Visualising Clusters in High-Dimensional Data Sets by Intersecting Spheres

Frank Höppner and Frank Klawonn

***Abstract*— In this paper, we re-consider the problem of mapping a high-dimensional data set into a low-dimensional visualisation. We adopt the idea of multidimensional scaling but instead of projecting a high-dimensional point to a low-dimensional representation, we project a cluster in the high-dimensional space to a 3D-sphere. Rather than preserving distances from the high-dimensional space we aim at preserving the cluster interdependencies and try to recover them by the arrangement of the spheres. Using clusters and spheres rather than single data objects makes the method much more suitable for larger data sets. Our method can also be considered as a visual technique for cluster validity investigations. Strongly overlapping clusters or spheres in the visualisation are indicators for an unsuitable clustering result.**

I. Introduction

In this paper, we consider the problem of visualising a high-dimensional data set. Information visualisation has a long history and various methods for representing high dimensional data in a low-dimensional space (usually 2D or 3D) have been proposed, such as principal component analysis (PCA) [1] or multidimensional scaling (MDS) [2] or FastMap [3]. The idea of MDS is to map each high-dimensional data point to a corresponding point in the 2D plane such that the pairwise distances between any two data points is preserved best. The original MDS algorithm determines the data projections by minimizing an objective function which measures the difference between the high- and low-dimensional distance. The original approach suffers from the large number of parameters that have to be optimized ($2n$ parameters for n data objects).

The computational effort can be reduced by subsampling the original data set and projecting the smaller subsample instead of the complete original set. Rather than drawing a random subsample, some thoughts should be spent on finding a representative subset. This may be obtained, for instance, by performing a cluster analysis: the k-means clustering algorithm calculates c representative prototypes out of the n original data points. Since $c \ll n$, applying MDS to the cluster centres is much faster.

In this particular application the clustering algorithm is used as a kind of data reduction method – the number of clusters k therefore needs not to be the *true number of clusters* in the data set. Instead, we may select k depending on the computational effort we are willing to spend. Representing a data set of size $n = 1000$ by only some, say $c = 15$, typical MDS representatives is much faster, but also looses a lot of information: Nothing is said about the number of data objects that are represented by the prototype. A single prototype may represent a complete or only a portion of a cluster, but from the MDS projection we cannot tell whether the two clusters are well-separated or close neighbours.

Our aim is to adapt the MDS approach to overcome these problems. Rather than representing a cluster by a point in a scatter plot, we use spheres and thereby have means to express the size of a cluster with the help of the sphere's radius. (In this paper, we consider the case of a three-dimensional, interactive display.) We use the fuzzy c-means clustering algorithm (FCM), which is quite similar to k-means. It also seeks for the c most representative data points but, unlike k-means, every data object belongs to *all* clusters simultaneously – but to a different degree. We use this *degree of belongingness* to measure the *degree of overlap* between clusters and arrange the spheres in such a way that their intersection corresponds to the overlapping degree in the original clusters.

The outline of the paper is as follows: In the next section, we briefly introduce the fuzzy c-means clustering algorithm. In section III we discuss the transformation of the FCM result into a sphere configuration (radius and pairwise overlap). Next, in section IV, we discuss the objective function that has to be minimized to locate the spheres in such a way that their overlap corresponds to the overlap of FCM clusters. Finally, we present some results.

II. The Fuzzy c-Means Algorithm

The fuzzy c-means algorithm partitions a data set $X := \{x_1, ..., x_n\} \subset \mathbb{R}^d$ into c clusters. Each cluster is represented by a prototype $p_i \in \mathbb{R}^d$, $1 \le i \le c$. The data-prototype relation is fuzzy, that is, a membership degree $u_{i,j} \in [0, 1]$ 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 $U \in \mathbb{R}^{c \times n}$. In the classical FCM we can interpret the membership degrees as "probabilistic memberships", we have

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

The clustering process is carried out by minimizing the objective function

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

Frank Höppner is with the Department of Information Systems, University of Applied Sciences BS/WF, Wolfsburg, Germany, Email: f.hoeppner@fh-wolfsburg.de

Frank Klawonn is with the Department of Computer Science, University of Applied Sciences BS/WF, Wolfenbüttel, Germany, Email: f.klawonn@fh-wolfenbuettel.de

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 optimization, 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:

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

and for the membership degrees:

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

where $I_j = \{k \in \mathbb{N}_{\le c} \,|\, x_j = p_k\}$. The FCM algorithm is depicted in figure 1. For a more detailed discussion of FCM and examples we refer to the literature, e.g. [4], [5], [6].

