# Prototype-less Fuzzy Clustering

Christian Borgelt

***Abstract*—In contrast to standard fuzzy clustering, which optimizes a set of prototypes, one for each cluster, this paper studies fuzzy clustering without prototypes. Starting from an objective function that only involves the distances between data points and the membership degrees of the data points to the different clusters, an iterative update rule is derived. The properties of the resulting algorithm are then examined, especially w.r.t. to schemes that focus on a constrained neighborhood for each data point. Corresponding experimental results are reported that demonstrate the merits of this approach.**

## I. Introduction

Fuzzy clustering algorithms [7], [2], [3], [15] are very popular methods for finding groups in data, especially in domains where groups are imperfectly separated and thus a crisp assignment of data points to clusters is inappropriate. These algorithms are usually prototype-based: they try to optimize a set of prototypes, one for each cluster, which consist of a cluster's location, size, and shape parameters. The goal of fuzzy clustering is then defined by an objective function, which involves the data points, the prototypes, and the membership degrees of the data points to the clusters, and is usually to be minimized. The most common objective function is

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

where $\mathbf{X} = \{\vec{x}_j \mid 1 \leq j \leq n\}$ is the given data set, consisting of $n$ vectors ($m$-dimensional) and $\mathbf{C} = \{\mathbf{c}_i \mid 1 \leq i \leq c\}$ is the set of cluster prototypes. $d_{ij}$ denotes the distance between datum $\vec{x}_j$ and the $i$-th cluster (where this distance may depend not only on a cluster center, but also on cluster-specific parameters describing the cluster's size and shape [12], [11], [5]). $u_{ij} \in [0, 1]$ is the degree of membership to which data point $x_j$ belongs to the $i$-th cluster. The $c \times n$ matrix $\mathbf{U} = (u_{ij})_{1 \leq i \leq c, 1 \leq j \leq n}$ combines the individual assignments and is called the *(fuzzy) partition matrix*. Finally $w$ is the so-called *fuzzifier*, which controls the crispness of the assignment: the higher its value, the softer are the cluster boundaries.

In order to rule out the trivial (but useless) solution $\forall i, j; u_{ij} = 0$ and to ensure that no cluster is empty, one introduces the constraints

$$\forall j; 1 \leq j \leq n : \sum_{i=1}^{c} u_{ij} = 1, \quad \forall i; 1 \leq i \leq c : \sum_{j=1}^{n} u_{ij} > 0.$$

Different fuzzy clustering algorithms are then distinguished based on the cluster prototypes and the distance measure.

Christian Borgelt is with the European Center for Soft Computing, Edificio Científico-Tecnológico, c/ Gonzalo Gutiérrez Quirós s/n, 33600 Mieres, Asturias, Spain (email: christian.borgelt@softcomputing.es).

The most common fuzzy clustering algorithm is a straightforward generalization of classical $k$-means clustering [1], [14], [19] to fuzzy membership degrees: the fuzzy $c$-means algorithm [2], [3], [15] is based on point prototypes and uses the Euclidean distance. More sophisticated variants introduce cluster-specific covariance matrices (to describe ellipsoidal shapes), sizes, and weights (see, for example, [12], [11], [5]).

The optimization scheme, derived by exploiting the necessary condition that all partial derivatives of the objective function w.r.t. the parameters (membership degrees, cluster parameters) must vanish at a minimum, is usually alternating, so that membership degrees and cluster prototypes are optimized separately, while the other group of parameters is fixed.

In this paper, however, I present an algorithm that does not employ prototypes to describe the clusters, but uses only a partition matrix. In addition to the basic algorithm (Section II), the derivation of which follows the standard paths for fuzzy clustering, I study schemes that focus on a constrained neighborhood for the update of the fuzzy membership degrees (Section III). Among these approaches those that are based on a neighborhood graph are particularly interesting.

## II. The Basic Algorithm

The basic idea of the presented algorithm is that data points that are far away from each other should not have high degrees of membership to the same cluster, while for data points that are close together, high degrees of membership to the same cluster are not only acceptable, but actually desirable. The scheme has some relation to the reformulation approach [13], which, if it is used to eliminate the update of the prototype parameters rather than the update of the membership degrees, leads to a similar, but more complex objective function, and to fuzzy $k$-nearest neighbor algorithms [17], from which one may also derive a candidate update rule.

