Pergamon

Pattern Recognition, Vol. 30, No. 7, pp. 1109–1119, 1997
© 1997 Pattern Recognition Society. Published by Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0031-3203/97 $17.00+.00

PII: S0031-3203(96)00140-9

CLUSTERING BY COMPETITIVE AGGLOMERATION

HICHEM FRIGUI and RAGHU KRISHNAPURAM*
Department of Computer Engineering and Computer Science, University of Missouri-Columbia, Columbia, MO 65211, U.S.A.

(*Received* 8 *September* 1995; *in revised form* 18 *April* 1996; *received for publication* 10 *September* 1996)

Abstract—We present a new clustering algorithm called Competitive Agglomeration (CA), which minimizes an objective function that incorporates the advantages of both hierarchical and partitional clustering. The CA algorithm produces a sequence of partitions with a decreasing number of clusters. The initial partition has an over specified number of clusters, and the final one has the "optimal" number of clusters. The update equation in the CA algorithm creates an environment in which clusters compete for feature points and only clusters with large cardinalities survive. The algorithm can incorporate different distance measures in the objective function to find an unknown number of clusters of various shapes. © 1997 Pattern Recognition Society. Published by Elsevier Science Ltd.

Unsupervised clustering   Fuzzy clustering   Competitive agglomeration
Cluster validity   Line detection   Curve detection   Plane fitting

1. INTRODUCTION

Clustering, also known as unsupervised classification, is a process by which a data set is divided into different clusters such that elements of the same cluster are as similar as possible and elements of different clusters are as dissimilar as possible. Most existing clustering algorithms can be classified into the following two categories:(1) hierarchical clustering and partitional clustering. Hierarchical clustering procedures provide a nested sequence of partitions with a graphical representation known as the dendrogram. Hierarchical procedures can be either agglomerative or divisive. Agglomerative procedures start with each sample as a cluster and form the nested sequence by successively merging clusters. Divisive procedures start with all samples in one cluster and successively split clusters. Besides providing a graphical representation of the data that can easily be interpreted, the main advantage of hierarchical procedures is that the clustering is not influenced by initialization and local minima. Moreover, the number of clusters need not be specified a priori. The user can analyze the dendrogram, select a threshold, and cut the graph at a suitable level to identify the clusters. However, for large data sets, dendrograms are impractical, and if the expected number of clusters is small, much of the computation needed to obtain a complete dendrogram can be wasteful. The main disadvantage of hierarchical clustering procedures is that they consider only local neighbors when merging/splitting clusters and they cannot incorporate a priori knowledge about the global shape or size of clusters. Moreover, hierarchical clustering is static in the sense that points which are committed to a cluster in the early stages, cannot move to another cluster, since the sequence of partitions is nested.

Partitional clustering procedures generate a single partition (as opposed to a nested sequence) of the data in an attempt to recover the natural grouping present in the data. Prototype-based clustering algorithms are the most popular class of partitional clustering methods. In prototype-based clustering algorithms, each cluster is represented by a prototype, and the sum of distances from the feature vectors to the prototypes is usually used as the objective function. These algorithms themselves can be divided into two classes: hard and fuzzy. In hard clustering, each data point is assigned to one and only one of the clusters with a degree of membership equal to 1, which assumes well defined boundaries between the clusters. This model often does not reflect the description of real data, where boundaries between subgroups might be fuzzy. In fuzzy clustering, each data point belongs to each cluster with a certain degree of membership or belonging in the interval [0,1]. Because there is no total commitment of a given point to a given cluster, fuzzy clustering algorithms require more memory storage and are computationally somewhat more expensive. However, they are less likely to get trapped in local minima of the objective function being minimized than the hard clustering algorithms. Objective-function based clustering methods have been used extensively in pattern recognition and computer vision applications.(2,3) They have the advantage of being dynamic in the sense that points can move from one cluster to another to reduce the objective function. Another advantage of prototype-based clustering algorithms is that they can incorporate the knowledge about the global shape or size of clusters by using an appropriate distance measure in the objective function. In fact, these algorithms have been extended recently to

* Author to whom correspondence should be addressed. E-mail: raghu@ece.missouri.edu.

detect lines, planes, circles, ellipses, curves and curved surfaces by simply modifying the distance measure used in the objective functions.[3–6]

The best theoretical solution to a partitional clustering problem is to find all possible partitions containing C clusters and choose the partition that minimizes the objective function. However, the enumeration of all possible partitions is impractical even for data sets of moderate sizes.[1] Therefore, as an alternative, most partitional approaches use an iterative optimization technique. One starts with a reasonable initialization, and points are moved from one cluster to another to lower the value of the objective function in each iteration. This approach is compuationally attractive and typically converges in a few iterations for reasonable data sets. However, it guarantees only a local and not a global minimum. In other words, the final partition is sensitive to the initialization. Another drawback of the partitional approach is the difficulty in determining the optimal number of clusters. The proposed solutions to this problem usually rely on validity measures, and it is difficult to devise a unique measure that takes into account the variability in cluster shape, density, and size. Moreover, these procedures are computationally expensive since they require solving the optimization problem repeatedly for different values of C.

