Soft Comput (2018) 22:1719–1730
https://doi.org/10.1007/s00500-016-2435-0

METHODOLOGIES AND APPLICATION

Fuzzy extensions of the DBScan clustering algorithm

Dino Ienco[1,2] · **Gloria Bordogna**[3]

Published online: 15 November 2016

Abstract The DBSCAN algorithm is a well-known density-based clustering approach particularly useful in spatial data mining for its ability to find objects' groups with heterogeneous shapes and homogeneous local density distributions in the feature space. Furthermore, it can be suitable as scaling down approach to deal with big data for its ability to remove noise. Nevertheless, it suffers for some limitations, mainly the inability to identify clusters with variable density distributions and partially overlapping borders, which is often a characteristics of both scientific data and real-world data. To this end, in this work, we propose three fuzzy extensions of the *DBSCAN* algorithm to generate clusters with distinct fuzzy density characteristics. The original version of *DBSCAN* requires two precise parameters ($minPts$ and ϵ) to define locally dense areas which serve as seeds of the clusters. Nevertheless, precise values of both parameters may be not appropriate in all regions of the dataset. In the proposed extensions of *DBSCAN*, we define soft constraints to model approximate values of the input parameters. The first extension, named *Fuzzy Core DBSCAN*, relaxes the constraint on the neighbourhood's density to generate clusters with fuzzy core points, i.e. cores with distinct density; the second extension, named *Fuzzy Border DBSCAN*, relaxes ϵ to allow the generation of clusters with overlapping borders. Finally, the third extension, named *Fuzzy DBSCAN* subsumes the previous ones, thus allowing to generate clusters with both fuzzy cores and fuzzy overlapping borders. Our proposals are compared w.r.t. state of the art fuzzy clustering methods over real-world datasets.

Keywords Fuzzy clustering · Density-based clustering · DBSCAN clustering

Communicated by V. Loia.

✉ Dino Ienco
dino.ienco@irstea.fr

Gloria Bordogna
bordogna.g@irea.cnr.it

[1] IRSTEA, UMR TETIS, Montpellier, France

[2] LIRMM, Montpellier, France

[3] CNR IREA, Milan, Italy

1 Introduction

The advent of the big data era has launched new challenges to the research community who reacted either by introducing new algorithms or by extending existing algorithms to manage large datasets. Specifically, the first approaches focus on the "scaling up" objective to deal with big data sets. Nevertheless, they risk to become useless in a short time, due the data continuous growth. CISCO[1] estimated that the data on the Internet will increase at a compound annual growth rate of 25% by the year 2017. Thus, to deal with datasets continuously growing in size it will be necessary to frequently scale up algorithms. The second kind of approach aims at scaling down, i.e. at synthesizing, the data sets by reducing their size, and to use existing algorithms on the reduced data. Although scaling down may risk to cancel important information it has the chance of reducing the datasets by eliminating noise or redundant data. Clustering techniques can be categorized as scaling down approaches, since their objective is to identify groups of items within the dataset with common characteristics in a feature space, while removing outliers and noise which are considered uninteresting for further analysis.

[1] http://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/Cloud_Index_White_Paper.html.

Many real-world applications such as social network community identification and satellite image analysis need effective means to identify regions that are characterized by locally dense areas in the feature space representing the objects of interests. For instance, such regions may represent communities of users linked by the friend of friend relationships in social networks, ecosystems may appear as regions characterized by homogeneous feature' values in satellite images. To detect such objects, density-based clustering algorithms have been widely applied. They evaluate a local criterion to group objects: clusters are regarded as regions in the feature space where the objects are densely distributed and separated by regions with sparse objects, which are considered noise or outliers.

Indeed, in many real applications, such as in satellite image analysis, one needs to cope with noise invariably affecting data. Furthermore, one does not have any knowledge about the number of clusters, the possible clusters' shapes and the objects distribution in the feature space. Finally, crisp clustering algorithms fail to detect the variable and fuzzy nature of cluster borders which are often faint and overlapped one another. Among the proposed crisp density-based clustering algorithms *DBSCAN* (Ester et al. 1996) is a well-known and widely applied approach as it does not require to specify in input the number of clusters, it can detect clusters of any shape, and it can remove noisy points. Furthermore, this algorithm is suitable to process big data when adopting a spatial index, such as an R-tree, since its complexity varies as $O(N * \log N)$ (Sander et al. 1998).

Nevertheless, it suffers for some drawbacks: first, to drive the process, this algorithm needs two numeric input parameters, *minPts*, i.e. the neighbourhood density, and ϵ, i.e. the distance to define the local neighbourhood size, which together define the desired local density of the generated clusters. Specifically, *minPts* is a positive integer specifying the minimum number of objects that must exist within a maximum distance ϵ in the feature space in order for an object to belong to a cluster. Second, the results of *DBSCAN* are strongly dependent on the setting of these input parameters that must be chosen with great accuracy (Ester et al. 1996) by considering both the scale of the dataset and the closeness of the objects in order to achieve both speed and effectiveness of the results. To set the right values of these parameters one generally engages a trial and error exploratory phase in which the algorithm is run several times with distinct values of the input parameters. These repeated trials are costly when dealing with big data volumes. A final drawback of *DBSCAN* is that it detects clusters with sharp boundaries, which is a common limitation of all crisp clustering algorithms when used to group objects whose distribution has a faint and smooth density profile in the feature space. They draw crisp boundaries to separate clusters, which are often somewhat arbitrary. To cope with undesired crisp boundaries, soft clustering approaches have been defined which generate clusters with fuzzy overlapping boundaries (Pal et al. 2005; Ji et al. 2014; Yager and Filev 1994). Most of the soft clustering approaches detect fuzzy clusters with same shape, with each object of the dataset belonging to all clusters to a distinct degree. Moreover, even the fuzzy extensions of *DBSCAN* generate fuzzy clusters with the same characteristics of fuzziness, i.e. all clusters have same fain borders (Ulutagaya and Nasibov 2012; Smiti and Eloudi 2013). In this paper, we investigate new extensions of the *DBSCAN* algorithm, defined within the framework of fuzzy set theory, with the aim to cope with the limitation of both classic *DBSCAN* and soft clustering algorithms. The idea is to define distinct *DBSCAN* extensions capable to manage approximate values of the input parameters, thus less sensible to the input parameter setting, and capable to detect possibly fuzzy overlapping clusters with distinct density characteristics and profiles.

There are several real applications in which it could be useful specifying approximate values of the input parameters and detecting fuzzy clusters with both distinct shapes and distinct density profiles. Consider the detection of communities of users in a social network based on the *FoF* relationships: while one can specify easily that the users must have a given number of degrees of separation from other users on the network to belong to the community, it may be questionable defining the precise minimum number of users that constitutes a community. In this case, it can be useful to apply a clustering algorithm, as in our first extension of *DBSCAN* in Bordogna and Ienco (2014), by allowing the specification of an approximate density, i.e. an approximate number of users and detecting non-overlapping fuzzy communities where a user can belong to a community to a degree.

On the other side, in the case, one has to detect stars and galaxies in astronomical optical images, appearing with a crisp nucleus and faint borders, it can be easier to specify an approximate local neighbourhood size, as in our second extension of *DBSCAN* proposed in this paper, and thus detecting objects with crisp core and faint border.