### A. Objective Function

A natural way to code the intuitive fuzzy clustering goal outlined above is the objective function

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

which is to be minimized subject to the usual constraints

$$\forall j; 1 \leq j \leq n : \sum_{i=1}^{c} u_{ij} = 1, \quad \forall i; 1 \leq i \leq c : \sum_{j=1}^{n} u_{ij} > 0.$$

Here $d_{jk}$ is the distance between the data points $x_j$ and $x_k$ and $u_{ij}$ and $u_{ik}$ are the degrees of membership to which the data points $x_j$ and $x_k$, respectively, belong to the $i$-th cluster.

The *fuzzifier* $w$ controls again the crispness of the assignment: the higher its value, the softer is the clustering result.

Clearly the value of this objective function is the higher, the more distant data points are assigned to the same cluster, while assigning close data points to the same cluster is relatively harmless (that is, does not increase the function value much). Hence minimizing the above function can be expected to yield an appropriate fuzzy partition matrix.

### B. Update Procedure

The derivation of the update rule for the membership degrees follows the standard lines known from prototype-based fuzzy clustering, for example, fuzzy $c$-means. The constraint that the membership degrees of each data point must add up to one is introduced into the objective function with the help of Lagrange multipliers, yielding the Lagrange function

$$\mathcal{L}(\mathbf{X}, \mathbf{U}, \Lambda) = \sum_{i=1}^{c} \sum_{j=1}^{n} \sum_{k=1}^{j-1} u_{ij}^{w} u_{ik}^{w} d_{jk}^{2} + \sum_{k=1}^{n} \lambda_k \left(1 - \sum_{i=1}^{c} u_{ik}\right).$$

This Lagrange function is then minimized instead of the objective function, thus implicitly respecting the constraint. One exploits that a necessary condition for a minimum is that the partial derivatives w.r.t. the parameters (here only the membership degrees) vanish. That is, at a minimum of the Lagrange function we have $\forall a, 1 \leq a \leq c : \forall b, 1 \leq b \leq n :$

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial u_{ab}} &= \sum_{\substack{k=1 \\ k \neq b}}^{n} w u_{ab}^{w-1} u_{ak}^{w} d_{kb}^{2} - \lambda_b \\ &= w u_{ab}^{w-1} \sum_{k=1}^{n} u_{ak}^{w} d_{kb}^{2} - \lambda_b \quad = \quad 0. \end{aligned}$$

(Note that the index condition $k \neq b$ can be dropped, because $\forall b : d_{bb} = 0$ and thus the corresponding term always vanishes.) This condition leads to $\forall i, 1 \leq i \leq c : \forall j, 1 \leq j \leq n :$

$$u_{ij} = \left( \frac{\lambda_j}{w \sum_{k=1}^{n} u_{ik}^{w} d_{jk}^{2}} \right)^{\frac{1}{w-1}}.$$

Summing these equations over the clusters (in order to be able to exploit the corresponding constraint on the membership degrees: they must sum to 1), we obtain

$$1 = \sum_{i=1}^{c} u_{ij} = \sum_{i=1}^{c} \left( \frac{\lambda_j}{w \sum_{k=1}^{n} u_{ik}^{w} d_{jk}^{2}} \right)^{\frac{1}{w-1}}.$$

Consequently, the $\lambda_j$, $1 \leq j \leq n$, are

$$\lambda_j = \left( \sum_{i=1}^{c} \left( w \sum_{k=1}^{n} u_{ik}^{w} d_{jk}^{2} \right)^{\frac{1}{1-w}} \right)^{1-w}.$$

Inserting this result into the equations for the membership degrees yields $\forall i, 1 \leq i \leq c : \forall j, 1 \leq j \leq n :$

$$u_{ij} = \frac{\left( \sum_{k=1}^{n} u_{ik}^{w} d_{jk}^{2} \right)^{\frac{1}{1-w}}}{\sum_{l=1}^{c} \left( \sum_{k=1}^{n} u_{lk}^{w} d_{jk}^{2} \right)^{\frac{1}{1-w}}}.$$

which for the special case $w = 2$ (which is the most common choice for prototype-based fuzzy clustering) simplifies to

$$u_{ij} = \frac{\left( \sum_{k=1}^{n} u_{ik}^{2} d_{jk}^{2} \right)^{-1}}{\sum_{l=1}^{c} \left( \sum_{k=1}^{n} u_{lk}^{2} d_{jk}^{2} \right)^{-1}}.$$

