Prog Artif Intell (2015) 4:21–36
DOI 10.1007/s13748-015-0066-9

REGULAR PAPER

RADDACL2: a recursive approach to discovering density clusters

Daniel Avila[1] · **Iren Valova**[1]

Received: 29 May 2015 / Accepted: 13 November 2015 / Published online: 28 November 2015

Abstract Discovering connected regions in data space is a complex problem that is extremely demanding on the user. Datasets often require preprocessing and postprocessing before they are fit for algorithm and user consumption. Existing clustering algorithms require performance parameters to achieve adequate results. Typically, these parameters are either empirically optimized or scanned using brute force, which ultimately adds additional burden to the user. We present RADDACL2, a density-based clustering algorithm, with the intent of reducing overall user burden. The algorithm requires no information other than the dataset to identify clusters. In addition, the algorithm is deterministic, meaning the results will always be the same. Both of these features reduce user burden by decreasing the number of passes one must make to get an outcome. A number of experiments are performed using toy and real datasets to verify the capabilities of RADDACL2 as compared to existing algorithms.

Keywords Clustering · Discovering connected regions in data space · Density · Non-linearly separable · Centroid · Segmentation · Deterministic · Neighborhood · Precluster-ing · Statistics

List of symbols

Centroid [C_X]	The centroid of the current region. The value of X associates the centroid with a specific region
Observation [O]	A series of attributes representing a single observation in the dataset
Region [R_X]	A region represents a subdivision of the dataset. These are created during Density Discovery
Precluster [PC]	A set of related observations that contain a very density. They are identified during Density Discovery
Average Absolute Deviation [AAD]	An average of the absolute deviations from a measure ofcentral tendency
Standard Deviation [STD]	The variability of data from a measure of central tendency
Neighborhood Function [NF]	An algorithm that is used to identify if a pair of preclusters are neighbors
Neighborhood Radius [NR]	A threshold assigned to each precluster. These are used by the neighborhood function to determine if two preclusters should be merged

1 Introduction

Clustering algorithms are an unsupervised learning technique, and represent statistical pattern recognition. The primary purpose of a clustering algorithm is to identify relationships within a dataset of observations [1]. Clustering is often used to classify data with no a priori labels or structure. Clustering approaches are ubiquitous in nature, and used in a variety of fields [2]. Typically, clustering approaches are used in the broad field of data sciences: pattern recognition [3–5], machine and statistical learning [6–8], and data mining [9,10], but are applied in specific fields that range from image processing [11,12], to surveys and questionnaires [13,14], to genetics [15–17].

✉ Iren Valova
ivalova@umassd.edu

[1] Computer and Information Science, University of Massachusetts Dartmouth, Dartmouth, MA, USA

There exist several broad categories of clustering algorithms [18] of which hierarchical vector representation, density, and, in general, partitional methods are most relevant to this paper. Each uses different criteria and assumptions to group observations into clusters or categories. Furthermore, with respect to the observations, all methods compute either soft (e.g., fuzzy) or hard clustering. Soft clustering often uses a probabilistic model to assign observations to groups based on likelihood [18]. More relevant to this paper, and more often used, is hard clustering, where observations map to only one group.

In this paper we introduce RADDACL2, a hybrid approach between partitional vector representation and density-based clustering algorithms. RADDACL2 is a more reliable extension of the RADDACL that introduces a number of usability and performance improvements over its predecessor [19]. The primary goal of RADDACL2 is to reduce the complexity of implementation for the user, while still achieving adequate results on a variety of real datasets. In order to achieve this goal, RADDACL2 uses no performance parameters when generating clusters from a dataset. The only input is the dataset itself. The algorithm calculates all necessary thresholds based on the data provided to it. Second, the results of the algorithm are deterministic. Therefore, the algorithm only has to be run once per dataset. Briefly, RADDACL2 first performs density discovery stage, identifying areas of high density, called preclusters, while separating out outliers into noise. RADDACL2 then invokes a rebuild function, to merge preclusters into their final connected regions. The paper is organized as follows: Sect. 1 provides an overview of particular forms within the taxonomy of clustering approaches, Sect. 2 presents a review of clustering methods and state-of-the-art algorithms that are related to the concept of RADDACL2; Sect. 3 describes RADDACL2 algorithm in detail. Section 4 illustrates RADDACL2's performance via several experiments. Finally, Sect. 5 presents the conclusion and evaluation, future directions, and limitations of RADDACL2.

2 Review of related clustering methods

Hierarchical clustering is performed in one of two ways: agglomerative, which joins observations in an iterative fashion until all items have been paired, or divisive, which splits observations into groups [20]. Hierarchical clustering algorithms differ from the proposed algorithm in that they generate a dendrogram rather than an actual clustering. The dendrogram provides a visualization representing a holistic view of dissimilarity among observations. One advantage of hierarchical clustering is that it is deterministic: algorithms only need to be run once and will produce the same results. The main disadvantage of hierarchical clustering is the complexity of the results. Extracting clusters from a dendrogram can be difficult [21] especially when the dimensionality of the dataset is high. In addition, results are affected by outliers, and are, at times, subjective (i.e., the threshold for cutting a dendrogram). Implementations [22] include single-linkage, complete-linkage, and average-linkage. In recent years, there have been numerous advancements on hierarchical clustering [23,24].

Vector-representation methods (referred to as centroid methods by [18]) partition the observations into groups, in which a group is represented by a specific vector. These approaches vary considerably and include classical methods such as KMeans [4,25] (and derivatives [26–29]), follow-the-leader approaches [30], self-organizing maps [31,32], and vector quantization [33]. Clustering is complete when every point in the original dataset has been assigned to a central observation, that is, a representative vector. Usually, this occurs through some convergence criteria. There are several drawbacks of vector-representation (and similar) methods. First, they cannot handle non-linearly separable data that many real datasets contain. Second, they require pre- and/or postprocessing to produce user-consumable results. Finally, these methods often require users to decide, a priori, how many vectors will be used to represent the data. These performance parameters force the user to either preprocess the data to determine the ideal initial parameters or run the algorithm a multitude of times to identify the best set of results. In either case, the user is burdened with additional work unrelated to analyzing the results. Most of these algorithms are also non-deterministic, so multiple runs with the same parameters may be required to find an ideal clustering.

Distribution clustering algorithms assume clusters in the dataset can be modeled by a number of known distributions. The dataset is fitted against the provided distribution(s), which are optimized iteratively until the results converge to the local optimum. If improperly tuned, the algorithm is likely to suffer from overfitting, which will manifest itself as merging multiple clusters into a single, large cluster. Like partition-based algorithms, the results are non-deterministic, so multiple runs are required for ideal results. The greatest challenge with distribution clustering lies in its assumption that the observations in a dataset map to a distribution, which is often not the case for real datasets. These algorithms tend to be more challenging to implement, since the user must decide on the distribution of the data. Often there exists a more complex model that would better describe a particular dataset. A common implementation of distribution clustering is Guassian Mixture Models [34], which is a form of expectation-maximization clustering.

Density-based vector-representation clustering algorithms both partition space, but under different criteria [1,2,18]. Essentially, cluster boundaries are determined by near-

contiguous distributions of points. A cluster becomes a chain of observations and the borders of that cluster are created by drops in observation density. Density algorithms can identify clusters that form unique shapes, including non-linearly separable clusters. One known problem with existing architectures is that they require a number of performance tuning parameters to achieve results. For example, DBSCAN [20], a common variation, requires users to set a minimum cluster size, and that the observations in a cluster must be within a certain distance from another observation in that cluster. Different algorithms have addressed this problem in various ways. Identifying the appropriate values for these performance parameters imposes an additional burden on algorithm usability. Perhaps the greatest challenge for density clustering is the identification of borders between clusters that are adjacent to one another. Density algorithms expect a sharp decrease in density around the edges of a cluster. Borders become difficult to identify when clusters are adjacent or overlapping within the dataset. Several variants of DBSCAN, such GDBSCAN [35], OPTICS [36], and DeLiClu [37], solve some of the problems discussed above. Furthermore, density-based algorithms have been of particular interest in recent years [38,39], especially as databases become larger and, at times, more sparse.

To note, spectral clustering techniques [40] represent a method in which eigenspaces are used to begin the clustering process. However, a number of the aforementioned methods are also used within spectral techniques in order to partition the eigenspace.

