) representing various stages of detecting clusters at various vicinity thresholds. Clusters are differentiated by different *grayscale colors*. Note that as a result of applying VBCD, complete feature space is partitioned into clusters. After complete process, the data points in one particular cluster are colored by one color

region in the curve begins. And that too for only those flat regions which are considerably bigger in range (higher in magnitude) than the others.

Similarly for the feature space of Fig. 10d, the curve is shown in Fig. 9. It can be verified that the number of clusters in the stable region is always one. However, due to the fact that two stable regions are separated by one peak, the density varies in the cluster. And hence two different nested clusters are detected. It is shown in results in Fig. 4.

So, to select a vicinity threshold, one must keep in mind that for a data set with no to very low outliers, almost all data points need to be classified in one or the other clusters. For this to happen, threshold must be selected from the left side of the said curve. But if certain other regions are stable (shown by high red marker values) and lie to the right side of the curve too, one may threshold the vicinity array multiple number of times and combine the results (demonstrated in Sect. 4) to obtain nested clusters.

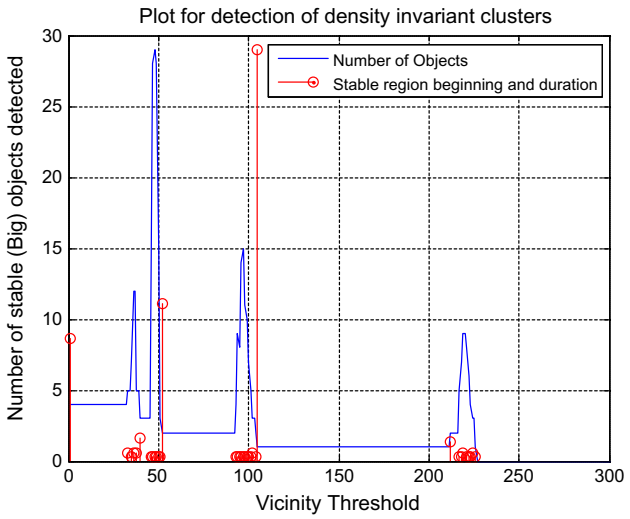


Fig. 8 Graph of number of objects versus vicinity threshold (in *blue color*) and markers for beginning of flat regions (horizontal in *blue graph*) with their magnitude representing the relative length of flat regions (color figure online)

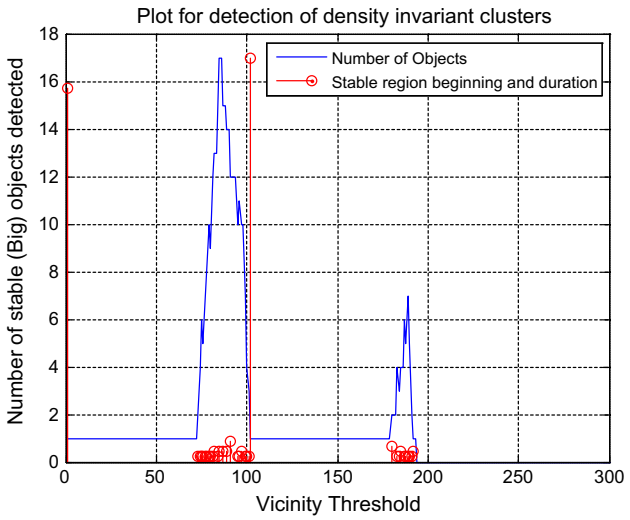


Fig. 9 Graph for feature space shown in Fig. 10d having nested clusters

3.3 Algorithm speed up

Step 4 in the algorithm discussed above is computationally expensive. Vicinity of every point in feature space (and not just data space) is to be calculated for this purpose as vicinity values for non-feature points in feature space helps in establishing the connectedness in thresholded image obtained after using vicinity threshold. This is good when data points are all different and we have a very large data set. But when few clusters are hanging in feature space, this step is very time-consuming. To speed up the algorithm, we make use of the following property of distance metric and one very important observation.

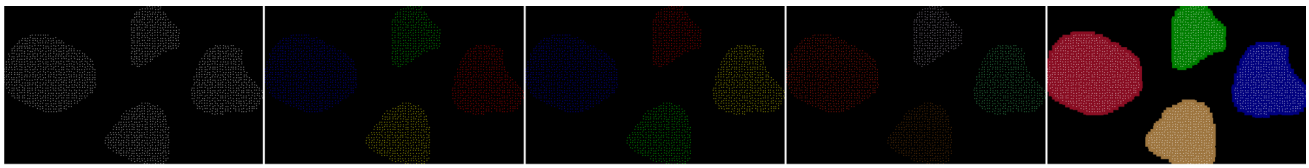
Property $d(x, y) = d(y, x)$

Observation In the light of above property the following two processes are equivalent.

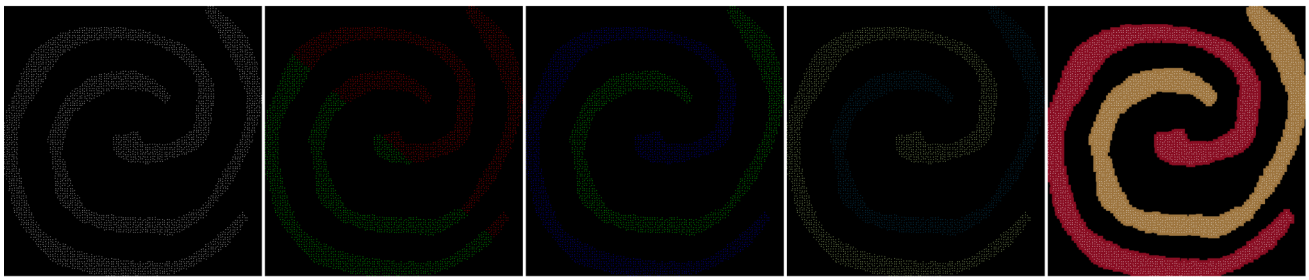
(i) *Spatial filtering approach for calculation of vicinity values in feature space* Based on the *Eps*-neighborhood selected, design a suitable spatial filter which meets the requirement of Eq. 1 and by using it, filter the complete feature space to obtain vicinity values of every point in feature space.

