Agglomerative Fuzzy Clustering

Christian Borgelt and Rudolf Kruse

Abstract The term *fuzzy clustering* usually refers to prototype-based methods that optimize an objective function in order to find a (fuzzy) partition of a given data set and are inspired by the classical *c*-means clustering algorithm. Possible transfers of other classical approaches, particularly hierarchical agglomerative clustering, received much less attention as starting points for developing fuzzy clustering methods. In this paper we strive to improve this situation by presenting a (hierarchical) agglomerative fuzzy clustering algorithm. We report experimental results on two well-known data sets on which we compare our method to classical hierarchical agglomerative clustering.

1 Introduction

The objective of *clustering* or *cluster analysis* is to divide a data set into groups (so-called *clusters*) in such a way that data points in the same cluster are as similar as possible and data points from different clusters are as dissimilar as possible (see, e.g., [5, 10]), where the notion of similarity is often formalized by defining a distance measure for the data points. Even in classical clustering the resulting grouping need not be a partition (that is, in some approaches not all data points need to be assigned to a group and the formed groups may overlap), but only if points are assigned to different groups with different degrees of membership, one arrives at *fuzzy clustering* [14, 2, 3, 8].

However, the term *fuzzy clustering* usually refers to a fairly limited set of methods, which are prototype-based and optimize some objective function to find a good (fuzzy) partition of the given data. Although classical clustering comprises many more methods than the well-known *c*-means algorithm (by

School of Computer Science, Otto-von-Guericke-University Magdeburg
Universitätsplatz 2, 39106 Magdeburg, Germany
rudolf.kruse@ovgu.de, christian@borgelt.net

which most fuzzy clustering approaches are inspired), these other methods are only rarely “fuzzified”. This is particularly true for hierarchical agglomerative clustering (HAC) [16], of which only few fuzzy versions have been proposed.

Exceptions include [7], [11] and [1]. [7] describes HAC for trapezoidal fuzzy sets with either single or complete linkage, but is restricted to one dimension due to its special distance function. [11] proposes an HAC algorithm for crisp data based on fuzzy distances, which are effectively distances weighted by a function of membership degrees. It mixes single and complete linkage. [1] merges clusters in the spirit of HAC, but keeps the original clusters for possible additional mergers, so that a hierarchy in the form of a directed acyclic graph results (while standard HAC produces a tree). Also noteworthy is [15], which suggest a mixed approach, re-partitioning the result of fuzzy c-means clustering and linking the partitions of two consecutive steps.

Related approaches include [6] and [12] as well as its extension [9]. The first uses a competitive agglomeration scheme and an extended objective function for fuzzy c-means in order to reduce an overly large initial number of clusters to an “optimal” number. The latter two change the term in the objective function that penalizes many clusters from a quadratic expression to an entropy expression. Although fairly different from hierarchical agglomerative clustering approaches, they share the property that clusters are merged to find a good final partition, but they do not necessarily produce a hierarchy.

Our approach is closest in spirit to [11], as it also relies on the standard scheme of hierarchical agglomerative clustering, although we treat the original data points as clusters already, while [11] keeps data points and clusters clearly separate. Furthermore, [11] focuses on single and complete linkage while we use a centroid scheme. Our approach also bears some relationship to [15] concerning the distances of fuzzy sets, which [15] divides into three categories: (1) comparing membership values, (2) considering spatial characteristics, and (3) characteristic indices. While [15] relies on (2), we employ (1).

The remainder of this paper is structured as follows: in Sections 2 and 3 we briefly review standard fuzzy clustering and hierarchical agglomerative clustering, indicating which elements we use in our approach. In Section 4 we present our method and in Section 5 we report experimental results. Finally, we draw conclusions from our discussion in Section 6.

