An Extended Objective Function for Prototype-less Fuzzy Clustering

Christian Borgelt[1] and Rudolf Kruse[2]

[1] European Center for Soft Computing
Edificio Científico-Tecnológico, c/ Gonzalo Gutiérrez Quirós s/n, 33600 Mieres, Asturias, Spain
E-mail: christian.borgelt@softcomputing.es

[2] Dept. of Knowledge Processing and Language Engineering
Otto-von-Guericke-University of Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany
Email: rudolf.kruse@cs.uni-magdeburg.de

***Abstract*—While in standard fuzzy clustering one optimizes a set of prototypes, one for each cluster, we study fuzzy clustering without prototypes. We define an objective function, which only depends on the distances between data points and the membership degrees of the data points to the clusters, and derive an iterative membership update rule. The properties of the resulting algorithm are then examined, especially w.r.t. to an additional parameter of the objective function (compared to the one proposed in [7]) that can be seen as a more flexible alternative to the fuzzifier. Corresponding experimental results are reported that demonstrate the merits of our approach.**

***Keywords*—fuzzy clustering, prototype-less clustering, fuzzifier**

I. Introduction

Fuzzy clustering algorithms [9], [2], [3], [4], [16] 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 (data points), 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 [13], [12], [6]). $u_{ij} \in [0, 1]$ 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.

The most common fuzzy clustering algorithm is a straightforward generalization of classical k-means clustering [1], [15], [21] to fuzzy membership degrees: the fuzzy c-means algorithm [2], [3], [16] 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, [13], [12], [6]).

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, we investigate an approach that does not employ prototypes to describe the clusters, but uses only a partition matrix. It has the advantage that the data points need not be embedded in a metric space, but that it suffices to know a distance matrix. Following the standard paths for fuzzy clustering, we derive the basic algorithm in Section II. In Section III we present experimental results for the standard version of the algorithm (that is, without the additional parameter introduced here) and compare them in Section IV to experiments in which the additional parameter is used. It turns out that the additional parameter is a more flexible alternative to the fuzzifier and thus can be seen as being related to the approach presented in [19].

II. The Basic Algorithm

The basic idea of our approach 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 [14], 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 neighbors algorithms [18], from which one may also derive a candidate update rule for prototype-less fuzzy clustering. However, as discussed in [7], the latter does not lead to useful results, as it tends to equalize the membership degrees.

A. Objective Function

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

$$\begin{aligned} J(\mathbf{X},\mathbf{U}) &= \sum_{i=1}^{c}\sum_{j=1}^{n}\sum_{k=1}^{j-1}(u_{ij}^{w}u_{ik}^{w}+\alpha(u_{ij}^{w}+u_{ik}^{w}))d_{jk}^{2} \\ &= \sum_{i=1}^{c}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(\frac{1}{2}u_{ij}^{w}u_{ik}^{w}+\alpha u_{ij}^{w}\right)d_{jk}^{2}, \end{aligned}$$

which is to be minimized subject to the usual constraints

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

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.

Regardless of the value of the second parameter α, the value of this objective function is clearly the higher, the more distant data points are assigned to the same cluster. On the other hand, assigning data points that are close to each other to the same cluster is relatively harmless (that is, does not increase the function value much). Hence minimizing this function can be expected to yield an appropriate fuzzy partition matrix.

Choosing $\alpha = 0$ yields a fairly standard objective function, which was explored in detail in [7], in particular w.r.t. local neighborhood schemes. However, the goal of this paper is to investigate the influence of a non-vanishing α (this parameter is not present in [7]). As our investigation shows (see the experiments in Section IV), negative values are particularly interesting. They lead to a behavior similar to the approach in [19], which introduced an alternative to the fuzzifier in the prototype-based setting: The more negative α is, the harder are the data point assignments. A look at the second form of the objective function already makes this plausible, since the term containing α penalizes, for negative α values, an equal distribution to the different clusters, especially for data points that are far away from (most) other data points.