(ii) *Gravity inspired approach* Inspired by the gravitational field, if a mass exists in isolation, its effect in the regions surrounding it can be felt by test mass in gravitational field produced by original mass. The field is known to follow inverse square law, but we are using only inverses for our calculations in Eq. 1. For several discrete masses co-existing in the same space, the resultant at a location is the vector sum due to all masses. By using the same analogy, instead of doing calculation (spatial filtering in Observation 1) for every point in the feature space, we do calculations for labeled points only (i.e., the points belonging to data set). With the help of these points transformed to feature space, vicinity value of every other non-data point in feature space can also be calculated as follows—(please note that the total number of these points will be either less than or equal to the total points in data set, equality when all data points are different in feature space). For each such labeled point, and using the same spatial filter described earlier in observation (i), first multiply the filter by the label value (scalar) and then add the filter coefficients so obtained to the corresponding positions in 'vicinity feature space' (an n-dimensional array containing vicinity values initialized to zero) keeping center of the filter matched to the point being processed location.

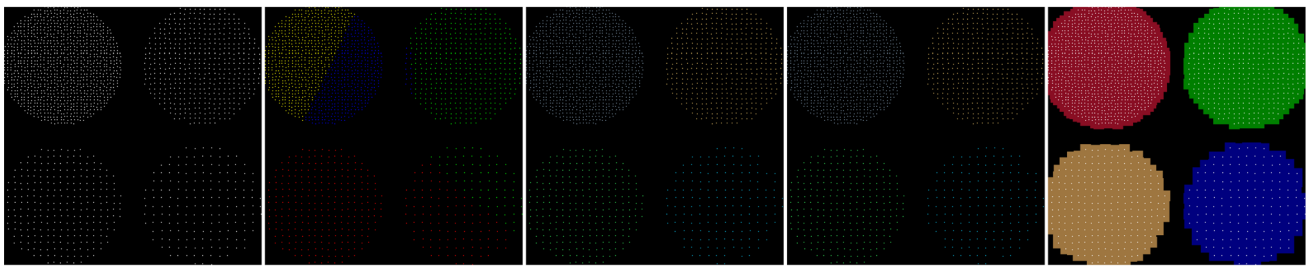
This will reduce the complexity drastically as now, we are making calculations for number of points either equal to or less data points. Also in spatial filtering method, for each point, filter elements have to be multiplied with all corresponding locations in 'feature space' then they have to be added together and finally divided by normalizing factor to give a single value for each location in feature space (normal filter operation). But by the new method, only addition of filter elements to the corresponding elements is to be done. The complexity is thus less than or equal to linear ($\leq O(M)$ in 'Big O' notation, $M =$ total number of data points). Before the speed up process, the complexity



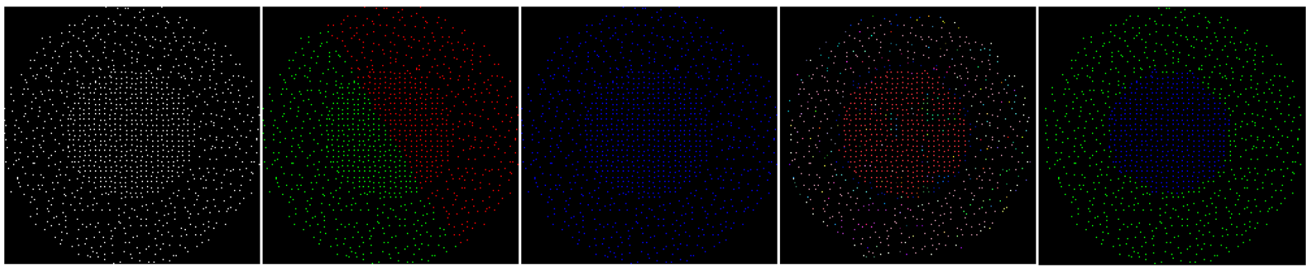
(a) Data set Name WELL SEPERATED DATASET (WSD)



(b) Data set name NON-CONVEX DATASET (NCD)



(c) Data set name VARIABLE DENSITY DATASET (VDD)



(d) Data set name NESTED DATASET (ND)

Fig. 10 Original and result of FCM, DBSCAN, EnDBSCAN and VBCD on **a** well-separated, uniform density clusters, **b** uniform density, non-convex clusters, **c** well-separated, non-uniform density clusters and **d** nested clusters (color figure online)

was $O(K)$ where K is the total number of points in the feature space which is significantly larger than M.

4 Experimental results

Experimental setup Results here are produced on MATLAB software installed on desktop machine having Intel core i7 processor with 4 GB RAM. The results are organized in 4 set of two-dimensional feature space data set named WELL SEPERATED DATASET (WSD) (having well-separated, uniform-density clusters), NON-CONVEX DATASET (NCD) (having uniform density, non-convex clusters), VARIABLE DENSITY DATASET (VDD) (having well-separated, non-uniform density clusters) and NESTED DATASET (ND) (having nested clusters). They are shown in Fig. 10a–d, respectively, with following organization: In each figure, the leftmost tile shows the feature space having clusters followed by tiles showing result of application of FCM, DBSCAN, EnDBSCAN and VBCD. A cluster in results is marked by a single color. Note that in the result of application of VBCD, the cluster is shown with a background color. The background color represents the extent of connectedness of feature points thresholded at automatically detected vicinity threshold.

