FDBSCAN-APT: A Fuzzy Density-based Clustering Algorithm with Automatic Parameter Tuning

Alessio Bechini*, Martina Criscione*, Pietro Ducange*, Francesco Marcelloni*, Alessandro Renda*†
* Department of Information Engineering, University of Pisa, Largo Lucio Lazzarino 1, 56122 Pisa, Italy
† Department of Information Engineering, University of Firenze, Via di S. Marta 3, 50139, Firenze Italy
Email: {alessio.bechini, pietro.ducange, francesco.marcelloni}@unipi.it, martina.c090@gmail.com, alessandro.renda@unifi.it

***Abstract*—Density-based clustering algorithms represent a convenient approach when the number of clusters is not known in advance and their shapes are arbitrary. Nevertheless, they are highly sensitive to the input parameter setting, especially when clusters' borders are close to each other, or even overlap. In this paper we propose FDBSCAN-APT, a fuzzy extension of the DBSCAN algorithm. FDBSCAN-APT is able to discover clusters with fuzzy overlapping borders and relies on the automatic setting of input parameters thanks to the definition of a novel heuristic based on the statistical modelling of the density distribution of objects. An extensive experimental analysis carried out on synthetic datasets shows that FDBSCAN-APT always finds reasonable parameter configurations and produces good clustering results in a variety of challenging scenarios.**

***Index Terms*—Fuzzy Clustering, Density-based Clustering, Automatic parameter setting**

I. Introduction

Among clustering algorithms, density-based methods are well suited to produce a partition when the shapes of cluster are arbitrary and no assumption can be made on the number of clusters, and\or when the ability to handle noise and outliers is required. The most popular density-based clustering algorithm is DBSCAN, which was proposed in [1]. DBSCAN uses a minimum density level, defined as the minimum number of objects ($MinPts$) within a radius (ε), to discover clusters as areas of high density, separated from one another by areas of low density [2]. In particular an object x is defined as a *core object* when at least $MinPts$ objects lie in its ε-neighborhood, i.e. at a distance less than or equal to ε from x. A non-core object y that lies in the ε-neighborhood of a core object, is defined as a *border object*. Finally, an object z that satisfies neither the core condition nor the border condition, is marked as *noise* or *outlier*. The clustering process in DBSCAN is based on two concepts: density reachability and density-connectedness. Let x_1 and x_n be two objects; then, x_n is *density reachable* from x_1 if x_1 is a core object and there exists a chain of core objects such that x_{i+1} is directly density-reachable from x_i, that is, x_{i+1} lies within the ε-neighborhood of x_i. Two objects x_1 and x_2 are *density-connected* if there exists a core object x_3, such that both x_1 and x_2 are density reachable from x_3. The notion of density connectedness is used in DBSCAN to find connected dense regions as clusters.

The original DBSCAN algorithm is not able to capture clusters with overlapping borders [3]. To enhance DBSCAN with this capability, fuzzy extensions of the algorithm have been proposed in the last years [3]. A notable example of fuzzy extension (FuzzyBorder), which represents the starting point of our work, relaxes the constraint on the neighborhood size for the definition of fuzzy borders by defining two distance thresholds: ε_{min} is considered for the evaluation of the *core condition*, while ε_{max} determines the maximum distance from a core object for which a border object can be assigned to a cluster. The two distance thresholds let us decouple the identification of core objects and the membership assessment of border objects and allow us modelling clusters with fuzzy overlapping borders.

However, as usual in clustering algorithms, DBSCAN and its extensions require the user to specify a set of input parameters that determine the behaviour and affect the outcome of the algorithms themselves. The importance of the choice of parameters in a density-based clustering algorithm can be easily understood by observing the example dataset (*square1*) shown in Fig. 1. Four clusters originate from four Gaussian distributions and their borders slightly overlap. In classical DBSCAN, a low value of the ε parameter would allow the algorithm to spot out four clusters, but border objects, characterized by a lower density, would not likely be included in clusters and would be labeled as outliers. On the other hand, high values of the ε parameter would likely result in the improper fusion of the four clusters into a single cluster. By introducing a further distance threshold for border objects, namely ε_{max}, the FuzzyBorder algorithm enhances the original DBSCAN with the capability of *broadening* cluster borders without affecting the core regions. Nevertheless, the choice of the two distance thresholds remains an open issue. Sometimes, even in the clustering literature, the parameter choice has been addressed through the optimization of an external validation measure, which requires the knowledge of the class labels. Although this approach may be useful to demonstrate the existence of an adequate parameter configuration, it becomes unworkable in real clustering applications, where the availability of labels cannot be assumed.

Although several heuristics have been proposed for param-

This work was partially supported by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence).

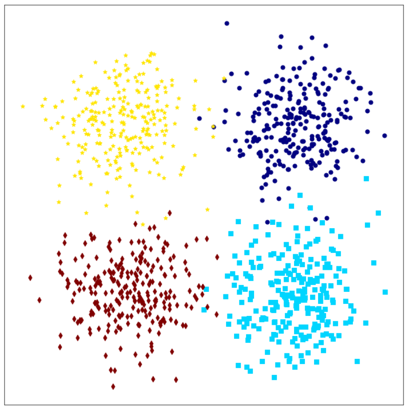


Fig. 1. Dataset *square1* as a scenario of critical parameter setting.

eter setting, to the best of our knowledge a fully automatic tuning procedure has not been described in the literature, and an analysis in the context of fuzzy DBSCAN algorithm has never been carried out as well. In this work we propose FDBSCAN-APT, a Fuzzy Density-based Clustering Algorithm with Automatic Parameter Tuning, an extension of the existing FuzzyBorder algorithm [3]. In FDBSCAN-APT proper values of the threshold distance parameters ε_{min} and ε_{max} are automatically inferred from data through the definition of a novel heuristic based on the statistical modelling of the density distribution of objects. To this aim, for each object x we define a data structure to store the list of objects in a *large enough* neighborhood of x, along with their distance from x itself. Information in such data structures can be promptly exploited to estimate proper values of ε_{min} and ε_{max}, and to obtain a final partition with the fuzzy DBSCAN procedure. Notably, the proposed heuristic can be applied also to the crisp DBSCAN algorithm to estimate a single parameter ε.

