Clustering with Size Constraints

Frank Höppner[1] and Frank Klawonn[2]

[1] Department of Economics
University of Applied Sciences Braunschweig /Wolfenbüttel
Robert Koch Platz 10-14
38440 Wolfsburg, Germany

[2] Department of Computer Science
University of Applied Sciences Braunschweig /Wolfenbüttel
Salzdahlumer Str. 46/48
38302 Wolfenbüttel, Germany
{f.hoeppner,f.klawonn}@fh-wolfenbuettel.de

Abstract. We consider the problem of partitioning a data set of n data objects into c homogeneous subsets or clusters (that is, data objects in the same subset should be similar to each other) with constraints on the number of data per cluster. The proposed techniques can be used for various purposes. If a set of items, jobs or customers has to be distributed among a limited number of resources and the workload for each resource shall be balanced, clusters of approximately the same size would be needed. If the resources have different capacities, then clusters of the corresponding sizes need to be found. We also extend our approach to avoid extremely small or large clusters in standard cluster analysis. Another extension offers a measure for comparing different prototype-based clustring results.

1 Introduction

Cluster analysis is a widely used technique that seeks for groups in data. The result of such an analysis is a set of groups or clusters where data in the same group are similar (homogeneous) and data from distinct groups are different (heterogeneous) [1]. In this paper, we consider variations of the clustering problem, namely the problem of subdividing a set X of n objects into c homogeneous groups with constraints on some or all clusters concerning the number of data they can contain. In contrast to the standard clustering problem, we abandon the heterogeneity between groups partly and introduce constraints on the number of data per cluster.

Applications of *uniform clustering* where all clusters should contain approximately the same number of data include for instance: (a) The distribution of n students into c groups of equal strength to obtain fair class sizes and with homogeneous abilities and skills to allow for teaching methods tailored to the specific needs of each group. (b) The distribution of n jobs to c machines or workers such that every machine has an identical workload and as similar jobs as possible to reduce the configuration time. (c) The placement of c sites such that goods from n locations can be transported to the c sites, while the total covered distance is minimized and queuing at the sites is avoided, that is, approximately the same number of goods should arrive at each site.

More general applciations include problems where the resources, for instance the lecturing halls in the above example (a), the machines in example (b) and the size of the sites in example (c), differ in capacity, so that the clusters still should have fixed size corresponding to the capcities, but not necessarily the same size for all clusters.

Due to the similarity of our problem with traditional clustering problems, we are going to modify an existing clustering algorithm, which will be reviewed in section 2. This objective function-based clustering algorithm – a variant of k-means – transforms the discrete, combinatorial problem into a continuous one, such that numerical problem solving methods can be applied. As proposed in [5], we modify the objective function such that the equi-sized clusters are considered in section 3 and discuss the results in section 4. Section 5 extends the approach to clusters with different size constraints. In section 6 our approach is exploited to define a similarity measure to compare different clustering results.

2 The FCM Algorithm

The fuzzy c-means (FCM) clustering algorithm partitions a data set $X := \{x_1, ..., x_n\} \subset \mathbf{R}^d$ into c clusters. A cluster is represented by a prototype $p_i \in \mathbf{R}^d$, $1 \le i \le c$. The data-prototype relation is not binary, but a membership degree $u_{ij} \in [0, 1]$ indicates the degree of belongingness of data object x_j to prototype p_i or cluster number i. All membership degrees form a membership matrix $U \in \mathbf{R}^{c \times n}$. We can interpret the membership degrees as “probabilistic memberships”, since we require

$$\forall 1 \le j \le n : \qquad \sum_{i=1}^{c} u_{ij} = 1 \, . \tag{1}$$

The clustering process is carried out by minimizing the objective function

$$J_m = \sum_{j=1}^{n} \sum_{i=1}^{c} u_{ij}^m d_{ij} \quad \text{with} \quad d_{ij} = \|x_j - p_i\|^2 \, . \tag{2}$$

under constraint (1). If the Euclidean distance between datum x_j and prototype p_i is high, J_m is minimized by choosing a low membership degree near 0. If the distance is small, the membership degree approaches 1. J_m is effectively minimized by alternating optimisation, that is, we alternatingly minimize (2) with respect to the prototypes (assuming memberships to be constant) and then with respect to the membership degrees (assuming prototypes to be constant). In both minimization steps, we obtain closed form solutions, for the prototypes:

$$\forall 1 \le i \le c : \qquad p_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m} \tag{3}$$

and for the membership degrees:

$$u_{ij} = \begin{cases} \frac{1}{\sum_{l=1}^{c} \left(\frac{\|x_j - p_i\|^2}{\|x_j - p_l\|^2} \right)^{\frac{1}{m-1}}} & \text{in case } I_j = \emptyset \\ \frac{1}{|I_j|} & \text{in case } I_j \neq \emptyset, i \in I_j \\ 0 & \text{in case } I_j \neq \emptyset, i \notin I_j \end{cases} \tag{4}$$

where $I_j = \{k \in \mathbf{N}_{\leq c} \,|\, x_j = p_k\}$. The FCM algorithm is depicted in Fig. 1. For a more detailed discussion of FCM and examples we refer to the literature, e.g. [2,3].

```
choose $m > 1$ (typically $m = 2$)
choose termination threshold $\varepsilon > 0$
initialize prototypes $p_i$ (randomly)
repeat
  update memberships using (4)
  update prototypes using (3)
until change in memberships drops below $\varepsilon$
```

Fig. 1. The FCM algorithm.

3 Equi-sized Clusters

It is often said that the k-means (as well as the FCM) algorithm seeks for clusters of approximately the same size, but this is only true if the data density is uniform. As soon as the data density varies, a single prototype may very well cover a high-density cluster and thereby gains many more data objects than the other clusters. This leads to large differences in the size of the clusters. Examples for this phenomenon are shown in Fig. 2 for two data sets: On the left image, there is a very high density cluster in the top left corner, on the right image, the density decreases from left to right, so the rightmost cluster has only some data.

The idea of our modification is to include an additional constraint in the objective function (2) that forces the clusters to cover the same number of data objects. The size of cluster i (number of data objects) corresponds to the sum of the membership values $\sum_{j=1}^{n} u_{ij}$. In fact, since we have continuous membership degrees we may require

$$\sum_{j=1}^{n} u_{ij} = \frac{n}{c} \tag{5}$$

for all $i \in \{1, \ldots, c\}$ even if n is not a multitude of c. This additional constraint (5) is – together with the constraint (1) – integrated into the objective function (2) via Lagrange multipliers. We then solve for the cluster prototypes and Lagrange multipliers by setting the partial derivatives to zero. This turns out to be

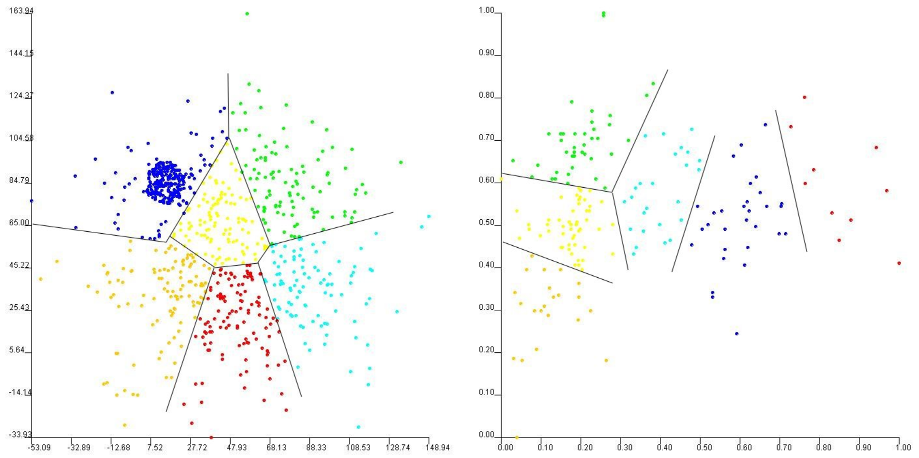


Fig. 2. Results of the FCM algorithm on two data sets.

a difficult problem for the general case of an arbitrary value of m, therefore we restrict ourselves to the case of $m = 2$, which is the most frequently used value of m in FCM. Given our Lagrange function

$$L \;=\; \sum_{i=1}^{c}\sum_{j=1}^{n} u_{ij}^2 d_{ij} \;+\; \sum_{j=1}^{n}\alpha_j\left(1-\sum_{i=1}^{c}u_{ij}\right) \;+\; \sum_{i=1}^{c}\beta_i\left(\frac{n}{c}-\sum_{j=1}^{n}u_{ij}\right) \tag{6}$$

we obtain as partial derivatives

$$\frac{\partial L}{\partial u_{ij}} \;=\; 2u_{ij}d_{ij} - \alpha_j - \beta_i \;=\; 0 \tag{7}$$

These equations, together with the constraints (1) and (5), lead to the following system of $(c \cdot n + c + n)$ linear equations for the variable u_{ij}, α_i and β_j ($i \in \{1,\ldots,c\}$, $j \in \{1,\ldots,n\}$). Empty entries indicate the value zero, RHS stands for the right hand side of the equation.