B. Update Procedure

Not surprisingly, the update rule for the membership degrees is derived along the same lines known from prototype-based fuzzy clustering, for example, fuzzy c-means. The constraint that the membership degrees of each data point must sum to 1 is incorporated into the objective function with the help of Lagrange multipliers, yielding the Lagrange function

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

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 $\forall a, 1\le a\le c: \forall b, 1\le b\le n:$

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

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

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

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

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

Consequently, the λ_j, $1\le j\le n$, are

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

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

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

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

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

Since this (non-linear) equation system is technically highly difficult to solve (due to the somewhat complicated interdependence of the membership degrees), we draw on the same trick that is exploited in prototype-based fuzzy clustering: alternating optimization. That is, we 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 we confine ourselves 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: sum 1). 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.

C. Initialization

An iterative update needs a starting point. Here we need an initial (fuzzy) assignment of the data points to the clusters. We 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, we did not observe much of a difference between them in our experiments: the results were basically the same. Hence we confine ourselves to the former method here.

III. Behavior for $\alpha = 0$

For a basic evaluation of the algorithm we used two classic benchmarks, namely the Iris data [11] (150 data points), with all four descriptive attributes (petal length and width and sepal length and width), and the Wine data [5] (178 data points), using only the descriptive attributes 7, 10, and 13, which are the most informative w.r.t. the class structure. The class attribute was, of course, not used as an input. For both data sets all used attributes were normalized to mean 0 and standard deviation 1 in order to rule out scaling effects.

We consider first the results for the Iris data. 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 cluster, even though some data points cannot be assigned unambiguously. (The degree of membership to a cluster is the higher, the darker the grey.) These clusters correspond reasonably well to the classes: if the clustering result is used as a classifier, 26 data points are misclassified.

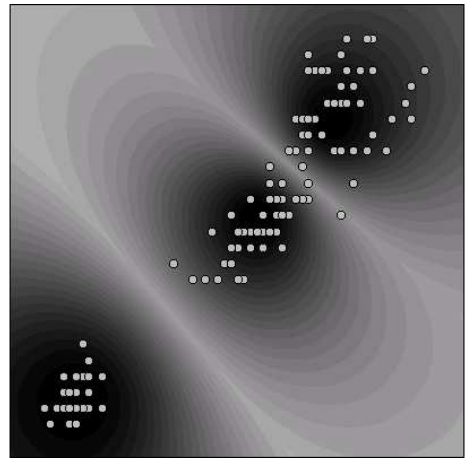

Fig. 1. Iris data clustered with fuzzy c-means ($w = 2$).

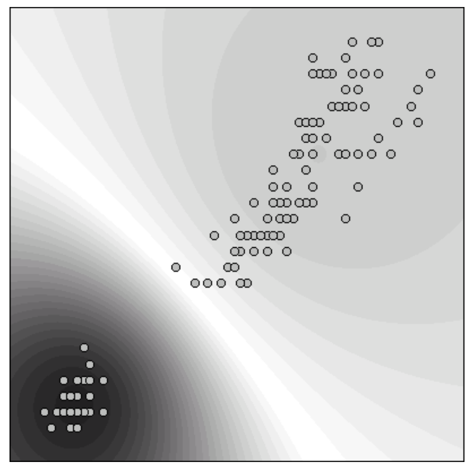

Fig. 2. Iris data clustered with prototype-less algorithm ($w = 2$, $\alpha = 0$).

The prototype-less algorithm, however, identifies only one of these classes (Iris Setosa, lower left), while the rest of the data points have almost 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 is not known to the algorithm, as it 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.