Since this (non-linear) equation system is technically highly difficult to solve (due to the somewhat complicated interdependence of the membership degrees), I draw on the same trick that is exploited in prototype-based fuzzy clustering: alternating optimization. That is, I use the above equation as an update rule that is applied iteratively in order to approach a (possibly only local) optimum.

In principle, this may even be done in two ways: an online fashion, in which the updated membership degrees immediately replace the old membership degrees and thus are used directly for updating other membership degrees, and in a batch fashion, where a full new set of membership degrees is computed in one step from the old membership degrees. However, several experiments revealed that a batch update is not feasible in practice, regardless of the initialization (see below): a batch update process is highly unstable and often ends with a fairly random crisp assignment of the data points.

As a consequence I confine myself in this paper to an online update, which cycles through the data points. That is, in each step all membership degrees of one data point are recomputed (which is necessary due to the normalization involved in the computation of the membership degrees). In order to avoid effects that could result from a special order of the data points, the update order is changed after every epoch, that is, the data points are shuffled after each traversal.

Note that the update rule refers only to pairwise distances between the data points and thus is also applicable to data that are not embedded in a metric space: only a distance matrix is needed. In contrast to this, prototype-based approaches refer to the data points directly when computing new cluster parameters (e.g. centers) and thus require a metric space.

### C. Initialization

An iterative update needs a starting point. Here we need an initial (fuzzy) assignment of the data points to the clusters. I tried two different schemes: in the first, all membership degrees are initialized to random values from the unit interval and then normalized for each data point (that is, they are divided by the sum of the membership degrees for the data point in order to achieve that this sum is 1 afterwards). Secondly, one may initialize all data points to the same value $\frac{1}{c}$ ($c$ is the number of clusters) and then seed the clusters by randomly choosing a data point for each of them, which is assigned crisply to it. Of course, in this case it is advisable to make sure that the data points used as seeds are updated last in the first epoch, so that the seeding does not get lost.

Although these two schemes appear to be considerably different, I did not observe much of a difference between them in my experiments: the results were basically the same. Hence I confine myself to the former method here.

### D. Fuzzy $k$-Nearest Neighbor

As an alternative to the update rule derived above, I considered an online update based on the classification rule for a fuzzy $k$-nearest neighbor classifier [17]:

$$u_{ij} = \frac{\sum_{l=1}^{k} u_{il} \, d_{jl}^{\frac{2}{1-w}}}{\sum_{l=1}^{k} d_{jl}^{\frac{2}{1-w}}}.$$

Here $d_{jl}$ is the distance between a data point $x_l$ from the training data set and a new data point $x_j$, which is to be classified. $u_{ij}$ and $u_{il}$ are the degrees of membership of the data points $x_j$ and $x_l$, respectively, to the $i$-th class. ([17] uses a heuristic formula to fuzzify the crisp class assignments that are usually to be found in training data sets. However, this formula is of no relevance here and therefore omitted.) The sums are assumed to run over the $k$ closest neighbors of the data point $x_j$ in the training data set, that is, they are assumed to refer to the $k$ smallest distances $d_{jl}$, $1 \leq l \leq k$.

At first sight it seems plausible to use this rule also as an update rule for fuzzy clustering, using either only the $k$ closest neighbors of a data set or even all other data points. However, my experiments revealed that this is not the case. Regardless of the number of neighbors that are considered (even if all data points are taken as neighbors), the fuzzy $k$-nearest neighbor rule equalizes the membership degrees (that is, in the end each data point has the same degree of membership to all clusters). Hence it is not usable as an update rule.

Nevertheless, one can extract from this approach the idea to constrain the neighborhood that is taken into account for the update of the membership of a data point. Such update schemes are studied in the next section (Section III).

