Fuzzy Density Based Clustering with Generalized Centroids

Christian Braune Member, IEEE
Institute for Intelligent Cooperating Systems
Department for Computer Science,
Otto von Guericke University Magdeburg
Universitätsplatz 2, 39106 Magdeburg, Germany
Email: christian.braune@ovgu.de

Rudolf Kruse Fellow, IEEE
Institute for Intelligent Cooperating Systems
Department for Computer Science,
Otto von Guericke University Magdeburg
Universitätsplatz 2, 39106 Magdeburg, Germany
Email: rudolf.kruse@ovgu.de

Abstract—Fuzzy density-based clustering has been a challenge.
Yet, from an implementation’s point of view these points are all assigned to a noise cluster. So the argument might still hold. However, if clusters are actually overlapping or the spatial structure of the data does not justify an unambiguous cluster assignment, we can use fuzzy clustering. Here the points are not assigned to a single cluster but rather belong to every cluster found at the same time but to a different degree. This membership degree is in inverse proportion the the distance of the clusters’ centers, i.e. if a point has a small distance to a cluster center then it has a high degree of membership to that cluster and vice versa.

For centroid-based clustering algorithms often fuzzy version exist. The best-known algorithm of this family appears to be the fuzzy c-means algorithm [4]. One advantage of fuzzy clustering algorithms is, that they a more robust than their crisp counterparts. They do not end in local minima of their respective objective function as often because of the smoother structure of this function.

Like its original, crisp version, the fuzzy c-means algorithm tends to find globular cluster shapes for which the cluster center acts as a natural representative. Because of that the membership degree can be calculated easily. Density-based clustering algorithms lack such a distinguished point. Fuzzifying a density-based clustering algorithm therefore comes much less naturally than in the centroid-based case. We try to solve this problem with our approach.

We propose that for fuzzy clustering of such non-convex clusters – as they may result from density-based clustering algorithms – another kind of centroid should be used. Single points cannot capture the shape of the clusters anymore once they contain too many or too strong concavities. Instead we want to generate a skeleton that represents the cluster structure and that can be used to assign fuzzy membership degrees.

The remaining paper will be organized as follows: In Section II we will briefly explain the DBSCAN algorithm and in Section III how the k-means algorithm has been fuzzified. Section IV will give an overview over existing techniques for fuzzifying density-based clustering algorithms, mainly the existing DBSCAN algorithm. In Section V we will present our approach. We evaluate our approach in Section VI and conclude in Section VII.
II. DBSCAN

DBSCAN is a well-known clustering algorithm [5] whose properties have been extensively studied over the past years. It is able to find nearly arbitrarily shaped clusters – as long as the connection between two different clusters is weak enough. Its main parameters are \( \varepsilon \) and \( \text{minPts} \). The first is used to describe the size of a hypersphere centered around each data point while the second parameter is used as a threshold to identify cluster cores. If the number of data points inside of its neighborhood exceeds this threshold it becomes the core of a cluster.

Points that both fulfill the core condition and are mutually contained in each others neighborhoods are said to be \textit{directly density reachable}. Points are \textit{density reachable} if they fulfill the core condition and can be linked to each other by a series of points that are each pairwise directly density reachable. At last, two points are density connected if there exists a series of points where the first and last point of the series are density reachable and the two points in question are contained in the first’s and second’s neighborhood respectively. A cluster is then identified as a subset of the dataset where each pair of points is density connected.

DBSCAN is very sensitive to the choice of parameters. Improper values might lead to either (almost) all points becoming core points or none at all [6]. This problem can, however, be overcome by trying to estimate the parameters [7] or a completely unsupervised alternating optimization approach [8].

III. FUZZY CLUSTERING

Sometimes points cannot be unambiguously assigned to one cluster alone [9]. In the case of the \( k \)-means algorithm if a point is just halfway between two cluster centers, a race condition occurs and the point usually is assigned to the cluster center that is handled first by the algorithms implementation. However, even in less strict cases where points lie roughly in the middle between two clusters one could be hesitant to assign a point to a single cluster. In such cases fuzzy clustering becomes a proper tool to handle these situations.