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 than for the standard fuzzy c-means

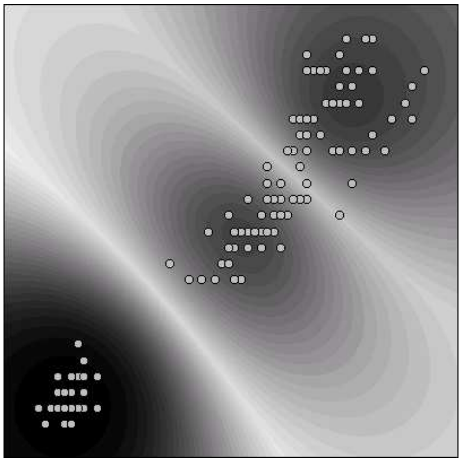

Fig. 3. Iris data with prototype-less algorithm ($w = \frac{5}{3}$, $\alpha = 0$).

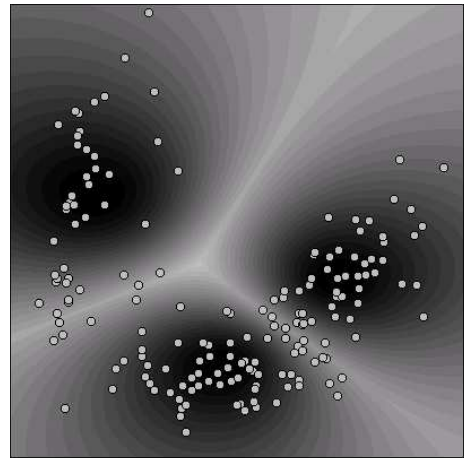

Fig. 5. Wine data clustered with fuzzy c-means ($w = 2$).

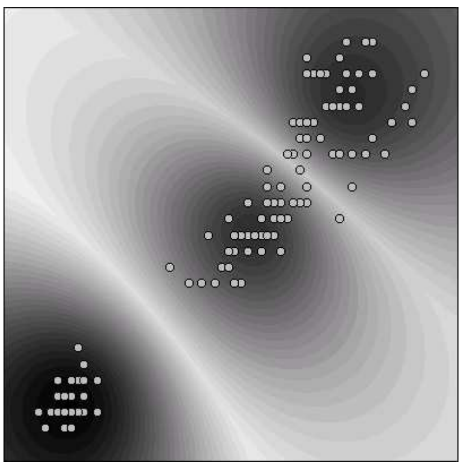

Fig. 4. Iris data with prototype-less algorithm (only petal length and width).

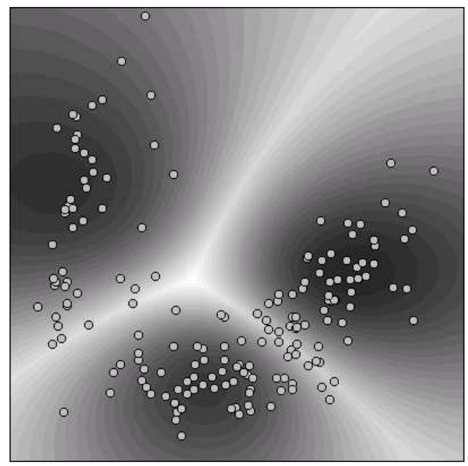

Fig. 6. Iris data with prototype-less algorithm ($w = \frac{5}{3}$, $\alpha = 0$).

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.

Similar observations can be made on the Wine data. Figure 5 shows, for comparison purposes, the result that is obtained with the standard fuzzy c-means algorithm ($w = 2$) with attribute 7 on the horizontal and attribute 10 on the vertical axis. It shows a fairly clear division into three clusters, which—like for the Iris data—correspond fairly well with the classes of this data set: if the clustering result is used as a classifier only 15 of the 178 data points are misclassified.

However, from Figure 5 one guess (and a 3-dimensional view on the data set confirms this) that the clusters are not well separated. Hence one may already suspect, judging from the result obtained on the Iris data, that the prototype-less algorithm may have trouble to find a similar structure with the default setting. And indeed, with a fuzzifier $w = 2$ the membership degrees are completely equalized. However, as it was the case for the Iris data, lowering the classifier amends the problem: with $w = \frac{3}{5}$ the result shown in Figure 6 is obtained, which is fairly similar to the fuzzy c-means result. Nevertheless the assignment is still less crisp as in the fuzzy c-means result, reflecting the overlapping classes. In order to fully reach the crispness of the fuzzy c-means result, an even lower fuzzifier would be needed. Generally, we found in our experiments that the prototype-less algorithm seems to require a lower fuzzifier than prototype-based fuzzy clustering in order to yield comparable results. In this sense, prototype-less fuzzy clustering is "fuzzier" than its prototype-based counterpart.

