Fuzzy and Probabilistic Clustering with Shape and Size Constraints

Christian Borgelt and Rudolf Kruse

Department of Knowledge Processing and Language Engineering
School of Computer Science, Otto-von-Guericke-University of Magdeburg
Universitätsplatz 2, 39106 Magdeburg, Germany
Email: {borgelt,kruse}@iws.cs.uni-magdeburg.de

Abstract: More sophisticated fuzzy clustering algorithms, like the Gustafson–Kessel algorithm [11] and the fuzzy maximum likelihood estimation (FMLE) algorithm [10] offer the possibility of inducing clusters of ellipsoidal shape and different sizes. The same holds for the expectation maximization (EM) algorithm for a mixture of Gaussians. However, these additional degrees of freedom can reduce the robustness of the algorithm, thus sometimes rendering their application problematic. In this paper we suggest methods to introduce shape and size contraints that handle this problem effectively.

Keywords: Fuzzy Clustering, Expectation Maximization, Cluster Size, Cluster Shape, Regularization

1 Introduction

Prototype-based clustering methods, like fuzzy clustering [1, 2, 12], expectation maximization (EM) [6] of a mixture of Gaussians [9], or learning vector quantization [15, 16], often employ a distance function to measure the similarity of two data points. If this distance function is the *Euclidean distance*, all clusters are (hyper-)spherical. However, more sophisticated approaches rely on a cluster-specific *Mahalanobis distance*, making it possible to find clusters of (hyper-)ellipsoidal shape. In addition, they relax the restriction (as it is present, e.g., in the fuzzy c-means algorithm) that all clusters have the same size [13]. Unfortunately, these additional degrees of freedom often reduce the robustness of the clustering algorithm, thus sometimes rendering their application problematic.

In this paper we consider how shape and size parameters of a cluster can be constrained, that is, modified in such a way that extreme cases are ruled out and/or a bias against extreme cases is introduced, which effectively improves robustness. The basic idea of constraining shape is the same as that of Tikhonov regularization for linear optimization problems [18, 8], while size and weight constraints can be based on a bias towards equality as it is well-known from Laplace correction or Bayesian approaches to probability estimation.

This paper is organized as follows: in Sections 2 and 3 we briefly review some basics of mixture models and the expectation maximization algorithm as well as fuzzy clustering. In Section 4 we discuss our methods to constrain shape, size, and weight parameters in clustering. In Section 5 we present experimental results on well-known data sets and finally, in Section 6, we draw conclusions from our discussion.

2 Mixture Models and EM Algorithm

In a mixture model [9] it is assumed that a given data set $\mathcal{X} = \{\vec{x}_j \mid j = 1, \ldots, n\}$ has been sampled from a population of c clusters. Each cluster is characterized by a probability distribution, specified as a prior probability and a conditional probability density function (cpdf). The data generation process may then be imagined as follows: first a cluster i, $i \in \{1, \ldots, c\}$, is chosen for a datum, indicating the cpdf to be used, and then the datum is sampled from this cpdf. Consequently the probability of a data point $\vec{x}$ can be computed as

$$p_{\vec{X}}(\vec{x};\Theta) = \sum_{i=1}^{c} p_C(i;\Theta_i) \cdot f_{\vec{X}|C}(\vec{x}|i;\Theta_i),$$

where C is a random variable describing the cluster i chosen in the first step, $\vec{X}$ is a random vector describing the attribute values of the data point, and $\Theta = \{\Theta_1, \ldots, \Theta_c\}$ with each Θ_i containing the parameters for one cluster (that is, its prior probability $\theta_i = p_C(i;\Theta_i)$ and the parameters of the cpdf).

Assuming that the data points are drawn independently from the same distribution (i.e., that the probability distributions of their underlying random vectors $\vec{X}_j$ are identical), we can compute the probability of a data set $\mathcal{X}$ as

$$P(\mathcal{X};\Theta) = \prod_{j=1}^{n} \sum_{i=1}^{c} p_{C_j}(i;\Theta_i) \cdot f_{\vec{X}_j|C_j}(\vec{x}_j|i;\Theta_i),$$