Note that the fact that the fuzzy $k$-nearest neighbor rule is not feasible does not invalidate the method in [23]. That method relies on a completely different approach, which is actually prototype-based as it combines fuzzy $k$-nearest neighbor and (classical) fuzzy $c$-means. It is therefore not comparable with the algorithm presented in this paper.

### E. Experiments

For a basic evaluation of the algorithm I used a classic benchmark, namely the Iris data [9], with all four descriptive attributes (petal length and width and sepal length and width). For comparisons, the result of the standard fuzzy $c$-means algorithm with $c = 3$ and $w = 2$ is shown in Figure 1. It yields a clear division into three classes, even though some data points cannot be assigned unambiguously. (The degree of membership to a cluster is the higher, the darker the grey.)

The prototype-less algorithm, however, identifies only one of these classes (Iris Setosa, lower left), while the rest of the data points have roughly equal membership degrees to the remaining two clusters (see Figure 2). One may see this as a failure, but actually the division of the data points belonging to Iris Virginica and Iris Versicolor (upper right) into two clusters is rather arbitrary (if the class label is not known to the algorithm, as is the case here). It can rather be argued that there are actually only two clearly separated clusters and then the result of the prototype-less algorithm would simply indicate that the number of clusters was chosen inappropriately.

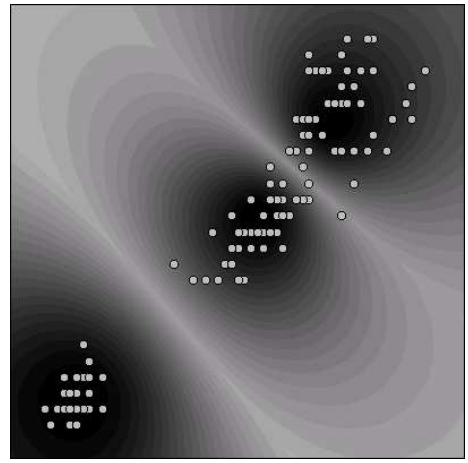

Fig. 1. Iris data clustered with fuzzy $c$-means ($w = 2$).

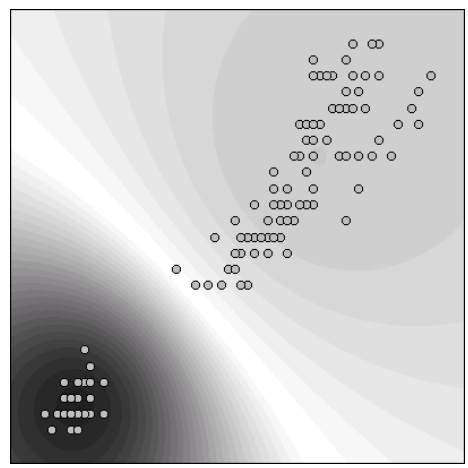

Fig. 2. Iris data clustered with prototype-less algorithm ($w = 2$).

On the other hand, the prototype-less algorithm can be made to yield a division into three clusters if the fuzzifier is reduced. As an example, Figure 3 shows the result for $w = \frac{5}{3}$. Even though the division of the Iris Virginica and Iris Versicolor data points is still less crisp as for the standard fuzzy $c$-means algorithms (since the grey does not get as dark in the upper right, thus still providing information that these clusters are not well separated), the cluster structure is almost the same.

It is also worth noting that the result becomes closer to the fuzzy $c$-means result if only the two most informative attributes (petal width and length) are used (see Figure 4). Nevertheless the cluster division stays less crisp than for fuzzy $c$-means, thus maintaining that the clusters are badly separated.

Generally, I found in my experiments (also with other data sets) that the prototype-less algorithm seems to require a slightly lower fuzzifier than prototype-based fuzzy clustering in order to yield comparable results. In this sense, prototype-less fuzzy clustering is "fuzzier" than prototype-based.

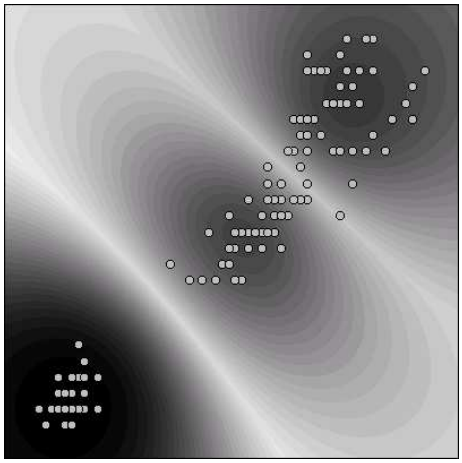

Fig. 3. Iris data with prototype-less algorithm ($w = \frac{5}{3}$).

