Fuzzy Core DBScan Clustering Algorithm

Gloria Bordogna[1] and Dino Ienco[2]

[1] CNR IdA detached at IREA, Via Bassini 15, Milano, Italy
bordogna.g@irea.cnr.it
[2] Irstea, UMR TETIS, Montpellier, France
LIRMM, Montpellier, France
dino.ienco@irstea.fr

Abstract. In this work we propose an extension of the *DBSCAN* algorithm to generate clusters with fuzzy density characteristics. The original version of *DBSCAN* requires two parameters (*minPts* and ϵ) to determine if a point lies in a dense area or not. Merging different dense areas results into clusters that fit the underlined dataset densities. In this approach, a single density threshold is employed for all the datasets of points while the distinct or the same set of points can exhibit different densities. In order to deal with this issue, we propose *Approx Fuzzy Core DBSCAN* that applies a soft constraint to model different densities, thus relaxing the rigid assumption used in the original algorithm. The proposal is compared with the classic *DBSCAN*. Some results are discussed on synthetic data.

1 Introduction

Density based clustering algorithms have a wide applicability in data mining. They apply a local criterion to group objects: clusters are regarded as regions in the data space where the objects are dense, and which are separated by regions of low object density (noise). Among the density based clustering algorithms *DBSCAN* is very popular due both to its low complexity and its ability to detect clusters of any shape, which is a desired characteristics when one does not have any knowledge of the possible clusters' shapes, or when the objects are distributed heterogenously such as along paths of a graph or a road network. Nevertheless, to drive the process, this algorithm needs two numeric input parameters, *minPts* and ϵ which together define the desired density characteristics of the generated clusters. Specifically, *minPts* is a positive integer specifying the minimum number of objects that must exist within a maximum distance ϵ of the data space in order for an object to belong to a cluster.

Since *DBSCAN* is very sensible to the setting of these input parameters they must be chosen with great accuracy [2] by considering both the scale of the dataset and the closeness of the objects in order not to affect too much both the speed of the algorithm and the effectiveness of the results. To fix the right values of these parameters one generally engages an exploration phase of trials and errors in which the clustering is run several times with distinct values of the parameters.

In fact, a common drawback of all crisp flat clustering algorithms used to group objects whose distribution has a faint and smooth density profile is that they draw crisp boundaries to separate clusters, which are often somewhat arbitrary.

There are applications in which the positions of the objects is ill-known, such as in the case of databases of moving objects, whose locations are recorded at fixed timestamps with uncertainty on their intermediate positions, or in the case of objects appearing in remote sensing images having a coarse spatial resolution so that a pixel is much greater than the object dimension, and thus uncertainty is implied when one has to detect the exact position of the object within an area. Last but not least, when one has to detect communities of users in a social network, while one can specify easily that the users must have at most a given number of degrees of separation on the network, it my be questionable to define the precise minimum number of elements defining a social community.

In this contribution we investigate an extension of the *DBSCAN* algorithm defined within the framework of fuzzy set theory whose aim is to detect fuzzy clusters with approximate density characteristics. In the literature a fuzzy extension of *DBSCAN* has been proposed, named FN-SBSCAN, with the objective of allowing the specification of an approximate distance between objects instead of a precise ϵ value [6]. Our proposal is dual since we want to leverage the setting of the precise value *minPts* by allowing the specification of an approximate minimum number of objects for defining a cluster. In our proposal a user drives the clustering algorithm by specifying a soft condition that can be expressed as follow: *group at least approximately* $minPts_{min} - minPts_{max}$ *objects which are within a maximum distance* ϵ *".*