Last but not least, there are applications in which objects are characterized by distinct local densities and faint, possibly overlapping, borders, such as in remote sensing images, where distinct ecosystems have distinct densities of trees (objects) and they may appear merged one another; in this case, it would be useful to allow specifying both an approximate neighbourhood density, and an approximate local neighbourhood size to generate fuzzy overlapping clusters as in our third extension of *DBSCAN*.

In the literature, several fuzzy extensions of *DBSCAN* have been proposed with the objective of leveraging the setting of the precise input parameters (Ulutagaya and Nasibov 2012). Nevertheless, none of them have tackled the objective of generating fuzzy clusters modelling distinct kind of fuzziness as we do in this paper. We leverage the setting of either one or

both the input parameters of *DBSCAN* by allowing the specification of soft constraints on both the number of objects and the closeness (reachability) between objects. Specifically, the precise value *minPts* is replaced by a soft constraint defined by a pair $(minPts_{min}, minPts_{max})$ that specifies an approximate minimum number of objects for defining a cluster, i.e. there is a tolerance on the crisp limit $minPts_{max}$ defined by $minPts_{max} - minPts_{min}$; in the same way, the precise distance ϵ, is replaced by a soft constraints $(\epsilon_{min}, \epsilon_{max})$ on the closeness of objects so that again on the crisp limit ϵ_{min} we have a tolerance defined by $\epsilon_{max} - \epsilon_{min}$.

The three extensions of *DBSCAN* generate clusters with either a fuzzy core, i.e. clusters whose elements are associated with a numeric membership degree in [0,1] but not overlapping one another, clusters with fuzzy overlapping border and a crisp core, and clusters with both fuzzy core and overlapping borders. Having three extensions producing clusters with distinct fuzzy and overlapping properties one can choose the most appropriate for task to accomplish.

Furthermore, fuzzy clusters allow several advantages: for instance, with a single run of the clustering it is possible to summarize several distinct runs of the original approach by specifying distinct thresholds on the membership degrees of the objects to the clusters. For this reason, it can be employed as intelligent reduction strategy for big data. In our case, this allows an easy exploration of the spatial distribution of the objects avoiding the tedious exact setting of the *DBSCAN* parameters (Ester et al. 1996).

The paper is organized as follow: Sect. 2 discusses related work, in Sect. 3 we recall the classic *DBSCAN* algorithm. The clustering algorithm generating clusters with fuzzy core points *Fuzzy Core DBSCAN*, firstly introduced in Bordogna and Ienco (2014), the extension generating clusters with fuzzy overlapping borders *Fuzzy Border DBSCAN* and the most general strategy generating fuzzy overlapping clusters (*Fuzzy DBSCAN*) are introduced in Sect. 4.

After the definitions of the three algorithms, Sect. 6 discusses and compares the performance of the different approaches over real-world data sets in comparison with those yielded by Fuzzy C-Means and *Soft-DBSCAN* fuzzy clustering algorithms. Section 7 concludes and summarizes the main achievements.

2 Related work

The relevant works to our proposal are those related to the literature on soft density-based clustering algorithms. Soft clustering algorithms are modelled within either fuzzy set, probability theory or possibilistic typicalities to allow assigning objects to clusters with a full or a partial membership degree, in this latter case with the possibility for an object to simultaneously belong to several clusters (Ji et al. 2014; Pal et al. 2005).

Density-based clustering algorithms grow clusters around seeds located in regions of the feature space which are locally dense of objects. *DBSCAN* (Ester et al. 1996) is one of the most popular density-based methods used in data mining due to both its ability to detect irregularly shaped clusters by copying with noise data, and to its relatively low complexity that varies as $O(N * \log N)$ when adopting a spatial index, thus making it suitable to process big data (Sander et al. 1998). Nevertheless, its effectiveness in detecting clusters is strongly dependent on the parameters setting, and this is the main reason that leads to its soft extensions. Besides this motivation we argue that, in order to properly adopt a soft density-based clustering approach with respect to another one, one should be able to understand the properties of the generated soft clusters. This is the reason that leads us to define three distinct extensions of *DBSCAN* each one generating fuzzy clusters with distinct characteristics.

Ulutagaya and Nasibov (2012) reports a survey of the main fuzzy density-based clustering algorithms, while Shamshirband et al. (2014) presents a study in which they show that density-based clustering algorithm coupled with fuzzy logic can efficiently deal with the task of intrusion detection in wireless sensor networks.

The most cited paper (Nasibov and Ulutagay 2009) proposes a fuzzy extension of the *DBSCAN*, named fuzzy neighbourhood *FN-DBSCAN*, whose main characteristic is to use a fuzzy neighbourhood size definition. In this approach, the authors address the difficulty of the user in setting the values of the input parameter ϵ when the distances of the points are in distinct scales, as it happens in astronomical images. Thus, they first normalize the distances between all points in [0, 1], and then they allow computing distinct membership degrees on the distance to delimit the neighbourhood of points, i.e. the decaying of the membership degrees as a function of the distance. Then, they select as belonging to the fuzzy neighbourhood of a point only those points belonging to the support of the membership function. This extension of *DBSCAN* uses a level-based neighbourhood set, instead of a distance-based neighbourhood size, and it uses the concept of fuzzy cardinality, instead of classical cardinality, for identifying core points. This last choice causes the creation (within the same run of the algorithm) of fuzzy clusters with very heterogeneous local density characteristics: both fuzzy clusters with cores having a huge number of sparse points (points located at the border of the local neighbourhood of each other), and fuzzy clusters with small cores, constituted only by a few close points. This approach can be considered dual to our first extension, the *Fuzzy Core DBSCAN* algorithm (Bordogna and Ienco 2014), since we fuzzify the minimum number of points $minPts$ defining the local neighbourhood density, while the distance ϵ is maintained crisp.

As a consequence, the membership degree of a point to the fuzzy core depends on the number of points in its crisp neighbourhood. By this choice, and the computation of the local density based on the classic set cardinality, a point is assigned to only one cluster to a distinct extent, thus generating non-overlapping clusters with possibly fuzzy cores. Clusters may have cores with faint profiles reflecting low density of the clusters' nucleus.

FNDBSCAN (Parker and Downs 2013) is closer to our second extension, named *FBorder*, in which we fuzzify only the membership of objects belonging to the border of clusters, this way allowing their partial overlapping. Nevertheless, differently than *FNDBSCAN*, *FBorder* grows clusters' cores around points characterized by homogeneous local density, thus generating clusters with crisp, not overlapping and homogeneously dense cores.

Kriegel and Pfeifle (2005) algorithm is employed to cluster objects whose position is ill-known. The authors propose the *FDBSCAN* algorithm in which a fuzzy distance measure is defined as the probability that an object is directly density-reachable from another objects. This problem can be modelled by our third extension, named *FDBScan*, that allows defining the local neighbourhood density of any object by specifying an approximate number of objects within an approximate maximum distance, thus capturing the uncertainty on the positions of the moving objects, and generating fuzzy clusters with both faint cores and fuzzy overlapping border. Finally, the most recent soft extension of *DBSCAN* has been proposed in Smiti and Eloudi (2013) where the authors combine the classic *DBSCAN* with the Fuzzy C-Means algorithm (Bezdek et al. 1984) proposing a method called *soft-DBSCAN*. They detect seeds points by the classic *DBSCAN* and in a second phase they compute the degrees of membership to the clusters around the seeds by relying on the Fuzzy C-Means clustering algorithm. A similar objective of selecting the seeds to feed the Fuzzy C-Means is pursued by the mountain method proposed in Yager and Filev (1994).