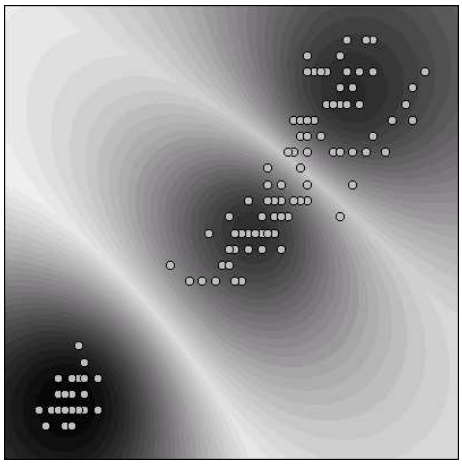

Fig. 4. Iris data with prototype-less algorithm (only petal length and width).

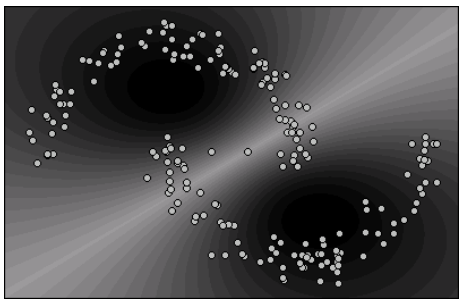

Fig. 5. Two half circles with fuzzy $c$-means.

## III. Constrained Neighborhood

In constrained neighborhood clustering only a subset of the data points are used for updating the membership degree of a given data point. I study three approaches: using only the closest neighbors (Section III-A), using only the farthest neighbors (Section III-B), and constructing a neighborhood graph from which new distances are derived (Section III-C).

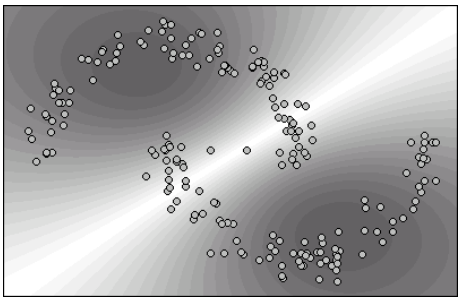

Fig. 6. Two half circles with clustered prototype-less algorithm.

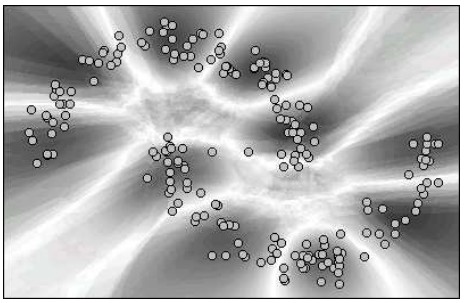

Fig. 7. Two half circles with prototype-less algorithm (50 closest neighbors).

For studying the effects of such constraints I use an artificial, two-dimensional data set that is shown in Figures 5 and 6 together with the results of fuzzy $c$-means and standard prototype-less clustering, respectively. It basically consists of two half circles of data points. However, in the middle there are two data points (artificial, intentional "noise") that destroy the clear separation of the two half circles. The goal is to identify the two half circles as one cluster each.

### A. Closest Neighbors

The most natural approach is to consider only the closest neighbors of a data point for updating its membership degrees. The intuition underlying this scheme is that the local neighborhood can remove the tendency of the basic algorithm to find compact clusters. (Due to the goal to minimize the objective function, all data points having high degrees of membership to the same cluster must be close together and thus form a compact cluster.) The hope is that then chain-like structures like those of the two half circles can be identified.

Unfortunately, this approach did not work quite as I hoped. Even though the tendency towards compact clusters is indeed removed, the effect is rather as shown in Figure 7, which depicts the result of prototype-less clustering with the update rule constrained to the 50 closest neighbors of each data point. Rather then combining the points in a chain into one cluster, each chain is split into smaller regions, which belong alternatingly to the two clusters. At second thought, this result is actually plausible: it is advantageous for the value of the (modified) objective function if neighboring points that are distant are assigned to different clusters.