2 Fuzzy Clustering

The input to our clustering algorithm is a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with n data points, each of which is an m-dimensional real-valued vector, that is, $\forall j; 1 \leq j \leq n : \mathbf{x}_j = (x_{j1}, \ldots, x_{jm}) \in \mathbb{R}^m$. Although HAC usually requires only a distance or similarity matrix as input, we assume metric data, since a centroid scheme requires the possibility to compute new center vectors.

In standard fuzzy clustering one tries to minimize the objective function

$$J(\mathbf{X}, \mathbf{C}, \mathbf{U}) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{w} \, d_{ij}^{2},$$

where $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_c\}$ is the set of *prototypes* (often merely cluster centers), the $c \times n$ matrix $\mathbf{U} = (u_{ij})_{1 \leq i \leq c; 1 \leq j \leq n}$ is the *partition matrix* containing the degrees of membership with which the data points belong to the clusters, the d_{ij} are the distances between cluster $\mathbf{c}_i$ and data point $\mathbf{x}_j$, and w, $w > 1$, is the *fuzzifier* (usually $w = 2$), which controls the "softness" of the cluster boundaries (the larger w, the softer the cluster boundaries). In order to avoid the trivial solution of setting all membership degrees to zero, the constraints $\forall j; 1 \leq j \leq n : \sum_{i=1}^{c} u_{ij} = 1$ and $\forall i; 1 \leq i \leq c : \sum_{j=1}^{n} u_{ij} > 0$ are introduced.

A fuzzy clustering algorithm optimizes the above function, starting from a random initialization of either the cluster prototypes or the partition matrix, in an alternating fashion: (1) optimize membership degrees for fixed prototypes and (2) optimize prototypes for fixed membership degrees. From this scheme we take the computation of membership degrees for $w = 2$, namely

$$u_{ij} = d_{ij}^{-2} \Big/ \textstyle\sum_{k=1}^{c} d_{kj}^{-2}.$$

We compute membership degrees for fuzzy clusters in an only slightly modified fashion, which are then compared to decide which clusters to merge.

3 Hierarchical Agglomerative Clustering

As its name already indicates, *hierarchical agglomerative clustering* produces a hierarchy of clusters in an agglomerative fashion, that is, by merging clusters (in contrast to divise approaches, which split clusters). It starts by letting each data point form its own cluster and then iteratively merges those two clusters that are most similar (or closest to each other).

While the similarity (or distance) of the data points is an input to the procedure, how the distances of (non-singleton) clusters are to be measured is a matter of choice. Common options include (1) *single linkage* (cluster distances are minimum distances of contained data points), (2) *complete linkage* (maximum distances of contained data points), and (3) *centroid* (distances of cluster centroids). Note that the centroid method requires that one can somehow compute a cluster center (or at least an analog), while single and complete linkage only require the initial similarity or distance matrix of the data points. Because of this we assume metric data as input.

In the single and complete linkage methods, clusters are merged by simply pooling the contained data points. In the centroid method, clusters are merged by computing a new cluster centroid as the weighted mean of the centroids of the clusters to be merged, where the weights are provided by the relative number of data points in the clusters to be merged.

4 Agglomerative Fuzzy Clustering

Our algorithm builds on the idea to see the given set of data points as the initial cluster centers (as in standard HAC) and to compute membership degrees of all data points to these cluster centers. However, for this the membership computation reviewed in Section 2 is not quite appropriate, since it leads to each data point being assigned to itself and to itself only (only one membership degree is 1, all others are 0). As a consequence, there would be no similarity between any two clusters (at least in the initial partition) and thus no proper way to choose a cluster merger. In order to circumvent this problem, we draw on the concept of a "raw" membership degree, which is computed from a distance via a radial function, where "raw" means that its value is not normalized to sum 1 over the clusters [4]. Possible choices for such a radial function (with parameters α and σ^2, respectively) are

$$f_{\text{Cauchy}}(r;\alpha) = \frac{1}{r^2+\alpha} \qquad \text{and} \qquad f_{\text{Gauss}}(r;\sigma^2) = e^{-\frac{r^2}{2\sigma^2}},$$