The rest of the paper is organized as follows: Section II discusses relevant related works. Section III provides the details of our proposed approach. Section IV describes the experimental study. Section V shows and discusses the experimental results, while Section VI draws some conclusion.

II. Related Works

In this section we first recall the most significant contributions related to the issue of parameter setting in density-based clustering algorithms, and then we describe the main idea behind FuzzyBorder DBSCAN, an algorithm proposed in [3], which inspired our approach.

A. The issue of parameter setting in density-based clustering algorithms

Along with the proposal of the DBSCAN algorithm, authors in [1] proposed a first heuristic to determine the parameters ε and $MinPts$. The heuristic is based on the definition of a function k-$dist$ that associates each object in the dataset with the distance from its k-th nearest neighbor. The distribution of such distances provides information about the density distribution of the dataset: by plotting distance values after sorting them in descending order, a user should be able to identify the index of the first "valley" associated to a distance threshold ε_k. This valley theoretically divides noise objects (high k-$dist$ values) from objects within clusters (low k-$dist$ values). By setting $\varepsilon = \varepsilon_k$ and $MinPts = k + 1$, the core condition will be verified for all the objects with a k-$dist$ value less than or equal to ε_k. Nevertheless, according to the authors, the identification of such index could benefit from an assumption about the percentage of noise objects in the dataset, but, in general, it requires user interaction, since the automatic detection of the "valley", or "elbow", is difficult. Furthermore, authors observe that k-$dist$ graphs are somehow insensitive to the value of k for $k \geq 4$: the problem of setting parameters ε and $MinPts$ can indeed be solved by fixing $MinPts$ and estimating ε with a visual analysis of the k-$dist$ graph. To the best of our knowledge, no methods have ever been proposed to automatically determine a proper value of the distance threshold parameter from the k-$dist$ graph.

In [4], the authors of DBSCAN delve into the issue of how to specify k. They state that, for 'reasonable' values of k (e.g. $\leq$ 10 in 2D space), the respective k-$dist$ graphs do not significantly differ from each other and the results of DBSCAN obtained with respective (k, ε_k) pairs are similar as well. Indeed the clustering algorithm is deemed rather stable across the choice of k, provided that it takes into account the dimensionality of the problem. As a heuristic, they propose to set $k = 2 * dimension - 1$, and $MinPts = 2 * dimension$ accordingly. The same heuristic is recommended in [5], with the remark that very large or high dimensional datasets, or datasets characterised by a lot of noise or duplicates may require higher values of $minPts$. As it concerns ε_k, it seems most appropriate to choose it as low as possible according to the k-$dist$ graph.

The recently proposed approaches DSets-DBSCAN [6] and KNN-DBSCAN [7] exploit the information from preliminary dominant sets clustering and k-nearest neighbors, respectively, to make DBSCAN parameter-free. The combination of the two algorithms, however, increases the complexity of the overall clustering. The AA-DBSCAN approach [8] aims to deal with multi-density datasets: first, it builds a density layer tree, and subsequently it finds clusters with an approximate adaptive ε distance for each layer.

OPTICS (Ordering Points To Identify the Clustering Structure) [9] overcomes the problem of setting global parameters; it stems from the observation that density-based clusters are monotonic with respect to the ε value [2]. OPTICS creates an ordering of the objects that represents the density-based clustering structure of the dataset. This ordering is equivalent to the partitions obtained for a wide range of values of the distance threshold parameter ε up to a 'generating distance' ε_{gen}. To reach its goal, OPTICS evaluates two attributes for each object: the *core-distance*, i.e. the distance for which the core condition is satisfied, and the *reachability-distance*, i.e. the smallest distance such that it is directly density-reachable from a core object. Starting from an arbitrary object in the dataset, the algorithm expands any found cluster by analysing (and storing) objects in a specific order such that clusters of higher density are finished first. The order is determined according to the reachability distance. However, OPTICS does not *explicitly* produce a partition of the dataset.

B. Background on FuzzyBorder DBSCAN

The FDBSCAN-APT algorithm stems from the FuzzyBorder solution, inheriting from it the capability of detecting clusters with fuzzy borders. The FuzzyBorder algorithm relaxes the constraint on the distance threshold ε and defines a new membership function, depicted in Figure 2. The parameters $MinPts$, ε_{min} and ε_{max} must be set by the user in the initialization step.

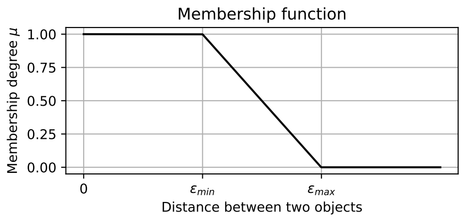


Fig. 2. The function for the fuzzy membership of an object to the neighborhood of another object.