The two values $minPts_{min}, minPts_{max}$ indicate the approximate number of objects defining the density of the clusters and define a soft constraint with a not decreasing membership function on the basic domain of positive integers. The algorithm uses this approximate input to generate clusters with a fuzzy core, i.e., clusters whose elements are associated with a numeric membership degree in [0,1]. Having fuzzy clusters allows several advantages: with a single run of the clustering it is possible to perform a sensitivity analysis by generating several distinct crisp partions obtained by specifying distinct thresholds on the membership degrees of the objects to the clusters. This allows an easy exploration of the spatial distribution of the objects without the need of several runs of the clustering as it occurres when one uses the classic *DBSCAN*. The contribution first recalls the classic *DBSCAN* algorithm; then, the *Approx Fuzzy Core DBSCAN* is defined. Section 4. discusses the results obtained by the extended algorithm; section 5 compared the proposal with related literature, and the conclusions summarize the main achievements.

2 Classic DBScan Algorithm

For sake of clarity in the following we will consider a set of objects represented by distinct points defined in a multidimensional domain. These objects can be either actual entities located on the bidimensional spatial domain such as cars, taxi cabs, airplanes, or virtual entities, such as web pages and tweets represented in the virtual n-dimensional space of the terms they contain. DBSCAN can be applied to group these objects based on their local densities in the space. This makes it possible to identify traffic jams of cars on the roads, or to identify groups of web pages and tweets that deal with same topics.

DBSCAN assigns points of a spatial domain defined on RxR to particular clusters or designates them as statistical noise if they are not sufficiently close to other points. DBSCAN determines cluster assignments by assessing the local density at each point using two parameters: distance (ϵ) and minimum number of points ($minPts$). A single point which meets the minimum density criterion, namely that there are $minPts$ located within distance ϵ, is designated a core point. Formally, Given a set P of N points $p_i = (x_{i_1}, x_{i_2}, ..., x_{i_n})$ with x_{i_j} defined on the n-dimensional domain R^n. $p \in P$ is a core point if at least a minimum number $minPts$ of points $p_1, , p_{minPts} \in P \exists s.t || p_j - p || < \epsilon$, Two core points p_i and p_j with $i, j s.t || p_i - p_j || < \epsilon$ define a cluster c, $p_i, p_j \in c$ and are core points of c, i.e., $p_j, p_j \in core(c)$ All not core points within the maximum distance ϵ from a core point are considered non-core members of a cluster, and are boundary or border points: $p \notin core(c)$ is a boundary point of c if $\exists p_i \in core(c)$ with $|| p - p_i || < \epsilon$. Finally, points that are not part of a cluster are considered noise: $p \notin core(c)$ are noise if $\forall c, \nexists p_i \in core(c)$ with $|| p - p_i || < \epsilon$. In the following the classic DBSCAN algorithm is described:

Algorithm 1. $DBSCAN(D, \epsilon, MinPts)$

```
Require: $P$: dataset of points
Require: $\epsilon$: the maximum distance around a point defining the point neighbourhood
Require: $MinPts$: density, in points, around a point to be considered a 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.     neighborsPts = regionQuery($p$,$\epsilon$)
 6.     if $(sizeof(neighborsPts) <= MinPts)$ then
 7.         mark $p$ as NOISE
 8.     else
 9.         $C$ = next cluster
10.         $Clusters = Clusters \cup expandCluster(p, neighborsPts, C, \epsilon, MinPts)$
11.     end if
12. end for
13. return $Clusters$
```

3 Generating Clusters with Fuzzy Cores

The extension of the classic $DBSCAN$ algorithm we propose, named fuzzy core DBSCAN, is obtained by considering crisp the distance, as in the classic approach, and by introducing an approximate value of the desired cardinality of the neighborhood of a point $minPts$. This can be done by substituting the numeric value $minPts$ with a soft constraint defined by a non decreasing membership function on the domain of the positive integers. This soft constraint specifies