3 RADDACL2

RADDACL [19] (and RADDACL2) fall under several of the reviewed broad clustering categories. RADDACL recursively partitions space based on density criteria. These partitions are called preclusters, which are represented, essentially, by a single vector (i.e., as would be found in KMeans). Next is a rebuilding step to correctly re-classify neighboring preclusters (in which the idea of a neighborhood function is borrowed from self-organizing maps [31]). RADDACL2 performs these steps in (a) a deterministic fashion and (b) with no performance parameters that require tuning. These two features combined, drastically reduce the number of runs required to produce a clustering. RADDACL, however, executes these steps in two separate functions and heavily depends on a threshold parameter to facilitate the recombination of the isolated dense regions. RADDACL was initially conceived as a teaching tool to present three difficult concepts: recursion, clustering, and density-based data processing. While the concept delivered excellent results, limitations in the form of custom dataset parameters and user-determined need for building of clusters kept it from being fully utilized beyond the classroom. Here we present RADDACL2 as a complete remodel of its predecessor. We have utilized RADDACL as a concept, but developed RADDACL2 as a full-fleshed clustering algorithm, which eliminates any parameter and threshold dependencies. The two fundamental steps—recursive discovery of dense regions and combining these into final clusters—are realized in two stages. The two phases of the algorithm, i.e., recursive density discovery and rebuild, are working in tandem by first isolating preclusters by subdividing the dataset in recursive fashion. The recursion stops when the average distance from center of cluster is reached. This parameter is calculated from the dataset based on the subdivided regions and the average deviation from the forming precluster centroids. Once the subdivision stops and all preclusters are identified, the rebuilding phase begins. This process relies on chaining preclusters in the formation of the final clusters. The basis here is the neighborhood function. While the neighborhood threshold depends on the neighborhood threshold, this parameter is calculated from the dataset being clustered, meaning no user involvement. Here, we present the full details of RADDACL2, a robust implementation of its predecessor's concept.

3.1 Phase 1: recursive density discovery

The first phase of RADDACL2 is responsible for identifying preclusters. Preclusters are obtained by recursively dividing the dataset into smaller groups of datapoints around a measure of central tendency. Figure 1 demonstrates a single iteration of the division process. The division process stops when the density of the remaining points exceeds the density threshold. That group of points becomes a precluster for the following phase. Figure 2 shows the results of the preclustering phase for two of the toy datasets used in experiments. The phase completes when all datapoints have been assigned to a precluster. Outliers manifest in this phase as preclusters with a small observation count.

In this phase, the initial centroid C_I is calculated for the dataset as well as a pairwise distance matrix for the observations in the dataset. We initialize the current region R_I as the entire dataset. The next step is to split R_I into R_N regions around C_I. The first region, R_1, contains all observations with attributes less than C_I. The second region, R_2, contains observations with first attribute greater than C_I, and the remaining attributes are less than C_I. This process repeats until all datapoints in R_I have been assigned to a region. Then, for each new region R_X, we set $R_X = R_I$ and repeat the process. A region becomes a precluster once its density falls below the precluster threshold T. This is calculated from R_I using absolution average deviation (AAD) from the centroid. When eligible, the Region R_I is not split, and we continue to explore

the remaining regions until all remaining regions are preclusters. The AAD is calculated as follows:

AAD $= \sum |x - \text{AVE}| / n$ for each x distance from centroid in the dataset of size n

where AVE is average distance to the centroid. While this metric is similar to standard deviation, it allows for parameter/independent preclustering.

Figure 3 presents the pseudocode describing the algorithm for phase 1. The initial value of R_I is the entire dataset.

It should be noted that the precluster stopping condition is calculated as AAD/AVE. If the improvement in the process is minimal (25 % defined as the universal threshold), then the preclustering process is terminated.

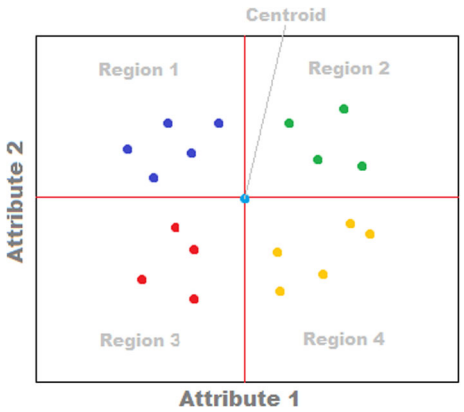

Fig. 1 Dividing the dataset to discover preclusters

3.2 Phase 2: rebuild

The second phase, called rebuild, is responsible for generating the final set of clusters. Each precluster identified in the previous phase is assigned to a cluster. Pairwise comparisons for the set of preclusters are made using a neighborhood function. When two preclusters are considered neighbors, they will be assigned to the same cluster. A visualization of the neighborhood function is shown in Fig. 4. The result is a series of chained preclusters where each chain represents a cluster.

The neighborhood function works as follows: we identify two preclusters PC_1 and PC_2. For each pair of datapoints between PC_1 and PC_2, we check to see if the distance between the pair is less than the neighborhood threshold NT. If it is, then we mark these two clusters as belonging to each other. Each unassigned precluster that is processed becomes a new cluster in the result set.

The pseudocode for the rebuild is provided in Fig. 5 and the one for the neighborhood function is presented in Fig. 6. The DP refers to datapoints in the preclusters.

4 Experiments

As with presenting every novel algorithm, comparative analysis is of crucial importance. We have utilized the ELKI framework to implement RADDACL2. ELKI [41] is a Java-based framework containing implementations of various

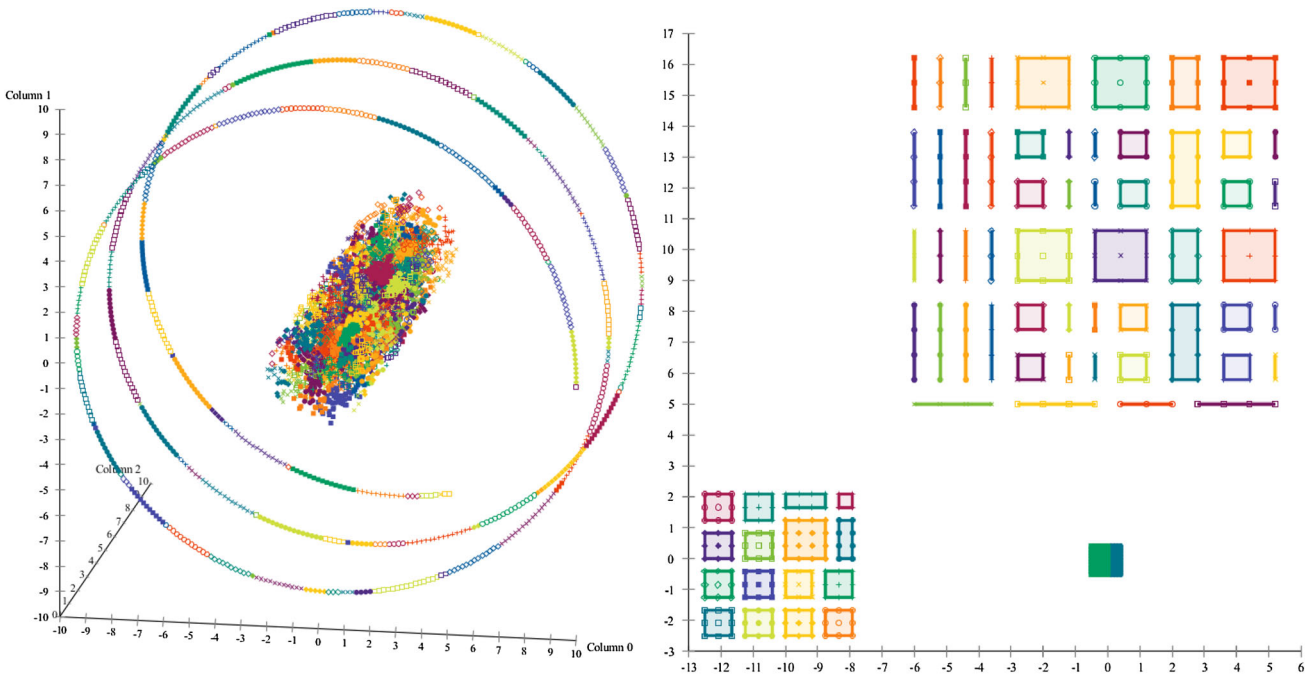

Fig. 2 Precluster pairs from two toy datasets

Fig. 3 Phase 1 algorithm pseudocode

```
1. Calculate C_I for R_I
2. Calculate AAD for R_I
3. Calculate STD for R_I
4. If RI exceeds density threshold
     a. Store R_I as a precluster
5. Else
     a. For each O_I in R_I
          i. Assign O_I to R_N, where N is an index based on O's
             location relative to C_I
     b. For each R_X generated in the previous step
Repeat steps 1-5 where R_I = R_X
```

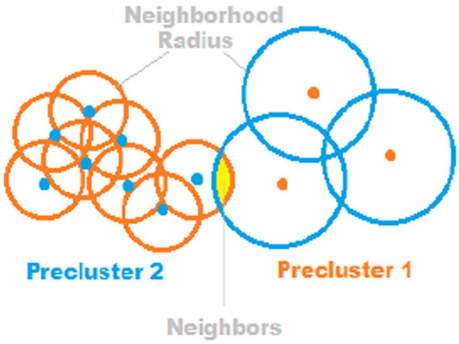

Fig. 4 Shows two neighboring clusters, identified by overlapping densities

clustering and outlier detecting algorithms. These implementations are designed for comparative analysis within ELKI; they remain true to their peer-reviewed publications. Common tasks such as dataset indexing and parsing are provided out of the box.