Note, however, that we do not know which value the random variable C_j, which indicates the cluster, has for each example case $\vec{x}_j$. Fortunately, though, given the data point, we can compute the posterior probability that a data point $\vec{x}$ has been sampled from the cpdf of the i-th cluster using Bayes' rule:

$$\begin{aligned} p_{C|\vec{X}}(i|\vec{x};\Theta) &= \frac{p_C(i;\Theta_i) \cdot f_{\vec{X}|C}(\vec{x}|i;\Theta_i)}{f_{\vec{X}}(\vec{x};\Theta)} \\ &= \frac{p_C(i;\Theta_i) \cdot f_{\vec{X}|C}(\vec{x}|i;\Theta_i)}{\sum_{k=1}^{c} p_C(k;\Theta_k) \cdot f_{\vec{X}|C}(\vec{x}|k;\Theta_k)}. \end{aligned}$$

This posterior probability may be used to complete the data set w.r.t. the cluster, namely by splitting each datum $\vec{x}_j$ into c data points, one for each cluster, which are weighted with the posterior probability $p_{C_j|\vec{X}_j}(i|\vec{x}_j;\Theta)$. This idea is used in the well-known expectation maximization (EM) algorithm [6], which consists in alternately computing these posterior probabilities

and estimating the cluster parameters from the completed data set by maximum likelihood estimation.

For clustering numeric data it is usually assumed that the cpdf of each cluster is an m-variate normal distribution (so-called *Gaussian mixture model* [9, 3]). That is,

$$\begin{aligned} f_{\vec{X}|C}(\vec{x}|i;\Theta_i) &= N(\vec{x};\vec{\mu}_i,\Sigma_i) \\ &= \frac{1}{\sqrt{(2\pi)^m|\Sigma_i|}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu}_i)^\top \Sigma_i^{-1}(\vec{x}-\vec{\mu}_i)\right), \end{aligned}$$

where $\vec{\mu}_i$ is the mean vector and Σ_i the covariance matrix of the normal distribution, $i = 1,\ldots,c$, and m is the number of dimensions of the data space. In this case the maximum likelihood estimation formulae are

$$\theta_i = \frac{1}{n}\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\Theta)$$

for the prior probability θ_i,

$$\vec{\mu}_i = \frac{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\Theta)\cdot\vec{x}_j}{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\Theta)}$$

for the mean vector $\vec{\mu}_i$, and

$$\Sigma_i = \frac{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\Theta)\cdot(\vec{x}_j-\vec{\mu}_i)(\vec{x}_j-\vec{\mu}_i)^\top}{\sum_{j=1}^{n} p_{C|\vec{X}_j}(i|\vec{x}_j;\Theta)}$$

for the covariance matrix Σ_i of the i-th cluster, $i = 1,\ldots,c$.

3 Fuzzy Clustering

While most classical clustering algorithms assign each datum to exactly one cluster, thus forming a crisp partition of the given data, fuzzy clustering allows for *degrees of membership*, to which a datum belongs to different clusters [1, 2, 12]. Most fuzzy clustering algorithms are objective function based: they determine an optimal (fuzzy) partition of a given data set $\mathbf{X} = \{\vec{x}_j \mid j = 1,\ldots,n\}$ into c clusters by minimizing an objective function

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

subject to the constraints

$$\sum_{j=1}^{n} u_{ij} > 0, \qquad \text{for all } i \in \{1,\ldots,c\}, \qquad \text{and} \tag{1}$$

$$\sum_{i=1}^{c} u_{ij} = 1, \qquad \text{for all } j \in \{1,\ldots,n\}, \tag{2}$$

where $u_{ij} \in [0,1]$ is the membership degree of datum $\vec{x}_j$ to cluster i and d_{ij} is the distance between datum $\vec{x}_j$ and cluster i. The $c \times n$ matrix $\mathbf{U} = (u_{ij})$ is called the *fuzzy partition matrix* and $\mathbf{C}$ describes the set of clusters by stating location parameters (i.e. the cluster center) and maybe size and shape parameters for each cluster. The parameter w, $w > 1$, is called the *fuzzifier* or *weighting exponent*. It determines the "fuzziness" of the classification: with higher values for w the boundaries between the clusters become softer, with lower values they get harder. Usually $w = 2$ is chosen. Hard clustering results in the limit for $w \to 1$. However, a hard assignment may also be determined from a fuzzy result by assigning each data point to the cluster to which it has the highest degree of membership.