Nevertheless, these extensions do not grow the clusters by applying density-reachable criteria as in our proposed approaches. Distinct density characteristics of clusters: faint cores and not overlapping distributions are modelled by *Fuzzy Core DBSCAN*; semi-overlapping distributions with homogeneous dense cores by our *Fuzzy Border DBSCAN* extension; finally, faint cores and semi-overlapping distributions by the third extension *Fuzzy DBSCAN*.

Another important issue when using a clustering algorithm on big data is its scalability. In this respect, Parker et al. (2010) proposes a scalable implementation of the *FN-DBSCAN*, named *SFN-DBSCAN*, with the objective of improving the efficiency when dealing with big data sets. Another efficient implementation is proposed in Ester et al. (1996). It tackles the problem of clustering a huge number of objects strongly affected by noise when the scale distributions of objects are heterogeneous. To remove noise they first map the distance of any point from its k-neighbours and rank the distance values in decreasing order; then they determine the threshold θ on the distance which corresponds to the first minimum on the ordered values. All points in the first ranked positions having a distance above the thresholds θ are deemed noisy points and are removed, while the remaining points will belong to a cluster. Only these latter points are clustered with the classic *DBSCAN* providing as input parameters $minPts = K$ and $\epsilon = \theta$. By adopting this same procedure, we can implement the proposed algorithms: we can determine the most appropriate distance $\epsilon_M ax = \theta$ (which delimits the support of the membership function defining the approximate size of the local neighbourhood). This way, depending on the data set, we remove noise and then apply one of the proposed algorithms on the remaining points.

Finally, the extension of *DBSCAN* with fuzzy logic reported in Shamshirband et al. (2014) shares with our extension the idea of generating clusters with distinct fuzziness properties, as specified by the fuzzy rules: specifically, a hybrid clustering method is introduced, namely a density-based fuzzy imperialist competitive clustering algorithm (D-FICCA), to detect malicious behaviours in wireless sensor networks (WSNs) with the aim to enhance the accuracy of malicious detection. A density-based clustering algorithm helps to improve the imperialist competitive algorithm for the formation of arbitrary cluster shapes as well as handling noise. The fuzzy logic controller is introduced to avoid possible errors of the worst imperialist action selection strategy. The results demonstrate that the proposed framework achieves higher detection accuracy compared to existing approaches.

3 Classic DBScan algorithm

For sake of clarity in the following, we will consider a set of objects represented in multidimensional feature space. We can figure out these objects as either cars, taxi cabs, airplanes represented in the feature space defined by their geographic coordinates (both 2D or 3D). *DBSCAN* can be applied to group these objects based on their local densities in the feature space. For example, this makes it possible to identify traffic jams of cars on the roads.

Specifically, *DBSCAN* assigns points of the feature space defined on $R \times R \times R \cdots \times R$ to particular clusters or designates them as outliers or noise if they are not sufficiently close to other points. It determines cluster assignments by assessing the local density at each point using two parameters: distance radius (ϵ) and minimum number of points ($minPts$) that must exists within the neighbourhood ϵ of the point. A single point which meets the minimum density criterion, namely it has $minPts$ located within distance

ϵ, is designated a *core point*. Formally, given a set of points $P = (p_1, p_2, \ldots, p_n)$, p is a core point if at least a minimum number $minPts$ of points exist s.t $p_j \in P$ and $||p - p_j|| < \epsilon$, where $||x||$ is the Euclidean distance in the n-dimensional feature space. Two core points p_i and p_j with $i \neq j$ belong to the same cluster c if $||p_i - p_j|| < \epsilon$. Both are core points of c ($p_i, p_j \in core(c)$). All the points that are not core points and lie within the maximum distance ϵ from a core point of a cluster c are defined as border points of c: $p \notin core(c)$ is a border point of c if $\exists p_i \in core(c)$ with $||p - p_i|| < \epsilon$. Finally, the points that are not part of any cluster are considered noisy points: $p \notin core(c)$ is noise if $\forall c, \nexists p_i \in core(c)$ with $||p - p_i|| < \epsilon$. In the following, the classic *DBSCAN* algorithm is formalized:

Algorithm 1 ***DBSCAN***$(P, \epsilon, MinPts)$

Require: P: dataset of points
Require: ϵ: the maximum distance around a point defining its local neighbourhood
Require: $MinPts$: minimum local density, in points, around a point to be a candidate core point
1: $C = 0$
2: $Clusters = \emptyset$
3: **for all** $p \in P$ s.t. p is unvisited **do**
4: mark p as visited
5: neighboursPts = regionQuery(p,ϵ)
6: **if** ($sizeof(neighboursPts) <= MinPts$) **then**
7: mark p as NOISE
8: **else**
9: C = next cluster
10: $Clusters = Clusters \cup expandCluster(p, neighboursPts, C, \epsilon, MinPts)$
11: **end if**
12: **end for**
13: **return** $Clusters$

Algorithm 2 $expandCluster(p, neighboursPts, C, \epsilon, MinPts)$

Require: p: the point just marked as visited
Require: $neighboursPts$: the neighbourhood of p
Require: C: the actual cluster
Require: ϵ the distance around a point to compute its local density
Require: $MinPts$: local density, in points, defining the minimum cardinality of the neighbourhood of a point to be a candidate core point
1: add p to cluster C
2: **for all** $p' \in neighboursPts$ **do**
3: **if** p' is not visited **then**
4: mark p' as visited
5: $neighboursPts'$ = regionQuery(p',ϵ)
6: **if** $sizeof(neighboursPts') > MinPts$ **then**
7: $neighboursPts = neighboursPts \cup neighboursPts'$
8: **end if**
9: **end if**
10: **if** p' is not yet member of any cluster **then**
11: add p' to cluster C
12: **end if**
13: **end for**
14: **return** C

4 Generating clusters with distinct fuzzy characteristics

4.1 Generating clusters with fuzzy cores

The first extension of the classic *DBSCAN* algorithm we proposed in Bordogna and Ienco (2014), named *Fuzzy Core DBSCAN*, for short *FCore*, is obtained by considering crisp distances and by introducing an approximate value of the minimum cardinality of the local neighbourhood of a point. This can be done by substituting the value $minPts$ with a soft constraint defined by a monotonic non-decreasing membership function on the domain of the positive integers. This soft constraint specifies the minimum approximate number of points that are required in the local neighbourhood of a point for starting the generation of a fuzzy core. Let us define the piecewise linear membership function as follows:

$$\mu_{minP}(\hat{n}) \begin{cases} 1, & \text{if } \hat{n} \geq Mpts_{Max} \\ \frac{\hat{n} - Mpts_{Min}}{Mpts_{Max} - Mpts_{Min}}, & \text{if } Mpts_{Min} < \hat{n} < Mpts_{Max} \\ 0, & \text{if } \hat{n} \leq Mpts_{Min} \end{cases} \quad (1)$$

This membership function gives the value 1 when the number $\hat{n}$ of elements in the neighbourhood of a point is greater than $Mpts_{Max}$, a value 0 when $\hat{n}$ is below $Mpts_{Min}$ and intermediate values when $\hat{n}$ is in between $Mpts_{Min}$ and $Mpts_{Max}$.