Table 1 Details of FCM

Data set	*N*	Time	Results
WSD	4	0.166	Good
NCD	2	0.285	Unacceptable
VDD	4	0.271	Unacceptable
ND	2	0.173	Unacceptable

N no. of clusters desired

Table 2 Details of DBSCAN

Data set	*Eps*	*MinPts*	Time	Results
WSD	11.909	10	0.329	Good
NCD	8.97	5	0.681	Good
VDD	29.574	10	0.204	Unacceptable
ND	10.3515	10	0.345	Unacceptable

Table 3 Details of EnDBSCAN

Data set	*Geps*	*MinPts*	*Alpha*	Time	Results
WSD	30	20	2	0.225	Good
NCD	30	15	2	0.574	Good
VDD	20	2	2	0.193	Good
ND	30	15	1	0.152	Unacceptable

Table 4 Details of VBCD

Data set	*W*	λ	T_1	T_2	Results
WSD	11	0.04	0.15	1.58	Good
NCD	11	0.04	0.521	2.822	Good
VDD	13	0.04	0.199	1.439	Good
ND	15	0.1, 1.8	0.162	1.03	Good

T_1 is the processing time without window size and threshold detection and T_2 is with it

However, in data set ND, since there are two thresholds, such a representation is not shown.

Tables 1, 2, 3 and 4 summarizes the details of tunable parameters used, processing times (in seconds) and the acceptability of results for algorithms FCM, DBSCAN, EnDBSCAN and VBCD.

The corresponding threshold curves are shown in the following figures.

In feature space WSD, well-separated, uniform density clusters are present. All algorithms produce the desired results. However, with or without automatic detection of window size and vicinity threshold, VBCD is the fastest. For feature space NCD, all algorithms accept FCM produces correct results. This is expected because FCM is not a suitable algorithm for non-convex cluster detection. Similarly for variable density clusters, DBSCAN, EnDBSCAN and VBCD produce the correct results, VBCD being the fastest. For the typical case of ND only VBCD tends to produce correct results. For EnDBSCAN, if small clusters and feature points at boundary between two clusters are ignored, it produces correct results otherwise one can notice several small clusters there. The threshold curve shown for feature space of VDD in Fig. 11c is different from what is shown in Fig. 8. This is because in Fig. 8, there was no automatic detection of the window size. It is to be noted that automatic detection of tunable parameters in algorithms other than VBCD is also an open area of research. Results in this paper are produced using those tunable parameters with which the output matches the desired intuitive clustering. Recently, for DBSCAN, in [32] automation for tunable parameters of DBSCAN was done using differential evolution techniques producing good results. Our experiments reveal that VBCD performs best out of the tested algorithms.

4.1 Evaluation of results using internal cluster validity indices

Internal evaluation of clustering is done by considering the results of clustering on a particular data set alone. No prior classification is necessary. There are various internal cluster validity indices in the literature. Popular ones are Dunn index and Silhouette index. We first discuss the mathematical formulation of these indices, discuss the problems in them and then propose the new SDI.

Dunn Index Let the result of clustering a data set is such that there are m clusters $C_1, C_2, C_3, \ldots, C_m$. The Dunn index is then defined as

$$DI_m = \frac{\min_{1 \le i < j \le m} \delta(C_i, C_j)}{\max_{1 \le k \le m} \Delta_k}$$

where $\delta(C_i, C_j)$ is inter-cluster distance metric and Δ_k is intra-cluster distance metric. A high Dunn index usually indicates better clustering. However, as the data size grows larger, the calculation of Dunn index becomes computationally complex. One of the drawbacks of using this index is the denominator term containing Δ_k. This intra-cluster distance metric (frequently called as diameter of the cluster) can be defined in many ways. In Euclidian space, it can be the distance between two most distant points in the cluster or average of all pair wise distances or distance of each data point from cluster centroid, etc. This creates a problem when the clusters are either nested or non-convex. For example, in feature space NCD, both the clusters have same centroid approximately. Even if one calculates the diameter, since the cluster is non-convex, the diameter will