Let x, y and $d(x, y)$ be a core object, a border object, and the distance between the two, respectively; the fuzzy membership of an object y to the neighborhood of x, namely $\mu_x(y)$, is defined as follows:

$$\mu_x(y) = \begin{cases} 1 & \text{if } d(x,y) < \varepsilon_{min} \\ \frac{\varepsilon_{max} - d(x,y)}{\varepsilon_{max} - \varepsilon_{min}} & \text{if } \varepsilon_{min} \leq d(x,y) \leq \varepsilon_{max} \\ 0 & \text{if } d(x,y) > \varepsilon_{max} \end{cases}$$

The *core-condition* in the FuzzyBorder algorithm is consistent with DBSCAN: only objects within ε_{min} are considered for the evaluation of the neighborhood cardinality and the determination of core objects. The cluster membership assessment of border objects in FuzzyBorder, instead, is different from DBSCAN: a border object can belong to a cluster even if it lies at a distance higher than ε_{min}, but lower than ε_{max}, from its closest core object. The algorithm allows an object to belong to the border of multiple clusters (optionally with a degree lower than 1), thus it yields a data grouping in clusters possibly with fuzzy overlapping borders.

III. The Proposed Algorithm

The pseudocode for FDBSCAN-APT is reported in Algorithms 1, 2, 3, 4 and it is described in the following.

FDBSCAN-APT requires the user to specify the following parameters: $minPts$, ε_{gen}, and α. Similarly to DBSCAN, $minPts$ represents the minimum number of objects required to lie within a threshold distance from an object p to define p as a *core* object. Unlike traditional DBSCAN, such threshold distance is not given as an input parameter, but it is automatically inferred from data. For this purpose, the user is just required to specify a value for ε_{gen}, a *generating* radius that

Algorithm 1 FDBSCAN-APT(D, ε_{gen}, $minPts$, α)

```
Given: $D$ - input dataset
Given: $\varepsilon_{gen}$ - generating radius
Given: $minPts$ - minimum number of objects to define a core
Given: $\alpha$ - coefficient for fuzzy border. Default = 1
1: $pList = \emptyset$
2: for $p \in D$ do
3:     $p.neighbors$ = distanceQuery($p$, $pList$, $\varepsilon_{gen}$)
4:     for $n \in p.neighbors$ do
5:         $n.neighbors = n.neighbors \cup \{(p, dist(p, n))\}$
6:     $pList = pList \cup \{p\}$
7: ($\hat{\varepsilon}_{min}$,$\hat{\varepsilon}_{max}$) = parametersEstimation($pList, minPts, \alpha$)
8: $Clusters$ = applyFuzzyDBSCAN($pList, \hat{\varepsilon}_{min}, \hat{\varepsilon}_{max}, minPts$)
9: return $Clusters$
```

Algorithm 2 parametersEstimation($pList$, $minPts$, α)

```
Given: $pList$ - list of objects
Given: $minPts$ - minimum number of objects to define a core
Given: $\alpha$ - coefficient for fuzzy border. Default = 1
1: $kDistList = \emptyset$
2: for $p \in pList$ do
3:     sort($p.neighbors$) by the distance from $p$
4:     $p.\varepsilon_{core}$ = distance of minPts-th point far from p
5:     $kDistList = kDistList \cup \{p.\varepsilon_{core}\}$
6: $gmm$ = GaussianMixtureModel($n\_components = 2$)
7: $(\mu_H, \sigma_H, w_H)$, $(\mu_L, \sigma_L, w_L)$ = $gmm$.fit($kDistList$)
8: $\hat{\varepsilon}_{min} = \mu_H + 2 * \sigma_H$
9: $\hat{\varepsilon}_{max} = \alpha * \hat{\varepsilon}_{min} * \frac{\mu_L}{\mu_L - \mu_H}$
10: return ($\hat{\varepsilon}_{min}$, $\hat{\varepsilon}_{max}$)
```

defines a neighborhood that *easily* contains more than $minPts$ objects for most objects in the dataset. Finally, α is a real number greater than 1 that can be used to fine-tune the degree of fuzziness of cluster borders.

The entry point of our procedure is Algorithm 1, in which instances of the input dataset are read and analyzed sequentially. For each instance p we store the set of neighbors in $pList$ within ε_{gen} ($p.neighbors$), along with their distance from p (distanceQuery, line 3), and we add it to the list of collected objects ($pList$). Likewise, the neighbors set of each object n in $p.neighbors$ is updated with a tuple given by the newly read object p and the distance between p and n.

When all instances have been analyzed, the procedure parametersEstimation (Algorithm 2) exploits the previously computed distances (stored in elements of $pList$) to estimate proper values of ε_{min} and ε_{max}. Finally, the applyFuzzyDBSCAN procedure (Algorithm 3) performs the clustering as in the FuzzyBorder algorithm [3]. Notably, no further distance computation is needed, as all the required information is already available for each object in the relative $neighbors$ data structure. The core condition for an object p is simply evaluated by comparing its core-distance ($p.\varepsilon_{core}$) with the estimated $\hat{\varepsilon}_{min}$ (line 5). The set of objects that lie in the $\hat{\varepsilon}_{min}$-neighborhood of p, which are considered for cluster expansion (Algorithm 4), and the set of objects that lie at a distance between $\hat{\varepsilon}_{min}$ and $\hat{\varepsilon}_{max}$, considered for the fuzzy border assessment, are immediately available from $neighbors$ via the $getMinSet$ and $getMinMaxSet$ procedures, respectively.