Results are gathered from a number of toy and real datasets using two other related clustering algorithms—KMeans and DeLiClu. KMeans belongs to the class of partitioning algorithms and, in spite of its many drawbacks, it considered something of a classic. DeLiClu is the latest and, arguably, the best in density-based clustering, hence its utilization for comparison. The results generated by these algorithms are compared against an expected outcome provided for that dataset. A number of statistical measures are used to generate metrics, which indicate the quality of the clustering.

ELKI provides pair-counting metrics based on the expected and actual outcome. A contingency table [42] is calculated between the two outcomes. An example of a contingency table is provided in Fig. 4. In the example, the expected outcome has three classes, each with a single observation. This is represented by the column to the right of the table. The algorithm produced two clusters, one with two observations and another with a single observation. This is shown in the last row of the table in Fig. 7.

In our analysis of the compared algorithms, we utilize several measures. From the contingency table, we are able to generate a confusion matrix, which describes the relationship between the pairings.

- TP—The number of observation pairs correctly labeled by the algorithm.
- FP—The number of observation pairs incorrectly labeled by the algorithm.

Fig. 5 Rebuild (Phase 2) algorithm pseudocode

```
1. Check if PC_1 belongs to a cluster
     a. If it does not, assign PC_1 to a new cluster C
     b. If it does, C = PC_1's cluster
2. For each PC_2 that has not be compared to PC_1
     a. Check if PC_1 and PC_2 are neighbors using NF.
          i. If they are, assign PC_2 and all of its neighbors
             to C
3. Remaining clusters with a single datapoint are merged
   together as noise.
```

Fig. 6 Neighbor function pseudocode

```
1. Calculate radius NT by adding NT_1 and NT_2
2. For each pair of DP in PC_1, PC_2
     a. If distance between DP_1 and DP_2 <= NT
          i. PC_1 and PC_2 are neighbors
3. PC_1 and PC_2 are not neighbors.
```

Results

Expected	R1	R2	
E1	0	1	1
E2	1	0	1
E3	1	0	1
	2	1	3

TP	TN	FP	FN
3	0	2	4

Fig. 7 The contingency table (*above*) and associated confusion matrix (*below*)

- FN—The number of observation pairs missing from the results.

$$\mathrm{TP} = \sum_{i=0}^{r}\sum_{k=0}^{c} X_{ik}^2;$$

$$\mathrm{FP} = \mathrm{TP} - \sum_{i=0}^{c}\left(\sum_{k=0}^{r} X_{ik}\right)^2;$$

$$\mathrm{FN} = \mathrm{TP} - \sum_{i=0}^{r}\left(\sum_{k=0}^{c} X_{ik}\right)^2$$

- r = the number of rows in the contingency table.
- c = the number of columns in the contingency table.
- X_{ij} = a cell in the contingency table.

Precision and recall are also useful for identifying the relationship between the number of clusters generated by the algorithm and the expected outcome. A high score for precision with a low score for recall typically indicates that the algorithm generated fewer clusters than was expected. Likewise, a high score for recall with a low score for precision indicates that the algorithm generated more clusters than was expected. The F-Measure provides a weighted average of precision and recall. The experiments in the paper assume F Measure with $\beta = 1$, which means equal weight between recall and precision. The F-Measure is used to compare overall clustering results of the various algorithms used in the experiments. These measures are summarized in Table 1.

Precision and recall are not directly used to compare the results between algorithms, but they do evaluate how well an algorithm clustered something. The F-Measure builds on that by combining the two. This value can effectively be used in comparison of two algorithms.

Reachability plots are a dendrogram produced by OPTICS and DeLiClu, representing density reachability between datapoints in the dataset. Clusters must be manually extracted from the plot. They are provided to describe results generated from the aforementioned algorithms. Figure 8 demonstrates how valleys or "dents" correlate to regions of high density in the dataset. They are not produced or required by RADDACL2, since the actual clustering is generated by the algorithm.

Table 1 Measuring the performance of RADDACL2 and other tested algorithms

Metric	Equation	Definition
Precision	$P = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$	A measure indicating the number of datapoints that belong to the same cluster in the expected and actual clustering
Recall	$R = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$	A measure indicating the number of datapoints assigned to a cluster in the that exists in the expected clustering
F-Measure	$F_\beta = \frac{(\beta^2+1)*P*R}{\beta^2*P+R}$	A measure indicating a weighted combination Precision and Recall. $\beta = 1$ indicates equal weight between Precision and Recall, and is often call the $F1$-Measure

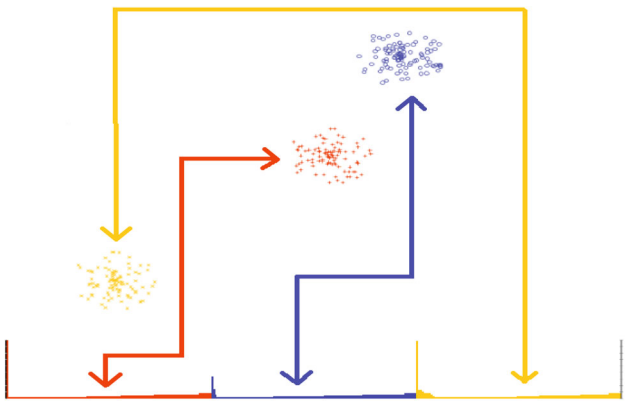

Fig. 8 A reachability plot on a synthetic dataset. Demonstrates the valleys or "dents" that separate areas of data point density within the dataset

DBSCAN [20] uses density rings to merge together nearby points into clusters. DBSCAN iterates through each point in the dataset, either assigning it to a noise cluster or an existing cluster depending on its neighbors. Points without neighbors end up in the noise cluster. Points with neighbors are assigned to an existing cluster (if its neighbors already belong to one) or a new cluster, along with its neighbors. This process continues until every point has been explored. A neighborhood is determined using the ε value. It works like a radius around a point to create a neighborhood. MinPTS, the other user adjustable parameter, represents the minimum number of points required to create a cluster. Clusters with elements less than minPTS are included as noise.

Density-linked clustering (DeLiClu) is considered an improvement over OPTICS as it eliminates a parameter, which is notoriously difficult to adjust. Nonetheless, the principle of operation is very similar. For that reason, here we will provide a brief overview of DeLiClu. This is a density-based clustering method, which depends on one parameter for its operation—minimum points per cluster—designated as density smoothing parameter. The algorithm utilizes R-trees, density distance, and hierarchical principles via priority queues to "sort" the dataset into clustered datapoints. The idea is very similar to hierarchical clustering. The hierarchy is maintained through a priority queue, where pairs of close pair points are being stored ranked by density distance, where DeLiClu parameter comes in. The result of the algorithm is a reachability plot, same as OPTICS, albeit more readable. This plot then needs to be "translated" into the actual clusters. Although DeLiClu is dealing with only one parameter, that still depends on the nature of the dataset and requires preprocessing and user intervention. The reachability plot requires postprocessing to display the clusters. RADDACL2 is free of all three requirements, which allows it to deliver faster performance at a better or at least comparable performance. Additional details on the calculation and interpretation of the plots can be found in the OPTICS [36] and DeLiClu [37] articles.

4.1 Toy datasets

Three datasets were created to test the capabilities of RADDACL2. They are designed specifically for density-based techniques. The same set of metrics and figures are provided for each dataset. Each test maintains algorithm parameters such as distance function and index structures when possible. If a distance function is required, Euclidian distance is assumed. Some algorithms, such as DeLiClu, require a special index structure to calculate results. The performance parameters for these indexes were maintained between experiments to reduce variability. If a dataset needed to be scaled prior to clustering for any reason, the same scale was used for each algorithm in the experiment. As both OPTICS and DeLiClu produce similar results and reachability plots, the results of OPTICS (the base algorithm) were omitted. DeLiClu operates with minimum points per cluster parameter of 25 % for all toy datasets. DBSCAN has the same value for its minPTS and $\varepsilon = 4$. It must be noted that Figs. 9, 12, and 16 represent the RADDACL2 results. RADDACL2 does not need preset parameters as they are calculated based on the dataset characteristics.

4.1.1 Spiral dataset

The spiral dataset consists of three dimensional datapoints with two clusters. A visualization of the clusters is shown in Fig. 9. There are a total of 6000 datapoints in the dataset.

This dataset is developed to test the algorithm's ability to identify clusters within clusters. It differs from the ring dataset (Fig. 9) by adding a dimension to the datapoints and varying the shapes. As shown through the provided metrics in Table 2, RADDACL2 and the other density algorithms can identify the two clusters.