	$u_{1,1}$	$\dots$	$u_{1,n}$	$\dots$	$u_{c,1}$	$\dots$	$u_{c,n}$	α_1	$\dots$	α_n	β_1	$\dots$	β_c	RHS
$\frac{\partial L}{\partial u_{1,1}}$	$2d_{1,1}$							-1			-1			
$\vdots$		$\ddots$							$\ddots$		$\vdots$			
$\frac{\partial L}{\partial u_{1,n}}$			$2d_{1,n}$							-1	-1			
$\vdots$				$\ddots$					$\ddots$			$\ddots$		
$\frac{\partial L}{\partial u_{c,1}}$					$2d_{c,1}$			-1					-1	
$\vdots$						$\ddots$			$\ddots$				$\vdots$	
$\frac{\partial L}{\partial u_{c,n}}$							$2d_{c,n}$			-1			-1	
$\sum u_{i,1}$	1			$\dots$	1									1
$\vdots$		$\ddots$				$\ddots$								$\vdots$
$\sum u_{i,n}$			1				1							1
$\sum u_{1,j}$	1	$\dots$	1											n/c
$\vdots$				$\ddots$										$\vdots$
$\sum u_{c,j}$					1	$\dots$	1							n/c

In principle, this system of linear equations could be solved by a suitable numerical algorithm. Even for small data sets with 200 data objects and 5 clusters, this would mean that we have to solve a system of 1205 equations in each iteration step of the clustering algorithm, which is not acceptable in terms of computational costs. However, it is possible to solve this system of equations in a more efficient way.

When multiplying the equations for $u_{k1}, \dots, u_{kn}$ by $\frac{1}{2d_{k1}}, \dots, \frac{1}{2d_{kn}}$, respectively, and then subtracting the resulting equations from the equation for $\sum_j u_{kj}$, we obtain

$$\sum_{j=1}^{n} \frac{\alpha_j}{2d_{kj}} \;+\; \beta_k \sum_{j=1}^{n} \frac{1}{2d_{kj}} \;=\; \frac{n}{c}. \tag{8}$$

From equation (7), we obtain

$$u_{ij} \;=\; \frac{\alpha_j + \beta_i}{2d_{ij}}. \tag{9}$$

Taking constraint (1) into account, yields

$$1 \;=\; \sum_{i=1}^{c} u_{ij} \;=\; \frac{\alpha_j}{2} \sum_{i=1}^{c} \frac{1}{d_{ij}} + \frac{1}{2} \sum_{i=1}^{c} \frac{\beta_i}{d_{ij}},$$

leading to

$$\alpha_j \;=\; \frac{2 - \sum_{i=1}^{c} \frac{\beta_i}{d_{ij}}}{\sum_{i=1}^{c} \frac{1}{d_{ij}}}. \tag{10}$$

Inserting (10) into (8), we obtain

$$\sum_{j=1}^{n} \frac{2 - \sum_{i=1}^{c} \frac{\beta_i}{d_{ij}}}{2 \sum_{i=1}^{c} \frac{d_{kj}}{d_{ij}}} + \beta_k \sum_{j=1}^{n} \frac{1}{2 d_{kj}} = \frac{n}{c}$$

and thus

$$-\sum_{j=1}^{n} \frac{\sum_{i=1}^{c} \frac{\beta_i}{d_{ij}}}{2 \sum_{i=1}^{c} \frac{d_{kj}}{d_{ij}}} + \beta_k \sum_{j=1}^{n} \frac{1}{2 d_{kj}} = \frac{n}{c} - \sum_{j=1}^{n} \frac{1}{\sum_{i=1}^{c} \frac{d_{kj}}{d_{ij}}}. \qquad (11)$$

This induces a system of c linear equations for the β_k with coefficients

$$a_{k\ell} = \begin{cases} -\sum_{j=1}^{n} \frac{\frac{1}{d_{\ell j}}}{2 \sum_{i=1}^{c} \frac{d_{kj}}{d_{ij}}} & \text{if } k \neq \ell \\ -\sum_{j=1}^{n} \frac{\frac{1}{d_{\ell j}}}{2 \sum_{i=1}^{c} \frac{d_{kj}}{d_{ij}}} + \sum_{j=1}^{n} \frac{1}{2 d_{kj}} & \text{if } k = \ell. \end{cases} \qquad (12)$$

This system of linear equations can be solved by a suitable numerical algorithm. The computation time is acceptable, since the number of equations is equal to the number of clusters and therefore independent of the number of data. Once the β_i have been determined, we can compute the α_j using equation (10) and finally obtain the membership degrees based on equation (9). After all, we arrive at the clustering algorithm depicted in Fig. 3.