Since users may find it difficult to specify the two values $Mpts_{Min}$ and $Mpts_{Max}$ in the case of big data, we can try to automatically suggest two appropriate values. This can be done by mapping the number of points of the data sets which are at a maximum distance among each other below ϵ, for increasing values of ϵ. This function is monotonically not decreasing: we then suggest the values of the functions corresponding to the first two flexes as the appropriate values of $Mpts_{Min}$ and $Mpts_{Max}$.

Another approach is to allow a user to specify two percentage values, $\%Mpts_{Min}$ and $\%Mpts_{Max}$ on the total dataset size, measured in number of objects, and then convert these percentages to determine $Mpts_{Min}$ and $Mpts_{Max}$ as follows:

$Mpts_{Min} = \text{round}(\%Mpts_{Min} * N)$ and $pts_{Max} = \text{round}(\%Mpts_{Max} * N)$, in which N is the total number of objects in the data set and $round(m)$ returns the closest integer to m.

Let us now define the fuzzy core. Considering a set P of N objects represented by N points $p_1, p_2, \ldots, p_N$ in the n-dimensional space R^n, so that each p_i has the coordinates $x_{i_1}, x_{i_2}, \ldots, x_{i_n}$.

Given a point $p \in P$, if $\hat{n}$ points p_i exist in the local neighbourhood of point p, i.e. with $\|p_i - p\| < \epsilon$, s.t. $\mu_{minP}(\hat{n}) > 0$ then p is a fuzzy core point with membership degree to the fuzzy core given by $Fuzzycore(p) = \mu_{MinP}(\hat{n})$ If two fuzzy core points p_i, p_j with $Fuzzycore(p_i) > 0$ and $Fuzzycore(p_j) > 0$ $\exists$ with $i \neq j$ s.t. $\|p_i - p\| < \epsilon$ then they belong to the same cluster c ($p_i, p_j \in c$) and both are fuzzy core points of c, ($p_i, p_j \in fuzzycore(c)$) with membership degrees $fuzzycore_c(p_i) = Fuzzycore(p_i)$ and $fuzzycore_c(p_j) = Fuzzycore(p_j)$. They belong to the cluster with membership degree $\mu_c(p_i) = Fuzzycore(p_i)$ and $\mu_c(p_j) = Fuzzycore(p_j)$.

A point p of a cluster c is a border point if it is not a fuzzy core point and $\exists p_i \in fcore(c)$ s.t. $\|p_i - p\| < \epsilon$ then p gets a membership degree to c defined as:

$$\mu_c(p) = min_{p_i \in neighcore(p)} fuzzycore_c(p_i) \quad (2)$$

where $neighcore(p) = \{p_i \text{ s.t. } fuzzycore_c(p_i) > 0 \wedge \|p_i - p\| < \epsilon\}$.

Finally, points p that are neither fuzzy core points nor border points are considered as noisy points

Notice that, the points belonging to a cluster c get distinct membership values to the cluster reflecting the number of their neighbours within a maximum distance ϵ. This definition allows generating fuzzy clusters with a fuzzy core, where the membership degrees represent the variable cluster density.

Moreover, a border point p can partially belong to a single cluster c since its membership degree is upperbounded by the minimum membership degree of its neighbours fuzzy core points. Notice that, this algorithm does not generate overlapping fuzzy clusters, but the support of the fuzzy clusters is still a crisp partition as in the classic *DBSCAN*: $c_i \cap c_j = \oslash$

Further property, the *FCore* DBSCAN reduces to the classic DBSCAN when the input values $Mpts_{Min} = Mpts_{Max}$: in this case *FCore* DBSCAN produces the same results of the classic *DBSCAN* with $minPts = Mpts_{Min} = Mpts_{Max}$ and same distance ϵ. In fact, the level-based soft condition imposed by μ_{minP} is indeed a crisp condition $\mu_{MinP}(x) \in \{0, 1\}$ on the minimum number of points defining the local density of the neighbourhood: $\mu_{minP}(\hat{n}) = 0$ when the number of points $\hat{n}$ within a maximum distance ϵ of any point p is less than $minPts = Mpts_{Min} = Mpts_{Max}$, on the contrary $\mu_{minP}(\hat{n}) = 1$. In this case, the membership degrees of all fuzzy core points is 1, and thus the fuzzy core reduces to a crisp core as in the classic *DBSCAN*.

The border points are thus defined as in the classic approach too, since their membership degrees are the minimum of the membership degrees of the core points in their neighbourhood, which in the crisp case is always 1.

The fuzzy procedure is sketched in Algorithms 3 and 4. Considering the outer loop of the process (Algorithm 3), the difference with the original version (Algorithm 1) lies at line 6.

In the fuzzy version, a point is marked as *NOISE* if its neighbourhood size is less than or equal to $Mpts_{Min}$, otherwise it will be a fuzzy core point with a given membership value. Once the point is recognized as fuzzy core point the procedure $expandClusterFuzzyCore$ is called (Algorithm 4).

As in the classical *DBSCAN*, this procedure is devoted to find all the reachable points from p and to mark them as core or border points. In the original version, the assignment of the point p is crisp, while we introduce a fuzzy assignment (line 1) modelled by the fuzzy function $\mu_{MinP}()$ defined in Eq. 1. The same function is employed when a new fuzzy core point is detected (line 8). Also in this case, firstly we verify the density around a given point p' w.r.t. $Mpts_{Min}$ and then, if the point satisfies the soft constraint to a positive degree, we add the point to the fuzzy core of cluster C with its associated membership value.

Algorithm 3 ***Fuzzy Core DBSCAN***$(P, \epsilon, Mpts_{Min}, Mpts_{Max})$

Require: P: dataset of points
Require: ϵ: the maximum distance around a point defining the point neighbourhood
Require: $Mpts_{Min}, Mpts_{Max}$: soft constraint on the density around a point to be a candidate fuzzy core point
1: $C = 0$
2: $Clusters = \emptyset$
3: **for all** $p \in P$ s.t. p is unvisited **do**
4: mark p as visited
5: nPts = regionQuery(p,ϵ)
6: **if** $(sizeof(nPts) \leq Mpts_{Min})$ **then**
7: mark p as NOISE
8: **else**
9: C = next cluster
10: $Clusters = Clusters \cup expandClusterFuzzyCore(p, nPts, C, \epsilon, Mpts_{Min}, Mpts_{Max})$
11: **end if**
12: **end for**
13: **return** $Clusters$

Algorithm 4 $expandClusterFuzzyCore(p, nPts, C, \epsilon, Mpts_{Min}, Mpts_{Max})$