Algorithm 3 applyFuzzyDBSCAN($pList, \hat{\varepsilon}_{min}, \hat{\varepsilon}_{max}, minPts$)

```
Given: $pList$ - list of objects
Given: $\hat{\varepsilon}_{min}$ - radius for identification of core objects
Given: $\hat{\varepsilon}_{max}$ - radius for identification of border objects
Given: $minPts$ - minimum number of objects needed to define a core, as in DBSCAN
 1: $C = 0$
 2: $Clusters = \emptyset$
 3: for $p \in pList$ do
 4:     mark $p$ as visited
 5:     if $p.\varepsilon_{core} > \hat{\varepsilon}_{min}$ then
 6:         label $p$ as outlier
 7:     else
 8:         $C = nextCluster$
 9:         add $p$ to $C$ as core object
10:         $Clusters = Clusters \cup$ expandCluster($p$, getMinSet($p$, $\hat{\varepsilon}_{min}$), $C$, $\hat{\varepsilon}_{min}$, $\hat{\varepsilon}_{max}$)
11: return $Clusters$
```

Algorithm 4 expandCluster(p, $pMin$, C, $\hat{\varepsilon}_{min}$, $\hat{\varepsilon}_{max}$)

```
Given: $p$ - object marked as visited, just recognized as core point
Given: $pMin$ - set of objects in the $\hat{\varepsilon}_{min}$-neighborhood of $p$
Given: $C$ - cluster
Given: $\hat{\varepsilon}_{min}$ - radius for identification of core objects
Given: $\hat{\varepsilon}_{max}$ - radius for identification of border objects
 1: $borders$ = getMinMaxSet($p$, $\hat{\varepsilon}_{min}$, $\hat{\varepsilon}_{max}$)
 2: for $s \in$ pMin do
 3:     if $s$ is not visited then
 4:         mark $s$ as visited
 5:         if $s.\varepsilon_{core} \leq \hat{\varepsilon}_{min}$ then
 6:             add $s$ to C as core
 7:             pMin = pMin $\cup$ getMinSet($s$, $\hat{\varepsilon}_{min}$)
 8:             $borders = borders \cup$ getMinMaxSet($s$, $\hat{\varepsilon}_{min}$, $\hat{\varepsilon}_{max}$)
 9:         else
10:             $borders = borders \cup \{s\}$
11: $borders = borders \setminus C$
12: for $b \in borders$ do
13:     add $b$ to C as border object with membership $\mu_{bc}$ where c is the nearest core of C
14: return C
```

A. Parameter Estimation: Gaussian Mixture Model

The parameters estimation procedure aims to determine proper values of ε_{min} and ε_{max} parameters, and it is described in Algorithm 2. For each object p we evaluate its core distance $p.\varepsilon_{core}$ (line 4), which is defined as the distance which makes p a core object. Given $k = MinPts - 1$, it is evaluated as the distance from its k-th nearest neighbor. Note that such a distance may be undefined for objects with less than $minPts$ objects in its ε_{gen}-neighborhood: in this case, we simply do not account for their contribution to the estimation of $\hat{\varepsilon}_{min}$ and $\hat{\varepsilon}_{max}$. As noted since the proposal of DBSCAN [1], the knowledge of all core distances (kept in $kDistList$ in Algorithm 2) can be particularly helpful in identifying a proper value for the distance threshold parameter.

The idea behind our approach for automatic parameter setting stems from the analysis of the unidimensional array of all core distances, and relies on the following assumptions:

- the density distribution of the dataset is assumed to be *unimodal*, i.e. all clusters have roughly the same density of objects.
- values in $kDistList$ can be modeled as a mixture of two unidimensional Gaussian components: the first one models the contribution of cluster objects within a high density region (roughly speaking, core objects); the second accounts for the contribution of border objects, and is affected by the presence of noise and outliers.

In order to visually convey the idea, the k-*dist* graph for dataset *square1* is reported in Fig. 3. The histogram of the $kDistList$ values evaluated on 100 bins is plotted on the right, sharing the axis of k-*dist* values with the k-*dist* graph.

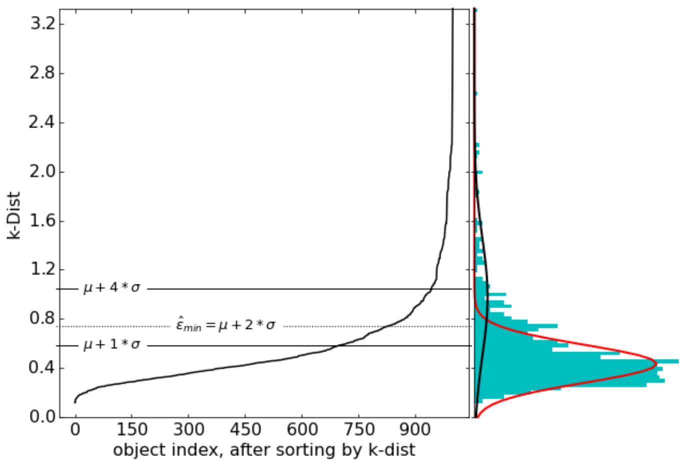


Fig. 3. The k-*dist* graph for *square1* dataset. On the right, the histogram of the $kDistList$ values is reported, and the Gaussian Mixture Model fitting is superimposed. Best viewed in color.

The GaussianMixtureModel-based algorithm (lines 6 and 7) evaluates the parameters of the two Gaussian components that best fit the $kDistList$ array with the expectation-maximization algorithm. Each component is fully described by three parameters, μ, σ and w, which represent the mean, the standard deviation and the weight of the component, respectively. In Fig. 3 the Gaussian fitting is superimposed to the histogram: the high density component (μ_H, σ_H and w_H) is displayed with a solid red line, and the low density component (μ_L, σ_L and w_L) is displayed with a solid black line.

Under the assumption that the high density Gaussian component models the contribution of *dense* cluster objects, we can estimate the value for ε_{min} in terms of its parameters, according to the following equation (line 8):

$$\hat{\varepsilon}_{min} = \mu_H + 2\sigma_H \tag{1}$$

By adopting such value we implicitly establish that the core condition will be met by approximately 98% of the high density Gaussian distribution. The $\hat{\varepsilon}_{min}$ value estimated for *square1* dataset is reported in Fig. 3 as a horizontal dashed line. Notably, according to the Gaussian fitting, we can express a lower bound and an upper bound reasonable values of ε_{min}: horizontal solid lines in Fig. 3 delimit such "elbow" and are drawn at values $\mu_H + \sigma_H$ and $\mu_H + 4\sigma_H$.