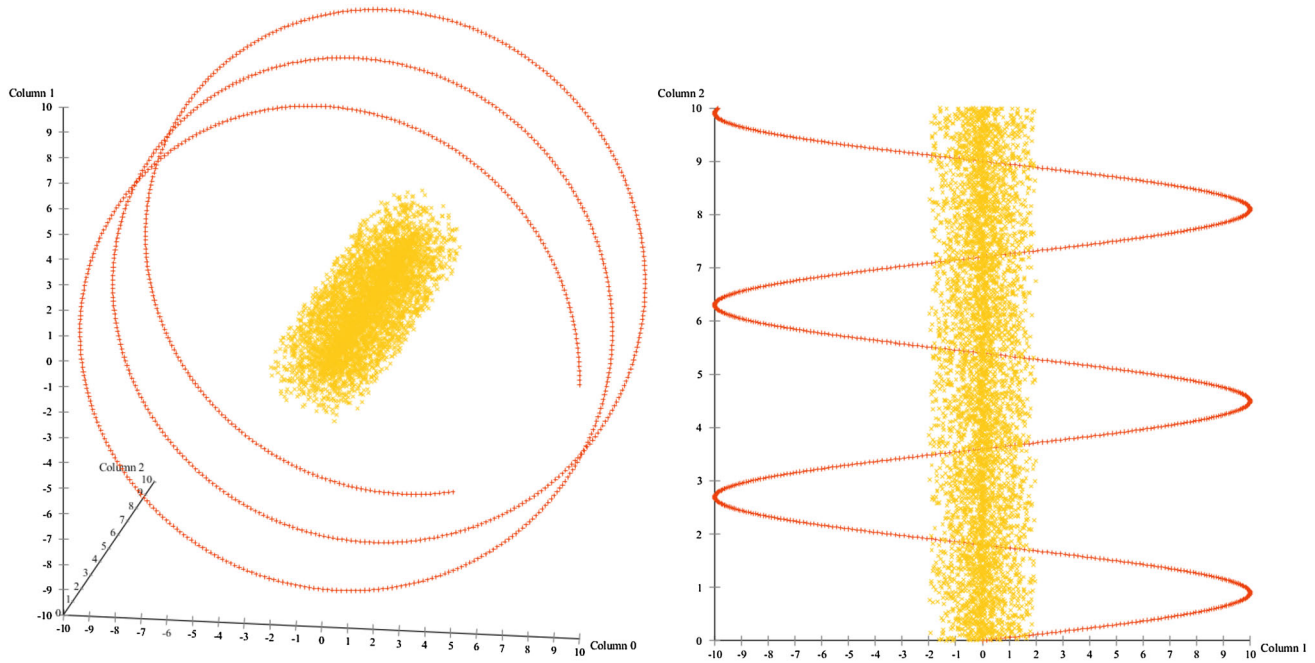

Fig. 9 3D visualization of the two expected clusters for the spiral dataset and produced by RADDACL2

Table 2 Toy dataset results comparison utilizing the defined measures

Algorithm	Spiral dataset			Ring dataset			Grid dataset		
	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall
RADDACL2	1	1	1	1	1	1	1	1	1
DBSCAN	1	1	1	1	1	1	1	1	1
KMEANS	0.76	0.77	0.8	0.44	0.67	0.33	0.78	0.8	0.77
DELICLU	1	1	1	1	1	1	1	1	1

Fig. 10 Reachability plot produced by DeLiClu for the spiral dataset

During the experiment, the datapoint attributes were scaled by various constants, to see how many runs it would take to re-identify the clusterings (structure is retained when scaling by a constant). RADDACL2 required a single run to identify the correct clustering when the dataset was scaled. DBSCAN required multiple runs to properly tune the density threshold. OPTICS and DeLiClu required one run and additional postprocessing to interpret the reachability plot.

Metrics were collected on the performance of the algorithms used in this experiment (Table 2). The density algorithms performed well, as expected, while KMeans, on average, could not separate the two clusters. In terms of usability, RADDACL2 surpasses the other algorithms. As shown in Fig. 10, the valleys indicating cluster separation are not readily observable. The same can be said for the parallel plot in Fig. 11, which would look like a big mass of lines if not for the color-coding. The proposed algorithm shows superiority in the lack of tunable parameters as well as the easiness of reading the results without requiring additional postprocessing.

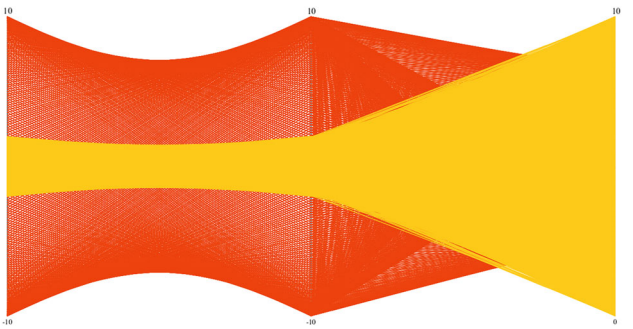

Fig. 11 Parallel plot for the two clusters in the spiral dataset. The *yellow* represents the cylinder, while the *red* is the spiral (color figure online)

4.1.2 Ring dataset

The ring dataset consists of 2-dimensional datapoints with three clusters. A visualization of the clusters is shown in Fig. 12. There are a total of 5050 datapoints in the dataset.

This dataset was developed to test unbalanced representation of clusters, as well as clusters encapsulated by other clusters. The outer ring contains 4000 datapoints, which is just under 80 % of the data. The middle ring contains 1000 datapoints, representing 19.8 % of the data. The center ring contains 50 datapoints, which is less than 1 % of dataset. It is obvious that the outer-most ring encapsulates the remaining rings, which makes for a heavily non-linearly separable dataset. As shown through the provided metrics in Table 2, RADDACL2 and the other density algorithms can identify the three clusters.

The parallel coordinate plot, seen in Fig. 13, is once again too dense to provide any useful information about the structure of the data. The reachability plot (Fig. 14) does offer some insight into the formation of clusters in the dataset. We can see several valleys in the plot representing sub-clusters within the outer and middle rings. It is difficult to identify the borders between the outer and middle rings.

As with the spiral dataset, the datapoint attributes were scaled by various constants, to see how many runs it would take to re-identify the clusters. Once again, RADDACL2 required a single run to identify the correct clustering even when scaling the dataset. DBSCAN required multiple runs to properly tune the density threshold. OPTICS and DeLiClu required one run and additional postprocessing to interpret the reachability plot demonstrating RADDACL2's usability once again. Other algorithms, such as KMeans, were unable to provide an adequate clustering (Fig. 15). Identifying clusters in a non-linearly separable dataset has proven to be a challenging problem for algorithms using non-density models.

4.1.3 Grid dataset

The grid dataset consists of 2-dimensional datapoints with three clusters. A visualization of the clusters is shown in

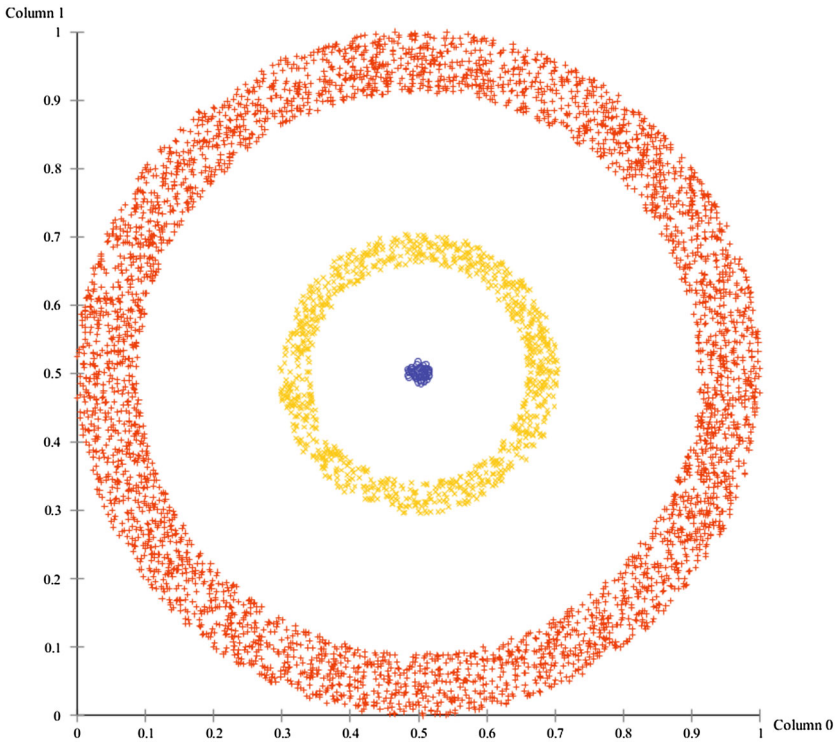

Fig. 12 A 2D visualization of the dataset and the clustering result by RADDACL2

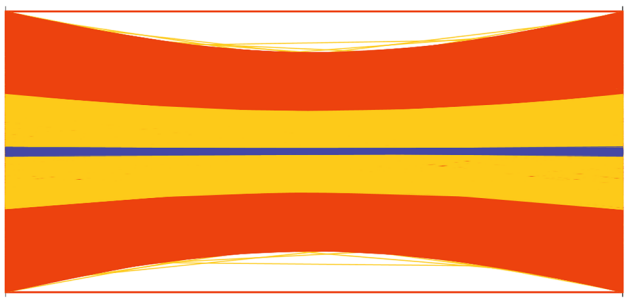

Fig. 13 The parallel coordinates for the ring dataset. *Color* shows the clusters (color figure online)

Fig. 16. There are 469 datapoints in the dataset, which is developed to test clusters with varying levels of density within the same dataset. Each datapoint within a cluster is equidistant from their neighbors, and the borders between clusters are well defined.