In the fuzzy \( c \)-means algorithm the membership of a point to a single cluster is 1.0 if and only if its distance to the cluster center is 0. In all other cases the membership is inversely proportional to the distance of the point to the cluster center. It can be easily calculated by the formula

\[
\mu(x_i, c_j) = \frac{d_{ij}^{-2}}{\sum_{k=1}^{c} d_{kj}^{-2}} = \frac{1}{\sum_{k=1}^{c} \left( \frac{d_{ij}}{d_{kj}} \right)^{2\alpha}} \tag{1}
\]

where \( d_{ij} \) is the distance between the point \( x_i \) and the cluster center \( c_j \). \( \omega \) is called \textit{fuzzifier} and influences how crisp the clustering around the centroids is. Higher values for \( \omega \) produce crisp clustering only in the closest vicinity of the clusters’ centroids, see Figure 1.

Fuzzy clustering can also be used when there is no ambiguity since it tends to produce more robust results than crisp clustering [11].

![Fig. 1: Membership degrees for two cluster centers (one-dimensional case). Left with lower, right with higher fuzzifier [10].](image)

IV. RELATED WORK

To obtain fuzzy results from a clustering algorithm’s result one can use the crisp clustering algorithm and simply apply a fuzzification formula to obtain membership degrees. This works well in the case of centroid-based clustering algorithms – the fuzzy \( c \)-means algorithm [4] as extension of the \( k \)-means algorithm [3] is a prime example. In density-based clustering the only cluster representatives available are the core points. This is where most of the previous work aims at when fuzzifying the clustering process.

First attempts to create fuzzy density-based algorithm have been based on the DBSCAN algorithm in [12]–[14]. Here the fuzziness is introduced into the core condition. Points that are within the \( \varepsilon \)-neighborhood of a point contribute differently to the cardinality of the neighborhood. Theoretically DBSCAN does not differentiate between points that are close to the center of the neighborhood and points that are close to its border. Central points should however contribute more to the centers qualification as cluster core. Thus the cardinality of the neighborhood set is fuzzified. However, the cluster assignment remains crisp in the end. Clusters

Another approach is described in [15]. In contrast to the fuzzy neighborhood approach mentioned above, cluster assignments can become fuzzy here. Core points are identified in the same way as in the classical DBSCAN algorithm with one distinction: There are two thresholds for \( \text{minPts} \). If the lower threshold is exceeded the point starts to become a cluster core. The membership to being a cluster core rises linearly until the upper threshold is exceeded. Here the point fully becomes a core point. For non-core points the membership degree to the respective cluster is simply the maximum of all the membership degrees of those core points in whose neighborhoods the point lies. This may however lead to membership assignments that do not sum up to 1.0 if the point is part of several overlapping neighborhoods from different clusters but not a core point itself. Also, points that were (possibly incorrectly) identified as noise do not receive any membership to any cluster at all. Lastly, the clustering is not truly fuzzy: due to the linear interpolation of membership degrees (a) only a discrete amount of fuzzy membership values can be obtained, and (b) the distance between a point and its associated core points does not matter. The obtained membership degrees resemble more or less the \( \alpha \)-cuts of a fuzzy membership degree distribution of a more general cluster representation. The advantage of this approach however is, that clusters of varying density can be found in a more robust way.

A completely different approach is given in [16] in which DBSCAN is used to find an initialization for the fuzzy \( c \)-means algorithm. By choosing the parameters \( \varepsilon \) and \( \text{minPts} \)
sufficiently small or high respectively, a lot of small and very dense clusters are retrieved. These act as seeds for the fuzzy $c$-means algorithm. Instead of euclidean distance the authors use the Mahalanobis distance [17]. However, instead of adjusting the centroid calculation accordingly, the authors use the arithmetic mean (weighted by the membership degrees of the associated points) thus using an inappropriate estimator for the centroids given the distance metric used.

V. GENERALIZED CENTROIDS

Essentially the idea behind assigning fuzzy memberships is sound and good as it is. It just cannot be easily applied to density-based clustering. If we recall Equation 1 then we see that the membership degree of a point to a cluster is always calculated by its distance to the cluster’s centroid. In the previous section we reviewed some ways that try to overcome the need for this calculation by either introducing fuzziness into the neighborhood definition or the core points themselves.