where r is the distance to a cluster center. Using these functions (with $\alpha > 0$) prevents singularities at the cluster centers that occur with the simple inverted squared distance (that is, for $\alpha = 0$) and thus allows us to compute suitable membership degrees even for the initial set of clusters, namely as

$$u_{ij}^{(\alpha)} = \frac{f_{\text{Cauchy}}(d_{ij};\alpha)}{\sum_{k=1}^{c} f_{\text{Cauchy}}(d_{kj};\alpha)} \qquad \text{or} \qquad u_{ij}^{(\sigma^2)} = \frac{f_{\text{Gauss}}(d_{ij};\sigma^2)}{\sum_{k=1}^{c} f_{\text{Gauss}}(d_{kj};\sigma^2)}.$$

Based on these membership degrees two clusters $\mathbf{c}_i$ and $\mathbf{c}_k$ can now be compared by aggregating (here: simply summing) point-wise comparisons:

$$\delta_{ik} = \sum_{j=1}^{n} g(u_{ij}, u_{kj}),$$

where g is an appropriately chosen difference function. Here we consider

$$g_{\text{abs}}(x,y) = |x-y|, \quad g_{\text{sqr}}(x,y) = (x-y)^2 \quad \text{and} \quad g_{\text{wgt}}(x,y) = |x-y|(x+y).$$

The first function, g_{abs}, may appear the most natural choice, while g_{sqr} generally weights large differences more strongly and g_{wgt} emphasizes large differences of large membership degrees and thus focuses on close neighbors.

A fuzzy HAC algorithm can now be derived in a standard fashion: compute the initial membership degrees by using each data point as a cluster center. Compute the cluster dissimilarities δ_{ik} for this initial set of clusters. Merge the two clusters $\mathbf{c}_i$ and $\mathbf{c}_k$, for which δ_{ik} is smallest, according to

$$\mathbf{c}_* = \frac{1}{\sum_{j=1}^{n}(u_{ij}+u_{kj})}\left(\mathbf{c}_i \sum_{j=1}^{n} u_{ij} + \mathbf{c}_k \sum_{j=1}^{n} u_{kj}\right).$$

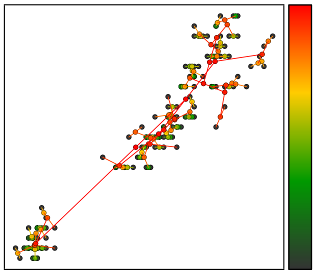


Fig. 1 Result of standard hierarchical agglomerative clustering (i.e. crisp partitions) with the centroid method on the well-known Iris data, attributes petal_length (horizontal) and petal_width (vertical). The colors encode the step in which clusters are merged (from bottom to top on the color bar shown on the right); the data points are shown in gray.