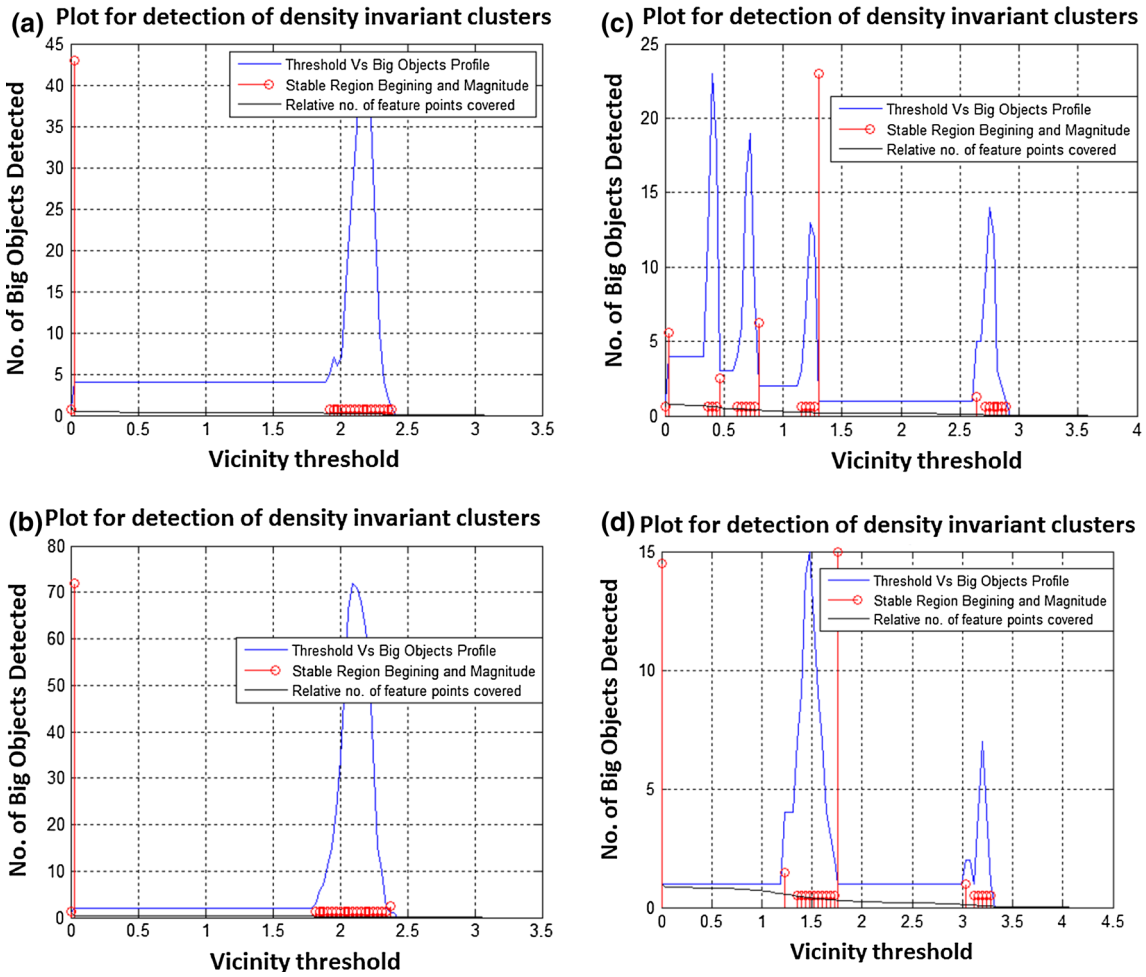


Fig. 11 Threshold plot for feature space of data set **a** WSD, **b** NCD, **c** VDD, **d** ND

pass through non-cluster region also and will be unnecessarily large.

Silhouette Index For clusters $C_1, C_2, C_3, \ldots C_m$ of a particular data set, the silhouette value for ith data point is defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where $a(i)$ is the average dissimilarity measure of the ith data point to other points in the same cluster and $b(i)$ is the minimum dissimilarity measure of the ith data point to all the other clusters. Obviously $-1 \leq s(i) \leq 1$ and it is a measure of how well ith data point is assigned to a particular cluster. $s(i)$ close to 1 means that the ith data point is properly clustered. A value of -1 indicates that it could have been assigned to the neighboring cluster and a value of 0 means that the point is on the boundary of two clusters. The mean value of $s(i)$ is taken as a measure of tightness of the grouped data. This index also suffers the problem of 'bad indication' when the clusters are nested or non-convex. We now show two cases of failure of Silhouette index in Fig. 17.

In Fig. 12, results of Silhouette index evaluation are plotted for feature space of NCD for FCM and VBCD. On the X-axis, we have silhouette values and on Y-axis, we have clusters and their corresponding elements. It is known beforehand that results on FCM are unacceptable and that of VBCD are correct. But silhouette plots show otherwise. As seen from FCM results in Fig. 12, there are negligible values in the range ≤ 0 for $s(i)$ values but for VBCD, there are plenty values in the negative range. The average silhouette values for FCM and VBCD are 0.515 and 0.041 which implies, as per silhouette, FCM produces better clustering than VBCD. But the reality is just opposite. This happened due to non-convex nature of clusters in NCD and partial nesting.

Proposed Space Density Index (*SDI*) The concept behind SDI is that it demands clusters in the clustered results to possess 'good separation' in either 'space' or 'densities' or a combination of both. Separation in space means that the minimum spacing between two arbitrary clusters should be large enough. Same thing applies to separation in density. In

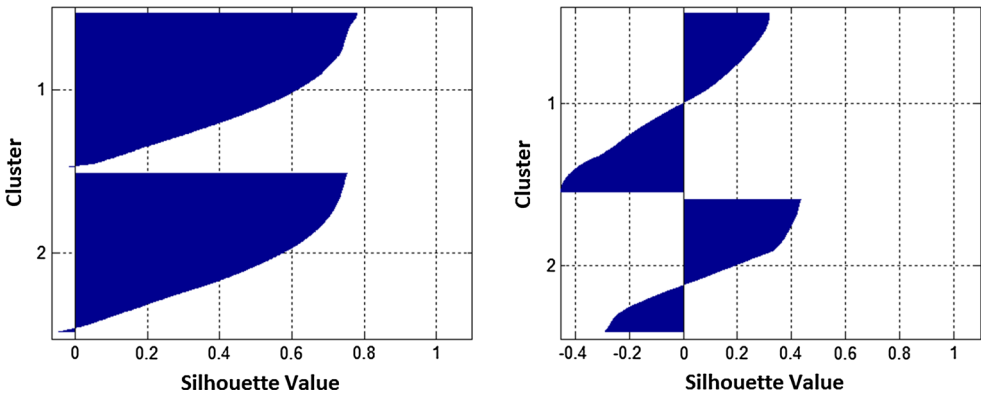


Fig. 12 Silhouette plots for feature space of NCD. First tile is for FCM and second is for VBCD

Table 5 Comparative performance evaluation of Silhouette index and SDI

D. set	Silhouette Index				Space-Density Index			
	FCM	DBSCAN	EnDBSCAN	VBCD	FCM	DBSCAN	EnDBSCAN	VBCD
WSD	0.84	0.84	0.84	0.84	44.52	44.52	44.52	44.52
	All algos. perform same				All algos. perform same			
NCD	0.51	0.04	0.04	0.04	3	36.87	36.87	36.87
	FCM > DBSCAN = EnDBSCAN = VBCD				FCM < DBSCAN = EnDBSCAN = VBCD			
VDD	0.5	0.77	0.77	0.77	6.04	29.42	29.42	29.42
	FCM < DBSCAN = EnDBSCAN = VBCD				FCM < DBSCAN = EnDBSCAN = VBCD			
ND	0.47	N.A.	−0.66	0.01	3	N.A.	3.67	9.316
	FCM > VBCD > EnDBSCAN				FCM < EnDBSCAN < VBCD			

N.A. indicates only one cluster detected

the light of above, let's revisit feature spaces of Fig. 10a–d. In data set WSD, before clustering, it is apparent that there are four same-density, well-separated clusters. After applying FCM however, the results do not match up to the expectations. We observe two clusters having same density and no separation. Results of VBCD however are as expected. So, VBCD should possess a high SDI value than FCM. Similarly, in data set ND, due to nesting of clusters, spatial separation is very low but separation in densities is very good. For NCD and VDD, clusters appear to be separated in density as well as space. We therefore propose the following definition for SDI.