Require: p: the point just marked as visited
Require: $nPts$: the points in the neighbourhood of p
Require: C: the actual cluster
Require: ϵ the distance around a point to compute its density
Require: $Mpts_{Min}, Mpts_{Max}$: soft constraint on the density around a point to be a candidate fuzzy core point
1: add p to C with membership $Fuzzycore(p) = \mu_{MinP}(|nPts|)$
2: **for all** $p' \in nPts$ **do**
3: **if** p' is not visited **then**
4: mark p' as visited
5: $nPts'$ = regionQuery(p',ϵ)
6: **if** $sizeof(nPts') > Mpts_{Min}$ **then**
7: $nPts = nPts \cup nPts'$
8: add p' to C with membership $Fuzzycore(p') = \mu_{MinP}(|nPts'|)$
9: **end if**
10: **if** p' is not yet member of any cluster **then**
11: add p' to C (as border point)
12: **end if**
13: **end if**
14: **end for**
15: **return** C

4.2 Generating clusters with overlapping fuzzy border and classic core points

A second extension of *DBSCAN*, named *Fuzzy Border DBSCAN* (*FBorder*), can be defined by allowing the specification of an approximate value of the maximum

distance instead of asking for a precise numeric parameter ϵ and in defining a soft constraint with a monotonic not increasing membership function on the positive real domain of distance values. The soft constraint defines the concept of fuzzy neighbourhood size, so that a point can belong to the fuzzy neighbourhood of another point to a degree in (0,1]. This allows computing a gradual membership to the clusters.

Differently than the proposal of Nasibov and Ulutagay (2009) we allow to specify the membership function on the distance as a soft constraint with piecewise linear shape defined by two values ϵ_{Min} and ϵ_{Max} so that when the distance is smaller than ϵ_{Min} the membership degree is maximum (1), when it is greater than ϵ_{Max} its membership is null (0) and it decreases linearly when it is in between ϵ_{Min} and ϵ_{Max}:

$$\mu_{dist}(p, pi) = \begin{cases} 1, & \text{if } \|p - p_i\| \leq \epsilon_{Min} \\ \frac{\epsilon_{Max} - \|p - p_i\|}{\epsilon_{Max} - \epsilon_{Min}}, & \text{if } \epsilon_{Min} < \|p - p_i\| < \epsilon_{Max} \\ 0, & \text{if } \|p - p_i\| > \epsilon_{Max} \end{cases} \tag{3}$$

In this definition, $\|p - p_i\|$ can be defined as either the Euclidean distance or the complement of a cosine similarity distance or any other distance measure more suitable in the application context.

We can then redefine a core point of a cluster with fuzzy border: given a point p if at least a number $minPts$ of points $\mathrm{P} = \{p_1, \ldots, p_{minPts}\} \exists$ s.t. $\forall p_i \in P, \mu_{dist}(pi, p) = 1$ then p is a core point.

If two core points $p_i p_j$ with $i \neq j$ and $\mu_{dist}(p_i, p_j) = 1$ then p_i, p_j belongs to c, i.e. they define a cluster c with fuzzy border and are core points of c, i.e. $p_i, p_j \in core(c)$ and thus they get a membership degree to the cluster $\mu_c(p) = 1$.

A point p of a cluster that is not a core point is a fuzzy border point if it satisfies the following: $\forall p$ s.t. $p \notin core(c)$ *and* $p_i \in core(c)$ *and* $\mu_{dist}(p_i, p) > 0$ then p gets a membership degree to the fuzzy border of cluster c defined as:

$$\mu_c(p) = min_{p_i \in neighcore(p)} \mu_{dist}(p, p_i) \tag{4}$$

where $neighcore(p) = \{p_i \in core(c) \wedge \mu_{dist}(p, p_i) > 0\}$

This definition allows generating fuzzy clusters with faint borders.

Moreover, a point p can partially belong to the fuzzy borders of more clusters at the same time with distinct membership values. This allows generating fuzzy clusters with overlapping boundaries, i.e. semi-overlapping fuzzy clusters. This is guaranteed by the condition for the selection of the points to be evaluated as border points of clusters which requires that $\mu_c(p) < 1$ for each generated c. While a point p is considered as noise if $\forall c \nexists p_i \in core(c)$ s.t. $\mu_{dist}(p_i, p) > 0$.

The strategy is outlined in Algorithm 5 and 6. The outer loop (Algorithm 5) starts the process. Given a point, the neighbourhood is selected considering ϵ_{Min}. If the $MinPts$ constraint is not satisfied the point is initially marked as NOISE otherwise the creation of a new cluster begins, and the procedure $expandClusterFuzzyBorder$ is called. Algorithm 6 tries to expand the current cluster C as much as possible. The difference with the original version of *DBSCAN* lies in the way the border points are managed and detected. Here, we employ a temporary structure $fuzzyBorderPts$ to collect the current set of border points. Border points are points with density lower than $MinPts$ (line 6) but, differently from the original algorithm, a point can be a border point if it is reasonably at a distance from the cluster in between ϵ_{Min} and ϵ_{Max}. To verify this second condition, we query the neighbourhood of a point p for both ϵ_{Min} and ϵ_{Max} distances (line 2 and 8). Formula 4 specifies that the membership of a border point is the minimum of the memberships μ_{dist} between the point and all the core points of the cluster directly reachable. In order to compute the minimum, we need first to detect all core points of the cluster and then compute the $\mu_c(\cdot)$ for all the border points (line 15–18). Line 15 is particularly important because a point that was inserted in the temporary structure $fuzzyBorderPts$, successively can verify the condition to be a core point. The difference between the two sets ($fuzzyBorderPts$ and C) guarantees that only the border points are considered after line 15. Note that when $\epsilon_{min} = \epsilon_{max}$ this extension reduces to the classic *DBSCAN* algorithm, since a point will get from Eq. 1 either a zero or a full (1) membership degree to the cluster. This extension is very similar to the approach proposed in Nasibov and Ulutagay (2009), since we fuzzify the input parameter ϵ too. Nevertheless, in our proposal, the core is still crisp and not fuzzy as in Nasibov and Ulutagay (2009). Further, differently than in the previous cited paper, *minPts* is still a numeric value that defines the local density of a core as in the classic *DBSCAN*. This allows generating fuzzy clusters with a crisp core, and a fuzzy border. More clearly, in this extension of *DBSCAN*, all generated clusters have cores with same density but that may differ for the density of their border, which may have faint overlapping profiles.