The parallel coordinate plot, seen in Fig. 17, could be used to easily distinguish the three clusters. The reachability plot shown in Fig. 18 also distinctly shows three different valleys, each representing a cluster.

As with the previous toy datasets, the datapoint attributes were scaled by various constants, to see how many runs it would take to re-identify the clusters. The results of the scaling test were consistent with the previous toy datasets.

Table 2 provides the metrics for this experiment, with similar results as the Ring and Spiral Dataset. RADDACL2's usability is ahead of the competition with this dataset. KMeans was unable to correctly identify clusters in this dataset each time, occasionally producing results such as the clustering in Fig. 19. Considering the amount of preprocessing usually required to bring the dataset into a useful format, only having to run the algorithm once certainly removes much of the burden on the postprocess analysis.

4.2 Real datasets

In these examples, we illustrate how RADDACL2 works with real-world examples. Metrics are not provided here as the expected clustering is not available. Each test tries to share as many common attributes between the algorithms as possible. When a distance function is required, Euclidian distance is used. Some algorithms, such as DeLiClu, require a special index structure to calculate results. The performance parameters for these indices were maintained between exper-

Fig. 14 The reachability plot for the ring dataset

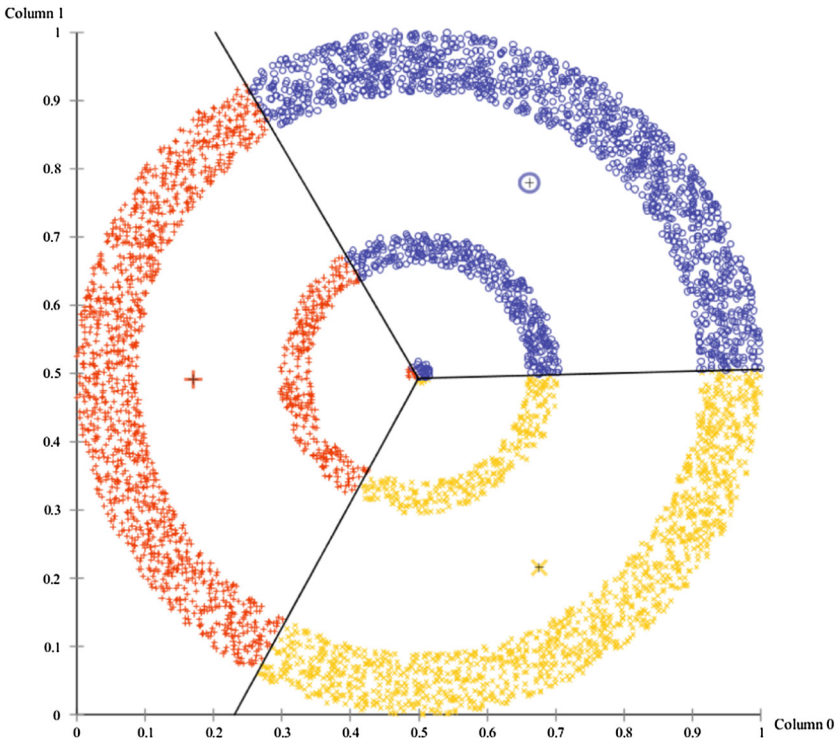

Fig. 15 The clustering results and centers of a KMeans run on the ring dataset

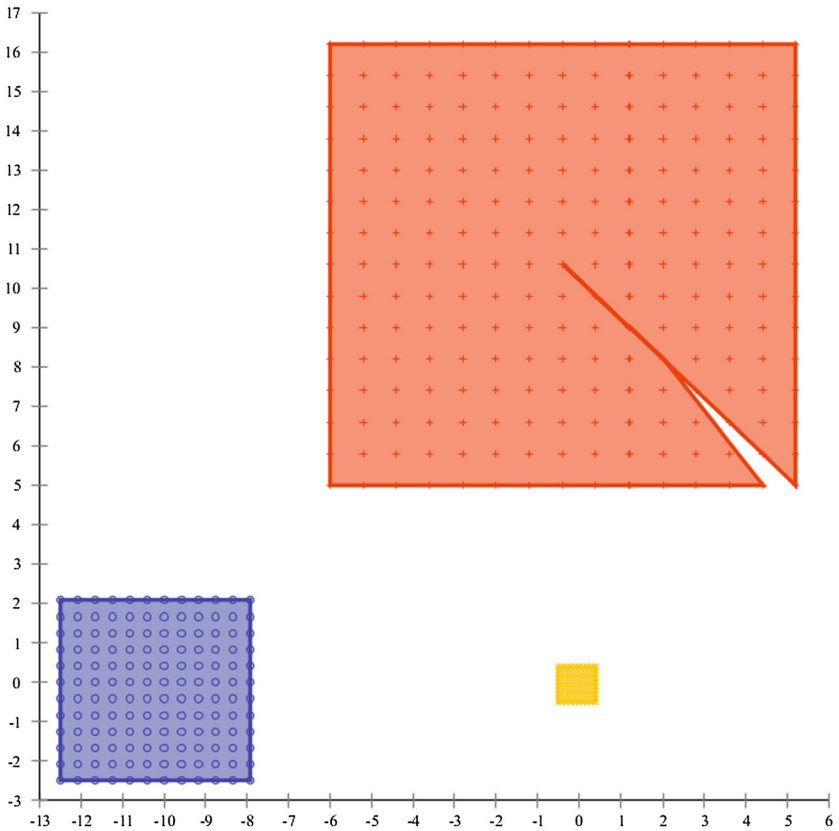

Fig. 16 A 2D visualization of the grid dataset and the clustering result produced by RADDACL2

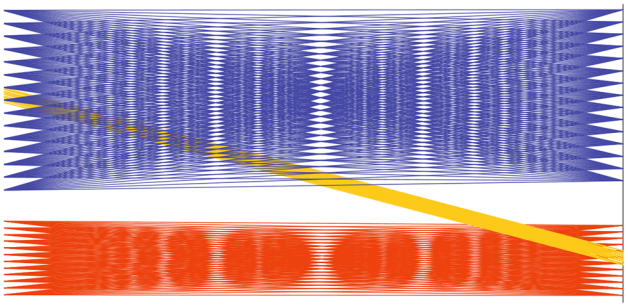

Fig. 17 The parallel coordinates for the grid dataset. *Color* shows the clusters (color figure online)

iments to reduce variability. If a dataset needed to be scaled prior to clustering for any reason, the same scale was used for each algorithm in the experiment. It must be noted that DeLiClu uses exorbitant amount of memory when processing the MRI dataset. When it is done processing, everything is determined to be one cluster. Hence, DeLiClu results for the faces and MRI dataset are not provided. DBSCAN's results, however, are included. Although this algorithm uses two parameters—the dreaded ε and minimum points required for cluster formation, thus demonstrating inferiority to RADDACL2, it is a benchmark representative of density-based clustering. Therefore, the comparison is only fitting.

4.2.1 ORL faces

In this case, RADDACL2 clusters faces from the ORL database (provided by AT&T Laboratories Cambridge) [43]. This dataset comprises 400 grayscale images of faces from 40 different individuals. Each individual has 10 different perspectives (i.e., angle of taken image) of their face in the database. Each picture is represented in 256 levels of gray with a size of 92×112.

RADDACL2 is applied to the intensity values of the image. Since we are looking for features, weight on the intensity is more important than location, as features are often separated by changes in contrast within the image. The resulting images in Figs. 20 and 21 are colored based on the resulting clusters provided by RADDACL2.