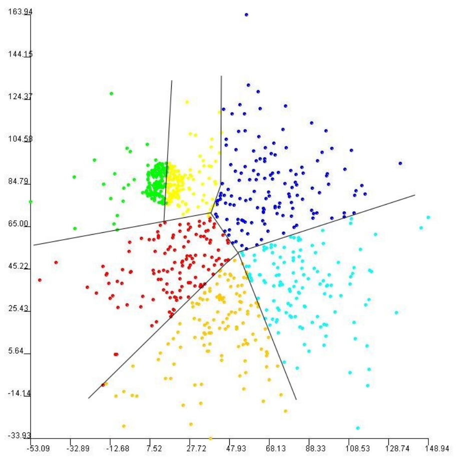
```
choose termination threshold $\varepsilon > 0$
initialise prototypes $p_i$
repeat
  solve linear equation system (11) for $\beta$
  using $\beta$, calculate $\alpha$ using (10), update memberships using (9)
  update prototypes using (3)
until change in memberships drops below $\varepsilon$
```

Fig. 3. The proposed algorithm.

Note that the boundedness of the membership degrees $u_{ij} \in [0, 1]$ represents an additional constraint on the objective function of FCM as well as the objective function of our new algorithm. In the original FCM, however, it was not necessary to consider it explicitly, because one can easily see from the resulting membership degrees (4) that this condition is satisfied. It is not possible to conclude this boundedness for the new membership degrees (9). It is clear, however, that the influence of negative memberships will be rather small: Since the objective function (2) and (6) uses only positive weights u_{ij}^2, large negative values cannot help in the minimization. We will comment on this in the following section.

4 Examples for Uniform Clustering

To illustrate the impact of our modified objective function, we show the results of the new algorithm for the data sets shown in Fig. 2, where the standard FCM algorithm yielded a result with high variation in the cluster size. The results are shown in the left images of Figs. 4 and 6. By comparison to Fig. 2 we see, that the high-density cluster has been split into two clusters (Fig. 4) and that the data on the left of Fig. 6 is now distributed among four rather than three clusters, such that the rightmost cluster gains more data. As expected, the sum of membership degrees for each individual cluster equals $\frac{n}{c}$.



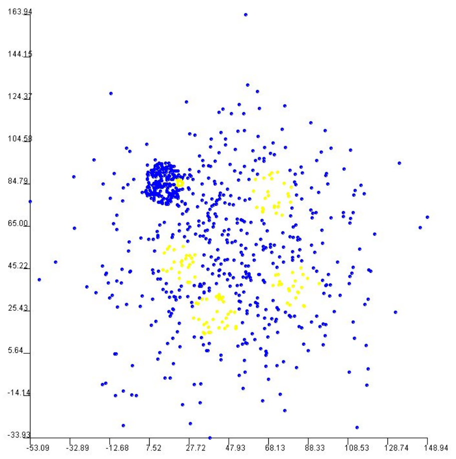


Fig. 4. Results of the new algorithm.

Regarding the boundedness of the membership degrees u_{ij} it turned out that they actually take negative values. This is, of course, an undesired effect, because then the interpretation of $\sum_{j=1}^{n} u_{ij}$ as the *size* or number of data objects is not quite correct. As conjectured in the previous section, it turned out on closer examination that the total sum of negative weights is rather small. In both data sets, the sum of all negative membership degrees was below 0.5% of the total data set size n.

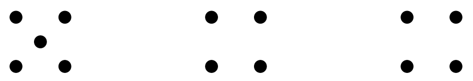


Fig. 5. A 'difficult' example data set.

We want to illustrate the kind of situation in which negative membership degrees occur with the help of the data set shown in Fig. 5. Consider the data set is partitioned into three clusters. Since the leftmost cluster has an additional data object x in the middle, it is not obvious how to distribute the data among all clusters in equal shares.

Regarding the minimization of the sum of weighted distances, it would be optimal to assign high membership degrees to all five data objects. This would, however, violate the constraint that all clusters must share the same size. To get membership degrees as high as possible, the membership of all other data objects (middle and right cluster) to this cluster are chosen slightly negative. Since the sum of all membership values is constrained to be one, negative values for the middle and right data allow us to have slightly higher degrees for the leftmost data. On the other hand, having a negative membership degrees for some data object x on the right forces us to increase other membership degrees of x (to guarantee a sum of 1). This is possible almost at no cost, if x is close to the centre of another cluster, because then we have a small distance value and increasing the membership degree to this cluster does no harm in the minimization of (2). (For a detailed discussion of the general influence of the membership weight u_{ij}^m see [4].)