4.3 Generating clusters with fuzzy cores and overlapping fuzzy border

In this subsection, we introduce how to model fuzziness over both cores and borders in order to subsume the previous proposed approaches into what is named *Fuzzy DBSCAN*, i.e. $FDBScan$. The two soft constraints defined in (1) and (3) replace both $minPts$ and ϵ to allow the definition of the

Algorithm 5 ***Fuzzy Border DBSCAN***$(P, \epsilon_{Max}, \epsilon_{Min}, MintPts)$

Require: P: dataset of points
Require: $MinPts$: the minimum density around a point to be a candidate core point
Require: $\epsilon_{Min}, \epsilon_{Max}$: soft constraint on the distance around a point defining the point fuzzy neighbourhood size
$C = 0$
$Clusters = \emptyset$
for all $p \in P$ s.t. p is unvisited **do**
  mark p as visited
  nPts = regionQuery(p,ϵ_{Min})
  if $(sizeof(nPts) \leq MinPts)$ **then**
    mark p as NOISE
  else
    C = next cluster
    $Clusters = Clusters \cup expandClusterFuzzyBorder(p, nPts, C, \epsilon_{Max}, \epsilon_{Min}, MinPts)$
  end if
end for
return $Clusters$

Algorithm 6 $expandClusterFuzzyBorder(p, nPts, C, \epsilon_{Max}, \epsilon_{Min}, MinPts)$

Require: p: the point just marked as visited
Require: $nPts$: the points in the neighbourhood of p
Require: C: the actual cluster
Require: $\epsilon_{Max}, \epsilon_{Min}$: soft constraint on the distance around a point defining its fuzzy neighbourhood size
Require: $MinPts$: the density around a point to be considered a core point
add p to C (as core point)
$fuzzyBorderPts = regionQuery(p, \epsilon_{Max}) \setminus nPts$
for all $p' \in nPts$ **do**
  mark p' as visited
  $nPts'$ = regionQuery(p',ϵ_{Min})
  if $sizeof(nPts') > MinPts$ **then**
    $nPts = nPts \cup nPts'$
    $fuzzyBorderPts' = regionQuery(p', \epsilon_{Max}) \setminus nPts'$
    $fuzzyBorderPts = fuzzyBorderPts \cup fuzzyBorderPts'$
    add p' to C (as core)
  else
    $fuzzyBorderPts = fuzzyBorderPts \cup p'$
  end if
end for
$fuzzyBorderPts = fuzzyBorderPts \setminus C$
for all $p' \in fuzzyBorderPts$ **do**
  add p' to C (as border point) with membership $\mu_c(p')$ Equation (4)
end for
return C

fuzzy local density and the fuzzy local neighbourhood size of points respectively:

- a soft constraint specified by two values ($Mpts_{min} \leq Mpts_{max}$) on the Natural domain defines a fuzzy local dense region;
- a soft constraint specified by a pair ($\epsilon_{min} \leq \epsilon_{max}$) on the positive reals defines the local fuzzy neighbourhood size of a point.

We define the local density of a point p as follows:

$$dens(p) = \sum_{p_i \in neigh(p, \epsilon_{max})} \mu_{dist}(p, p_i) \quad (5)$$

where $neigh(p, \epsilon_{Max}) = \{p_i \text{ s.t. } \|p_i - p\| < \epsilon_{Max}\}$

If $\mu_{minP}(dens(p)) > 0$ then the point p belongs to the fuzzycore of a certain cluster with a membership degree $Fuzzycore(p) = \mu_{minP}(dens(p))$

If $\mu_{minP}(dens(p)) = 0$, then p is a border or a noise point.

If in the local neighbourhood of a fuzzy core point p_i there exists another fuzzy core point p_j, then a cluster c is generated: $\exists p_i, p_j$, s.t. $\mu_{dist}(p_i, p_j) > 0 \wedge Fuzzycore(p_i) > 0 \wedge Fuzzycore(p_j) > 0$ then $fuzzycore_c(p_i) = Fuzzycore(p_i) \wedge fuzzycore_c(p_j) = Fuzzycore(p_j)$.

A point p that is not a fuzzy core point is a fuzzy border point of a cluster c, if it satisfies the following condition: $\exists p_i$ and $\exists p$ s.t. $fuzzycore(p_i) = 0 \wedge \mu_{dist}(p, p_i) > 0 \wedge fuzzycore_c(p) > 0$.

If a point is a border point it cannot be a fuzzy core point of any cluster:

$$\nexists c \text{ s.t. } fuzzycore_c(p) > 0$$

If all the conditions are respected we define p as a fuzzy border point of a cluster c with a membership function to the cluster defined as:

$$\mu_{b(p)} = min_{p_i \in neighfcore(p)}(min(fuzzycore_c(p_i), \mu_{dist}(p, p_i))) \quad (6)$$

where $neighfcore(p) = \{p_i \text{ s.t. } fuzzycore_c(p_i) > 0 \wedge \mu_{dist}(p, p_i) > 0\}$

The procedures are described in Algorithms 7 and 8. The general schema is similar to the original *DBSCAN*. The main difference concerns the decision between core and border points which is made by considering $\mu_{minP}(\cdot)$ and the possibility of a point to belong to multiple clusters. Note that, this algorithm reduces to either *FCore* when $Mpts_{Min} = Mpts_{Max}$ or to *FBorder* when $\epsilon_{Min} = \epsilon_{Max}$

Algorithm 7 ***Fuzzy DBSCAN***$(P, \epsilon_{Max}, \epsilon_{Min}, Mpts_{Max}, Mpts_{Min})$

Require: P: dataset of points
Require: $\epsilon_{Min}, \epsilon_{Max}$: soft constraint on the distance around a point defining the point fuzzy neighbourhood size
Require: $Mpts_{Min}, Mpts_{Max}$: soft constraint on the density around a point to be considered as fuzzy core point
$C = \emptyset$
$Clusters = \emptyset$
for all $p \in P$ s.t. p is unvisited **do**
  mark p as visited
  nPts = regionQuery(p,ϵ_{Max})
  dens(p) = as in equation (5)
  if $(\mu_{minP}(dens(p)) == 0)$ **then**
    mark p as NOISE
  else
    C = next cluster
    $Clusters = Clusters \cup expandClusterFuzzy(p, nPts, C, \epsilon_{Max}, \epsilon_{Min}, Mpts_{Max}, Mpts_{Min})$
  end if
end for
return $Clusters$

Algorithm 8 $expandClusterFuzzy(p, nPts, C, \epsilon_{Max}, \epsilon_{Min}, Mpts_{Max}, Mpts_{Min})$

Require: p: the point just marked as visited
Require: $nPts$: the points in the fuzzy neighbourhood of p
Require: C: the actual cluster
Require: $\epsilon_{Max}, \epsilon_{Min}$: soft constraint on the distance around a point to compute its fuzzy neighbourhood
Require: $Mpts_{Min}, Mpts_{Max}$: soft constraint on the density around a point to be considered a fuzzy core point
1: add p to C (as core) with membership $\mu_{MinP}(dens(p))$
2: **for all** $p' \in nPts$ **do**
3: mark p' as visited
4: **if** $\mu_{minP}(dens(p')) > 0$ **then**
5: $nPts' = \text{regionQuery}(p', \epsilon_{Max})$
6: $nPts = nPts \cup nPts'$
7: add p' to C (as core) with membership $\mu_{MinP}(dens(p'))$
8: **else**
9: add p' to C (as border point) with membership computed by equation (6)
10: **end if**
11: **end for**
12: **return** C

5 Computational complexity

In this section, we introduce a discussion about the computational complexity of the different approaches we propose. Regarding the time complexity of the three proposals: *Fuzzy Core DBSCAN*, *Fuzzy Border DBSCAN* and *Fuzzy DBSCAN*, all of them have the same complexity of the original *DBSCAN*. In the *DBSCAN* algorithm, the computational time is mainly influenced by the number of time the function $regionQuery(\cdot, \cdot)$ is invoked. If we support this operation with a spatial indexing structure like an R-Tree, we can avoid a linear search and perform such operation in $O(\log n)$, where n is the number of elements in the dataset. In the Fuzzy variants, it can happen to traverse the same object multiple times because we can reach it from different starting points. This means that the $regionQuery(\cdot, \cdot)$ can be applied more than once for the same element. To avoid extra computation, we can simply employ an hash table to store the set of retrieved elements. Before performing the costly $regionQuery(\cdot, \cdot)$ action, we check if the neighbour points are already in the hash table, otherwise we perform the query and we store the results in the hash table for future use. In the case of the original *DBSCAN* algorithm, the worst case complexity is $O(n^2)$ that involves the case in which no spatial indexing structure is employed or the parameter are not carefully set (e.g. all points are within a distance less than ϵ). Considering our three fuzzy extensions, for all three cases, the computational complexity is the same ($O(n^2)$) as the one of the original *DBSCAN* since the general schemas are very similar. From a practical point of view, we have observed that the three approaches behave similarly with an average computational complexity lower than the worst case.