Fig. 18 The reachability plot for the grid dataset

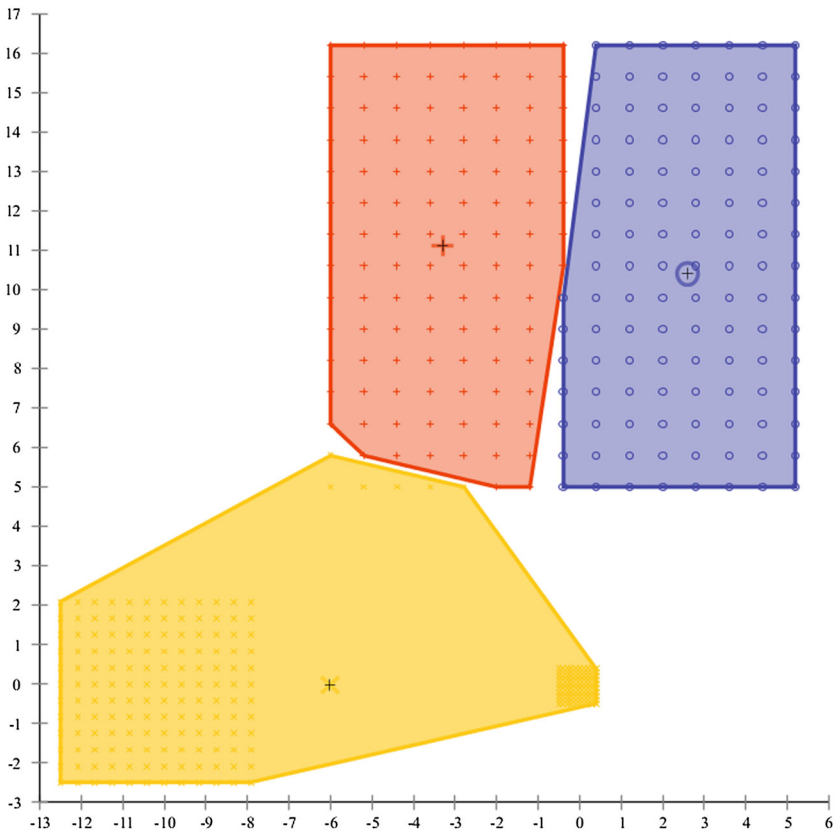

Fig. 19 The clustering results with centers of a KMeans run on the grid dataset

Fig. 20 Clustering of the same face with eyes open (*bottom row*) and closed (*top row*)

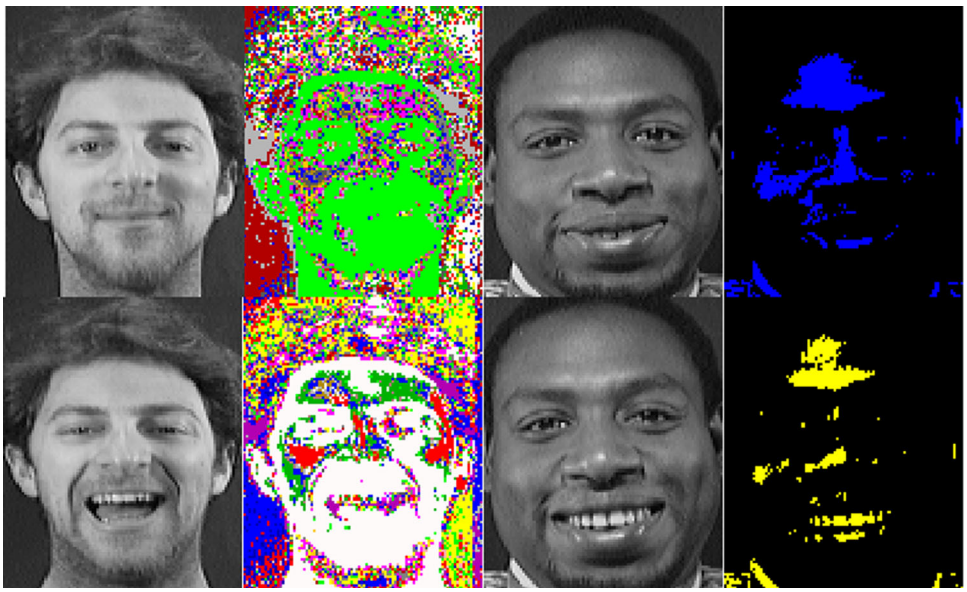

Fig. 21 Two examples demonstrating RADDACL2 differentiating between faces with an opened or closed mouth

The results demonstrate that RADDACL2 is capable of identifying structural changes in major features of the face, including the eyes a mouth. Figure 20 provides an example of the same face with both eyes open and closed. Using basic geometry, it is possible to extract this information. For the open eyes, the outline of the eye will be closer to the eyebrow, and the cluster should take up more area than a closed eye, for instance. For closed eyes, the distance between the eye and eyebrow will be larger, as the data for eyelid will belong to a different cluster. The eye cluster will be significantly flatter, having a much greater width than height. These distinctions can be used to identify open and closed eyes.

Open and closed mouth images exhibit a similar behavior. Figure 21 provides two examples of faces with open and closed mouths. In the first case, RADDACL2 demonstrates the difference by the number of clusters around the mouth area. We can see that when the mouth is closed, the mouth blends with the beard, as they have similar contrast. When the mouth is open, however, a large number of clusters form around the mouth region, which can be easily extrapolated. In the second example, the behavior is similar to the open and closed eye problem. The closed mouth manifests as a thinner cluster, while the open mouth has more height and draws a distinction to the features contained within (the teeth, in this case).

After this analysis, let us turn the attention to the DBSCAN results presented in Fig. 22. The experiments are at minPTS of 28 %, while the epsilon parameter value varied in the range 3–9.5. Figure 22 features the best clustering with the applicable epsilon value in the figure caption. What is important, that

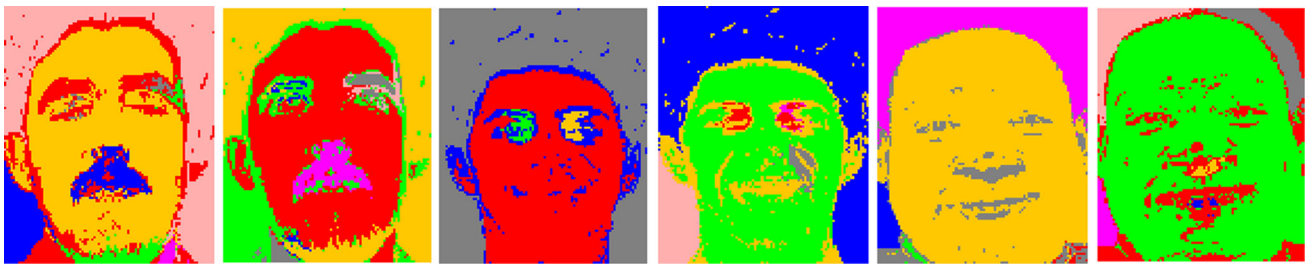

Fig. 22 DBSCAN results for the faces in Figs. 20 and 21. From *left* to *right*, the ε values are 8, 8.5, 8, 7.5, 8.5, 7.5; minPTS is 28 %

Fig. 23 KMeans results ($k = 20$) for the faces in Figs. 20 and 21

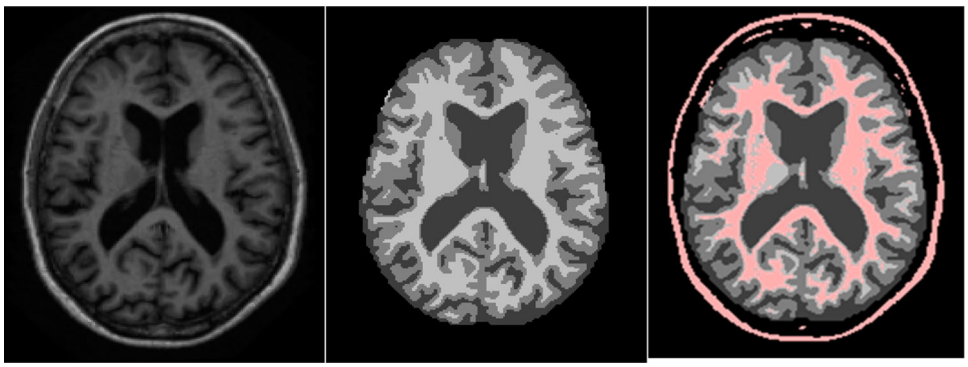

Fig. 24 From *left* to *right*, the atlus-generated image, the segmented result provided in the database, and the RADDACL2 cluster containing extracted white matter and skull on top of the segmented result

even under these most favorable circumstances, the geometry of the facial features is lost on some of the faces. It would be difficult for any postprocessing procedure to pick up the closed eyes or the smiling mouth on every face. The comparison between RADDACL2 and DBSCAN can be concluded with the fact that RADDACL2 does not require parameter adjustment, requires much less memory to run, and, finally, the result can be successfully subjected to geometric interpretation of clustered facial features. While we provide the KMeans results for the faces in Figs. 20 and 21, a geometric analysis will be impeded due to the blending of facial details (Fig. 23).