To summarise: In a situation where an equi-sized partition is difficult to obtain while minimizing at the same time the sum of weighted distances (2), the cluster with too many data objects 'borrows' some membership from data near the centres of the other clusters. Figures 4 and 6 show this effect for the two example data sets. The data for which negative membership values occur are shown in a lighter shading. These data objects are all close to the respective cluster prototype. And there is always one cluster without any negative membership degrees, which corresponds to the rightmost cluster in our example data set in Fig. 5.

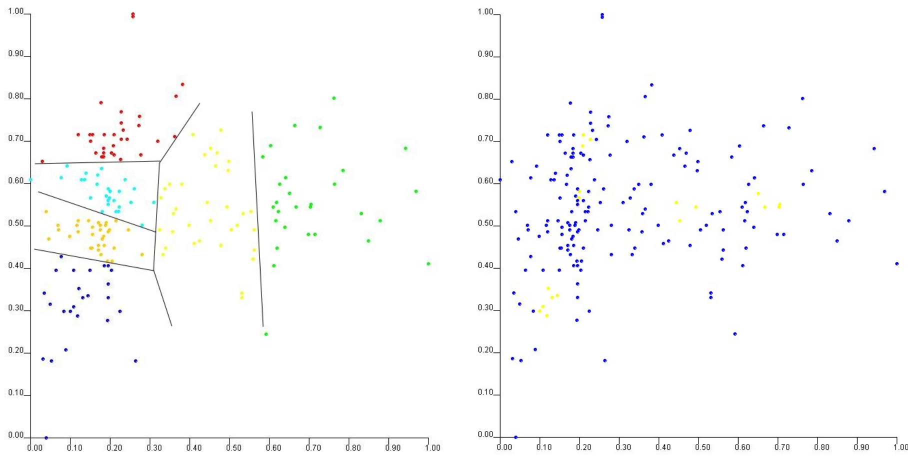


Fig. 6. Results of the new algorithm.

In all our experiments, the side effects of this trade off between minimizing (2) and satisfying (5) were quite small, so we do not consider this as a major drawback of our approach. We can even make use of this information: By analysing which cluster has no negative membership degrees at all, we can find out which cluster tends to be 'too big'. When breaking ties in the final assignment of data to clusters, it should be this cluster that gets more data objects than the other.

5 Limiting the Size of Single Clusters only

Instead of forcing all clusters to have the same size of $\frac{n}{c}$, it is also possible to have different requirements for each cluster individually. We introduce size parameters s_i and replace (5) by

$$\sum_{j=1}^{n} u_{ij} = s_i \tag{13}$$

The derivation of the linear equation systems still holds, we just have to replace the fraction $\frac{n}{c}$ in (11) by s_i.

For some of the clusters, we may have no constraint on their size. In such a case the condition (13) needs not to be guaranteed by means of a Lagrange multiplier as in equation (6). So we consider a linear equation system with a reduced number of Lagrange multipliers (or force the respective β_i to be zero, such that it has no influence on the remaining equations).

Two examples are given to illustrate the performance of the algorithm on two low-dimensional data sets. Figure 7 shows a set consisting of basically two clusters (50 data points each) and three outliers in the lower left corner. When clustered with $c = 3$ clusters, FCM assigns the three outliers to a single cluster (cf. figure 7, left). Then, we force two clusters to have a size of $s_i = 34$. This forces the outlier cluster to move closer to one of the larger clusters (cf. figure 7, right).

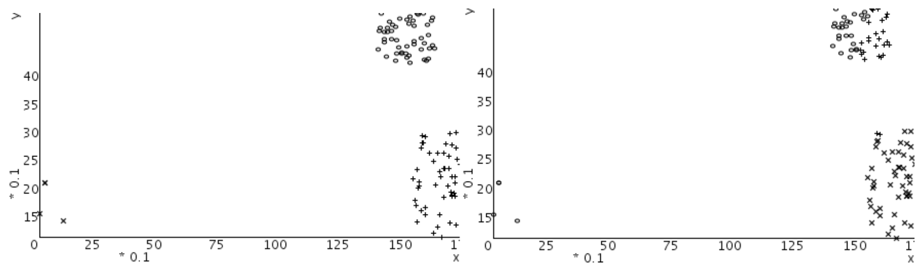


Fig. 7. Example 1. Left: Result of FCM. Right: Result of Size-Constrained FCM.