The value of the radius $\hat{\varepsilon}_{max}$ cannot be inferred directly from the low density Gaussian component, since it models the contribution of border objects but in general it may be affected by noise and outliers. In general, being it unrelated to the core condition, this parameter just influences the extension of cluster borders. When ε_{max} is set equal to ε_{min}, FDBSCAN-APT traces back to the crisp traditional DBSCAN algorithm; higher values of ε_{max} increase the degree of overlap of cluster borders and reduce the capability of detecting outliers. Hence, the following heuristic is proposed for estimating the value of ε_{max} (line 9):

$$\hat{\varepsilon}_{max} = \alpha \cdot \hat{\varepsilon}_{min} \cdot \frac{\mu_L}{\mu_L - \mu_H} \tag{2}$$

It is expressed as a function of $\hat{\varepsilon}_{min}$ and its value is determined by two coefficients: α is a real number greater than or equal to 1 and can be set by the user if an assumption can be made about the degree of overlap of clusters or about the percentage of noise. The second coefficient is strictly greater than 1 since it holds $\mu_L, \mu_H > 0$ and $\mu_L > \mu_H$. Notably, when a dataset is affected by noise, the contribution of noise will tend to increase the value of μ_L, resulting in a low value of the coefficient. On the other hand, in the absence of noise, the low density Gaussian just models the contribution of border objects: the value of μ_L will be closer to the value of μ_H, thus resulting in a high value of the coefficient. In other words, the heuristic aims to avoid an improper inclusion of noise objects in cluster borders. When noise detection is not a prime concern, high ε_{max} values can be used, adequately tuning the α parameter accordingly. For the purpose of our analysis, α is set to the default value of 1.

B. Computational Complexity

The *regionQuery* procedure is the most influential term in the analysis of the computational complexity of DBSCAN-based algorithms. In FDBSCAN-APT it is implemented as *distanceQuery* in Algorithm 1. As highlighted in [1], [3], [9], when no index support is provided, the complexity of each regionQuery is $\mathcal{O}(n)$, resulting in an overall complexity of $\mathcal{O}(n^2)$. It can be reduced to $\mathcal{O}(n \log n)$ if a spatial indexing structure is adopted. As for the space complexity, the definition of *neighbors* data structure for each object of the dataset implies a cost of $\mathcal{O}(n^2)$.

IV. Experimental Setup

An experimental study has been performed on several synthetic datasets to verify whether FDBSCAN-APT can generate reasonable clustering results. Since ground truth labels are available, the quality of the FDBSCAN-APT outcomes can be assessed by means of an external measure. In this work we adopt the Adjusted Rand Index (ARI) [10], upper-bounded by 1: the higher its value, the better the clustering result.

The goodness of the heuristics for the automatic parameters tuning is evaluated by exploring, with a grid search, the bi-dimensional parameter space defined by ε_{min} and ε_{max}. For what concerns ε_{min}, we explore ten evenly spaced values in the range $[\mu_H + \sigma_H, \, \mu_H + 4\sigma_H]$. As per ε_{max}, we explore the first five multiples of each ε_{min} value.

A. Datasets and Parameter Setting

The datasets used in this work have been obtained from a collection of clustering benchmarks available online[1]. A description of datasets is reported in Table I. Figures 1 and 4 show the representation of the datasets in the bi-dimensional attribute space. To evaluate the effect of noise on the proposed approach, four additional datasets are obtained by adding 1%, 5%, 10%, 20% of noise to the *banana* dataset, respectively.

TABLE I
Datasets description: all datasets are bi-dimensional.

Dataset	Sample	Clusters	Noise %
square1	1000	4	0
banana	4811	2	0
banana_1	4859	2	1
banana_5	5051	2	5
banana_10	5292	2	10
banana_20	5773	2	20
cluto-t4-8k	8000	6	∼10
cluto-t8-8k	8000	8	∼4
aggregation	788	7	0

For all datasets the parameter *MinPts* is set to 4, according to the heuristic proposed by authors in [1]. The parameter ε_{gen} is set as the average value, over 100 repetitions, of the distance of two objects randomly sampled from the dataset. We have verified that for most objects in all datasets such a value is large enough to include more than $minPts$ objects in the ε_{gen}-neighborhood. An alternative heuristics has been proposed in [9] for setting the value of ε_{gen} in OPTICS.

V. Results and Discussion

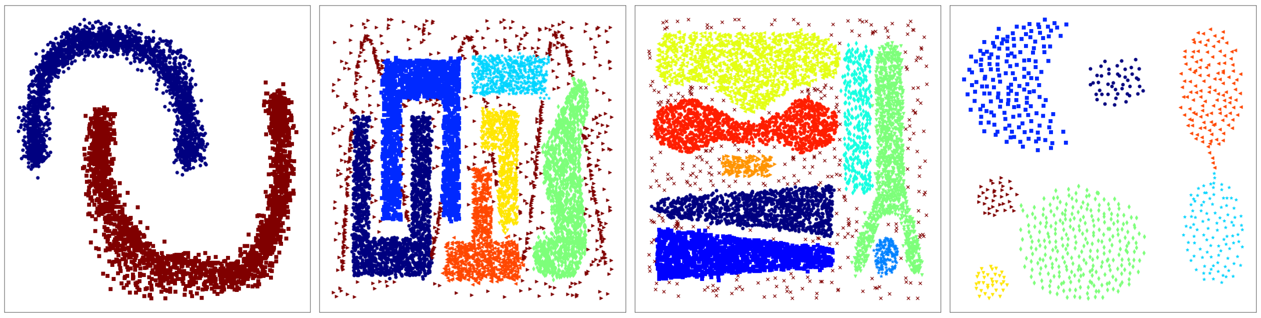
The k-$dist$ graphs of each dataset, along with relative histograms and Gaussian fitting, are reported in Fig. 5. A visual analysis of the figures suggests that our approach allows spotting out reasonable values (i.e. in the 'elbow' region) for parameter ε_{min} for each dataset, independently of the noise percentage. The final parameter configurations adopted in FDBSCAN-APT are reported in Table II.