Our approach differs from these in a sense that we consider a closer approach to the fuzzy $c$-means approach. Instead of trying to find a way to assign fuzzy membership degrees to a data point by changing the neighborhood or core point definitions of DBSCAN we aim the centroid definition itself. From Figure 2 can be seen, that the centroids of density-based clusters do not always lie in the set of points forming the clusters anymore. Instead they are closer to the points of the other cluster.

Additionally our new algorithm allows to calculate interpretable fuzzy membership degrees to arbitrary points inside the data space. This may be useful if not all points are known during the modeling phase or if we are interested in regions where there is maximal ambiguity between the different clusters (e.g. for an active learning approach, see Figure 6).

A better way to describe the clusters’ shape would be cluster skeletons. There are several ways to calculate a cluster’s skeleton. By first calculating the concave hull of each cluster (see Figure 3 for a depiction of the generation process, following [18]) we can obtain the medial axis [19] which we use for simplicity here. Another way of obtaining the concave hull would be by using $\alpha$-shapes [20] or $\chi$-shapes [21]. Our proposed method has the slight advantage that the resulting hulls are more stable over a broader parameter range than the other methods. As such we do not need to put too much effort into the parametrization of the skeletonization process.

For a polygon (such as the concave hull) in the plane the medial axis is defined as the set of all points which do not have a unique nearest neighbor on the polygon. In the case of convex polygons these points would all lie inside the polygon. Since we are only interested in the part of the medial axis which spans the data set or cluster, we filter those parts of the medial axis which are not contained inside the polygon. The resulting skeleton undergoes an additional filtering step. For each edge of the skeleton at least two of the three following conditions must hold for it to be contained in the final, generalized centroid:

a) The edge must not end on the border polygon. Necessarily this happens when calculating the medial axis, but these parts of the axis can be seen as an overfitting of the cluster’s skeleton to the data. This condition must always be fulfilled.

b) The edge must be contained in the longest path along the medial axis. If we consider the data set shown in Figure 4 then the medial axis will contain an elongated series of edges that span a the complete cluster. Additionally several shorter edges will be part of the medial axis that run from the main axis towards indentations of the border polygon.

c) The edge must lie on any path that exists between each pair of degree 3 or more. These edges that run toward the indentations of the border polygon branch off of the main axis (thus creating nodes of degree 3 or possibly higher). In the case of the data set shown in Figure 5 these edges bear additional information about the clusters’ shapes, i.e. where clusters are broader or where clusters deviate from a single axis structure.

By pruning the branches we can avoid overfitting of the skeleton to the data while still getting a proper representation of the cluster structure.

With these skeletons we are now able to calculate fuzzy membership degrees in a very similar fashion to the fuzzy $c$-means membership degree calculation. The membership calculation (see Equation 1) requires the only the distance from a point to the cluster’s centroid. By replacing the previous centroid by the generalized centroid (or the cluster’s skeleton) we need to calculate the distance from the point to the skeleton instead.

Fig. 2: Two non-convex clusters. The centroids are no proper representatives for each cluster.

Fig. 3: Iterative refinement of a cluster’s hull. Starting from the convex hull (top left) edges on the hull are replaced one by one until the resulting hull fits the data points better [18].

V. GENERALIZED CENTROIDS

Essentially the idea behind assigning fuzzy memberships is sound and good as it is. It just cannot be easily applied to density-based clustering. If we recall Equation 1 then we see that the membership degree of a point to a cluster is always calculated by its distance to the cluster’s centroid. In the previous section we reviewed some ways that try to overcome the need for this calculation by either introducing fuzziness into the neighborhood definition or the core points themselves.

Our approach differs from these in a sense that we consider a closer approach to the fuzzy $c$-means approach. Instead of trying to find a way to assign fuzzy membership degrees to a data point by changing the neighborhood or core point definitions of DBSCAN we aim the centroid definition itself. From Figure 2 can be seen, that the centroids of density-based clusters do not always lie in the set of points forming the clusters anymore. Instead they are closer to the points of the other cluster.