The second example in figure 8 shows the case of two long-stretched clusters of 100 data points each. Using 4 clusters FCM covers each of the two long-stretched clusters by 2 prototypes (figure 8, left). Now we force one of the bottom clusters to have exactly a size of 100 data points. To achieve this, the other prototype (supported by 50 data points) has to be repelled completely from the clusters and the data from the second cluster must now be shared among three prototypes. This is exactly the case in figure 8 (right). The figure also shows how the position of the prototypes evolves over the iterations.

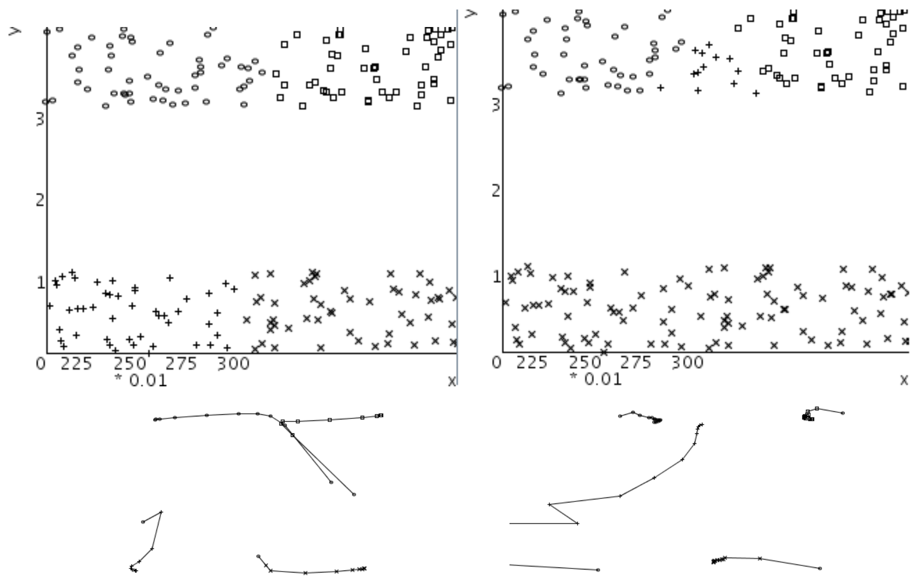


Fig. 8. Example 2: Left: Result of FCM. Right: Result of Size-Constrained FCM. (top: resulting partition, bottom: trace of prototypes to their final location)

It is also possible to apply our technique to avoid extremely small or large clusters in a cluster analysis. We simply formulate a fuzzy clustering algorithm with upper and lower bounds on the size of a cluster as shown in figure 9. We first run the original FCM and then impose constraints on cluster sizes in case a cluster's size is below the lower or above the upper bound on cluster sizes.

It should be mentioned, that the *size s_i of a cluster* has been interpreted in a fuzzy sense: It *may* happen that a prototype achieves quite a large size $\sum_{j=1}^{n} u_{i,j}$ although only in very few cases this prototype receives the highest membership degree. From our experiences such situations are more likely to occur, if the partition is very stable and the number of clusters is chosen wrongly.

6 A Distance Measure for Sets of Prototypes

In this section we consider the problem of defining a distance measure between two sets of prototypes. Figure 10 illustrates the problem by an example: Proto-

```
choose minimal and maximal cluster size $s_{\min}$ and $s_{\max}$.
initialize size constraint vector $s_i = (0, .., 0)$ (no size constraint for any cluster)
run traditional Fuzzy c-Means (cf. figure 1)
while (at least one cluster sizes $a_i$ does not satisfy $s_{\min} \le a_i \le s_{\max}$.
  for all clusters $i$ with $s_i = 0$
    if $a_i < s_{\min}$, set $s_i = s_{\min}$
    if $a_i > s_{\max}$, set $s_i = s_{\max}$
  end
  run algorithm in figure 3 using current constraints $s_i$ (with $s_i \neq 0$)
end
```

Fig. 9. Clustering with constraints on cluster size.

type set A consists of three prototypes $\{a, b, c\}$, set B consists of four prototypes $\{d, e, f, g\}$. We need a measure of divergence between the two sets. Intuitively we have to match each of the prototypes in set A to one (or more) prototype(s) in set B and then account for the distances between the associated pairs of prototypes. An intuitive matching is illustrated by dotted lines in figure 10.