IV. Influence of the Parameter α

In order to illustrate the results that can be obtained if the parameter α is not set to 0, Figures 7 and 8 show the result for slightly negative α ($\alpha = -0.02$ and $\alpha = -0.05$, respectively) for the Iris data. Note that these results are both

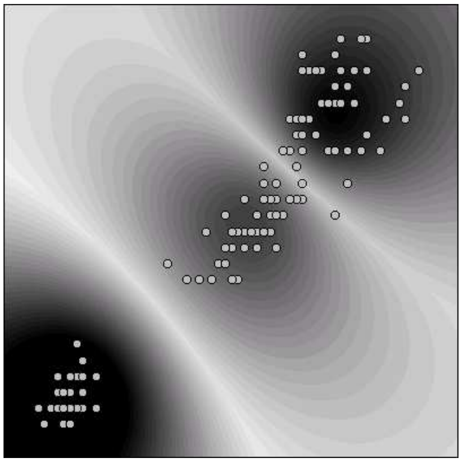

Fig. 7. Iris data with prototype-less algorithm ($w = 2$, $\alpha = -0.02$).

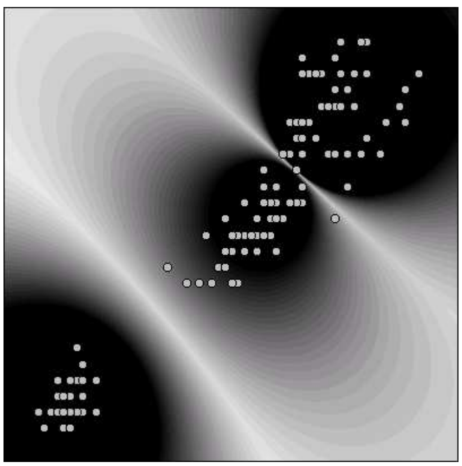

Fig. 8. Iris data with prototype-less algorithm ($w = 2$, $\alpha = -0.05$).

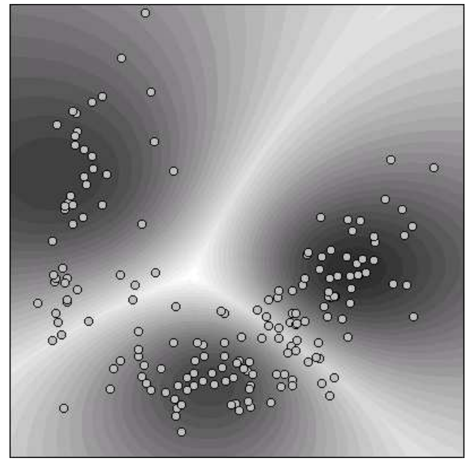

Fig. 9. Wine data with prototype-less algorithm ($w = 2$, $\alpha = -0.02$).

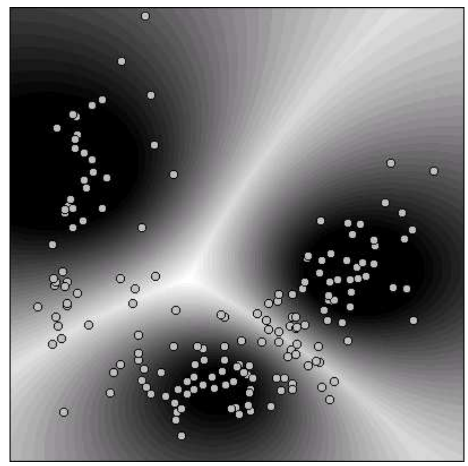

Fig. 10. Wine data with prototype-less algorithm ($w = 2$, $\alpha = -0.05$).