TABLE II
Automatically inferred parameter setting for FDBSCAN-APT

Dataset	$\hat{\varepsilon}_{min}$	$\hat{\varepsilon}_{max}$
square1	0.742	1.334
banana	0.0094	0.0158
banana_noise1	0.0102	0.0132
banana_noise5	0.0101	0.0123
banana_noise10	0.0099	0.0124
banana_noise20	0.0097	0.0128
cluto-t4-8k	5.009	7.302
cluto-t8-8k	6.901	10.561
aggregation	1.060	4.161

Figure 6 reports a comprehensive assessment of the clustering results in terms of ARI. Each image represents the results of the grid search for a given dataset. Indeed, each pixel in the image represents the ARI obtained by the applyFuzzyDBSCAN procedure with a given parameter configuration. The

[1]https://github.com/deric/clustering-benchmark (last visited May 2020)



(a) banana (b) cluto-t4-8k (c) cluto-t8-8k (d) aggregation

Fig. 4. Datasets used in this paper, in addition to square1 which is depicted in Fig. 1

parameter configuration automatically inferred with the heuristic proposed in FDBSCAN-APT is marked as a red circle; the best configuration among the grid search is marked as a black cross. The last row of each image represents configurations for a classic, crisp, DBSCAN algorithm, with $\varepsilon_{max} = \varepsilon_{min}$. Numerical results are reported in Table III in terms of ARI for FDBSCAN-APT, and for the best configuration over the whole grid (Grid best) and the best configuration for crisp DBSCAN (DBSCAN best).

Figure 6 and Table III suggest that FDBSCAN-APT always obtains reasonable partition ($ARI > 0.8$), and results are comparable to, or even better than, the best parameter configuration found by the grid search. It is worth underlining that grid-based optimization, besides being computationally expensive, requires the availability of class labels and therefore becomes impractical in real clustering applications. Furthermore, Figure 6 shows how crucial the choice of parameters actually is: it is evident that the range of 'good' parameter values for datasets *square1*, *cluto-t4-8k*, *cluto-t8-8k* is quite limited. Figure 6a, for example, confirms the observation reported in Section I: too low values of ε_{min} are unsuited to model the distribution of objects in the four clusters of *square1*; on the other hand, high values of ε_{min} lead to merging of clusters, with a dramatic drop of the ARI value. However, in the three mentioned datasets, FDBSCAN-APT successfully spots out a parameter configuration in the *acceptable* region.

The parameter tuning for the *banana* dataset is less problematic: as it can be observed in Fig. 4a, the inter-cluster distance is much higher than the intra-cluster distance among objects. Indeed a correct match with the ground truth can be obtained by setting high values of ε_{min} (Figures 6b, 6c, 6d and 6e). FDBSCAN-APT, instead, finds a more conservative parameter setting, which still leads to a correct modeling of the two clusters. Notably, as the percentage of noise increases, the optimal parameter configuration shifts towards a region of lower values of ε_{max} and, eventually (Fig. 6f), of lower values of ε_{min}.