Additionally our new algorithm allows to calculate interpretable fuzzy membership degrees to arbitrary points inside the data space. This may be useful if not all points are known during the modeling phase or if we are interested in regions where there is maximal ambiguity between the different clusters (e.g. for an active learning approach, see Figure 6).

A better way to describe the clusters’ shape would be cluster skeletons. There are several ways to calculate a cluster’s skeleton. By first calculating the concave hull of each cluster (see Figure 3 for a depiction of the generation process, following [18]) we can obtain the medial axis [19] which we use for simplicity here. Another way of obtaining the concave hull would be by using $\alpha$-shapes [20] or $\chi$-shapes [21]. Our proposed method has the slight advantage that the resulting hulls are more stable over a broader parameter range than the other methods. As such we do not need to put too much effort into the parametrization of the skeletonization process.

For a polygon (such as the concave hull) in the plane the medial axis is defined as the set of all points which do not have a unique nearest neighbor on the polygon. In the case of convex polygons these points would all lie inside the polygon. Since we are only interested in the part of the medial axis which spans the data set or cluster, we filter those parts of the medial axis which are not contained inside the polygon. The resulting skeleton undergoes an additional filtering step. For each edge of the skeleton at least two of the three following conditions must hold for it to be contained in the final, generalized centroid:

a) The edge must not end on the border polygon. Necessarily this happens when calculating the medial axis, but these parts of the axis can be seen as an overfitting of the cluster’s skeleton to the data. This condition must always be fulfilled.

b) The edge must be contained in the longest path along the medial axis. If we consider the data set shown in Figure 4 then the medial axis will contain an elongated series of edges that span a the complete cluster. Additionally several shorter edges will be part of the medial axis that run from the main axis towards indentations of the border polygon.

c) The edge must lie on any path that exists between each pair of degree 3 or more. These edges that run toward the indentations of the border polygon branch off of the main axis (thus creating nodes of degree 3 or possibly higher). In the case of the data set shown in Figure 5 these edges bear additional information about the clusters’ shapes, i.e. where clusters are broader or where clusters deviate from a single axis structure.

By pruning the branches we can avoid overfitting of the skeleton to the data while still getting a proper representation of the cluster structure.

With these skeletons we are now able to calculate fuzzy membership degrees in a very similar fashion to the fuzzy $c$-means membership degree calculation. The membership calculation (see Equation 1) requires the only the distance from a point to the cluster’s centroid. By replacing the previous centroid by the generalized centroid (or the cluster’s skeleton) we need to calculate the distance from the point to the skeleton instead.
Fig. 4: Cluster hulls \((p = 0.2, \text{according to } [18], \text{left})\) and the resulting generalized centroid for each cluster (right).

Fig. 5: Cluster hulls \((p = 0.2, \text{according to } [18], \text{left})\) with the original, unpruned skeleton (middle, all edges that belong to the medial axis except those that run onto the border) and the resulting, pruned generalized centroid for each cluster (right).

Fig. 6: Two non-convex clusters. The centroids are no proper representatives for each cluster.

Let \(H^* \subseteq C\) be the concave hull of a cluster \(C\). This can be either the shape descriptor obtained by [18], the \(\alpha\)-shape [20] or \(\chi\)-shape [21]. \(H^*_C\) can be represented by a set of edges \(e_H = \langle x_i, x_j \rangle\) between the points \(x_i, x_j\) on the border polygon in counter-clockwise order (CCW). The points on the border polygon are necessarily points of the data set \(X\).

Let \(\mathcal{M}_C = \{p \in \mathbb{R}^d | p \text{ has no unique nearest neighbor on } H^*_C\}\) be the cluster’s medial axis. This can be calculated as the Voronoi diagram of the border polygon’s edges (not the points, as one would do in \(k\)-means clustering). Since our polygon has only straight line segments as border the medial axis will also mostly consist of straight line segments \(e_{\mathcal{M}_C} = \langle p_i, p_j \rangle\). Under certain circumstances the edges may be arched. For simplicity in the following calculations we assume direct connections between these edges’ points. Note that the \(p_i\) forming the edges now do not need to be actual points of the data set. Let \(E(\mathcal{M}_C)\) denote the set of edges (straight line segments) that represent the medial axis.