Regarding the space complexity, the materialization of the pointwise distance matrix implies a cost of $O(n^2)$, while in the worst case, the hash table can be $O(n^2)$. Since the two quantities need to be summed up, the final space complexity is $O(n^2)$.

Table 1 Dataset characteristics

Dataset	No. of instances	No. of attributes	No. of classes
Breast	699	9	2
Diabetes	768	8	2
Ecoli	336	7	8
Glass	214	9	6
Iris	150	4	3
Parkinsons	197	23	2
Vehicle	846	18	4

6 Experiments

In this section, we discuss the evaluation of our proposals on real-world datasets by comparing them w.r.t. state of the art soft clustering approaches. We choose as competitors the Fuzzy C-Means algorithm (Bezdek et al. 1984) (**FCM**) due to its popularity, and the (*FN-DBSCAN*) (Smiti and Eloudi 2013) (**Soft-DBSCAN**) as representative of fuzzy density-based approaches extending *DBSCAN*. The comparison includes all the three fuzzy *DBSCAN* extensions we introduced: *Fuzzy Core DBSCAN* (**FCore**), *Fuzzy Border DBSCAN* (**FBorder**) and *Fuzzy DBSCAN* (**FDBScan**). In order to benchmark all the different approaches, we use datasets from the UCI Machine Learning Repository[2] whose characteristics are reported in Table 1. More in detail, we use seven datasets with different characteristics (number of instances, number of features and number of classes). We summarize the data characteristics in Table 1.

The behaviour of the different algorithms is evaluated by computing both external and internal measures of validity of the results, which express the conformity of the results with the a-priori classifications (external measures) and the optimization of an objective function (internal measures). For the **FCM** algorithm we use the implementation available under the R[3] statistical computing software. We set the value of the fuzzification parameter m equals to 2, that controls the fuzziness of cluster boundaries, and the number of clusters equals to the number of classes. This way we drive the **FCM** clustering with correct information, thus favourably biasing its results. We run the **FCM** 50 times and then average the results thus obtained.

For the **Soft-DBSCAN** approach we vary the $Mpts$ parameter between 2 and 15 and the ϵ threshold between 0.1 and 1.0 with step of 0.05.

[2] https://archive.ics.uci.edu/ml/datasets.html.

[3] http://www.r-project.org/.

For **FCore**, **FBorder** and **FDBScan** we vary the soft constraints considering all the possible values combination in the previous intervals. For each method we retain the solution with the least number of noise points for, in principle, the used datasets should not contain noise.

6.1 Internal and external clustering validity measures

The clustering results are assessed under both internal and external validity measures. As internal criteria we choose the *Partition Coefficient* (Guillén et al. 2007) and the *Fuzzy Performance Index* (Smiti and Eloudi 2013), while we employ the *Fuzzy F-Measure* as external one (Suanmali et al. 2009), which is a combination of Recall and Precision.

We define with D the dataset, with $|D|$ the size of the dataset, with D_{cl} the instances of the dataset belonging to class cl, and with C the obtained cluster solution. We indicate with μ_{ij} the membership degree of the ith object to the jth cluster.

The *Partition Coefficient* (Guillén et al. 2007) is calculated as follows:

$$PC = \frac{1}{|D|} \times \sum_{i=1}^{|D|} \sum_{j=1}^{|C|} \mu_{ij}^2$$

This internal evaluation measure allows to compute the amount of overlap between clusters. High values of this measure indicate more cluster cohesion and density.

As second internal measure we employ the *Fuzzy Partition Index* (Smiti and Eloudi 2013). This measure is defined as:

$$FPI = 1 - \left(\frac{|C|}{|C|-1}\right) \times \left(1 - \sum_{i=1}^{|D|} \sum_{j=1}^{|C|} \frac{\mu_{ij}^2}{|D|}\right)$$

This measure evaluates the degree of separation of the fuzzy partition produced by the clustering algorithm. More in detail, the *Fuzzy Partition Index* quantifies the average cohesion of the clusters according to the membership function of the element of each cluster. Also for this measure, high values indicate more cluster cohesion.

The external measure we use is the *Fuzzy F-Measure* (Suanmali et al. 2009). This measure is a fuzzy adaptation of the standard *F-Measure* commonly involved to compare clustering results with the reference classification. First of all we define the *Fuzzy F-Measure* for a cluster C_j given a class cl as:

$$FMeasure(C_j, cl) = 2 \times \frac{FPrecision(C_j, cl) * FRecall(C_j, cl)}{FPrecision(C_j, cl) + FRecall(C_j, cl)}$$

where

$$FPrecision(C_j, cl) = \frac{\sum_{i \in C_j \cap D_{cl}} \mu_{ij}}{|C_j|}$$

and

$$FRecall(C_j, cl) = \frac{\sum_{i \in C_j \cap D_{cl}} \mu_{ij}}{|D_{cl}|}$$

and the final *Fuzzy F-Measure* is defined as:

$$\sum_{C_j \in C} \frac{|C_j|}{|D|} \times \text{Fuzzy F-Measure}(C_j, cl)$$

Each cluster C_j is associated with the class cl that maximizes the corresponding Fuzzy F-Measure(C_j, cl). The final solution is a weighted sum between the Fuzzy F-Measure of a cluster C_j and its importance considering the clustering solution.

6.2 Results

We report the evaluation results of the different approaches in Tables 2, 3 and 4. Table 2 shows the results in term of *Fuzzy F-Measure*. We can observe that, most of the time, our proposals outperform the competitors. Considering the *Breast* and *(*Iris) datasets, **FCM** obtains the highest score, while our strategies still obtain reasonable and competitive results. Regarding the comparison among the three different fuzzy extensions we proposed, we can observe that the **FBorder** strategy always reaches the same or the best score in term of *Fuzzy F-Measure* w.r.t. the other extensions. This model, contrary to the others we proposed, allows a fuzziness degree only for border points, while it considers that core points can belong to only one cluster. The empirical results underline that the assumption behind the **FBorder** well fits the underlined data distribution of the real-world benchmark we considered.