Constraint (1) guarantees that no cluster is empty and constraint (2) ensures that each datum has the same total influence by requiring that the sum of the membership degrees of a datum must be 1. Due to the second constraint this approach is usually called *probabilistic fuzzy clustering*, because with it the membership degrees for a datum formally resemble the probabilities of its being a member of the corresponding clusters. The partitioning property of a probabilistic clustering algorithm, which "distributes" the weight of a datum to the different clusters, is due to this constraint.

Unfortunately, the objective function J cannot be minimized directly. Therefore an iterative algorithm is used, which alternately optimizes membership degrees and cluster parameters [1, 2, 12]. That is, first the membership degrees are optimized for fixed cluster parameters, then the cluster parameters are optimized for fixed membership degrees. The main advantage of this scheme is that in each of the two steps the optimum can be computed directly. By iterating the two steps the joint optimum is approached (although, of course, it cannot be guaranteed that the global optimum will be reached—one may get stuck in a local minimum of the objective function J).

The update formulae are derived by simply setting the derivative of the objective function J w.r.t. the parameters to optimize equal to zero (necessary condition for a minimum). Independent of the chosen distance measure we thus obtain the following update formula for the membership degrees [12]:

$$u_{ij} = \frac{d_{ij}^{-\frac{2}{w-1}}}{\sum_{k=1}^{c} d_{kj}^{-\frac{2}{w-1}}}, \tag{3}$$

that is, the membership degrees represent the relative inverse squared distances of a data point to the different cluster centers, which is a very intuitive result.

The update formulae for the cluster parameters, however, depend on what parameters are used to describe a cluster (location, shape, size) and on the chosen distance measure. Therefore a general update formula cannot be given. Here we briefly review the three most common cases: The best-known fuzzy clustering algorithm is the fuzzy c-means algorithm, which is a straightforward generalization of the classical crisp c-means algorithm. It uses only cluster centers for the cluster prototypes and relies on the *Euclidean distance*, i.e.,

$$d_{ij}^2 = (\vec{x}_j-\vec{\mu}_i)^\top(\vec{x}_j-\vec{\mu}_i),$$

where $\vec{\mu}_i$ is the center of the i-th cluster. Consequently it is restricted to finding spherical clusters of equal size. The resulting update rule is

$$\vec{\mu}_i = \frac{\sum_{j=1}^{n} u_{ij}^w \vec{x}_j}{\sum_{j=1}^{n} u_{ij}^w}, \tag{4}$$

that is, the new cluster center is the weighted mean of the data points assigned to it, which is again a fairly intuitive result.

The Gustafson–Kessel algorithm [11] uses a cluster-specific *Mahalanobis distance*, i.e.,

$$d_{ij}^2 = (\vec{x}_j-\vec{\mu}_i)^\top\Sigma_i^{-1}(\vec{x}_j-\vec{\mu}_i),$$

where $\vec{\mu}_i$ is the cluster center and Σ_i is a cluster-specific covariance matrix with determinant 1. It describes the shape of the cluster, thus allowing for ellipsoidal clusters of equal size. This distance measure leads to same update rule (4) for the clusters centers, while the covariance matrices are updated as

$$\Sigma_i = \frac{\Sigma_i^*}{\sqrt[m]{|\Sigma_i^*|}} \quad \text{where} \quad \Sigma_i^* = \frac{\sum_{j=1}^n u_{ij}^w (\vec{x}_j - \vec{\mu}_i)(\vec{x}_j - \vec{\mu}_i)^\top}{\sum_{j=1}^n u_{ij}^w} \quad (5)$$

and m is the number of dimensions of the data space. Σ_i^* is called the *fuzzy covariance matrix*, which is simply normalized to determinant 1 to meet the abovementioned constraint. Compared to standard statistical estimation procedures, this is also a fairly intuitive result. It should be noted that the restriction to clusters of equal size may be relaxed by simply allowing general covariance matrices. However, depending on the characteristics of the data, this additional degree of freedom can deteriorate the robustness of the algorithm.