Algorithm 2. *expandCluster*(p, *neighborsPts*, C, ϵ, *MinPts*)

```
Require: $p$: the point just marked as visited
Require: $neighborsPts$: the neighborhood of $p$
Require: $C$: the actual cluster
Require: $\epsilon$ the distance around a point to compute its density
Require: $MinPts$: density, in points, defining the minimum cardinality of the neighborhood of a
    point to be considered a core point
 1. add $p$ to cluster $C$
 2. for all $p^{'} \in neighborsPts$ do
 3.   if $p^{'}$ is not visited then
 4.     mark $p^{'}$ as visited
 5.     $neighborsPts^{'}$ = regionQuery($p^{'}$,$\epsilon$)
 6.     if $sizeof(neighborsPts^{'}) > MinPts$ then
 7.       $neighborsPts = neighborsPts \cup neighborsPts^{'}$
 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 $C$
```

the approximate number of points that are required in the neighborhood of a point for generating a fuzzy core of a cluster. Let us define the piecewise linear membership function as follows:

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

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

Since users may find it difficult to specify the two values $Mpts_{Min}$ and $Mpts_{Max}$ when they are not aware of the total number of objects involved in the process, they can specify two percentage values, $\%Mpts_{Min}$ and $\%Mpts_{Max}$ which are then converted into $Mpts_{Min}$ and $Mpts_{Max}$ as follows:

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

Let us redefine the fuzzy core . Given a set P of N objects represented by N points in the n-dimensional domain R^n $p_1, p_2, ...p_N$, where p_i has the coordinates $x_{i_1}, x_{i_2}, ..., x_{i_n}$.

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

A point p of a cluster that is not a fuzzy core point is a boundary or border point if it satifies the following: Given $p \notin fuzzycore(c)$ if $\exists p_i \in fuzzycore(c)$, i.e., with membership degree $fuzzycore_c(p_i) > 0$, s.t. $\|p_i - p\| < \epsilon$ then p gets a membership degree to c defined as: $\mu_c(p) = max_{p_i \in fuzzycore(c)} fuzzycore_c(p_i)$

Finally, points p that are not part of a cluster are considered noise: $\forall c$ if $\nexists p_i \in fuzzycore(c)$ s.t. $\|p_i - p\| < \epsilon$, then p is noise.

Notice that the points belonging to a cluster c gets 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 boundary point p can partially belong to a single cluster c since its membership degree is upperbounded by the maximum membership degree of its neighbouring 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 fuzzy core DBSCAN reduces to the classic DBSCAN when the input values $MinPts_{Min} = MinPts_{Max}$: in this case the fuzzy core DBSCAN produces the same results of the classic DBSCAN with $minPts = MinPts_{Min} = MinPts_{Max}$ 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: $\mu_{minP} = 0$ when the number of points within a maximum distance ϵ of any point p is less than $minPts = MinPts_{Min} = MinPts_{Max}$, on the contrary $\mu_{minP} = 1$. 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 degree is the maximum of their closest core points, i.e., it 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 neighborhood size is less than or equal to $MinPts_{Min}$ 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 $\mu_{MinP}()$. 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. $MinPts_{Min}$ and then, if the point verifies the soft constraint, we add the point to the fuzzy core of cluster C with its associated membership value. Differently from the original version, line 10 is only devoted to detect border points not yet assigned to any cluster.

Algorithm 3. ***Approx Fuzzy Core DBSCAN***(D,ϵ,$MinPts_Min$,$MinPts_{Max}$)