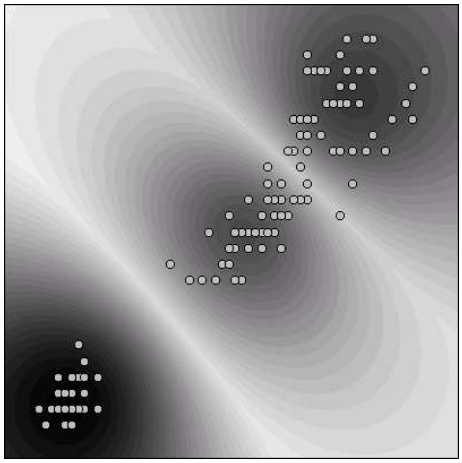

Fig. 8. Iris data with prototype-less algorithm (60 farthest neighbors).

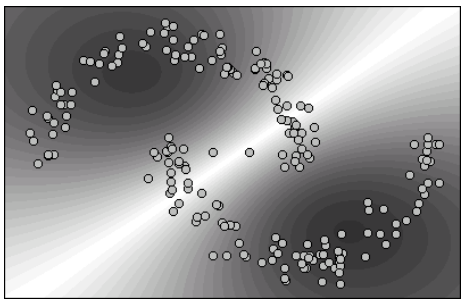

Fig. 9. Two half circles with prototype-less algorithm (60 farthest neighbors).

### B. Farthest Neighbors

A closer investigation of the reasons, why a closest neighbor approach fails, revealed that the terms of the objective function that refer to data points with a large distance are decisive. For minimizing the objective function it is vital that these points are assigned to different clusters, while points that are close together have only a comparatively small influence on the value of the objective function. This insight led to the idea to replace the closest neighbors with the *farthest* neighbors.

An update scheme that is based on the farthest neighbors of each data point actually works surprisingly well. Figure 8, for example, shows the result obtained for the Iris data if only the 60 farthest neighbors of each data point are used for updating the membership degrees of a data point. This figure is almost identical to Figure 3, which showed the result for the standard algorithm (considering all data points), but with a fuzzifier of $w = \frac{5}{3}$. This indicates that using only the farthest neighbors could have a similar effect as reducing the fuzzifier: the cluster division becomes harder. Other experiments confirmed this observation. For instance, Figure 9 shows the result for the data set of two half circles, with the update also based on the 60 farthest neighbors. Compared to Figure 6, in which the light grey shades indicated that the clusters are not well separated, the result is crisper (but, of course, the clusters still do not capture the chain-like structure of the data set).

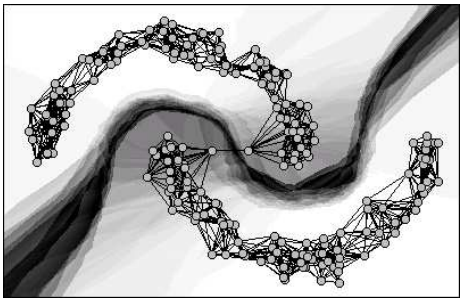

Fig. 10. Two half circles clustered with the prototype-less algorithm and a neighborhood graph with $k = 12$. (Note the inverted shading: the lighter the grey, the higher the membership degree.)

### C. Neighborhood Graph

As a final approach to use a constrained neighborhood, I tried a scheme based on a neighborhood graph. In such a graph each data point is connected by an edge to its $k$ closest neighbors, with a user-specified number $k$ of neighbors. For the data set with the two half circles, this neighborhood graph is shown for $k = 12$ in Figure 10 and for $k = 20$ in Figure 11: the black lines between the data points are the edges of the neighborhood graph. (Note the bridge between the two point sets in the middle of the figures, which is brought about by the two intentional "noise" data points.)