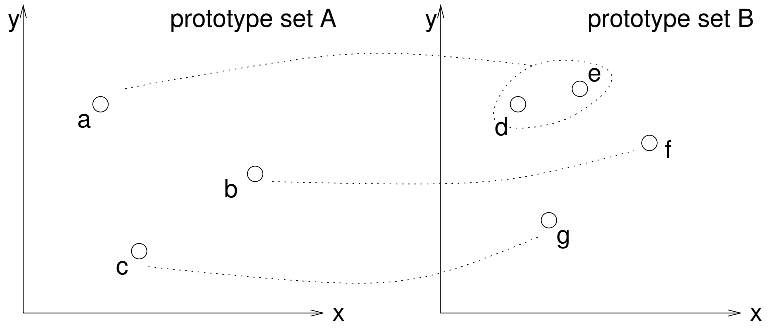


Fig. 10. Constraints on the membership matrix.

This problem is quite similar to clustering a small data set with $c \approx n$. If we interpret the membership degree $u_{i,j}$ as the strength of the relationship between prototype i (set A) and prototype j (set B), cf. dotted lines in figure 10, the sum of weighted distances (2) becomes a good indicator for the distance between the sets.

The membership constraints (1) and (5) assure that every data object / prototype has the same overall weight in the assignment, that is, every data object (or prototype, resp.) must be assigned to a prototype (or data objects, resp.) with equal shares. This is illustrated on the left side of figure 11. Since clusters may have different sizes, forcing each cluster to share the same number of data objects is a quite unrealistic requirement, therefore we have introduced cluster size s_i in the previous section. Since we want to measure the distance between two sets of clusters, we also replace the data objects in our membership matrix by cluster prototypes. Again, while it was reasonable to assign a constant weight of 1 to each of the data points, now that we have clusters of different sizes it makes more sense to consider their size as a constraint for the sum of membership degrees.

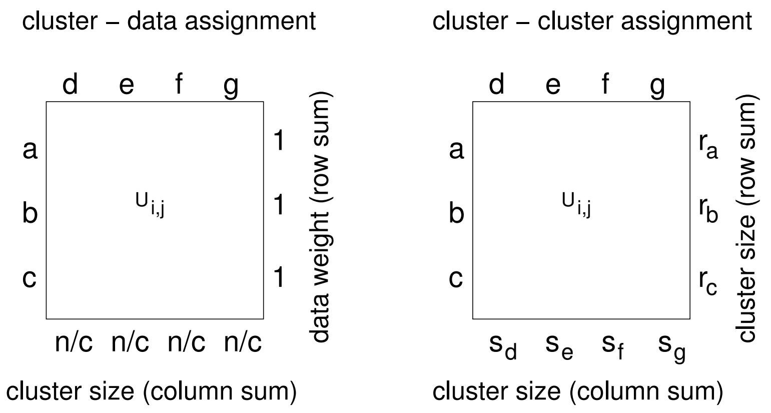


Fig. 11. Constraints on the membership matrix. Left: case of uniform clusters. Right: case of individual weights for data and prototypes.

We consider the membership matrix U as a data distribution matrix, where $u_{i,j}$ denotes how many data objects of cluster i of partition A shall be assigned to cluster j in partition B. Denoting the cluster size of set A by r_j, the cluster sizes r_j and s_i provide us with marginal distributions of the matrix. We are interested in deriving the joint distribution from the marginal distribution given that the total assignment costs are minimized. (Note that due to the possibility of negative entries for $u_{i,j}$ as discussed in section 4, the interpretation as a frequency distribution is not fully justified.)

Since we want to interpret the cluster sizes s_i and r_j as marginal distributions of the same matrix, we apparently require $S := \sum_i s_i = \sum_j r_j =: R$. If the prototypes are derived from data sets of different size such that $R \neq S$, we proportionally adjust the sizes of one data set, e.g. scale s_i by a factor of $\frac{R}{S}$.

To find the (unique) solution, we only have to replace constraint (1) by new constraints

$$\sum_{i=1}^{c} u_{i,j} = r_j \tag{14}$$

and we obtain a slightly modified linear system $A\beta = b'$ with identical $a_{k\ell}$ coefficients but new left hand side

$$b'_k \;=\; s_k - \sum_{j=1}^{n} \frac{r_j}{\sum_{i=1}^{c} \frac{d_{kj}}{d_{ij}}} \tag{15}$$

Determining the (minimal) distance between two sets of prototypes, consisting of their position and size, therefore involves solving this linear equation system and calculating the weighted sum of distances via the membership degrees (9) using

$$\alpha_j = \frac{2r_j - \sum_{i=1}^{c} \frac{\beta_i}{d_{ij}}}{\sum_{i=1}^{c} \frac{1}{d_{ij}}}. \tag{16}$$

We illustrate the performance of this distance measure using an example given in figure 12. One set of clusters consists of 4, the other of 3 prototypes. We examine the influence of the position of cluster a, which will move along the indicated line. All cluster sizes were 20, except the three clusters in the lower left corner. In experiment 1 the moving cluster has size 40 (and the two adjacent clusters of the other partition have size 20), in experiment 2 the moving cluster has size 10 (and the other two have size 5).