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

Algorithm 4. *expandClusterFuzzyCore*(p, $neighborsPts$, C, ϵ, $MinPts_{Min}$, $MinPts_{Max}$)

```
Require: $p$: the point just marked as visited
Require: $neighborsPts$: the points in the neighbourhood of $p$
Require: $C$: the actual cluster
Require: $\epsilon$ the distance around a point to compute its density
Require: $MinPts_{Min}, MinPts_{Max}$: soft constraint interval for the density around a point to be considered a core point
 1. add $p$ to $C$ with membership $Fuzzycore(p) = \mu_{MinP}(|neighborsPts|)$
 2. for all $p' \in neighborsPts$ do
 3.    if $p'$ is not visited then
 4.       mark $p'$ as visited
 5.       $neighborsPts'$ = regionQuery($p'$,$\epsilon$)
 6.       if $sizeof(neighborsPts') > MinPts_{Min}$ then
 7.          $neighborsPts = neighborsPts \cup neighborsPts'$
 8.          add $p'$ to $C$ with membership $Fuzzycore(p') = \mu_{MinP}(|neighborsPts'|)$
 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 Experiments

In this section we discuss the properties of our proposed algorithm by showing the results we can obtain when applying the fuzzy core DBSCAN to a synthetic data set in a bidimensional domain. Figure 1 (a) and 1(b) depict with distinct colors the clusters identified by the classic DBSCAN and the fuzzy core DBSCAN algorithms respectively. It can be noticed that the classic approach provided in input with the parameters $minPts = 9$ and $\epsilon = 12$ divides into two distinct clusters, cluster 1 and cluster 6, and also cluster 5 and cluster 3, while the fuzzy core clustering identifies as two clusters, cluster 2 and cluster 3 respectively. It can also be noticed that with the fuzzy DBSCAN the obtained conflated clusters have a variable density profile.

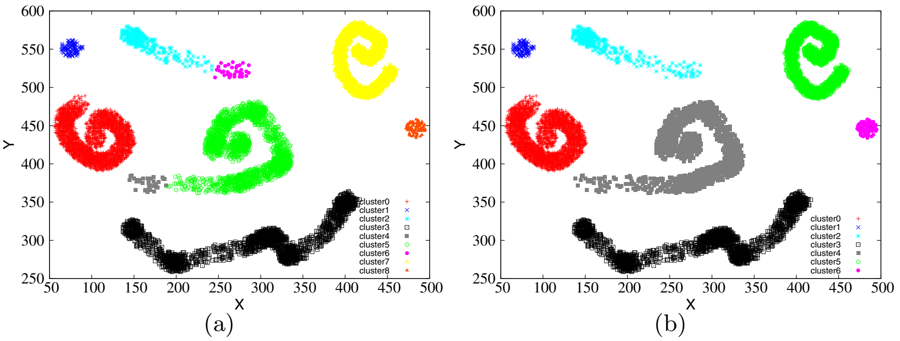

Fig. 1. Results of a) *DBSCAN* b) *Approx Fuzzy Core DBSCAN*. We set Mpts = 9 and $\epsilon = 12$ while for *Approx Fuzzy Core DBSCAN* the soft constraint over the minimum number of points ranges from 7 to 12 and ϵ is always equal to 12.

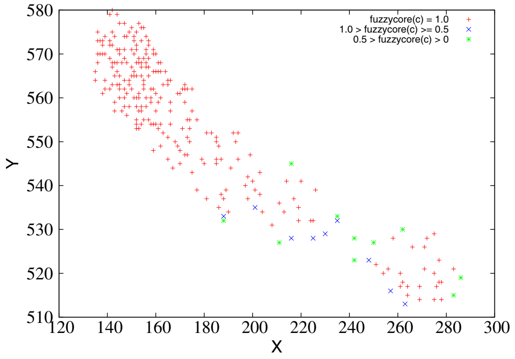

Fig. 2. Inspection of a cluster generated with the *Approx Fuzzy Core DBSCAN* approach ($Mpts$=(9,12), ϵ=12]). For the light blue cluster (Cluster2) shown in Figure 1b we visualize the fuzzy core points grouped in three category: fuzzy cores with membership equal to 1 (red cross), fuzzy cores with membership lesser than 1 and greater or equal to 0.5 (blue X) and fuzzy cores with membership lesser than 0.5 and bigger than 0 (green star).

This happens in the classic approach since the input parameters were not chosen accurately. To obtain a correct partition we would have to choose a smaller value $minPts < 9$. Nevertheless, to find the correct parameters we might have to re-run the classic DBSCAN several times before obtaining a satisfactory solution. Conversely, just with a single run of *Approx Fuzzy Core DBSCAN* we can identify the appropriate clusters, also the ones, cluster 2 and cluster 3, that are characterised by a variable density of their core.

Moreover, having the membership degrees associated with each pair $object-cluster$ we can inspect the partitions that we can obtain by associating with each range of membership degrees a distinct color. An example of this analysis is depicted in Figure 2. We plot the fuzzy core points of cluster 2 of Figure 1(b).

We avoid to plot border point as they are not influenced by the fuzzification process. Cluster 2 has the profile of a comet with a dense nucleous on the upper left side and a faint tail on the lower right side. The membership degrees to the fuzzy core are discretised into three bins: fuzzy core points with a membership value equals to 1.0, fuzzy core points with a membership in the range (1.0,0.5] and fuzzy core points with a membership in the interval (0.5,0). This way we quantify how the density distribution varies, having all full core points in the nucleus, and partial fuzzy core points on the tail.