Finally, the fuzzy maximum likelihood estimation (FMLE) algorithm [10] is based on the assumption that the data was sampled from a mixture of c multivariate normal distributions as in the statistical approach of mixture models (cf. Section 2). It uses a (squared) distance that is inversely proportional to the probability that a datum was generated by the normal distribution associated with a cluster and also incorporates the prior probability of the cluster. That is,

$$d_{ij}^2 = \left(\frac{\theta_i}{\sqrt{(2\pi)^m |\Sigma_i|}} \; \exp\left(-\frac{1}{2} (\vec{x}_j - \vec{\mu}_i)^\top \Sigma_i^{-1} (\vec{x}_j - \vec{\mu}_i) \right) \right)^{-1},$$

where θ_i is the prior probability of the cluster, $\vec{\mu}_i$ is the cluster center, Σ_i a cluster-specific covariance matrix, which in this case is not required to be normalized to determinant 1, and m the number of dimensions of the data space (cf. Section 2). For the FMLE algorithm the update rules are not derived from the objective function due to technical obstacles, but by comparing it to the expectation maximization (EM) algorithm for a mixture of normal distributions (cf. Section 2), which, by analogy, leads to the same update rules for the cluster center and the cluster-specific covariance matrix as for the Gustafson–Kessel algorithm [12], that is, equations (4) and (5). The prior probability θ_i is, also in analogy to statistical estimation (cf. Section 2), computed as

$$\theta_i = \frac{1}{n} \sum_{j=1}^n u_{ij}^w. \quad (6)$$

Note that the difference to the expectation maximization algorithm consists in the different ways in which the membership degrees (equation (3)) and the posterior probabilities in the EM algorithm are computed and used in the estimation.

Since the high number of free parameters of the FMLE algorithm renders it unstable on certain data sets, it is usually recommended [12] to initialize it with a few steps of the very robust fuzzy c-means algorithm. The same holds, though to a somewhat lesser degree, for the Gustafson–Kessel algorithm.

It is worth noting that of both the Gustafson–Kessel algorithm as well as the FMLE algorithm there exist so-called *axes-parallel* versions, which restrict the covariance matrices Σ_i to diagonal matrices and thus allow only axes-parallel ellipsoids [14]. These constrained variants have certain advantages w.r.t. robustness and execution time.

4 Constraining Cluster Parameters

The large number of parameters (mainly the elements of the covariance matrices) of the more flexible fuzzy and probabilistic clustering algorithms can render these algorithms less robust or even fairly unstable, compared to their simpler counterparts that only adapt the cluster centers. Common undesired results include very long and thin ellipsoids as well as clusters collapsing to a single data point. To counteract such undesired tendencies, we introduce shape and size constraints into the update scheme. The basic idea is to modify, in every update step, the parameters of a cluster in such a way that certain constraints are satisfied or at least that a noticable tendency (of varying strength, as specified by a user) towards satisfying these constraints is introduced. In particular we consider regularizing the (ellipsoidal) shape as well as constraining the (relative) size and the (relative) weight of a cluster.

4.1 Constraining Cluster Shapes

The shape of a cluster is represented by its covariance matrix Σ_i. Intuitively, Σ_i describes a general (hyper-)ellipsoidal shape, which can be obtained, for example, by computing the Cholesky decomposition or the eigenvalue decomposition of Σ_i and mapping the unit (hyper-)sphere with it.

Shape regularization means to modify the covariance matrix, so that a certain relation of the lengths of the major axes of the represented (hyper-)ellipsoid is obtained or that at least a tendency towards this relation is introduced. Since the lengths of the major axes are the roots of the eigenvalues of the covariance matrix, regularizing it means shifting the eigenvalues of Σ_i. Note that such a shift leaves the eigenvectors unchanged, i.e., the orientation of the represented (hyper-)ellipsoid is preserved. Note also that such a shift of the eigenvalues is the basis of the well-known Tikhonov regularization for linear optimization problems [18, 8], which inspired our approach. We suggest two methods:

Method 1: The covariance matrices Σ_i, $i = 1, \ldots, c$, of the clusters are adapted (in every update step) according to

$$\Sigma_i^{(\text{adap})} = \sigma_i^2 \cdot \frac{\mathbf{S}_i + h^2 \mathbf{1}}{\sqrt[m]{|\mathbf{S}_i + h^2 \mathbf{1}|}} = \sigma_i^2 \cdot \frac{\Sigma_i + \sigma_i^2 h^2 \mathbf{1}}{\sqrt[m]{|\Sigma_i + \sigma_i^2 h^2 \mathbf{1}|}},$$

where m is the dimension of the data space, $\mathbf{1}$ is a unit matrix, $\sigma_i^2 = \sqrt[m]{|\Sigma_i|}$ is the equivalent isotropic variance (equivalent in the sense that it leads to the same (hyper-)volume, i.e., $|\Sigma_i| = |\sigma_i^2 \mathbf{1}|$), $\mathbf{S}_i = \sigma_i^{-2} \Sigma_i$ is the covariance matrix scaled to determinant 1, and h is the regularization parameter.