Let \(S_C = \{e \in E(\mathcal{M}_C) | \text{the conditions a, b and c hold for } e\}\) be the cluster’s skeleton. With this skeleton given we are now able to compute the fuzzy membership of a data point to any given cluster (see Figure 6):

\[
\mu(x_j, S_C) = \frac{d_S^{s} \frac{2}{1-s}}{\sum_{k=1}^{c} d_S^{s} \frac{1}{1-s}} = \frac{1}{\sum_{k=1}^{c} (\frac{d_S^{s}}{d_S^{s}})^{\frac{1}{1-s}}}
\]  
(2)

The only thing that has changed in the calculation is that we do not compute the membership to the centroid of the cluster but to its skeleton. This – of course – changes how the \(d_S^{s}\) are computed. Instead of simply taking the euclidean distance between the points \(x_j, c_i\) we now have to deal with several line segments. The distance between a point and a straight line

\[
l(\lambda) = p_i + \lambda(p_j - p_i)
\]  
(3)

can be calculated by projecting the vector between the point and the base point of the line onto the directional vector of the line. By this we find the closest point on the line and can
For line segments however the closest point on the line carrying the line segment may lie outside of the segment itself. In this case we need to know whether the point lies before the line segment’s start or behind its end point. In that case the smallest distance between a point and the line segment would be the distance between the point and the starting or ending point respectively.

\[ \vec{v} = \vec{b} - \vec{a} \]
\[ \vec{w} = x - \vec{a} \]
\[ c_1 = \vec{v} \cdot \vec{w} \]
\[ c_2 = \vec{v} \cdot \vec{w} \]
\[ d(\vec{x}_j, e) = \begin{cases} d(\vec{x}_j, \vec{a}) & \text{if } c_1 < 0 \\ d(\vec{x}_j, \vec{b}) & \text{if } c_2 < c_1 , \\ d(\vec{x}_j, l(c_1/c_2)) & \end{cases} \]

with \( e = (\vec{a}, \vec{b}) \).

\( \vec{v} \) is the directional vector of the carrier line, \( \vec{w} \) the vector between the base point and the point for which we want to calculate the distance. \( c_1 \) gives us the value for \( \lambda \) such that the vector from \( l(\lambda) \) towards \( x \) is minimal. If \( c_1 \) is smaller than 0 then the point lies before the starting point of the line segment and we need to calculate the distance between \( x \) and \( p_i \). If it is larger than \( c_2 \) then \( x \) lies behind the end point of the line segment and we need to calculate the distance between \( x \) and \( p_f \). In all other cases we calculate the projection of \( \vec{w} \) onto \( \vec{v} \) to get the point closest to \( x \) and calculate their distance.

If now we want to calculate the distance between a point \( x \) and a cluster’s skeleton \( S_{C_i} \), we calculate the distance between \( x \) and every line segment building \( S_{C_i} \). The minimum over all these distances is the distance between \( x \) and \( S_{C_i} \):

\[ d_{S_j}^x = \min_{e \in S_{C_i}} d(x_j, e) \]

Because of the possibly non-convex structure of the skeleton and the varying lengths of the segments it is necessary to actually check every line segment. Heuristics proposing which edges are better candidates may fail even for the simplest skeletons.

VI. Evaluation

Internal validation measures allow to compare two different clusterings of the same data set and choose the better clustering. This can be either used to find the most appropriate number of clusters in a data set (if used with centroid or model-based methods) or to choose proper parameters for a density based clustering algorithm (cf. [8]). Here we will show that our algorithm yields appropriate partitioning matrices that can be used to evaluate clustering results obtained by density based algorithms.