If we apply a threshold so as to detect only the points with full membership 1 to the fuzzy core, we can observe that the tail of the comet splits into two parts as with the classic DBSCAN result in Figure 1(a).

5 Related Work

Clustering algorithms can be grouped into five main categories such as hierarchical, partition-based, grid-based, density based , and model-based and, furthermore, one can distinguish among crisp and soft (fuzzy or probabilistic) clustering according to the fact that elements belong to clusters with a full or a partial membership degree, in this latter case with the possibility for and element to simultaneously belong to several clusters. Among the partitional density based clustering algorithms, *DBSCAN* is the most popular one due to its ability to detect irregularly shaped clusters by copying with noise data sets. Nevertheless it performances are strongly dependent of the parameters setting and this is the main reason that lead to its soft extensions. In the literature there have been a few extensions of the *DBSCAN* algorithm in order to detect fuzzy clusters, as also discussed in the recent paper [3] in which the authors report a survey of the main density based clustering algorithms. The most cited paper [6] proposes a fuzzy extension of the *DBSCAN*, named $FN - DBSCAN$ (fuzzy neighborhood *DBSCAN*), whose main characteristic is to use a fuzzy neighborhood relation. In this approach the authors address the difficulty of the user in setting the values of the input parameters when the distances of the points are in distinct scales. Thus, they first normalize the distances between all points in [0,1], and then they allow specifying distinct membership functions on the distance to delimit the neighborhood of points, i.e., the decaying of the membership degree as a function of the distance. Then, they select as belonging to the fuzzy neighbourhood of a point only those points having a minimum membership degree greater than zero. This extension of *DBSCAN* uses a level-based neighborhood set instead of a distance-based neighborhood set, and it uses the concept of fuzzy cardinality instead of classical cardinality for identifying core points. This last choice causes the creation (with the same run of the algorithm) of both fuzzy clusters with cores having many sparse points and fuzzy clusters with cores having only a few close points. Thus the density characteristic of the generated clusters is very heterogeneous. Furthermore in this extensions points can belong to several clusters with distinct membership degree. This extension of the *DBSCAN* can be considered dual to our proposal since we fuzzify the minimum number of points in the neighbourhood of a point to be considered as part of the fuzzy core, while the maximum distance ϵ is still crisp, thus generating fuzzy non overlapping clusters. This kind of fuzziness of clusters reflects the variable densities of the clusters' cores: as a consequence the membership degree of a point to the fuzzy core depends on the number of points in its crisp neighborhood, and thus a point is assigned to only one cluster, the one with the greatest number of core points in its neighbourhood. With this semantics of fuzziness we want to favor the growing of denser clusters with respect to fainter ones. The utility of the fuzzy *DBSCAN* is pointed out in the paper [4] where the authors use FN-DBSCAN in conjunction with the computation of the convex hull of the generated fuzzy clusters to derive connected footprints of entities with arbitrary shape. Having fuzzy clusters allows generating isolines footprints. An efficient implementation is proposed in [2]. 