This modification of the covariance matrix shifts all eigenvalues by the value of $\sigma_i^2 h^2$ and then renormalizes the resulting matrix so that the determinant of the old covariance matrix is preserved (i.e., the (hyper-)volume of the cluster is kept constant). It tends to equalize the lengths of the major axes of the represented (hyper-)ellipsoid and thus introduces a tendency towards (hyper-)spherical clusters. (Algebraically, it makes the matrix "less singular", and thus "more regular", which explains the name *regularization* for this modification.) This tendency of equalizing the axes lengths is the stronger, the greater the value of h. In the limit, for $h \to \infty$, the clusters are forced to be exactly spherical; for $h = 0$ the shape is left unchanged.

Method 2: The above method always changes the length ratios of the major axes and thus introduces a general tendency towards (hyper-)spherical clusters. In this (second) method, however, a limit r, $r > 1$, for the length ratio of the longest to the shortest major axis of the represented (hyper-)ellipsoid is used and only if this limit is exceeded, the eigenvalues are shifted in such a way that the limit is satisfied.
Formally: let λ_k, $k = 1, \ldots m$, be the eigenvalues of the covariance matrix Σ_i. Set (in every update step)

$$h^2 = \begin{cases} 0, & \text{if } \dfrac{\max_{k=1}^m \lambda_k}{\min_{k=1}^m \lambda_k} \leq r^2, \\ \dfrac{\max_{k=1}^m \lambda_k - r^2 \min_{k=1}^m \lambda_k}{\sigma_i^2 (r^2 - 1)}, & \text{otherwise,} \end{cases}$$

and then execute Method 1 with this value of h^2.

4.2 Constraining Cluster Sizes

The size of a cluster can be described in different ways, for example, by the determinant of its covariance matrix Σ_i, which is a measure of the clusters squared (hyper-)volume, an equivalent isotropic variance σ_i^2 or an equivalent isotropic radius (standard deviation) σ_i (equivalent in the sense that they lead to the same (hyper-)volume, see above). The latter two measures are defined as

$$\sigma_i^2 = \sqrt[m]{|\Sigma_i|} \qquad \text{and} \qquad \sigma_i = \sqrt{\sigma_i^2} = \sqrt[2m]{|\Sigma_i|}$$

and thus the (hyper-)volume of a cluster may also be written as $\sigma_i^m = \sqrt{|\Sigma_i|}$.

Contraining the (relative) cluster size means to ensure a certain relation between the sizes or at least to introduce a tendency into this direction. We suggest three different versions of modifying cluster sizes, in each of which the measure that is used to describe the cluster size is specified by an exponent a of the equivalent isotropic radius σ_i. Special cases are

$$\begin{array}{ll} a = 1: & \text{equivalent isotropic radius,} \\ a = 2: & \text{equivalent isotropic variance,} \\ a = m: & \text{(hyper-)volume.} \end{array}$$

Method 1: The equivalent isotropic radii σ_i are adapted (in every update step) according to

$$\begin{aligned} \sigma_i^{(\text{adap})} &= \sqrt[a]{s \cdot \frac{\sum_{k=1}^c \sigma_k^a}{\sum_{k=1}^c (\sigma_k^a + b)} \cdot (\sigma_i^a + b)} \\ &= \sqrt[a]{s \cdot \frac{\sum_{k=1}^c \sigma_k^a}{cb + \sum_{k=1}^c \sigma_k^a} \cdot (\sigma_i^a + b)}. \end{aligned}$$

That is, each cluster size is increased by the value of the parameter b and then the sizes are renormalized so that the sum of the cluster sizes is preserved. However, the parameter s may be used to scale the sum of the sizes up or down (by default $s = 1$). For $b \to \infty$ the cluster sizes are equalized completely, for $b = 0$ only the parameter s has an effect. This method is inspired by Laplace correction or Bayesian estimation with an uniformative prior (see below).