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

Fig. 1. The fuzzy c-means algorithm.

III. Sphere Configuration

Once the membership matrix U and the cluster prototypes p_i, $i = 1, \ldots, c$, have been determined, we have to fix the radius r_i and sphere location q_i for each cluster in the 3D visualisation. A set of 3D spheres reflects the true data relationships well, if the intersection of the spheres corresponds to the overlap of the clusters. Before we can evaluate and optimize the position of the spheres, we therefore must define the desired overlap of the spheres.

A. Cluster Size

Since the volume of the spheres shall represent the *size* of the clusters, we calculate the radius r_i directly from the clustering result. We assume that the volume of a sphere corresponds directly to the number of data objects contained in a cluster. A canonical notion of the size of a cluster i in case of fuzzy clustering is

$$s_i = \sum_{j=1}^{n} u_{i,j}$$

The volume of a sphere with radius r is $V = \frac{4}{3}\pi r^3$. Fixing a constant data density ϱ (default $\varrho = 1$) for the visualisation, s_i data objects occupy a space of $\frac{s_i}{\varrho}$ and we thus obtain the radius

$$\frac{s_i}{\varrho} = V_i = \frac{4}{3}\pi r_i^3 \quad \Rightarrow \quad r_i = \sqrt[3]{\frac{3 s_i}{4 \varrho \pi}}$$

B. Cluster Overlap

We compute a squared $c \times c$ matrix $O = [o_{i,k}]$ encoding how much cluster i and cluster k overlap. Quite similar to the definition of the radius, we may define the degree of overlap between two clusters as

$$o_{i,k} = \sum_{j=1}^{n} \min\{u_{i,j}, u_{k,j}\} \tag{5}$$

This definition, however, leads to undesired results: Our goal is that a positive overlap value $o_{i,k}$ is reflected in the visualisation by intersecting spheres. Intuitively, if we have three clusters in a row, there will be now overlap between the two outer clusters, so their overlap values $o_{i,k}$ should be zero. But since in the fuzzy c-means algorithm a data object *always* belongs to *all* clusters simultaneously, none of the $o_{i,j}$ will actually be zero. In consequence, with this definition we request that *all* spheres should slightly intersect. This is an undesired effect, therefore we let only those data objects contribute to $o_{i,k}$ whose minimal membership degree $\min\{u_{i,j}, u_{k,j}\}$ lies above some threshold.

Fixing this threshold is not that easy: suppose the minimum of both membership degrees is 0.25. Should this data object contribute to $o_{i,k}$ or not? If the data object x_j belongs to one cluster with membership degree $u_{i,j} = 0.7$ probably not, because this indicates a quite clear assignment to cluster i. But if the highest membership degree is 0.4 only, the memberships degrees to both clusters are comparably high and we may say that x_j belongs to both clusters. Another problem is that the dimensionality of the data set influences the membership degrees: The higher the dimension, the more neighbours a cluster can have and the more clusters compete for the membership degree.

Therefore we use a dynamic threshold of $\alpha \cdot u_{*,j}$ with $u_{*,j} = \max_{i=1..c} u_{i,j}$, e.g. $\alpha = 0.6$. Only in case the minimum of $u_{i,j}$ and $u_{k,j}$ is at least 60% of the highest membership degree $u_{*,j}$, the assignment of the data object is considered as being ambiguous and x_j contributes to the overlap $o_{i,k}$. (We will present an alternative to this heuristic in section IV-D.)

Similar to the cluster size s_i, the overlap values $o_{i,k}$ correspond to the *(fuzzy) number of data objects* in the intersection of cluster i and k. Just as the cluster size has been transferred into a sphere's volume, we also transfer the number of data objects shared by clusters i and k into a corresponding space volume by replacing $o_{i,k}$ with $o_{i,k}/\varrho$.