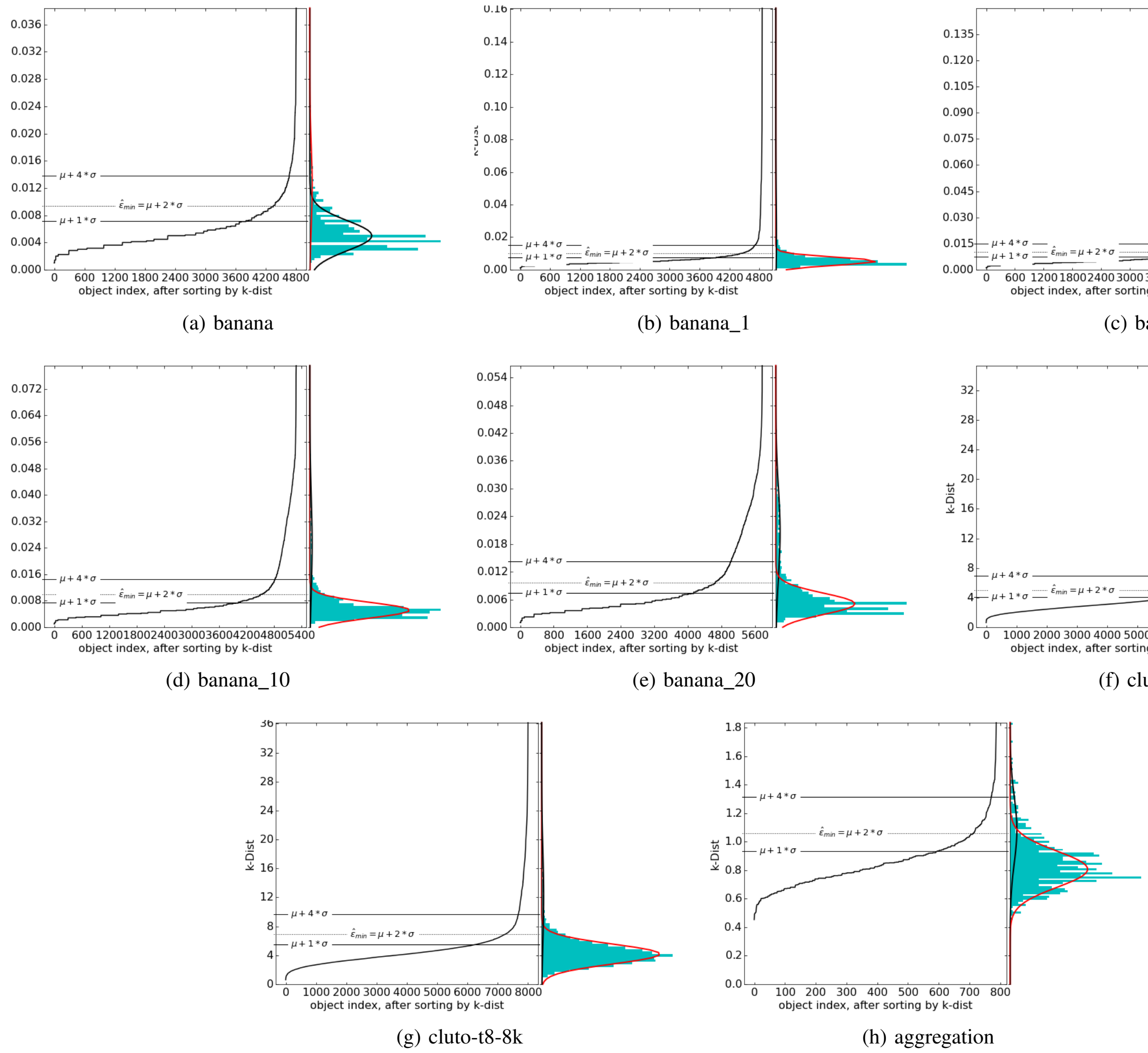
Finally, we can observe how the introduction of fuzzy borders leads in many cases to accurate outcomes, which cannot be obtained with the crisp version of DBSCAN algorithm. This benefit is particularly relevant in scenarios in which the border of clusters overlaps and the distribution of objects is not perfectly uniform within a cluster, as in *square1* dataset and, possibly, in a number of real-world applications. Intuitively, the benefit of fuzzy border decreases in presence of noise, (*cluto* and noisy versions of *banana* dataset): for such datasets the best parameter configuration can be always found in the last row of the respective plot of Fig. 6, where it holds $\varepsilon_{min} = \varepsilon_{max}$.

TABLE III
CLUSTERING RESULTS IN TERMS OF ARI. FDBSCAN-APT RESULTS ARE REPORTED ALONG WITH THOSE OBTAINED WITH THE BEST CONFIGURATION PARAMETER SETTING FOUND BY THE GRID SEARCH (GRID BEST) AND WITH THE BEST PARAMETER SETTING FOR THE CRISP DBSCAN (DBSCAN BEST).

Dataset	Grid best	DBSCAN best	FDBSCAN-APT
square1	0.943	0.853	0.844
banana	0.986	0.967	0.916
banana_1	0.989	0.979	0.931
banana_5	0.956	0.954	0.908
banana_10	0.930	0.929	0.882
banana_20	0.866	0.866	0.840
cluto-t4-8k	0.943	0.943	0.943
cluto-t8-8k	0.898	0.898	0.903
aggregation	0.943	0.905	0.809

VI. CONCLUSION

In this paper, we have proposed FDBSCAN-APT, a fuzzy extension of the DBSCAN clustering algorithm. Inspired to a recently proposed fuzzy DBSCAN algorithm, FDBSCAN-APT can discover clusters with fuzzy overlapping borders by relaxing the constraint on the distance threshold ε and exploiting a membership function based on two thresholds, namely ε_{min} and ε_{max}. In FDBSCAN-APT the crucial issue of input parameter setting is addressed thanks to the definition of a novel heuristic, which is able to automatically infer from data the reasonable values of the two distance thresholds: fixed the value of $minPts$, for each object we exploit a conveniently defined data structure to estimate the distance that let the object satisfy the core condition. Then, the distribution of such distances for all objects in the dataset is modelled as a mixture



(a) banana (b) banana_1 (c) banana_5

(d) banana_10 (e) banana_20 (f) cluto-t4-8k

(g) cluto-t8-8k (h) aggregation

Fig. 5. Visual representation of Gaussian Mixture Model fitting on $k\text{-}dist$ graph data for all datasets (Fig. 3 shows the results on square1 dataset). In each image, on the left the $k\text{-}dist$ graph is reported; the horizontal lines are drawn at values $\mu_H + \sigma_H$, $\mu_H + 4 * \sigma_H$ (boundaries for grid search) and $\mu_H + 2 * \sigma_H$ (distance threshold used for ε_{min} in FDBSCAN-APT). On the right, the histogram of the $kDistList$ values is reported, and the Gaussian Mixture Model fitting is superimposed. Best viewed in color.

of Gaussian functions, and the parameters of the Gaussian components are used to estimate the values of ε_{min} and ε_{max}. The effectiveness of FDBSCAN-APT in producing acceptable clustering results is shown on five synthetic datasets and evaluated in terms of Adjusted Rand Index. Furthermore, the automatically inferred parameter configuration is compared to a set of configurations obtained with a grid search over a range of values of ε_{min} and ε_{max}: results show that the proposed heuristic is effective in finding proper values for distance thresholds, even in presence of artificially added noise. Along with a broader experimental validation on real and big datasets, a possible development of the proposed approach would tackle the issue of parameter setting in multi-density datasets: this will be the focus of our future works.

References

[1] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," in *Proc. of the 2nd Int'l Conf. on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[2] J. Han, J. Pei, and M. Kamber, *Data Mining: concepts and techniques*. Elsevier, 2011.