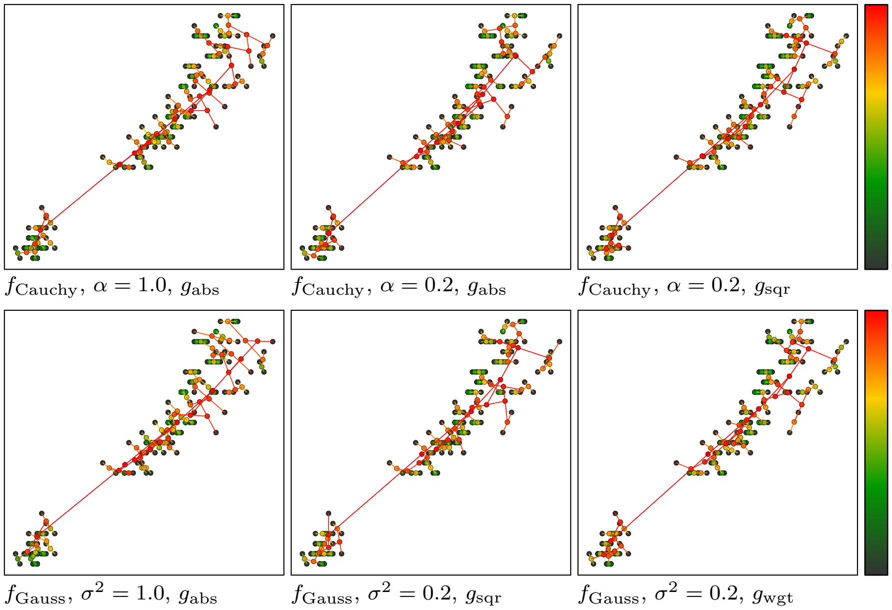


Fig. 2 Results of different versions of agglomerative fuzzy clustering on the Iris data, attributes petal_length (horizontal) and petal_width (vertical).

That is, the sum of membership degrees for each cluster is used as the relative weight of the cluster for the merging and thus (quite naturally) replaces the number of data points in the classical HAC scheme. The merged clusters $\mathbf{c}_i$ and $\mathbf{c}_k$ are removed and replaced by the result $\mathbf{c}_*$ of the merger.

For the next step membership degrees and cluster dissimilarities are recomputed and again the two least dissimilar clusters are merged. This process is repeated until only one cluster remains. From the resulting hierarchy a suitable partition may then be chosen to obtain a final result (if so desired), which may be further optimized by applying standard fuzzy c-means clustering.

Note that this agglomerative fuzzy clustering scheme is computationally considerably more expensive than standard HAC, since all membership degrees and cluster dissimilarities need to be re-computed in each step.

5 Experimental Results

We implemented our agglomerative fuzzy clustering method prototypically in Python, allowing for the two radial functions (Cauchy and Gauss, with parameters α and σ^2) to compute membership degrees and the three cluster dissimilarity measures (g_{abs}, g_{sqr} and g_{wgt}) to decide which clusters to merge. We applied this implementation in a simple first test of functionality to two well-known data sets from the UCI machine learning repository [13], namely the Iris data and the Wine data. For the clustering runs we used attributes petal_length and petal_width for the Iris data and attributes 7, 10 and 13 for the Wine data, since these are the most informative attributes w.r.t. the class structure of these data sets. This restriction of the attributes also allows us to produce (low-dimensional) diagrams with which the cluster hierarchies can be easily compared. The latter is important, since it is difficult to find an undisputed way of evaluating clustering results. Visual representations in diagrams at least allow to compare the results subjectively and provide some insight about the properties of the different variants.

As a comparison we applied standard hierarchical agglomerative clustering (HAC) with the centroid method for linking clusters. As it also produces a hierarchy of clusters (cluster tree) the results can be displayed in the same manner and thus are easy to compare. For both standard HAC and agglomerative fuzzy clustering we z-normalized the data (that is, we normalized each attribute to mean 0 and standard deviation 1) in order to avoid effects resulting from different scales (which is particularly important for attribute 13 of the Wine data set, which spans a much larger range than all other attributes and thus would dominate the clustering without normalization).

A selection of results we obtained are shown in Figures 1 and 2 for the Iris data and in Figures 3 and 4 for the Wine data. Since in our approach cluster dissimilarity basically depends on all data points, the distribution of the data points in the data space has a stronger influence on the mergers to be carried out. For example, for the Iris data, which is mainly located along a diagonal of the data space, mergers with our algorithm tend to be carried out more often in a direction perpendicular to this diagonal. How strong this effect is depends on the parameters: a smaller α or σ^2 reduces this effect. For the wine data set, which has a more complex data distribution, we believe that we can claim that the resulting cluster trees better respects the distribution of the data points than standard HAC does.