C. Cluster Size Correction

We have calculated the radius of the spheres on the basis of the cluster size in section III-A. Suppose we have a data set with two well-separated clusters A and B. After a cluster analysis with $c = 3$ we finally arrive at the following visualisation: sphere #1 covers cluster A, the second cluster B is covered by spheres #2 and #3. Since A and B are of equal size, the spheres that represent cluster A and B should have equal volumes. If spheres #2 and #3 are well separated, the occupied volume will be equal to that of the sphere #1. But if spheres #2 and #3 intersect very much, they occupy much less space than sphere #1 and the user will perceive cluster B as being much smaller than cluster A. Therefore, we *increase* the size s_i of a cluster by $\frac{1}{2}\sum_{k=1}^{c} o_{i,k}$ before we actually calculate the sphere's radius r_i. With this modification, the user will perceive both clusters as equal-sized.

IV. Fitness Function

The *desired* overlap $o_{i,k}$ has been fixed in the previous section on the basis of the clustering result. Next, given all the spheres' positions q_i we need to measure how well the spheres' intersections correspond to the $o_{i,k}$ values. Once this has been done, we can formulate the problem of finding the optimal position of the spheres as a minimization problem. We then solve this minimization by using an evolution strategy.

A. Evaluating a Configuration

Consider two spheres with $d = ||q_1 - q_2||$ being the distance between their centres. Without loss of generality let us assume that $r_1 < r_2$, that is, sphere 1 with radius r_1 is the larger one. The volume of sphere i is $V_i = \frac{4}{3}\pi r_i^3$. Depending on the radii r_i and the distance d between the centres of both spheres we distinguish four cases (cf. figure 2) to measure (resp. approximate) the intersection volume V of both spheres.

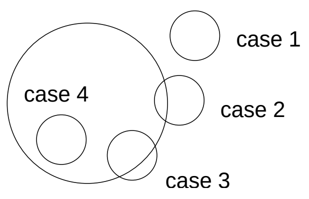

Fig. 2. Possible cases of intersection.

Case 1: $r_1 + r_2 \leq d$. Both spheres are separated and do not intersect. The intersection volume is therefore $V = 0$ (cf. figure 2).

Case 2: $r_1 \leq d \leq r_1 + r_2$. In this case both spheres intersect, but neither centre lies within the other sphere. The intersection of both spheres' surfaces defines a plane and the intersection volume can be considered as a composition of two spherical caps (cf. figure 3) sharing this plane as their base. Using the notation in figure 3, the height of the two caps is given by $r_i - d_i$. From the two right-angled triangles in the figure ($h^2 + d_1^2 = r_1^2$ and $h^2 + d_2^2 = r_2^2$) we derive

$$d_1 = \frac{d^2 + r_1^2 - r_2^2}{2d}$$

and obtain d_2 from $d_2 = d - d_1$. The volume of a spherical cap with height h and radius r is given by $\frac{1}{3}\pi h^2(3r - h)$, therefore the intersection volume is given by

$$V = \frac{1}{3}\pi \sum_{i=1}^{2} (r_i - d_i)^2 (2r_i + d_i)$$

Case 3: $r_1 - r_2 \leq d < r_1$. In this case, the centre of the smaller sphere falls within the larger sphere (cf. figure 4). We consider this being a rare case in our application, because this means that we have more data objects in the overlapping region than in the cluster itself.

Using the notation in figure 4, we construct the overlap volume from $V = V_2 - V_2' + V_1'$ with V_2 being the volume of sphere 2, $V_2' = \frac{1}{3}\pi h_2^2(3r_2 - h_2)$ being the spherical cap of sphere 2 with height h_2 and $V_1' = \frac{1}{3}\pi h_1^2(3r_1 - h_1)$ the spherical cap of sphere 1 with height h_1 (cf. figure 4). From $(d+x)^2 + y^2 = r_1^2$ and $x^2 + y^2 = r_2^2$ we obtain $x = \frac{r_1^2 - r_2^2 - d^2}{2d}$ and get the cap heights from

$$h_2 = r_2 - x \qquad \text{and} \qquad h_1 = r_1 - (d + x)$$

Case 4: $d < r_1 - r_2$ In this case, sphere 2 is completely absorbed by sphere 1 and the intersection volume is $V = V_2$. This case, however, is almost impossible when the overlapping volumes are derived from the result of a partitional cluster analysis.