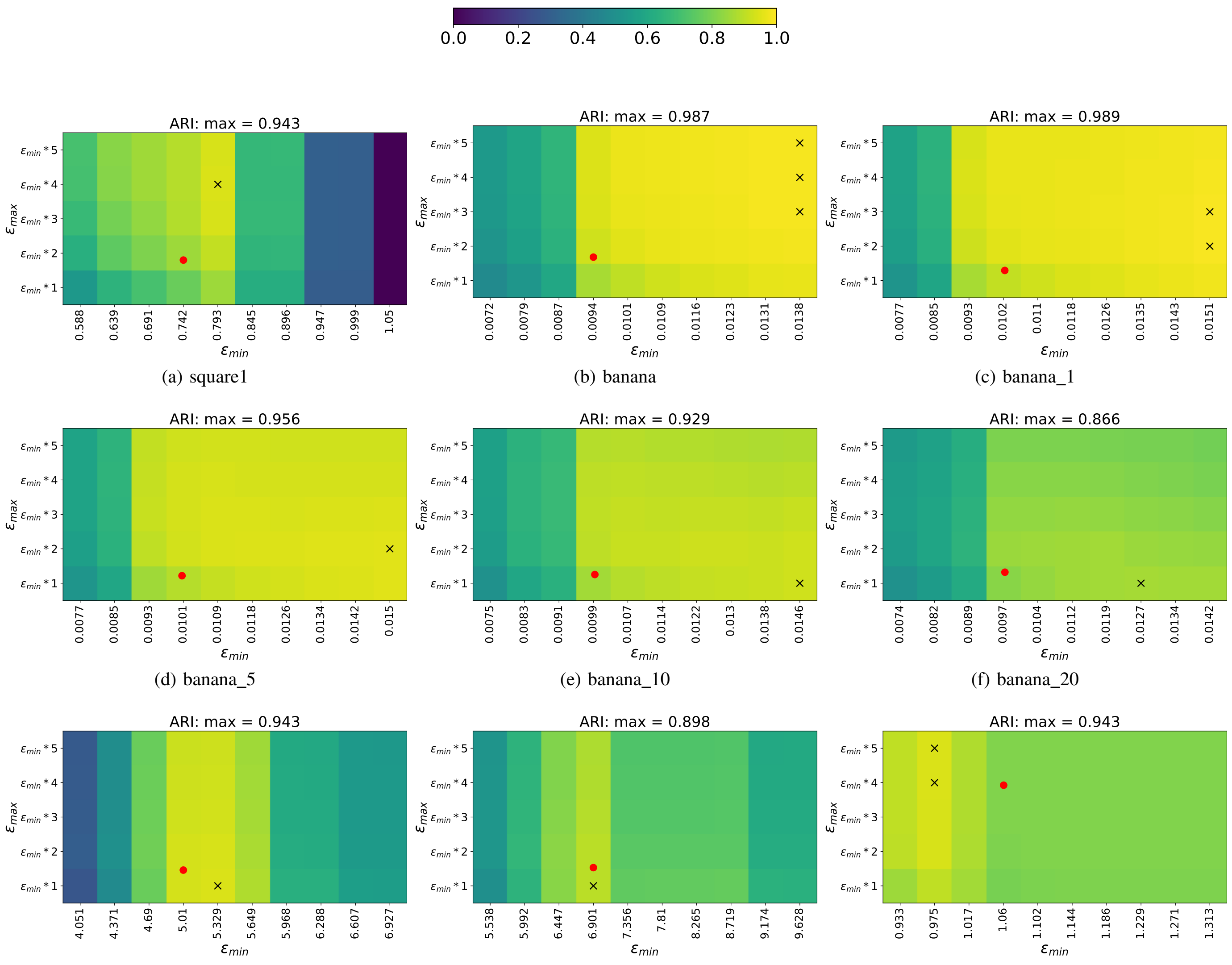
[3] D. Ienco and G. Bordogna, "Fuzzy extensions of the DBSCAN clustering algorithm," *Soft Computing*, vol. 22, no. 5, pp. 1719–1730, 2018.

[4] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 169–194, 1998.

[5] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN," *ACM Transactions on Database Systems (TODS)*, vol. 42, no. 3, p. 19, 2017.

[6] J. Hou, H. Gao, and X. Li, "DSets-DBSCAN: a parameter-free clustering algorithm," *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3182–3193, 2016.

[7] A. Sharma and A. Sharma, "KNN-DBSCAN: Using k-nearest neighbor information for parameter-free density based clustering," in *2017 Int'l Conf. on Intelligent Computing, Instrumentation and Control Technologies (ICICICT)*. IEEE, 2017, pp. 787–792.



(a) square1 (b) banana (c) banana_1

(d) banana_5 (e) banana_10 (f) banana_20

(g) cluto-t4-8k (h) cluto-t8-8k (i) aggregation

Fig. 6. ARI results of FDBSCAN-APT on synthetic datasets: for each dataset the best parameter configuration of the grid search is marked with a black cross; the configuration automatically selected with FDBSCAN-APT is marked with a red circle. Best viewed in color.

[8] J. Kim, J. Choi, K. Yoo, and A. Nasridinov, “AA-DBSCAN: an approximate adaptive DBSCAN for finding clusters with varying densities,” *Journal of Supercomputing*, vol. 75, no. 1, pp. 142–169, 2019.

[9] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, “OPTICS: ordering points to identify the clustering structure,” in *ACM Sigmod record*, vol. 28, no. 2. ACM, 1999, pp. 49–60.

[10] L. Hubert and P. Arabie, “Comparing partitions,” *Journal of classification*, vol. 2, no. 1, pp. 193–218, 1985.