6 Conclusions

We described a (hierarchical) agglomerative fuzzy clustering algorithm (fuzzy HAC) that is based on a cluster dissimilarity measure derived from aggregated point-wise membership differences. Although it is computationally more ex-

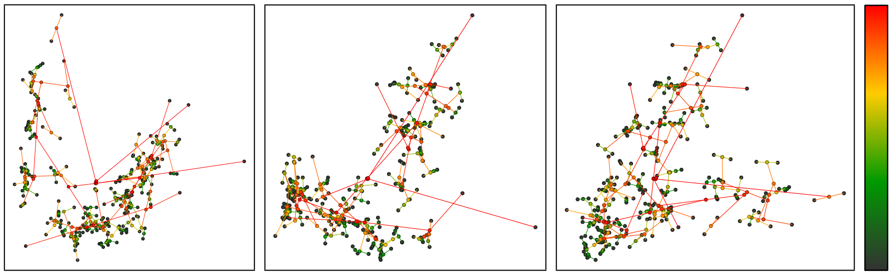
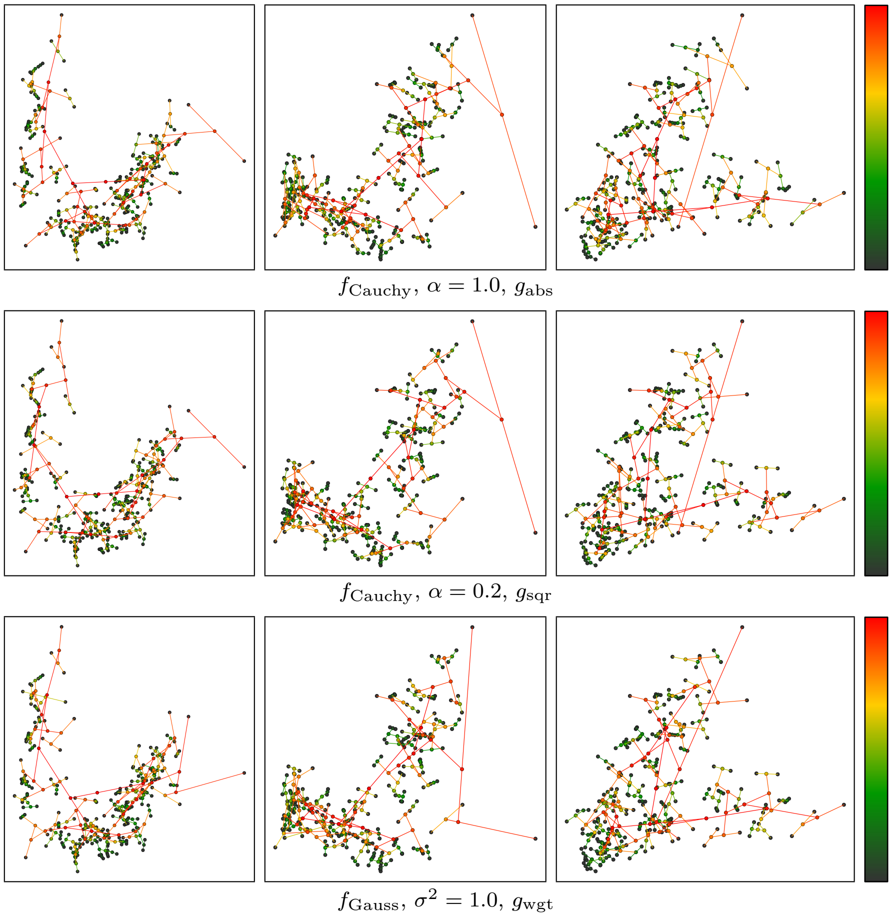


Fig. 3 Result of standard hierarchical agglomerative clustering (i.e. crisp partitions) with the centroid method on the Wine data, attributes 7 and 10 (left), 7 and 13 (middle) and 10 and 13 (right). The colors encode the step in which clusters are merged (from bottom to top on the color bar shown on the right); the data points are shown in gray.