B. Evolution Strategy

Let us denote the intersection volume of the spheres corresponding to cluster i and k by $v_{i,k}$. Then, our aim is to minimize

$$J = \sum_{i=1}^{c} \sum_{j=i+1}^{c} ||o_{i,j} - v_{i,j}||^2 \qquad (6)$$

The parameters to be optimized in this objective function are hidden in the $v_{i,k}$-values. Since the radii of our 3D-spheres are directly derived from the corresponding clusters, the only free parameters are the centres of the spheres. This means, the optimization has to adjust $3c$ parameters, because the centre of each sphere is determined by a point in the three-dimensional space. Since all parameters are real-valued, we apply a standard evolution strategy (ES) (for a detailed overview see for instance [7]). We use a $(10, 250)$-ES with adaption of the mutation rate. This means that we start with a random population of 10 configurations of c sphere centres. Each sphere centre will generate 25 children by adding a normal random variable with expectation value zero to each of its three coordinates. From the 250 children the 10 best are taken to the next generation. As a consequence, the best solution might get lost, when all children are worse than the best solution among the parents. Therefore, we always store the best solution, to recover it at the end of the evolution process. The adaption of the mutation rate, i.e. the choice of the variance of the normal distributions, is carried out according to Rechenberg's 1/5 rule of thumb [8]. This means that the mutation rate or average width of mutations is not changed, if approximately 1/5 of the mutations are successful. When the rate of successful mutations is significantly higher, the mutation rate is increased, when the success rate is significantly lower, the mutation rate is decreased. We stop the evolution process, when no improvement of the best solution could be achieved within the last 20 generations.

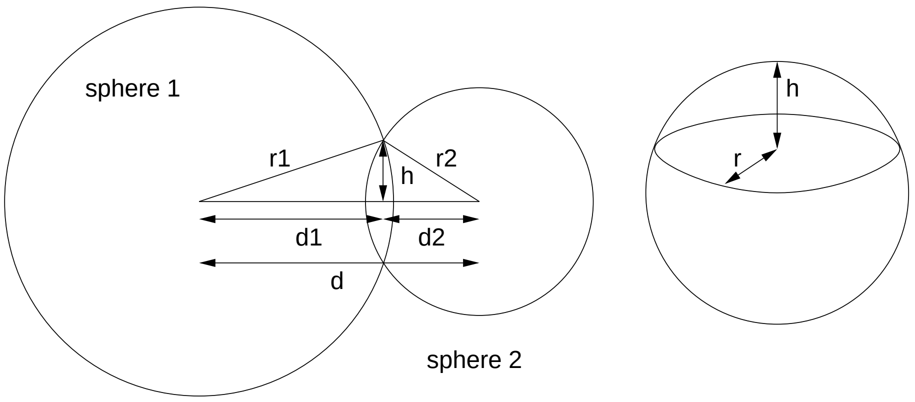

Fig. 3. Intersection volume in case 2.

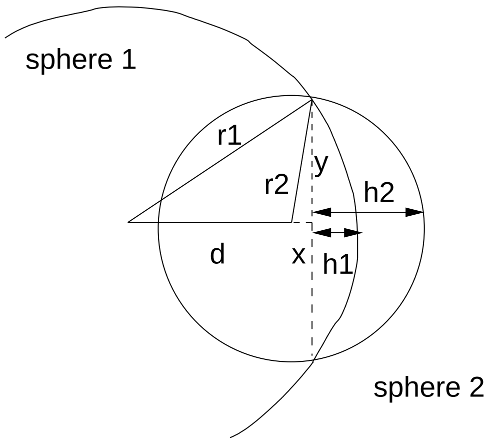

Fig. 4. Intersection volume in case 3.