Once a neighborhood graph is constructed, a new distance matrix (which contains all pairwise distances between data points) is computed for the data set. It is initialized by setting the distance of all data point pairs that are connected by an edge to their Euclidean distance. All other entries (except the diagonal entries, of course, which are always 0) are set to infinity. Then the Floyd-Warshall algorithm [10], [22] for computing shortest paths between node pairs in a graph is executed. That is, the distance between two data points that are not closest neighbors of each other is set to the length of the shortest path in the neighborhood graph connecting these two points. In addition, it is advisable to replace any entries in the matrix that remained infinite by a user-specified value or a user-specified multiple of the largest finite distance in the matrix. With this all distances become finite, which avoids numerical problems. The new distance matrix, which describes path lengths, is then used to do the clustering.

The results of such a neighborhood graph approach with $k = 12$ and $k = 20$ is shown in Figures 10 and 11, respectively. Note that no user-specified maximum distance or multiplier is needed in this case, because both neighborhood graphs are connected. Note also that the shading is inverted in these figures (that is, the lighter the grey, the higher the membership degree) in order to make the neighborhood graph and the grey shades more easily discernable.

Especially for $k = 12$ the chain-like structure of the half circles is nicely recognized, even though the bridge in the middle leads to a fuzzy assignment of the data points close to this bridge. However, this is actually a desirable result, because the bridge destroys the clear separation of the clusters and this

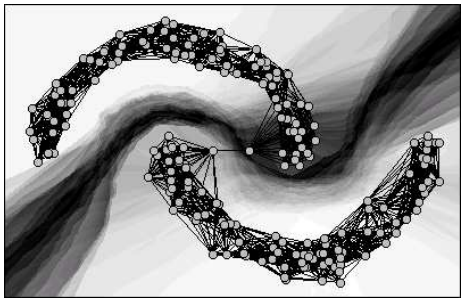

Fig. 11. Two half circles clustered with the prototype-less algorithm and a neighborhood graph with $k = 20$. (Note the inverted shading: the lighter the grey, the higher the membership degree.)

should be reflected in the result without spoiling it completely. For $k = 20$ the region of fuzzy assignments is, not surprisingly, bigger, but the fundamental structure is still captured well.

However, even though these results are very nice, I do not want to conceal that the success depends heavily on the symmetry of the structure. If one half circle is replaced by a quarter circle, the result is considerably worse: the correct division of the clusters is not recognized.

Nevertheless this example demonstrates the potential of the presented algorithm. A prototype-based algorithm would have considerable problems with the modified distances, since they do not allow for constructing cluster prototypes. One could amend this situation by using an approach based on medoids instead of means [16], [18], [20], but then the prototypes must be selected from the given data points, which may not contain appropriate cluster centers. A prototype-less algorithm, on the other hand, appears to be more natural and more flexible as it avoids the problem of constructing or selecting prototypes.

## IV. Conclusions

In this paper I presented a fuzzy clustering approach, which does not optimize a set of prototypes, but works solely with a fuzzy partition matrix. A core advantage of such a prototype-less scheme is that it only needs a distance matrix of the data objects, rather than the positions of the data points in a metric space. Neither is a procedure for computing prototypes needed. (It shares these advantages with (fuzzy) hierarchical agglomerative clustering [8], [16], [21], [6].) Therefore it can also be used in domains, in which the distances are non-metric, and thus has a wide potential application area.

The disadvantages of this approach are, of course, the higher computational complexity, which is $O(cn^2)$ for each update step (since all $n^2$ pairwise distances have to be evaluated for each of the $c$ clusters). However, this can be improved upon by using only the $k$ farthest neighbors as shown in Section III-B, which reduces the complexity to $O(cnk)$ for each step.

Future work includes to test the algorithm on pure distance data (that is, no embedding of data objects into a metric space) and to compare it to hierarchical agglomerative approaches. Furthermore graph structures other than the neighborhood graph may be worth to be investigated, like a minimum length spanning tree or a threshold graph, which contains only edges with lengths below a user-specified maximum.

### *Software*

The implementation of the algorithms described in this paper, which was also used for the experiments, is available at http://www.borgelt.net/ptless.html.