Method 2: This method, which is meant as a simplified and thus more efficient version of method 1, does not renormalize the sizes, so that the size sum is increased by cb. However, this missing renormalization may be mitigated to some degree by specifying a value of the scaling parameter s that is smaller than 1. Formally, the equivalent isotropic radii σ_i are adapted (in every update step) according to

$$\sigma_i^{(\text{adap})} = \sqrt[a]{s \cdot (\sigma_i^a + b)}.$$

Method 3: The above methods always change the relation of the cluster sizes and thus introduce a general tendency towards clusters of equal size. In this (third) method, however, a limit r, $r > 1$, for the size ratio of the largest to the smallest cluster is used and only if this limit is exceeded, the sizes are changed in such a way that the limit is satisfied. To achieve this, b is set (in every update step) according to

$$b = \begin{cases} 0, & \text{if } \dfrac{\max_{k=1}^c \sigma_k^a}{\min_{k=1}^c \sigma_k^a} \leq r, \\ \dfrac{\max_{k=1}^c \sigma_k^a - r \min_{k=1}^c \sigma_k^a}{r - 1}, & \text{otherwise,} \end{cases}$$

and then Method 1 is executed with this value of b.

4.3 Constraining Cluster Weights

A cluster weight θ_i appears only in the mixture model approach and the FMLE algorithm, where it describes the prior probability of a cluster. For cluster weights we may use basically the same adaptation methods as for the cluster size, with the exception of the scaling parameter s, since the θ_i are probabilities, i.e., we must ensure $\sum_{i=1}^c \theta_i = 1$. Therefore we have:

Method 1: The cluster weights θ_i are adapted (in every update step) according to

$$\theta_i^{(\text{adap})} = \frac{\sum_{k=1}^c \theta_k}{\sum_{k=1}^c (\theta_k + b)} \cdot (\theta_i + b) = \frac{\sum_{k=1}^c \theta_k}{cb + \sum_{k=1}^c \theta_k} \cdot (\theta_i + b),$$

where b is a parameter that is to be specified by a user. Note that this method is equivalent to a Laplace corrected estimation of the prior probabilities or a Bayesian estimation with an uninformative (uniform) prior.

Method 2: The value of the adaptation parameter b is computed (in every update step) as

$$b = \begin{cases} 0, & \text{if } \dfrac{\max_{k=1}^c \theta_k}{\min_{k=1}^c \theta_k} \leq r, \\ \dfrac{\max_{k=1}^c \theta_k - r \min_{k=1}^c \theta_k}{r - 1}, & \text{otherwise,} \end{cases}$$

with a user-specified maximum weight ratio r, $r > 1$, and then Method 1 is executed with this value of the parameter b.

5 Experiments

We implemented all methods suggested above as part of an expectation maximization and fuzzy clustering program written by the first author of this paper and applied it to several different data sets from the UCI machine learning repository [4]. In all data sets each dimension was normalized to mean value 0 and standard deviation 1 in order to avoid any distortions that may result from different scaling of the coordinate axes.