Definition 8 (*Space Density Index*) Let a clustering algorithm produces m clusters $C_1, C_2, C_3, \ldots, C_m$. Let $\delta_{\min}(C_i, C_j)$ be the minimum distance between cluster C_i and C_j. Let $\Omega(C_i)$ denote the average density of cluster C_i. Then the SDI is defined as follows

$$\mu(i) = \min_j \left(\sqrt{\delta_{\min}(C_i, C_j)^2 + \{\Omega(C_i) - \Omega(C_i)\}^2} \right)$$

$$\text{SDI} = \text{median}_{i=1}^{m} \{\mu(i)\}$$

$\mu(i)$ is the minimum space-density separation of cluster C_i with all the other clusters. Above definition of SDI is the worst-case indicator due to $\min_j(\cdot)$ term. However, to get the average case indicator, one should use $\text{avg}_j(\cdot)$ instead. Obviously $0 \leq \text{SDI} \leq \infty$. We now present comparative results of application of Silhouette index and SDI on the feature spaces of Fig. 10a–d (refer to Table 5).

It is clear form Table 5 that the performance evaluation is better when SDI is used instead of Silhouette index. In case when only one cluster is detected by a particular algorithm, internal evaluation becomes unnecessary. Such cases are marked with N.A. in Table 5.

4.2 External evaluation

To validate the method using external evaluation, we use 'KEEL' (Knowledge Extraction based on Evolutionary Learning) [33] which is an open-source data set repository. 'Banana' standard classification data set is used (An artificial data set where instances belong to several clusters with a banana shape). The validation procedure was suggested in [34]. It has two attributes, two classes (marked by experts of the field) and no missing values. Following are the details of attributes

Attribute 1 is real within range [−3.09, 2.81]
Attribute 2 is real within range [−2.39, 3.19]

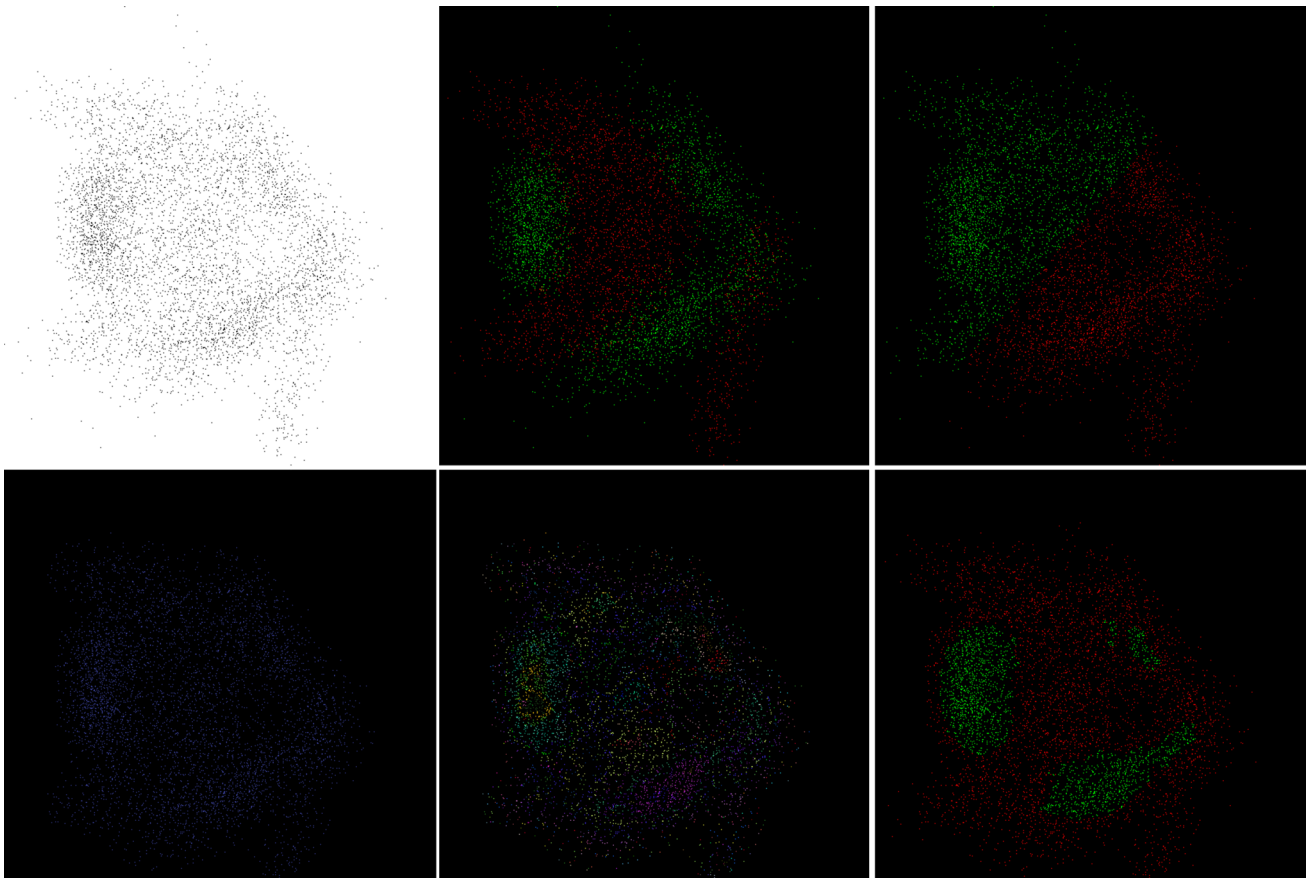


Fig. 13 Tile 1—feature space constructed from the 'Banana Dataset' obtained from KEEL data set repository. Tile 2—the desired classification (manually made). Tile 3, 4, 5, 6 are the results of application of FCM, DBSCAN, EnDBSCAN and VBCD. Clearly VBCD produces results close to the desired. Note that one color in the results indicates various clusters identified at same density scale (color figure online)