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 noise points and are removed, while the remaining will belong to a cluster. These latter points are clustered with the classic *DBSCAN* by providing as input parameters $minPts = K$ and $\epsilon = \theta$. Another motivation of defining fuzzy *DBSCAN* is to cluster objects whose position is ill-known, as in the paper [5] where 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 could be modeled in our approach by allowing the neighbouhood of any object as consisting of an approximate number of other objects, thus capturing the uncertainty on the positions of the moving objects, which could be inside or outside the radius ϵ. This way, the grouping of moving objects could be modeled with a simpler approach with respect to the proposal [5] that use fuzzy distance and probability distributions. Finally, the most recent soft extension of *DBSCAN* has been proposed in [1] where the authors combine the classic *DBSCAN* with the fuzzy C-means algorithm. 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. Nevertheless, this extension has the objective of determining seeds to feed the Fuzzy C-Means, like the approximate clustering algorithm based on the mountain method [7], which is different from our proposal since we do not grow the boundary by applying the fuzzy C-means, but rely on the DBSCAN. The result is not a fuzzy partition but is still a crip partition of the elements into distinct clusters, even if each element can belong to a single cluster with a distinct degree. In fact our aim is tofold: firstly of all we want to subsume several runs of the classic DBSCAN with different paramanater settings with a single run of our algorithm, and secondly we want to give a preference to the grouth of clusters with denser core with respect to clusters with fainter core.

6 Conclusion

In this work we present a new fuzzy clustering algorithm *Approx Fuzzy Core DBSCAN* that extends the original *DBSCAN* method. The main characteristics of this algorithm is to introduce a soft constraint to specify the approximate number of points that must exists in the neighbourhood of a point for generating a cluster. Specifically, *Approx Fuzzy Core DBSCAN* allows assigning a core point to a cluster with a membership value so clusters can contain core points with different membership values representing this way the distinct local densities. Beside leveraging the specification of the precise input, the proposal supplies with a single run of the clustering algorithm a solution that summarises multiple runs of the original classic DBSCAN algorithm: specifically, all the runs with a value of $MinPoints$ belonging to the support of the soft constraint. The visual quality of the results yielded by *Approx Fuzzy Core DBSCAN* is tested in the experimental section. A synthetic dataset is employed to highlight the benefit of representing the fuzziness of the local density around points in order to obtain meaningful results. As a future direction we would employ soft constraints also over the maximum distance ϵ and, then, over both parameters of the *DBSCAN* simultaneously in order to be more flexible over data exhibit heterogenous densities.

References

1. Smiti, A., Eloudi, Z.: Soft dbscan: Improving dbscan clustering method using fuzzy set theory. Human System Interaction 1, 380–385 (2013)
2. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD, vol. 160, pp. 226–231 (1996)
3. Ulutagay, G., Nasibov, E.N.: Fuzzy and crisp clustering methods based on the neighborhood concept: A comprehensive review. Journal of Intelligent and Fuzzy Systems 23, 1–11 (2012)
4. Parker, J.K., Downs, J.A.: Footprint generation using fuzzy-neighborhood clustering. Geoinformatica 17, 283–299 (2013)
5. Kriegel, H.P., Pfeifle, M.: Density-based clustering of uncertain data. In: KDD 2005, vol. 17, pp. 672–677 (2005)
6. Nasibov, E.N., Ulutagay, G.: Robustness of density-based clustering methods with various neighborhood relations. Fuzzy Sets and Systems 160(24), 3601–3615 (2009)
7. Yager, R.R., Filev, D.P.: Approximate clustering via the mountain method. IEEE Transactions on Systems, Man and Cybernetics 24(8), 1279–1284 (1994)