Tables 3 and 4 summarize the results in term of *Partition Coefficient* and *Fuzzy Partition Index* of the different algorithms. Both measures highlight the quality of our new fuzzy *DBSCAN* extensions. We can see that all the three extensions yield high values for the internal measures and outperform any of the competitors.

In order to explain this result, we deeply inspected the different clustering solutions. We observed that, first, the **Soft-DBSCAN** and the **FCM** algorithms assign each object to more clusters than the **FCore**, **FBorder** and **FDBScan**. Second, for an object its membership values distribution over all fuzzy clusters has often a multi-modal shape for both the **Soft-DBSCAN** and the **FCM**. This means that, several clusters share high membership values for the same object. This

Table 2 Achieved *Fuzzy F-Measure* of the different methods over the UCI datasets

Dataset	FCore	FBorder	FDBScan	Soft-DBSCAN	FCM
Breast	0.76	0.79	0.77	0.42	**0.88**
Diabetes	0.73	**0.75**	0.71	0.56	0.57
Ecoli	0.60	**0.61**	**0.61**	0.32	0.38
Glass	0.55	**0.74**	0.48	0.25	0.35
Iris	0.78	0.78	0.78	0.39	**0.8**
Parkinsons	0.79	**0.84**	0.78	0.46	0.6
Vehicle	**0.41**	**0.41**	**0.41**	0.30	0.34

The best results yielded by one of the tested algorithms on each dataset are marked in bold

Table 3 Achieved *Partition Coefficient* of the different methods over the UCI datasets

Dataset	FCore	FBorder	FDBScan	Soft-DBSCAN	FCM
Breast	0.95	**0.98**	0.97	0.58	0.84
Diabetes	0.96	**0.98**	0.97	0.50	0.57
Ecoli	**1.0**	**1.0**	**1.0**	0.38	0.31
Glass	0.89	0.81	**0.92**	0.46	0.43
Iris	**1.0**	**1.0**	**1.0**	0.68	0.74
Parkinsons	**0.97**	0.79	**0.97**	0.41	0.61
Vehicle	**1.0**	**1.0**	**1.0**	0.53	0.42

The best results yielded by one of the tested algorithms on each dataset are marked in bold

Table 4 Achieved *Fuzzy Performance Index* of the different methods over the UCI datasets

Dataset	FCore	FBorder	FDBScan	FN-DBSCAN	FCM
Breast	0.9	**0.96**	0.94	0.37	0.68
Diabetes	0.92	**0.96**	0.94	0	0.14
Ecoli	**1.0**	**1.0**	**1.0**	0.07	0.21
Glass	0.78	0.72	**0.84**	0.28	0.43
Iris	**1.0**	**1.0**	**1.0**	0.36	0.61
Parkinsons	**0.94**	0.72	**0.94**	0.12	0.22
Vehicle	**1.0**	**1.0**	**1.0**	0.04	0.23

is not the case for our fuzzy *DBSCAN* extensions where, in theory, an object can belong to multiple clusters but, in practice, it has membership degrees greater than zero for a limited number of clusters (usually no more than two or three clusters), which seems a reasonable characteristics of real data distributions.

7 Conclusion

In this contribution, we presented three fuzzy extensions of the *DBSCAN* clustering algorithm, to the aim of modelling distinct density-based characteristics of the objects spatial distributions in the feature space. The main characteristics of these algorithms are the definition of distinct soft constraints to specify the approximate local density of points needed for generating a cluster. Specifically, the first extension, *Fuzzy Core DBSCAN* allows assigning a core point to a cluster with a membership value; in doing so, clusters can contain core points with different membership values, thus allowing to detect clusters with heterogeneous densities of their nucleus with a single run of the algorithm. The second extension, *Fuzzy Border DBSCAN*, allows generating semi-overlapping clusters with fuzzy border and homogeneous dense cores. The third extension, *Fuzzy DBSCAN*, combines the previous ones thus detecting clusters with both fuzzy core and fuzzy border points, i.e. heterogeneous dense cores and overlapping borders.

The main novelty of the proposal is the intent to control distinct fuzzification characteristics of the clusters that can be generated when using a clustering algorithm, thus suiting distinct application domains, such as user community detection in social networks with partial membership either to disjoint communities *Fuzzy Core DBSCAN* or to semi-overlapped communities *Fuzzy Border DBSCAN*, and ecosystems detection in satellite images *Fuzzy DBSCAN*.

Furthermore, besides leveraging the specification of the precise input, the proposals supply with a single run a solu-

tion that summarizes multiple runs of the original classic *DBSCAN* algorithm. Experimental comparison w.r.t. state of the art fuzzy clustering approaches over real-world datasets underlined the higher quality of the results produced by our proposals, which better model the fuzzy characteristics of the real datasets.

Compliance with ethical standards

Conflict of interest Dino Ienco and Gloria Bordogna declares that they have no conflict of interest.

Ethical standard This article does not contain any studies with human participants or animals performed by any of the authors.

References

Bezdek JC, Ehrlich R, Full W (1984) FCM: the fuzzy c-means clustering algorithm. Comput Geosci 10(2):191–203

Bordogna, G., Ienco, D.: Fuzzy core dbscan clustering algorithm. In: IPMU, pp. 100–109 (2014)

Ester M, Kriegel H, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. KDD 160:226–231

Guillén A, González J, Rojas I, Pomares H, Herrera LJ, Valenzuela O, Prieto A (2007) Using fuzzy logic to improve a clustering technique for function approximation. Neurocomputing 70(16–18):2853–2860

Ji Z, Xia Y, Sun Q, Cao G (2014) Interval-valued possibilistic fuzzy c-means clustering algorithm. Fuzzy Sets Syst 253:138–156

Kriegel H, Pfeifle M (2005) Density-based clustering of uncertain data. In: KDD'05 vol 17, pp 672–677

Nasibov EN, Ulutagay G (2009) Robustness of density-based clustering methods with various neighborhood relations. Fuzzy Sets Syst 160(24):3601–3615

Pal NR, Pal K, Keller JM, Bezdek JC (2005) A possibilistic fuzzy c-means clustering algorithm. IEEE Trans Fuzzy Syst 13(4):517–530

Parker J, Downs J (2013) Footprint generation using fuzzy-neighborhood clustering. Geoinformatica 17:283–299

Parker J, Hall L, Kandel A (2010) Scalable fuzzy neighborhood dbscan. In: IEEE-fuzzy, pp 1–8

Sander J, Ester M, Kriegel H, Xu X (1998) Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications. Data Min Knowl Discov 2(2):169–194

Shamshirband S, Amini A, Anuar NB, Kiah LM, Wah TY, Furnell S (2014) D-ficca: a density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks. Measurement 55:212–226

Smiti A, Eloudi Z (2013) Soft dbscan: improving dbscan clustering method using fuzzy set theory. Hum Syst Interact 1:380–385

Suanmali L, Salim N, Binwahlan MS (2009) Fuzzy logic based method for improving text summarization. CoRR arXiv:0906.4690

Ulutagaya G, Nasibov E (2012) Fuzzy and crisp clustering methods based on the neighborhood concept: a comprehensive review. J Intell Fuzzy Syst 23:1–11

Yager R, Filev D (1994) Approximate clustering via the mountain method. IEEE Trans Syst Man Cybern 24(8):1279–1284