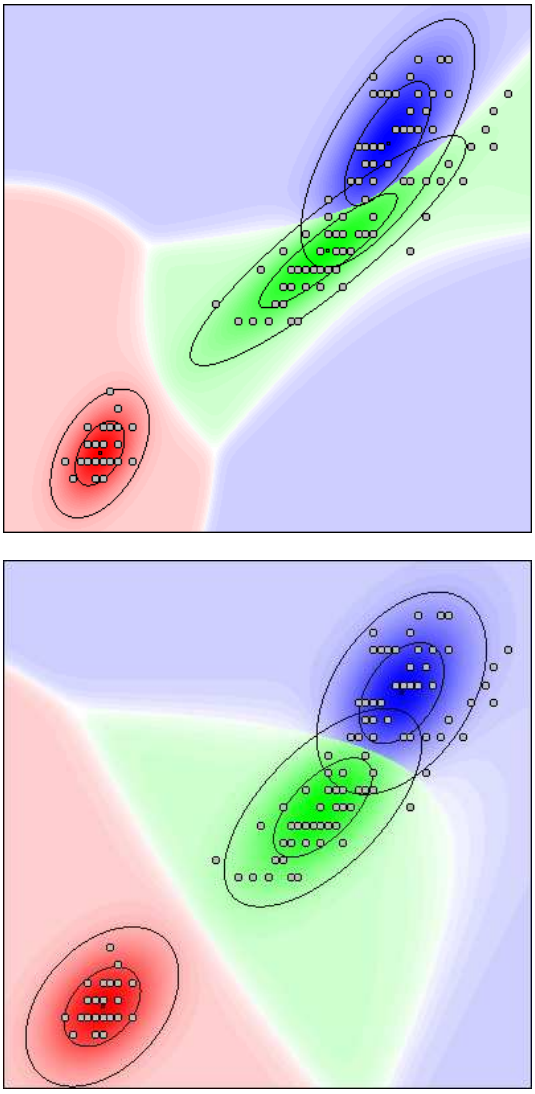

Figure 1: Result of Gustafson-Kessel algorithm on the iris data with fixed cluster size without (top) and with shape regularization (bottom, method 2 with $r = 4$). Both images show the petal length (horizontal) and the petal width (vertical). Clustering was done on all four attributes (sepal length and sepal width in addition to the above).

.

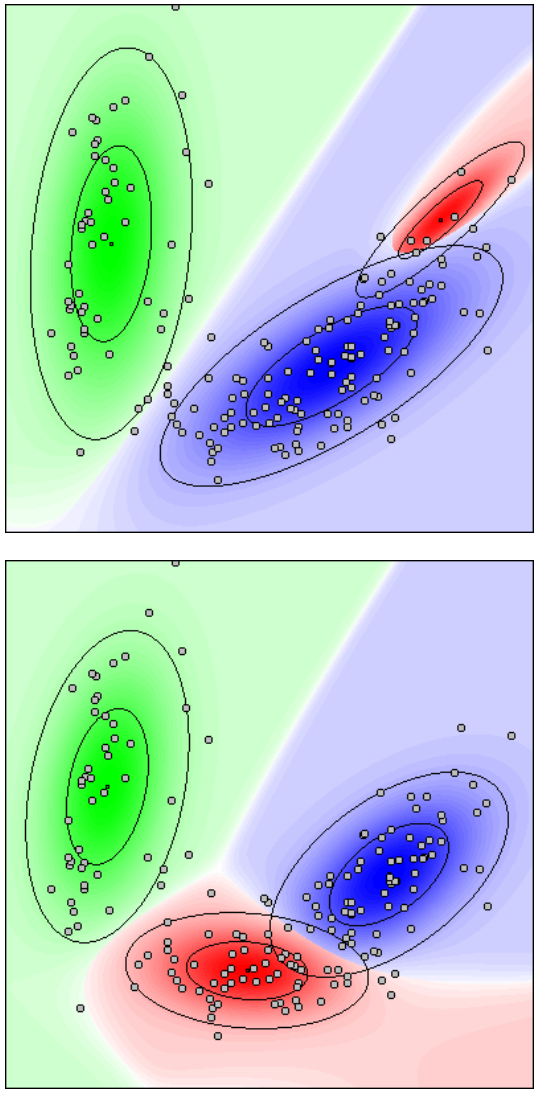

Figure 2: Result of fuzzy maximum likelihood estimation (FMLE) algorithm on the wine data with fixed cluster weight without (top) and with an adaptation of (relative) cluster sizes (bottom, method 3 with $r = 2$). Both images show attribute 7 (horizontal) and attribute 10 (vertical). Clustering was done on attributes 7, 10, and 13.

.

As one illustrative example, we present here the result of clustering the iris data (excluding, of course, the class attribute) with the Gustafson–Kessel algorithm using three cluster of fixed size (measured as the isotropic radius) of 0.4 (since all dimensions are normalized to mean 0 and standard deviation 1, 0.4 is a good size of a cluster if three clusters are to be found). The result without shape regularization is shown in Figure 1 at the top. Due to the few data points located in a thin diagonal cloud on the right border on the figure, the middle cluster is drawn into a fairly long ellipsoid. Although this shape minimizes the objective function, it may not be a desirable result, because the cluster structure is not compact enough. Using shape regularization method 2 with $r = 4$ the cluster structure shown at the bottom in Figure 1 is obtained. In this result the clusters are more compact and resemble the class structure of the data set.