f_{Cauchy}, $\alpha = 1.0$, g_{abs}

f_{Cauchy}, $\alpha = 0.2$, g_{sqr}

f_{Gauss}, $\sigma^2 = 1.0$, g_{wgt}

Fig. 4 Results of different versions of agglomerative fuzzy clustering on the Wine data, projections to attributes 7 and 10 (left), 7 and 13 (middle) and 10 and 13 (right).

pensive than classical (crisp) HAC, a subjective evaluation of its results seem to indicate that it may be able to produce cluster hierarchies that better fit the spatial distribution of the data points than the hierarchy obtained with classical HAC. Future work includes a more thorough investigation of the effects of its parameters (α and σ^2 and the choice of the dissimilarity function, as well as the fuzzifier, which we neglected in this paper). Furthermore, an intermediate (partial) optimization of the cluster centers with fuzzy c-means is worth to be examined and may make it possible to return to simple inverted squared distances to compute the membership degrees.

References

1. Bank M and Schwenker F (2012) Fuzzification of Agglomerative Hierarchical Crisp Clustering Algorithms. *Proc. 34th Ann. Conf. Gesellschaft für Klassifikation (GfKl 2010, Karlsruhe, Germany)*, 3–11. Springer, Heidelberg/Berlin, Germany
2. Bezdek JC (1981) *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, NY, USA
3. Bezdek JC, Keller J, Krishnapuram R, and Pal N (1999) *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing.* Kluwer, Dordrecht, Netherlands
4. Borgelt C (2005) *Prototype-based Classification and Clustering.* Habilitationsschrift, Otto-von-Guericke-University of Magdeburg, Germany
5. B.S. Everitt (1981) *Cluster Analysis.* Heinemann, London, United Kingdom
6. Frigui H and Krishnapuram R (1997) Clustering by Competitive Agglomeration. *Pattern Recognition* 30(7):1109–1119. Elsevier, Amsterdam, Netherlands
7. Ghasemigol M, Yazdi HS, and Monsefi R (2010) A New Hierarchical Clustering Algorithm on Fuzzy Data (FHCA). *Int. J. Computer and Electrical Engineering* 2(1):134–140. International Academy Publishing (IAP), San Bernadino, CA, USA
8. Höppner F, Klawonn F, Kruse R, and Runkler T (1999) *Fuzzy Cluster Analysis.* J. Wiley & Sons, Chichester, United Kingdom
9. Hsu M-J, Hsu P-Y, and Dashnyam B (2011) Applying Agglomerative Fuzzy K-Means to Reduce the Cost of Telephone Marketing. *Proc. Int. Symp. Integrated Uncertainty in Knowledge Modelling and Decision Making (IUKM 2011, Hangzhou, China)*, 197–208.
10. L. Kaufman and P. Rousseeuw (1990) *Finding Groups in Data: An Introduction to Cluster Analysis.* J. Wiley & Sons, New York, NY, USA
11. Konkol M (2015) Fuzzy Agglomerative Clustering. *Proc. 14th Int. Conf. Artificial Intelligence and Soft Computing (ICAISC 2015, Zakopane, Poland)*, 207-217. Springer, Heidelberg/Berlin, Germany
12. Li M, Ng M, Cheung Y-M, and Huang J (2008) Agglomerative Fuzzy K-Means Clustering Algorithm with Selection of Number of Clusters. *IEEE Trans. on Knowledge and Data Engineering* 20(11):1519–1534. IEEE Press, Piscataway, NJ, USA
13. Lichman M (2013) UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. University of California, Irvine, CA, USA
14. Ruspini EH (1969) A New Approach to Clustering. *Information and Control* 15(1):22–32. Academic Press, San Diego, CA, USA
15. Torra V (2005) Fuzzy c-means for Fuzzy Hierarchical Clustering. *Proc. IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE 2005, Reno, Nevada)*, 646–651. IEEE Press, Piscataway, NJ, USA
16. Ward JH (1963) Hierarchical Grouping to Optimize an Objective Function. J. American Statistical Association 58(301):236–244