The choices of the parameters for the evolution strategy were made on a purely heuristic basis, but turned out to be quite successful. Nevertheless, a detailed study of the influence of the parameters on the results and the time needed to find a good solution might yield an even better configuration. However, such investigations are not within the scope of this paper.

C. Further Modifications

Practical experiments revealed two problems. Firstly, if two spheres do not overlap (but they should because $o_{i,k} > 0$), there is no gradient information in J unless sphere i and k are already close to each other. The evolutionary strategy tries to improve the fitness function J by *mutating* the position vector, but if none of these mutations lead (incidentally) to an intersection of both spheres, the fitness function will not change. To make this kind of random search more directed, we add a penalty term: if $v_{i,k} = 0$ but $o_{i,k} > 0$, we add d^2 to introduce a tendency of moving both spheres closer to each other.

Secondly, in the opposite case, if two clusters do not overlap, from the point of minimizing J, we are perfectly fine if the corresponding spheres touch each other (but do not intersect). In this case, we have $o_{i,k} = v_{i,k} = 0$ and so the difference is also zero. However, when *looking* at the visualisation it is extremely difficult to see whether both spheres actually overlap or not. This makes the visual perception less valuable.

We may apply the same technique of introducing a penalty term. In case of $o_{i,k} = 0$, if both spheres are too close to each other (e.g. closer than $2(r_1 + r_2)$) we add a penalty term (e.g. $(2(r_1 + r_2) - d)^2$) to introduce a repelling tendency in the near range. There is, however, a drawback with this approach: If we have, for instance, three spheres A, B, C and there is some overlap between A and B as well as between B and C, but none between A and C, then this penalty term may still apply to A and C (depending on the radius of B). This is highly undesirable, because the penalty term will now shift both spheres apart from each other – thereby deteriorating the fit of A/B and B/C.

We therefore use a different approach. Let us consider the clusters as nodes in a graph and an overlap between clusters as an edge in this graph. We then identify all connected subgraphs in a first step. The goal of the optimization phase is to fit all the overlapping degrees, therefore subgraphs that are not connected among each other can be optimized independently. This does not only reduce the computational effort, but also solves the problem of perceiving unconnected subgraphs as being connected, because now we can place the individually optimized subgraphs separately.

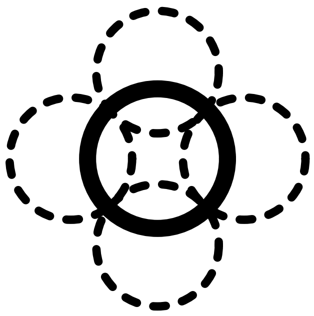

Fig. 5. Only a limited number of intersections can be achieved without further interference.

D. *Controlling the Complexity – Cluster Overlap Revisited*

By identifying the connected subgraphs and optimizing them individually, it is already possible to lower the complexity of the approach. Another kind of *controlled* complexity reduction is the following: Obviously a sphere can only intersect a limited number of other spheres (cf. figure 5) – and the higher the data space, the higher the number of possible intersections. Knowing that the evolution strategy can only reflect only a limited number, say *maxconn*, of overlapping clusters, we may discard all cluster intersections below some overlapping degree o_{cut}. We choose o_{cut} such that we obtain at most *maxconn* overlapping spheres:

1) We calculate $o_{i,k}$ directly from (5), ignoring the heuristic threshold mentioned in section III-B.
2) For each cluster i we sort the overlap volumes $o_{i,k}$ in decreasing order: $\hat{o}_{i,1}, \hat{o}_{i,2}, ..., \hat{o}_{i,c}$
3) Then, $\hat{o}_{i,\mathit{maxconn}}$ denotes the highest overlapping degree for cluster i that we might consider while staying below *maxconn* connected spheres.
4) We choose o_{cut} as

$$o_{\text{cut}} = \max_{i=1..c} \hat{o}_{i,\mathit{maxconn}}$$

5) All overlap volumes $o_{i,k} < o_{\text{cut}}$ will be set to zero (and thus ignored during the optimisation).

The advantage of this approach is that it becomes very clear, to what extent the visualisation simplifies the true cluster relationships: all intersections below o_{cut} are not shown. The threshold o_{cut} may be determined for all connected subgraphs (global constant) or for each subgraph individually.