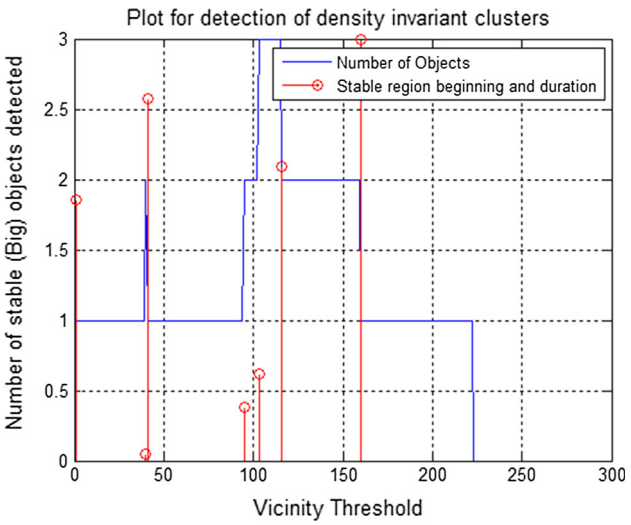


Fig. 14 Object versus vicinity threshold curve for Fig. 13. The results for VBCD in Fig. 12 are generated by using Thresholds 116 and 5. Because threshold at 42 shows some nested cluster and threshold at 160 is capable of identifying further high-density cluster which is not required as the already available classification data has clusters at only two densities

A feature space is constructed from the obtained data (as shown in Fig. 13, tile 1), and the results are then discussed.

Threshold curve is shown for feature space of Fig. 13 in Fig. 14.

The feature space in Fig. 13, tile 1 contains the feature points from the said database. The feature space so obtained has variable density. It is also not convex. The desired result, which is crafted by experts, is shown in tile 2. Same color in the result indicates clusters found at one density scale (with possibly some tolerance). The order of percentage match of results of various algorithms shown is

$$\text{VBCD} > \text{EnDBSCAN} > \text{DBSCAN} > \text{FCM}$$

4.3 Admissibility criteria analysis

Fisher and Van Ness [35] analyzed clustering algorithms with the objective of comparing them and providing guidance in choosing clustering procedure. They defined a set of admissibility criteria for clustering algorithms that test the sensitivity of clustering algorithms with respect to the changes that do not alter the essential structure of the

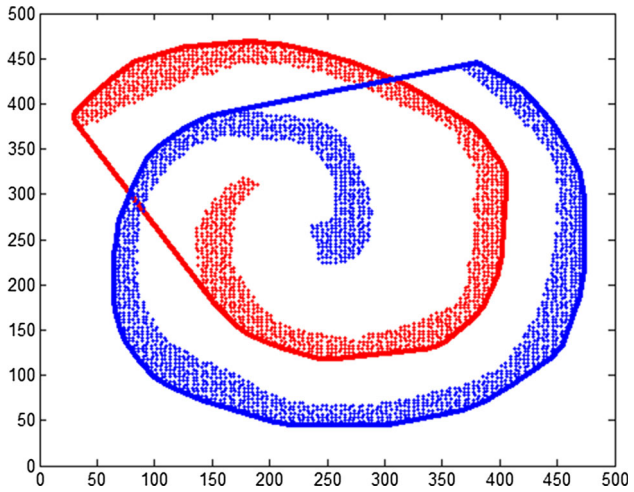


Fig. 15 Two clusters in two-dimensional feature space with their convex hulls (shown in *dark lines*) intersecting each other (color figure online)

data. A clustering is called 'A' admissible if it satisfies criterion 'A.' Similarly, Kleinberg [36] addressed a similar problem, where he defined three criteria: Scale invariance, richness and consistency. He showed that it is impossible to construct an algorithm that satisfies all these properties simultaneously. However, any two of them can be satisfied simultaneously.

The proposed work is

Not convex admissible Convex hulls of clusters can intersect for, e.g., in case of nested clusters. This is a desirable property. However, if convex hulls of clusters do not intersect, we call the algorithm convex admissible. Refer to Fig. 15 for two spiral clusters which are well separated (Red and Blue). But their convex hulls intersect. The proposed method separates the two clusters; however, FCM is convex admissible as it does not separate the two clusters.

Cluster proportion/omission admissible On replicating some of the clusters and/or deleting some of them, remaining cluster's boundaries do not get altered. This property is referred to as cluster proportion admissibility (for replicating) and cluster omission admissibility (for deletion of cluster). This property is also desirable and is possessed by the proposed work. The reason behind satisfaction of this property is vicinity values are calculated from a neighborhood instead of considering all data points. Even if we consider infinite neighborhood size, the effect of deletion of clusters (which are after a certain distance) is negligible. Refer to Fig. 16 for an example.

Scale covariance, consistency and richness It can be easily seen that the algorithm is scale covariant (invariance is a frequently used term in literature but it is more appropriate to use covariance), rotation invariant and mostly robust to affine changes. To prove this let us assume the affine transformation equation to be

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix}$$

which maps points (x, y) to points (x', y'). The distance between two points (x_1, y_1) and (x_2, y_2) before transformation is $d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$ and let the position and distance between those points after transformation is—(x'_1, y'_1), (x'_2, y'_2) and $d' = \sqrt{(x'_1 - x'_2)^2 + (y'_1 - y'_2)^2}$. If we choose the affine transform matrix as $\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} S & 0 \\ 0 & S \end{bmatrix}$ where S represents uniform scaling parameter, it can be easily shown that $d' = S \times d$ (independent of translation parameters e and f). When $S = 1$, only translation is present. Since the result is independent of translation parameters, so is the method. Similarly, if we choose $\begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ we can show that $d' = d$ (independent of translation parameters e and f). Now it is evident from Eq. 1 that vicinity values depend on distance between two points. So, in case of rotation as $d' = d$, we can say that VBCD is independent of rotating the feature space. In case of uniform scaling, relative vicinity values remain the same as $d' = S \times d$ except for scaling of vicinity threshold and window size.