4.2.2 MRI databases

RADDACL2 is also used on a variety of MRI scans provided by OASIS [44]. Each patient comes with three or four raw scans captured form the MRI and additional postprocessed images. For the experiments, the SUB_111 and T88_111 postprocessed images were used. The SUB_111 images are produced by averaging the provided raw scans and applying some basic processing techniques. T88_111 contain atlus-corrected imagery and provide a top-down MRI. The exact procedure is described in the fact sheet provided with the dataset. RADDACL2 is applied to the intensity values only. Position was omitted, since the features we are looking for are mostly divided by contrast.

Figure 23 contains the results of the RADDACL2 clustering on the T88_111 and the segmented imagery provided by the database. RADDACL2 can identify major features comparably to modern segmentation techniques, as shown in the overlay provided by Fig. 24. The difference is that RADDACL2 came to the result without making any strong assumptions about the data being processed, whereas the provided segmentation algorithm only works specifically on that type of imagery.

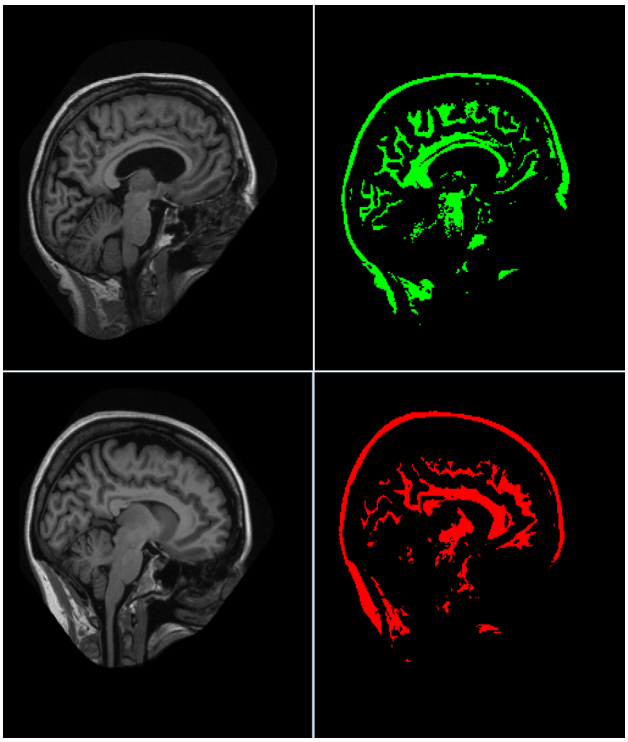

Fig. 25 Images on the *left* represent the original SUB_111 co-registered average images. The images on the *right* contain RADDACL clustering information, including white matter and skull location

Figure 25 demonstrates the results produced by RADDACL2 on the SUB_111 for the same patient. Again, RADDACL2 identifies the skull and white matter with sufficient accuracy. Using basic geometry, we can associate clusters generated by RADDACL2 to specific features of the brain.

Once the discovery of connected regions has been achieved, it becomes possible to automate a number of applications. One common application would be region segmentation. This subsequently allows for the extraction or removal of those features from the image, which would depend on the user's needs. These processed results aid in the diagnosis of a variety of diseases and behavioral patterns compared to the raw imagery [45,46].

The results from DBSCAN are shown in Fig. 26. The minPTS is at 29 % and the epsilon value is in the figure caption. In this experiment we have also included the noise generated from parts of the image that are determined to have too few points (below the minPTS threshold) to be considered clusters. It would seem that these regions contain more usable information than the actual clusters.

Although the experiments were performed on MRIs of the brain, it is likely that RADDACL2 will achieve similar results when applied to different areas of the body. Another common application would be to directly classify a disease from the imagery. RADDACL2 could be used to directly detect anomalies, such as lesions associated with breast cancer and potentially stage of cancer in the patient [47]. Here we also provide the results from KMeans, as DeLiClu delivers only one cluster at the end. Aside from needing long processing times (much longer in comparison to RADDACL2), it also utilizes a good amount of memory, which RADDACL2 does not need. Figure 27 shows the KMeans results with 20 predetermined clusters for the images in Figs. 24 and 25. Needless to analyze, any useful detail is lost in the process of partitioning.

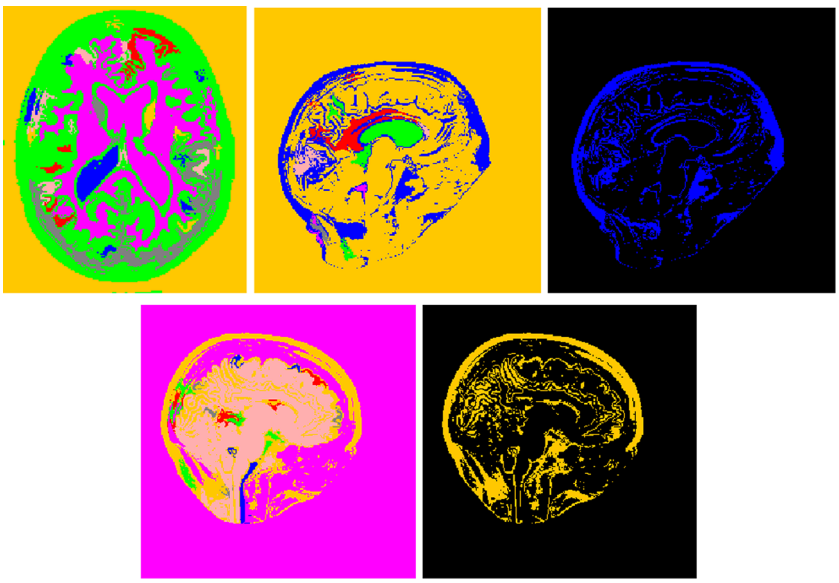

Fig. 26 DBSCAN results from the MRI database; from *left* to *right* the ε values are 5.5, 6.5, 6.5; the third and fifth images are the noise generated by the algorithm

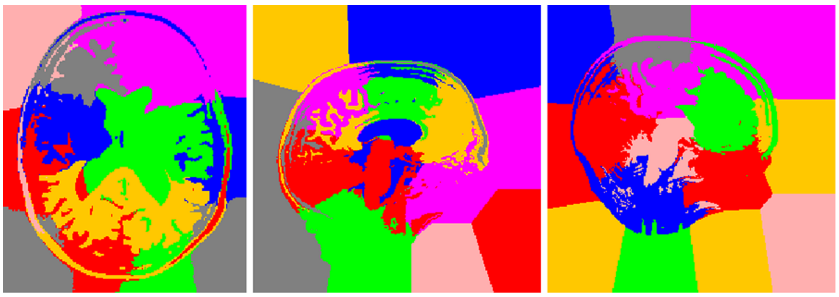

Fig. 27 KMeans clusters (k = 20) for the original images in Figs. 23 and 24

5 Conclusion

In this paper we present RADDACL2, a Recursive Algorithm for Density Discovery and Clustering. RADDACL2 offers several advantages over various other algorithms by satisfying the following two conditions. First, RADDACL2 provides a deterministic clustering for a given set of data. Second, RADDACL2 internally calculates thresholds and performance parameters based on the shape of the data. These two points significantly reduce the amount of time a user must spend tuning the algorithm to get correct results. However, it would only matter if RADDACL2 could perform comparably to existing algorithms. Therefore, a number of experiments are performed using toy and real datasets to verify RADDACL2's capabilities. The results indicate that RADDACL2 can be used in a number of clustering space tasks. Experiments performed on toy datasets confirm that RADDACL2 can properly classify non-linearly separable datasets. Traditional, and oft-used, algorithms, such as KMeans, are unable to successfully classify under these conditions without excessive preprocessing. Experiments on the face data provide evidence that RADDACL2 can be used for feature recognition and differentiation. Experiments with the MRI database demonstrate RADDACL2's ability to use a region of interest for segmentation applications. These two datasets confirm that RADDACL2 can be used to automate tasks, which normally require an expert, such as disease detection or target tracking. Considering the presented findings, we can conclude that RADDACL2 provides the deterministic results experts are looking for when working in the clustering domain while simultaneously reducing user burden by simplifying the problem constraints. These two key features are what separate RADDACL2 from the competition. It is interesting to follow this research with investigations of different distance metrics, e.g., Bregman Divergences [48].

Acknowledgments The authors wish to thank Mr. Derek Beaton for his comments and suggestions in developing RADDACL2. He is one of the co-authors on the original RADDACL algorithm.