The data sets we use for evaluation show several interesting properties. In Figure 2 we already see intersecting convex hulls, which leads to common centroids not being representative anymore. Figure 7 shows the aggregation data set [22]. Clusters in this data set are not properly separated (lower left) or connected by short bridges. Figure 8 shows the compound data set [23]. Clusters in this data set vary in density or are completely contained in each others convex hulls (lower left). Figure 9 shows the r15 data set [24]. Clusters are very close to each other in the data set’s central region. Figure 10 shows the t4.8k data set [25]. Clusters in this data set are overlapping and connected by a thin bridge of points following a sinusoidal curve. This data set contains – in addition – a high amount of noise.

To choose whether to use one specific partitioning of the data or another internal cluster validation measures can be used. In contrast to external validation measures that rely on knowledge about the ground truth of the data (or can be used to calculate to which extent two different clusterings agree), internal validation measures are solely based on the clustering
Fig. 9: r15 data set. With clusters found by DBSCAN ($\varepsilon = 1.4, \text{minPts} = 10$) and cluster skeletons. Black points indicate noise.

Fig. 10: t4.8k data set. With clusters found by DBSCAN ($\varepsilon = 10, \text{minPts} = 20$) and cluster skeletons. Black points indicate noise.

result and maybe some information about the data’s structure.

The partition coecfficent ($PC$, see Equation 10) [26] and the partition entropy ($PE$, see Equation 11) [27] can be used to compute scores solely based on the fuzzy partition matrix itself. While the $PE$ should be minimized, the optimal value for the $PC$ is 1.0. Both optima are actually reached for a crisp clustering where all membership degrees are either 1.0 or 0.0.

Partition Coecfficent [26]

$$PC = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^2 \tag{10}$$

Partition Entropy [27]

$$PE = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij} \cdot \log u_{ij} \tag{11}$$

An index that also uses structural information about the data set is the Xie-Beni-Index ($XB_\omega$, see Equation 12) [28].

$$XB_\omega = \frac{\sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij} \cdot d(S_{C_j}, S_{C_k})}{n \cdot \min_{C_j \neq C_k} d(S_{C_j}, S_{C_k})} \tag{12}$$

Table II shows the results for the test cases chosen for this evaluation. Compared to bad clusterings (i.e. low agreement with the ground truth) of the data set, our algorithm always yields the better or similar results. In the case of the r15 data set the worse clustering is produced by choosing the parameters in a way that the eight central clusters are placed under the same cluster label (i.e. only eight clusters in total were generated). This central structure seems to yield a better partitioning (w.r.t. the chosen validation indices) than the ground truth of the data set.

VII. CONCLUSION

In this paper we have presented a novel fuzzy density-based clustering algorithm. Our approach takes the crisp clustering result and calculates a cluster skeleton based on the concave hull of the cluster. The fuzzy membership degrees are then calculated in a way similar to the well-known fuzzy $c$-means algorithm. The distance calculations needed are not performed against the arithmetic mean of all points belonging to the cluster but instead against the cluster’s skeleton which acts as a new, generalized centroid. Problems can occur if one cluster is completely contained in another. This may lead to intersections between the skeletons (and the skeletons are – again – not suitable as representations for the clusters). Another problem that might occur is overfitting. If the hulls representing the clusters are too serrated, the resulting skeleton may become too dendritic and adapt to the data too well.

However, experimental results based on artificial data sets and selected, internal evaluation measures show, that our algorithm yields membership degrees that can be used to choose the a better set of parameters w.r.t. a fuzzy cluster definition given implicitly by the used validation index. This can be used to automatically determine the set of optimal parameters and perform fuzzy density-based clustering with only a cluster definition and no explicit parametrization of the density-based clustering algorithm itself.
<table>
<thead>
<tr>
<th>Name</th>
<th>ε</th>
<th>minPts</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggregation</td>
<td>2</td>
<td>20</td>
</tr>
<tr>
<td>compound</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>r15</td>
<td>2.75</td>
<td>30</td>
</tr>
<tr>
<td>t4.8k</td>
<td>10</td>
<td>30</td>
</tr>
</tbody>
</table>

**APPENDIX**

Table III shows the parameters used to obtain the bad clusterings for the evaluation. We used the DBSCAN implementation from [30]. The data sets were obtained from https://cs.joensuu.fi/sipu/datasets/.

**REFERENCES**