obtained with the standard fuzzifier $w = 2$. (This has the advantage that the computations can be carried out much faster than the one of Figure 3, since fractional exponents cause considerable computational costs.) While the result for $\alpha = -0.02$ (Figure 7) lies between the prototype-less result for $w = \frac{5}{3}$ and the fuzzy c-means result w.r.t. the crispness of the assignments, the result for $\alpha = -0.05$ (Figure 7) is even crisper than the fuzzy clustering result. Actually the membership degrees of data points that clearly belong to one of the classes become 1 for the correct class, while all other classes are assigned a membership degree of zero. Only at the boundaries of the clusters the membership degrees become fuzzy, thus nicely modeling the overlapping cluster structure.

Note also that the boundaries of the clusters (light grey areas) are slightly shifted compared to the fuzzy c-means result: they are minimally farther to the upper right. However, the class structure is still recognized: if the clustering result is used as a classifier, 26 data points are misclassified.

Similar observations can be made on the Wine data, for which Figures 9 and 10 show the results for $\alpha = -0.02$ and $\alpha = -0.05$, respectively. The former almost coincides with the result obtained with $w = \frac{5}{3}$ and $\alpha = 0$ (Figure 6), while the latter is again (as for the Iris data) even crisper than the fuzzy c-means result (Figure 5). Note also that the cluster boundaries are slightly different from the fuzzy c-means result, which is particularly clear for the two clusters on the left.

These results demonstrate that the effect of the parameter α is similar to the one that can be achieved with the approach of [19] for the prototype-based case, namely that the parameter α can be used as a (more efficient) alternative to the fuzzifier (needing only a simple addition and not the computation of fractional powers). The lower the value of α (or the higher its absolute value, since negative values produce the effect), the crisper the assignment of the data points.

In addition, our experiments indicate that this approach (almost) eliminates the drawback of fuzzy clustering (as discussed in [19]) that all data points have a non-vanishing membership to all clusters: with a sufficiently low α (sufficiently large negative value), data points that lie in the middle of the formed clusters become (almost) crisply assigned.

V. Conclusions

In this paper we 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 [10], [17], [23], [8].) 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), and that it produces softer assignments of the data points if the same fuzzifier is used as for prototype-based fuzzy clustering. However, we eliminated the latter disadvantage by extending the objective function in the way suggested in Section II-A, namely by adding a parameter α that penalizes an assignment of a data point to several clusters. With this additional parameter similar effects can be achieved as by changing the fuzzifier, thus making it a more efficient alternative that does not require fractional exponents.

Future work includes to test the algorithm on pure distance data (that is, no embedding of data objects into a metric space, only a distance matrix is given) and to compare it to hierarchical agglomerative clustering approaches. Furthermore it may be worthwhile to investigate other neighborhood schemes than those discussed in [7], since they may provide a simple and effective way to lower the computational costs.

Software

An 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.L. Blake and C.J. Merz. *UCI Repository of Machine Learning Databases*. University of California, Irvine, CA, USA 1998 http://www.ics.uci.edu/~mlearn/MLRepository.html

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

[7] C. Borgelt. Prototype-less Fuzzy Clustering. *Proc. 14th IEEE Int. Conference on Fuzzy Systems (FUZZ-IEEE'07, London, UK)*. IEEE Press, Piscataway, NJ, USA 2007

[8] 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

[9] 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

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

[11] 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

[12] 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

[13] 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

[14] 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

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

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

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

[18] 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

[19] F. Klawonn and F. Höppner. What is Fuzzy about Fuzzy Clustering? Understanding and Improving the Concept of the Fuzzifier. *Proc. 5th Int. Symposium on Intelligent Data Analysis (IDA 2003, Berlin, Germany)*, 254–264. Springer-Verlag, Berlin, Germany 2003

[20] 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

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

[22] 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

[23] 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

[24] 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