References

1. Grabmeier, J., Rudolph, A.: Techniques of cluster algorithms in data mining. Data Min. Knowl. Discov. **6**(4), 303–360 (2002)
2. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Comput. Surv. **31**(3), 264–323 (1999)
3. Jain, A.K., Duin, R.P.W., Mao, J.: Statistical pattern recognition: a review. Pattern Anal. Mach. Intell. IEEE Trans. **22**(1), 4–37 (2000)
4. Jain, A.K.: Data clustering: 50 years beyond K-means. Pattern Recognit. Lett. **31**(8), 651–666 (2010)
5. Bunke, H., Riesen, K.: Towards the unification of structural and statistical pattern recognition. Pattern Recognit. Lett. **33**(7), 811–825 (2012)
6. Hastie, T., Tibshirani, R., Friedman, J.J.H.: The Elements of Statistical Learning, vol. 1. Springer, New York (2001)
7. Curtin, R.R., Cline, J.R., Slagle, N.P., Amidon, M.L., Gray, A.G.: MLPACK: a scalable C++ machine learning library. J. Mach. Learn. Res. **14**, 801–805 (2013)
8. Chi, Y.: Multivariate methods. Wiley Interdiscip. Rev. Comput. Stat. **4**(1), 35–47 (2012)
9. Berkhin, P.: A survey of clustering data mining techniques. In: Kogan, J., Nicholas, C., Teboulle, M. (eds.) Grouping Multidimensional Data, pp. 25–71. Springer, Berlin (2006)
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. ACM SIGKDD Explor. Newsl. **11**(1), 10–18 (2009)
11. Plaza, A., Benediktsson, J.A., Boardman, J.W., Brazile, J., Bruzzone, L., Camps-Valls, G., Chanussot, J., Fauvel, M., Gamba, P., Gualtieri, A., et al.: Recent advances in techniques for hyperspectral image processing. Remote Sens. Environ. **113**, S110–S122 (2009)
12. Liming, X., Yanchao, Z.: Automated strawberry grading system based on image processing. Comput. Electron. Agric. **71**, S32–S39 (2010)
13. Bécue-Bertaut, M., Pagès, J.: Multiple factor analysis and clustering of a mixture of quantitative, categorical and frequency data. Comput. Stat. Data Anal. **52**(6), 3255–3268 (2008)
14. Chan, G., Gelernter, J., Oslin, D., Farrer, L., Kranzler, H.R.: Empirically derived subtypes of opioid use and related behaviors. Addiction **106**(6), 1146–1154 (2011)
15. Gottardo, F., Liu, C.G., Ferracin, M., Calin, G.A., Fassan, M., Fassan, M., Bassi, P., Sevignani, C., Byrne, D., Negrini, M., Pagano, F., Gomella, L.G., Croce, C.M., Baffa, R.: Micro-RNA profiling in kidney and bladder cancers. Urol. Oncol. **25**(5), 387–392 (2007)
16. Stahlberg, A., Elbing, K., Andrade-Garda, J., Sjogreen, B., Forootan, A., Kubista, M.: Multiway real-time PCR gene expression profiling in yeast *Saccharomyces cerevisiae* reveals altered transcriptional response of ADH-genes to glucose stimuli. BMC Genomics **9**(1), 170 (2008)

17. Tichopád, A., Pecen, L., Pfaffl, M.W.: Distribution-insensitive cluster analysis in SAS on real-time PCR gene expression data of steadily expressed genes. Comput. Methods Programs Biomed. **82**(1), 44–50 (2006)
18. Estivill-Castro, V.: Why so many clustering algorithms: a position paper. SIGKDD Explor. Newsl. **4**(1), 65–75 (2002)
19. Beaton, D., Valova, I.: RADDACL: a recursive algorithm for clustering and density discovery on non-linearly separable data. In: Neural Networks, 2007. IJCNN 2007. International Joint Conference on 2007, pp. 1633–1638
20. Dunham, M.H.: Data Mining: Introductory and Advanced Topics. Pearson Education India, Delhi (2006)
21. Rizzi, R., Mahata, P., Mathieson, L., Moscato, P.: Hierarchical clustering using the arithmetic-harmonic cut: complexity and experiments. PLoS ONE **5**(12), e14067 (2010)
22. Sileshi, M., Gamback, B.: Evaluating clustering algorithms: cluster quality and feature selection in content-based image clustering. 2009 WRI World Congress on Computer Science and Information Engineering, vol. 6, pp. 435, 441, March 31 2009–April 2 2009
23. Malik, H.H., Kender, J.R., Fradkin, D., Moerchen, F.: Hierarchical document clustering using local patterns. Data Min. Knowl. Discov. **21**(1), 153–185 (2010)
24. Xiong, T., Wang, S., Mayers, A., Monga, E.: DHCC: divisive hierarchical clustering of categorical data. Data Min. Knowl. Discov. **24**(1), 103–135 (2012)
25. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means, spectral clustering and normalized cuts. KDD '04 Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 551–556 (2004)
26. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 1027–1035, (2007)
27. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis, vol. 344. Wiley.com, New York (2009)
28. Kaufman, L., Rousseeuw, P.J.: Clustering my means of Medoids. Statistical Data Analysis Based on the L_1–Norm and Related Methods (2002)
29. Mahdavi, M., Abolhassani, H.: Harmony K-means algorithm for document clustering. Data Min. Knowl. Discov. **18**(3), 370–391 (2009)
30. Bellaachia, A., Bari, A.: Flock by leader: a novel machine learning biologically inspired clustering algorithm. In: Tan, Y., Shi, Y., Ji, Z. (eds.) Advances in Swarm Intelligence, pp. 117–126. Springer, Berlin (2012)
31. Kohonen, T.: Self-Organizing Maps. Springer, Berlin (2001)
32. Bação, F., Lobo, V., Painho, M.: Self-organizing maps as substitutes for k-means clustering. In: Sunderam, V.S., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) Computational Science—ICCS 2005, pp. 476–483. Springer, Berlin (2005)
33. Linde, Y., Buzo, A., Gray, R.: An algorithm for vector quantizer design. Commun. IEEE Trans. **28**(1), 84–95 (1980)
34. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B **39**(1), 1–31 (1977)
35. Sander, J., Ester, M., Kriegel, H.-P., Xu, X.: Density-based clustering in spatial databases: the algorithm gdbscan and its applications. Data Min. Knowl. Discov. **2**(2), 169–194 (1998)
36. Ankerst, M., Breunig, M.M., Kriegel, H.-P., Sander, J.: OPTICS: ordering points to identify the clustering structure. ACM SIGMOD Rec. **28**(2), 49–60 (1999)
37. Achtert, E., Böhm, C., Kröger, P.: DeLi-Clu: boosting robustness, completeness, usability, and efficiency of hierarchical clustering by a closest pair ranking. In: Advances in Knowledge Discovery and Data Mining, pp. 119–128. Springer, Berlin (2006)
38. Hinneburg, A., Gabriel, H.H.: DENCLUE 2.0: fast clustering based on kernel density estimation. In: Berthold, M.R., Shawe-Taylor, J., Lavrač, N. (eds.) Advances in Intelligent Data Analysis VII, pp. 70–80. Springer, Berlin (2007)
39. Bae, E., Bailey, J., Dong, G.: A clustering comparison measure using density profiles and its application to the discovery of alternate clusterings. Data Min. Knowl. Discov. **21**(3), 427–471 (2010)
40. Dhillon, I.S., Guan, Y., Kulis, B.: Kernel k-means: spectral clustering and normalized cuts. In: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 551–556 (2004)
41. Achtert, E., Kriegel, H.-P., Zimek, A.: ELKI: a software system for evaluation of subspace clustering algorithms. In: Scientific and Statistical Database Management, pp. 580–585 (2008)
42. Miller, I., Miller, M., John, E.: Freund's mathematical statics with applications. Pearson Prentice Hall, Upper Sadle River (2004)
43. Samaria, F.S., Harter, A.C.: Parameterisation of a stochastic model for human face identification. In: Proceedings of the Second IEEE Workshop on Applications of Computer Vision, pp. 138–142 (1994)
44. Marcus, D.S., Wang, T.H., Parker, J., Csernansky, J.G., Morris, J.C., Buckner, R.L.: Open access series of imaging studies (OASIS): cross-sectional mri data in young, middle aged, nondemented, and demented older adults. J. Cogn. Neurosci. **19**, 1498–150 (2007). http://www.oasis-brains.org/app/template/Index.vm
45. Hashioka, A., Kobashi, S., Kuramoto, K., Wakata, Y., Ando, K., Ishikura, R., Ishikawa, T., Hirota, S., Hata, Y.: Shape and appearance knowledge based brain segmentation for neonatal MR images. World Automation Congress (WAC), 2012, pp. 1–6, 24–28 June (2012)
46. Selvaraj, D., Dhanasekaran, R.: Novel approach for segmentation of brain magnetic resonance imaging using intensity based thresholding. IEEE International Conference on Communication Control and Computing Technologies (ICCCCT), pp. 502–507, 7–9 Oct. (2010)
47. Lee, S.H., Kim, J.H., Kim, K.G., Park, J.S., Park, S.J., Moon, W.K.: Optimal clustering of kinetic patterns on malignant breast lesions: comparison between k-means clustering and three-time-points method in dynamic contrast-enhanced MRI. EMBS 2007. 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 2089–2093, 22–26 Aug. (2007)
48. Zhang, M.-L., Zhou, Z.-H.: Exploiting unlabeled data to enhance ensemble diversity. Data Min. Knowl. Discov. **26**(1), 98–129 (2013)