## References


[1] G.H. Ball and D.J. Hall. ISODATA — An Iterative Method of Multivariate Data Analysis and Pattern Classification. *IEEE Int. Comm. Conf. (Philadelphia, PA)*. IEEE Press, Piscataway, NJ, USA 1966

[2] J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, NY, USA 1981

[3] J.C. Bezdek and N. Pal. *Fuzzy Models for Pattern Recognition*. IEEE Press, New York, NY, USA 1992

[4] J.C. Bezdek, J. Keller, R. Krishnapuram, and N. Pal. *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Kluwer, Dordrecht, Netherlands 1999

[5] C. Borgelt. *Prototype-based Classification and Clustering*. Habilitation thesis, University of Magdeburg, Germany 2005

[6] Y. Dong and Y. Zhuang. Fuzzy Hierarchical Clustering Algorithm Facing Large Databases. *Proc. 5th World Congress on Intelligent Control and Automation (WCICA 2004, Hangzhou, China)*, 4282–4286. IEEE Press, Piscataway, NJ, USA 2004

[7] J.C. Dunn. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters. *Journal of Cybernetics* 3(3):32–57. American Society for Cybernetics, Washington, DC, USA 1973 Reprinted in [3], 82–101

[8] B.S. Everitt. *Cluster Analysis*. Heinemann, London, United Kingdom 1981

[9] R.A. Fisher. The Use of Multiple Measurements in Taxonomic Problems. *Annals of Eugenics* 7(2):179–188. Cambridge University Press, Cambridge, United Kingdom 1936

[10] R.W. Floyd. Algorithm 97: Shortest Path. *Communications of the ACM* 5(6):345. ACM Press, New York, NY, USA 1962

[11] I. Gath and A.B. Geva. Unsupervised Optimal Fuzzy Clustering. *IEEE on Trans. Pattern Analysis and Machine Intelligence (PAMI)* 11:773–781. IEEE Press, Piscataway, NJ, USA 1989. Reprinted in [3], 211–218

[12] E.E. Gustafson and W.C. Kessel. Fuzzy Clustering with a Fuzzy Covariance Matrix. *Proc. of the IEEE Conf. on Decision and Control (CDC 1979, San Diego, CA)*, 761–766. IEEE Press, Piscataway, NJ, USA 1979. Reprinted in [3], 117–122

[13] R.J. Hathaway and J.C. Bezdek. Optimization of Clustering Criteria by Reformulation. *IEEE Trans. on Fuzzy Systems* 3:241–145. IEEE Press, Piscataway, NJ, USA 1995

[14] J.A. Hartigan and M.A. Wong. A $k$-means Clustering Algorithm. *Applied Statistics* 28:100–108. Blackwell, Oxford, United Kingdom 1979

[15] F. Höppner, F. Klawonn, R. Kruse, and T. Runkler. *Fuzzy Cluster Analysis*. J. Wiley & Sons, Chichester, England 1999

[16] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. J. Wiley & Sons, New York, NY, USA 1990

[17] J.M. Keller, M.R. Gray, and J.A. Givens Jr. A Fuzzy k-nearest Neighbor Algorithm. *IEEE Trans. on Systems, Man, and Cybernetics* 15(4):580–584. IEEE Press, Piscataway, NJ, USA 1985

[18] R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi. Low-Complexity Fuzzy Relational Clustering Algorithms for Web Mining. *IEEE Trans. on Fuzzy Systems* 9(4):595–607. IEEE Press, Piscataway, NJ, USA 2001

[19] S. Lloyd. Least Squares Quantization in PCM. *IEEE Trans. on Information Theory* 28:129–137. IEEE Press, Piscataway, NJ, USA 1982

[20] T.A. Runkler. Relational Gustafson–Kessel Clustering with Medoids and Triangulation. *Proc. 14th IEEE Int. Conf. Fuzzy Systems (FUZZ-IEEE'06, Reno, NV)*, 73–78. IEEE Press, Piscataway, USA 2005

[21] P.-C. Wang and J.-J. Leou. New Fuzzy Hierarchical Clustering Algorithms. *J. Information Science and Engineering* 9(3):461–489. Inst. of Information Science, Taipei, Taiwan, Chine 1993

[22] S. Warshall. A Theorem on Boolean Matrices. *Journal of the ACM* 9(1):11–12. ACM Press, New York, NY, USA 1962

[23] N. Zahid, O. Abouelala, M. Limouri, and A. Essaid. Fuzzy Clustering based on K-nearest Neighbours Rule. *Fuzzy Sets and Systems* 120:239–247. Elsevier Science, Amsterdam, Netherlands 2001