V. Experiments

In this section, we give some preliminary results. It cannot be overemphasized that the printed pictures appear quite unspectacular and unimpressive, because a three dimensional, animated scene *lives* from interaction. By changing the perspective the human observer gets a much better impression of how the clusters mutually intersect.

Nevertheless, figures 6, 7 and 8 show a screen shot of the visualisation of the IRIS data set, the auto-mpg and the wine data set from the UCI machine learning repository [9].

Table I gives an impression on the achieved accuracy in minimizing the fitness function (6) depending on the chosen *maxconn* parameter. For all datasets we observe a sharp increase in the fitness function at some *maxconn* value, e.g., for $4 \to 5$ in the auto-mpg data set or for $5 \to 6$ in the wine data set. These values correspond to our intuition, because we can expect that we can adjust the overlapping degrees of at most 4 to 6 spheres at the same time.

maxconn	iris	wine	auto
3	17.70	6.73	0.005
4	38.53	2.67	0.04
5	40.82	2.63	55.53
6	44.20	7.06	25.96
7	53.38	37.60	32.40

TABLE I
VALUE OF THE FITNESS FUNCTION (6) DEPENDING ON *maxconn*.

For the visualisation of the wine data set (figure 8), out of 25 sphere intersections, 12 matched the original overlap volume perfectly, in 4/1/2/1/1 cases we had an error below 5/10/15/20/25%, in 4 cases there was a sphere intersection although $o_{i,k} = 0$. Altogether, the approximation quality is quite good and the visualisation turns out to give a good impression of the structure in the data set.

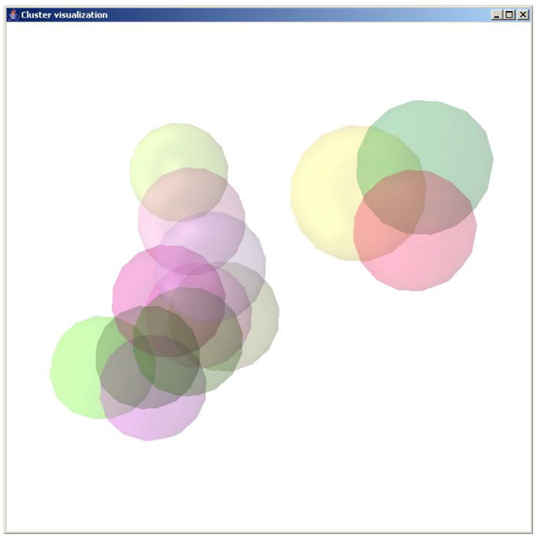

Fig. 6. Visualisation of the IRIS data set with $c = 15$, *maxconn*$= 5$.

VI. Conclusions

We have introduced a visualisation technique for higher-dimensional and larger data sets based on a fuzzy cluster analysis. The purpose of the clustering is not necessarily to find an optimal (fuzzy) partition of the original data. The function of the clusters is to cover the data. The visualisation then tries to represent the clusters in such a way that their overlap is best preserved. We have used an animated 3D visualisation, since it allows more freedom than a simple 2D representation. Of course, our approach can be applied in the same manner to obtain a 2D visualisation. However, in this case clusters would be represented by overlapping circles instead of spheres, making it more difficult to preserve the overlapping regions in the original space. A 2D representation would also affect the choice of our parameter *maxconn* that must be chosen much smaller than in the 3D case.

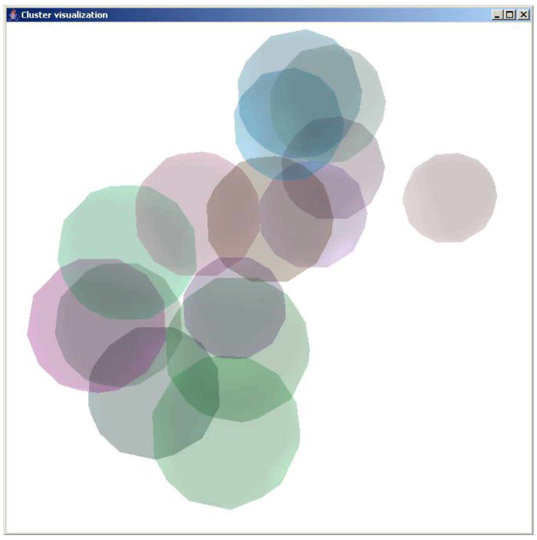

Fig. 7. Visualisation of the auto-mpg data set with $c = 15$, *maxconn*$= 4$.