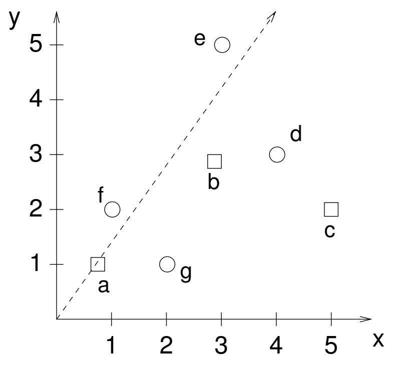


Fig. 12. Example data set. The prototype on the lower left will be moving along the dotted line towards the upper right corner.

Figure 13 shows the distance values depending on the x-coordinate of the moving cluster a. For experiment 1 (size 40) the best match is obtained if the prototype is close to the two smaller clusters f and g (cf. figure 13, right). If the size of clusters f and g is rather small (experiment 2), a good match for f and g is less valuable than a good match of cluster e: at $x \approx 3.5$ prototype a is very close to prototype e and the distance becomes minimal. This example shows that the consideration of the cluster sizes adds valuable information in the comparison.

7 Conclusions

In this paper, we have considered the problem of subdividing a data set into homogeneous groups with constraints on the number of data per group. Finding homogeneous groups is a typical task for clustering algorithms, however, if the

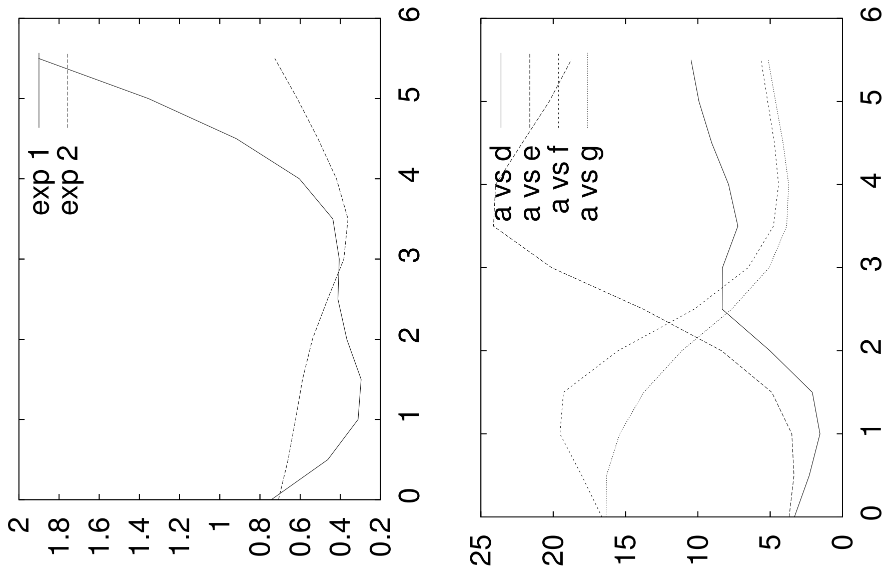


Fig. 13. Left: Distance values for both experiments. Right: distribution of the large cluster a (experiment 1) to clusters in partition $\{d, e, f, g\}$ ($u_{a,*}$).

data density is not uniform, such algorithms usually tend to deliver clusters of varying size, which is inappropriate for some applications. We have proposed an algorithm that outperforms a popular variant of k-means in that respect. We have discussed the case of equi-sized clusters as well as clusters of different sizes. Another interesting aspect of the proposed approach is the extension to the comparison of clustering results in general.

References

1. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall (1988)
2. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York (1981)
3. Höppner, F., Klawonn, F., Kruse, R., Runkler, T.A.: Fuzzy Cluster Analysis. John Wiley & Sons, Chichester, England (1999)
4. Klawonn, F., Höppner, F.: What is fuzzy about fuzzy clustering? – Understanding and improving the concept of the fuzzifier. In: Berthold, M.R., Lenz, H.-J., Bradley, E., Kruse, R., Borgelt, C. (eds.): Advances in Intelligent Data Analysis, Springer, Berlin (2003) 254–264
5. Klawonn, F., Höppner, F.: Equi-sized, homogeneous partitioning. In: Gabrys, B., Howlett, R.J., Jain, L.C.: Knowledge-Based Intelligent Information and Engineering Systems, Part II. Springer, Berlin (2006) 70–77