An example of affine robustness is shown in Fig. 17. No simple formulation for affine distortions in general can be derived as above. Still the algorithm is robust to affine changes.

In Fig. 17, as compared to the feature space of Fig. 16, the second cluster is shear mapped. Still the results remain unaltered. Similarly the algorithm is consistent. The clustering results are unaffected by shrinking within cluster distances and increasing between-cluster distances. The algorithm is, however, not rich as it does not produce all possible partitions of the input data set. This is also desired for clustering applications as all possible partitions are of little use.

5 Discussion, conclusion and future scope

Clustering is a subjective field. Two individuals or applications might differ on the definition of cluster. This is the reason why there are many clustering algorithms. Due to this fact, a universal judgment criteria is also absent for the clustered results. However, all clustering methods agree that a cluster is essentially that region in feature space where there is a high density of features. There are many

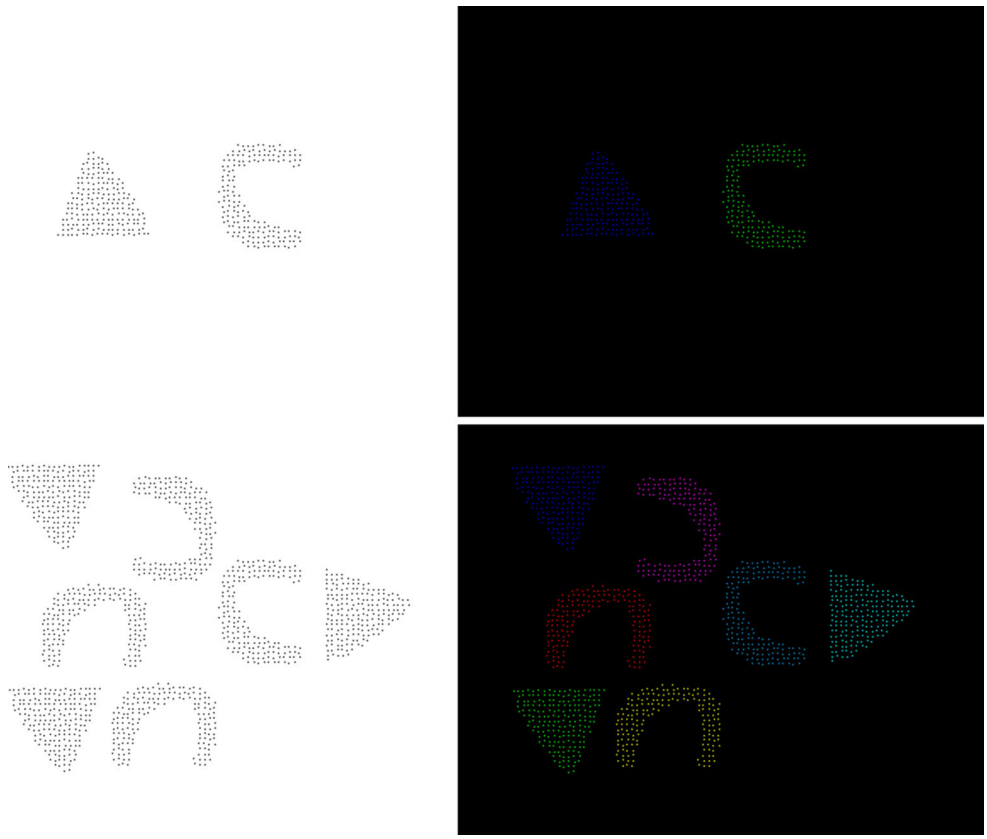


Fig. 16 Cluster proportion/omission admissibility. As seen, on replicating cluster arbitrary number of times or deleting some of them, the individual cluster boundaries are unaffected

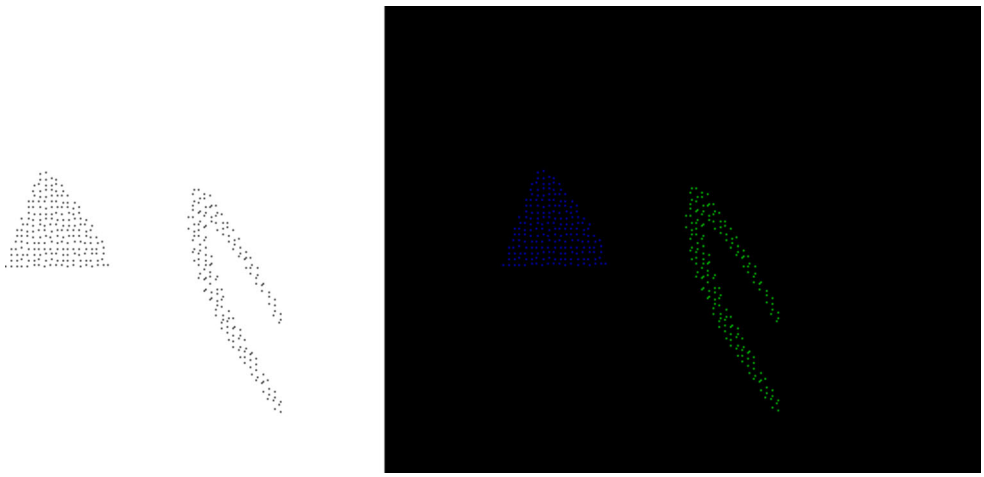


Fig. 17 Shear mapped clusters

approaches to discover natural groupings as discussed earlier. They have ad hoc goals to be accomplished—exhaustive partitioning of data set, identification of severely dense regions, computational complexity, density invariance, etc. Some approaches like FCM force the data to be clustered irrespective of any clustering tendency and some approaches let the data points to be free from belonging to any cluster by calling them noise points.

Through this paper a new approach to detect natural groupings in data is presented. The approach works on regular as well as typical cases of non-convex and nested or partially nested clusters on which previous approaches didn't give desired results. Technique for automatic detection of various tunable parameters is also suggested. New internal cluster validity index is also proposed which works well as shown in Sect. 4. The time complexity is also low if the technique is used without automatic threshold detection, otherwise it is comparable to previous methods.