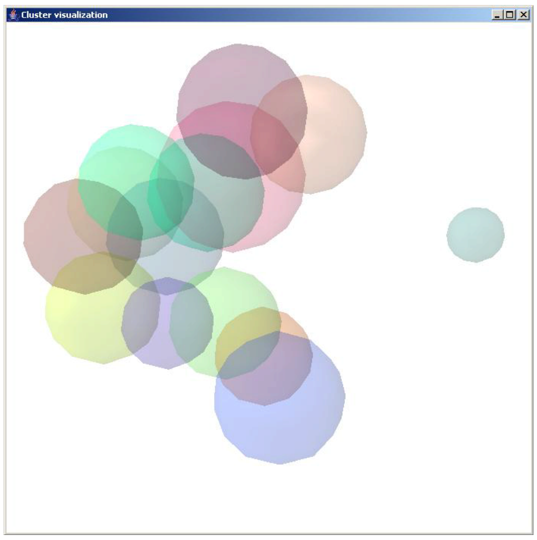

Fig. 8. Visualisation of the wine data set with $c = 15$, *maxconn*$= 5$.

It should be mentioned that there are also also ways to exploit fuzzy cluster analysis for visualisation purposes as they are for instance proposed in [10], [11].

Finally, it should be noted that our approach can also be used as a kind of visual cluster validity check. When we are interested in a suitable clustering of the original data instead of just partitioning them somehow for the visualisation, we can visualise the clusters and see how much they overlap. Global cluster validity measures like the partition coefficient or the partition entropy [4] are based on the overlap of clusters. The visualisation has the advantage that we do not only obtain a single number as an indication of how well the clusters fit the data – a problem already addressed in [12] – but we can also see which clusters might not be trusted due to their high overlap with others.

References

[1] G. Dunn and B. Everitt, *An Introduction to Mathematical Taxonomy*. Cambridge, MA: Cambridge University Press, 1982.

[2] R. Shepard and S. Nerlove, *Multidimensional Scaling*. New York: Seminar Press, 1972.

[3] C. Faloutsos and K.-I. Lin, “FastMap: A fast algorithm for indexing, data mining and visualisation of traditional and multimedia datasets,” in *Proc. of ACM SIGMOD Int. Conf. on Management of Data*, 1995, pp. 163–174.

[4] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

[5] J. Bezdek, J. Keller, R. Krishnapuram, and N. Pal, *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*. Boston: Kluwer Academic Publishers, 1999.

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

[7] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. Oxford: Oxford University Press, 1996.

[8] I. Rechenberg, *Evolutionsstrategie’94*. Stuttgart: Frommann-Holzboog, 1994.

[9] C. Blake and C. Merz, “UCI repository of machine learning databases,” [Online]. Available: http://www.ics.uci.edu/ mlearn/MLRepository.html, 1998.

[10] K. Yamamoto, T. Yoshikawa, and T. Furuhashi, “A proposal of fuzzy modeling on fusion axes considering the data structure,” in *Proc. of the IEEE Intl Conf. on Fuzzy Systems (FUZZIEEE2003)*. IEEE Press.

[11] K. Yamamoto, T. Furuhashi, and T. Yoshikawa, “A proposal of visualization method using fuzzy clustering and fuzzy multiple discriminant analysis,” in *Intl Workshop of Fuzzy Systems & Innovational Computing (FIC2004)*, 2004, pp. 356–361.

[12] F. Klawonn, V. Chekhtman, and E. Janz, “Visual inspection of fuzzy clustering results,” in *Advances in Soft Computing: Engineering Design and Manufacturing*, O. Benitez, J.and Cordón, F. Hoffmann, and R. Roy, Eds. London: Springer-Verlag, 2003, pp. 65–76.