As another example let us consider the result of clustering the wine data with the fuzzy maximum likelihood estimation (FMLE) algorithm using three clusters of variable size. We used attributes 7, 10, and 13, which are the most informative w.r.t. the class assignments. One result we obtained without constraining the relative cluster size is shown in Figure 2 at the top. However, the algorithm is much too unstable to present a unique result. Often enough clustering fails completely, because one cluster collapses to a single data point—an effect that is mainly due to the steepness of the Gaussian probability density function and the sensitivity of the algorithm to the initialization of the cluster parameters.

This situation is considerably improved by constraining the (relative) cluster size, a result of which (that sometimes, with a fortunate initialization, can also be achieved without) is shown at the bottom in Figure 2. It was obtained with method 3 with

$r = 2$. Although the result is still not unique and sometimes clusters still focus on very few data points, the algorithm is considerably more stable and reasonable results are obtained much more often than without size constraints. Hence we can conclude that constraining (relative) cluster size considerably improves the robustness of the algorithm.

6 Conclusions

In this paper we suggested a shape regularization method as well as methods to constrain the (relative) cluster size and weight for clustering algorithms that use a cluster-specific Mahalanobis distance to describe the shape and the size of a cluster. The basic idea is to introduce a tendency towards equal length of the major axes of the represented (hyper-)ellipsoid and towards equal cluster sizes. As the experiments show, these methods improve the robustness of the more sophisticated fuzzy clustering algorithms, which without them suffer from instabilities even on fairly simple data sets. Regularized and constrained clustering is so robust that it can even be used without an initialization by the fuzzy c-means algorithm. It should be noted that with a time-dependent shape regularization parameter one may obtain a soft transition from the fuzzy c-means algorithm (spherical clusters) to the Gustafson-Kessel algorithm (general ellipsoidal clusters).

Software

A free implementation of the described methods as command line programs for expectation maximization and fuzzy clustering (written in C) can be found at

http://fuzzy.cs.uni-magdeburg.de/
˜borgelt/software.html#cluster

References

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

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

[3] J. Bilmes. A Gentle Tutorial on the EM Algorithm and Its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. University of Berkeley, Tech. Rep. ICSI-TR-97-021, 1997

[4] C.L. Blake and C.J. Merz. UCI Repository of Machine Learning Databases. http://www.ics.uci.edu/˜mlearn/MLRepository.html

[5] H.H. Bock. *Automatische Klassifikation*. Vandenhoeck & Ruprecht, Göttingen, Germany 1974

[6] A.P. Dempster, N. Laird, and D. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society (Series B)* 39:1–38. Blackwell, Oxford, United Kingdom 1977

[7] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. J. Wiley & Sons, New York, NY, USA 1973

[8] H. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Kluwer, Dordrecht, Netherlands 1996

[9] B.S. Everitt and D.J. Hand. *Finite Mixture Distributions*. Chapman & Hall, London, UK 1981

[10] I. Gath and A.B. Geva. Unsupervised Optimal Fuzzy Clustering. *IEEE Trans. Pattern Analysis & Machine Intelligence* 11:773–781. IEEE Press, Piscataway, NJ, USA, 1989

[11] E.E. Gustafson and W.C. Kessel. Fuzzy Clustering with a Fuzzy Covariance Matrix. *Proc. 18th IEEE Conference on Decision and Control (IEEE CDC, San Diego, CA)*, 761–766, IEEE Press, Piscataway, NJ, USA 1979

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

[13] A. Keller and F. Klawonn. Adaptation of Cluster Sizes in Objective Function Based Fuzzy Clustering. In: C.T. Leondes, ed. *Database and Learning Systems IV*, 181–199. CRC Press, Boca Raton, FL, USA 2003

[14] F. Klawonn and R. Kruse. Constructing a Fuzzy Controller from Data. *Fuzzy Sets and Systems* 85:177-193. North-Holland, Amsterdam, Netherlands 1997

[15] T. Kohonen. *Learning Vector Quantization for Pattern Recognition*. Technical Report TKK-F-A601. Helsinki University of Technology, Finland 1986

[16] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, Heidelberg, Germany 1995 (3rd ext. edition 2001)

[17] R. Krishnapuram and J. Keller. A Possibilistic Approach to Clustering, *IEEE Transactions on Fuzzy Systems*, 1:98-110. IEEE Press, Piscataway, NJ, USA 1993

[18] A.N. Tikhonov and V.Y. Arsenin. Solutions of Ill-Posed Problems. J. Wiley & Sons, New York, NY, USA 1977