Obviously when the data size and dimensionality are increased, density-based methods become computationally complex. In VBCD technique, however, one additional

constraint is present construction of feature space, which requires large blocks of contiguous memory locations for sparse data. Big data handling techniques can be used to improve the method. Also density invariance requires the algorithm to run of all possible parameter values which is the topic of research for reduction in time complexity.

References

1. Jain AK, Dubes RC (1988) Algorithms for clustering data. Prentice Hall, Englewood Cliffs
2. Goshtasby AA (2012) Image registration: principles, tools and methods. Springer, Berlin
3. Jain AK (2010) Data clustering: 50 years beyond K-means. Pattern Recognit Lett 31(8):651–666
4. Ester M, Kriegel PH, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of 2nd KDD. AAAI Press
5. Roy S, Bhattacharyya DK (2005) An approach to find embedded clusters using density based techniques. In: ICDCIT 2005, LNCS 3816. Springer, Berlin, pp 523–535
6. Dunn JC (1973) A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. J Cybern 3(3):32–57
7. Davies DL, Bouldin DW (1979) A cluster separation measure. IEEE Trans Pattern Anal Mach Intell 1(2):224–227
8. Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Comput Appl Math 20:53–65
9. Chen J, Pi D (2012) A cluster validity index for fuzzy clustering based on non-distance. In: 2013 fifth international conference on computational and information sciences (ICCIS). IEEE
10. Kirkland O, De La Iglesia B (2013) Experimental evaluation of cluster quality measures. In: 2013 13th UK workshop on computational intelligence (UKCI). IEEE
11. Lu Z-M, Feng J-M, Fan D-M, Yang P, Tian Y (2014) Novel partitional clustering algorithm for large data processing. Syst Eng Electron 36(5):1010–1015
12. de Lope J, Maraval D (2013) Data clustering using a linear cellular automata-based algorithm. Neurocomputing 114:86–91
13. Liu AH, Poon LKM, Liu T-F, Zhang NL (2014) Latent tree models for rounding in spectral clustering. Neurocomputing 144:448–462
14. Kuo RJ, Huang YD, Lin C-C, Wud Y-H, Zulvia FE (2014) Automatic kernel clustering with bee colony optimization algorithm. Inf Sci 283:107–122
15. Liu X, Li M (2014) Integrated constraint based clustering algorithm for high dimensional data. Neurocomputing 142(22):478–485
16. Kirkland O, De La Iglesia B (2013) Experimental evaluation of cluster quality measures. In: 13th UK workshop on computational intelligence, UKCI
17. Meyer CD, Wessell CD (2012) Stochastic data clustering. SIAM J Matrix Anal Appl 33(4):1214–1236
18. Ferrari DG, de Castro LN (2012) Clustering algorithm recommendation: a meta-learning approach. In: SEMCCO 2012. Springer, Berlin
19. Cetişli B, Edizkan R (2015) Use of wavelet-based two-dimensional scaling moments and structural features in cascade neuro-fuzzy classifiers for handwritten digit recognition. Neural Comput Appl 26(3):613–624
20. Mahdipour H-A, Khademi M, Sadoghi HY (2012) Model-based fuzzy c-shells clustering. Neural Comput Appl 21(1):29–41
21. Niu B, Duan Q, Liang J (2013) Hybrid bacterial foraging algorithm for data clustering. In: Intelligent data engineering and automated learning—IDEAL 2013. Springer, Berlin, pp 577–584
22. Altameem T, Zanaty EA, Tolba A (2014) A new fuzzy C-means method for magnetic resonance image brain segmentation. Connect Sci 1:1–17. doi:10.1080/09540091.2014.970126
23. Wu S, Quan M, Feng X (2012) Spectral clustering algorithm based on local sparse representation. In: Intelligent data engineering and automated learning—IDEAL 2013. Springer, Berlin, pp 628–635
24. Wang Y, Chen L, Mei J-P (2014) Incremental fuzzy clustering with multiple medoids for large data. IEEE Trans Fuzzy Syst 22(6):1557–1568
25. Abdelghaffar NM, Lotfy HMS, Khamis SM (2014) A multi-agent-based approach for fuzzy clustering of large image data. J Real-Time Image Process. doi:10.1007/s11554-014-0473-3
26. Peng H et al (2015) An automatic clustering algorithm inspired by membrane computing. Pattern Recognit Lett 68:34–40
27. Stetco A, Zeng X-J, Keane J (2015) Fuzzy C-means++: fuzzy C-means with effective seeding initialization. Expert Syst Appl 42(21):7541–7548. doi:10.1016/j.eswa.2015.05.014
28. Babaeian A et al (2015) Nonlinear subspace clustering using curvature constrained distances. Pattern Recognit Lett 68:118–125
29. He R et al (2015) Robust subspace clustering with complex noise. IEEE Trans Image Process 24(11):4001–4013
30. İnkaya T (2015) A parameter-free similarity graph for spectral clustering. Expert Syst Appl 42(24):9489–9498
31. Ankerst M, Breuing MM, Kriegel HP, Sander J (1999) OPTICS: ordering points to identify the clustering structure. In: ACM-SIGMOD'99, pp 49–60
32. Karami A, Johansson R (2014) Choosing DBSCAN parameters automatically using differential evolution. Int J Comput Appl 91(7):1–11
33. Alcalá-Fdez J, Fernandez A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Multiple-Valued Logic Soft Comput 17(2–3):255–287
34. Moreno-Torres JG, Sáez JA, Herrera F (2012) Study on the impact of partition-induced dataset shift on k-fold cross-validation. IEEE Trans Neural Netw Learn Syst 23(8):1304–1313
35. Fisher L, VanNess J (1971) Admissible clustering procedures. Biometrika 58(1):91
36. Kleinberg J (2002) An impossibility theorem for clustering. NIPS 15:463–470