In this paper, we introduce a new approach called Competitive Agglomeration (CA), which combines the advantages of hierarchical clustering and partitional clustering techniques. The CA algorithm minimizes a fuzzy prototype-based objective function iteratively, so that it can be used to find clusters of various shapes. The objective function is designed so that it inherits the advantages of hierarchical clustering. The CA algorithm starts by partitioning the data set into a large number of small clusters. As the algorithm progresses, adjacent clusters compete for data points, and the clusters that lose the competition gradually become depleted and vanish. Thus, as the iterations proceed, we obtain a sequence of partitions with a progressively diminishing number of clusters. The final partition is taken to have the "optimal" number of clusters from the point of view of the objective function. However, in contrast to traditional hierarchical clustering, in each iteration points can move from one cluster to another, and the sequence of partitions is not necessarily nested. Thus, the problem due to premature commitment is obviated. Moreover, since we begin the procedure with a large number of small clusters, the final result is far less sensitive to initialization and local minima. Thus, the CA algorithm combines the advantages of hierarchical clustering and partitional clustering.

The organization of the rest of the paper is as follows. In Section 2, we define the objective function for the CA approach, derive necessary conditions to update the fuzzy memberships, and present the CA algorithm. In Section 3, we illustrate the performance of the CA algorithm for the special cases of spherical, ellipsoidal, linear, and shell clusters. Finally, in Section 4 we summarize the conclusions.

2. PARTITIONAL AGGLOMERATIVE CLUSTERING

Let $X = \{\mathbf{x}_j | j = 1, \ldots, N\}$ be a set of N vectors in an n-dimensional feature space with coordinate axis labels $(x_1, \ldots, x_n)$. Let $\mathbf{B} = (\beta_1, \ldots, \beta_C)$ represent a C-tuple of prototypes each of which characterizes one of the C clusters. Each β_i consists of a set of parameters. The CA algorithm minimizes the following objective function:

$$J(\mathbf{B}, \mathbf{U}, X) = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^2 d^2(\mathbf{x}_j, \beta_i) - \alpha \sum_{i=1}^{C} \left[\sum_{j=1}^{N} u_{ij} \right]^2. \tag{1}$$

Subject to

$$\sum_{i=1}^{C} u_{ij} = 1, \quad \text{for } j \in \{1, \ldots, N\}. \tag{2}$$

In equation (1), $d^2(\mathbf{x}_j, \beta_i)$ represents the distance from feature vector $\mathbf{x}_j$ to the prototype β_i, u_{ij} represents the degree of membership of feature point $\mathbf{x}_j$ in cluster β_i, and $U = [u_{ij}]$ is a $C \times N$ matrix called a constrained fuzzy C-partition matrix. It should be noted that the number of clusters C in equation (1) is dynamically updated in the CA algorithm. The objective function in equation (1) has two components. The first component, which is similar to the Fuzzy C-Means (FCM) objective function,[7] is the sum of squared distances to the prototypes weighted by constrained memberships. This component allows us to control the shapes and sizes of the clusters and to obtain compact clusters. The global minimum of this component is achieved when the number of clusters C is equal to the number of samples N, i.e. each cluster contains a single data point. The second component in equation (1) is the sum of squares of the cardinalities of the clusters which allows us to control the number of clusters. The global minimum of this term (including the negative sign) is achieved when all points are lumped in one cluster, and all other clusters are empty. When both components are combined and α is chosen properly, the final partition will minimize the sum of intra-cluster distances, while partitioning the data set into the smallest possible number of clusters. The clusters which are depleted as the algorithm proceeds will be discarded, as explained later.

To minimize the objective function in equation (1) with respect to $\mathbf{U}$, we apply Lagrange multipliers and obtain

$$J(\mathbf{B}, \mathbf{U}; X) = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^2 d^2(\mathbf{x}_j, \beta_i) - \alpha \sum_{i=1}^{C} \left[\sum_{j=1}^{N} u_{ij} \right]^2 - \sum_{j=1}^{N} \lambda_j \left(\sum_{i=1}^{C} u_{ij} - 1 \right).$$

We then fix $\mathbf{B}$ and solve

$$\frac{\partial J}{\partial u_{st}} = 2u_{st} d^2(\mathbf{x}_t, \beta_s) - 2\alpha \sum_{j=1}^{N} u_{sj} - \lambda_t = 0,$$
$$\text{for } s \in \{1, \ldots, C\}, t \in \{1, \ldots, N\}, \tag{3}$$

to obtain an updating equation for the memberships u_{st}. Equation (3) is a set of $N \times C$ linear equations with $N \times C + N$ variables (u_{st} and λ_t). Using them in conjunction with the N equations resulting from the constraint in equation (2), one can solve for the $N \times C + N$ variables. However, the solution can be simplified considerably by assuming that the membership values do not change significantly from one iteration to the next and by computing the term $\sum_{j=1}^{N} u_{sj}$ in equation (3) using the membership values from the previous iteration. With this assumption, equation (3) reduces to

$$u_{st} = \frac{2\alpha \times N_s + \lambda_t}{2d^2(\mathbf{x}_t, \beta_s)}, \qquad (4)$$

where

$$N_s = \sum_{j=1}^{N} u_{sj} \qquad (5)$$

is the cardinality of cluster s. Using equation (4) and the constraint in equation (2), we obtain

$$\sum_{k=1}^{C} \frac{2\alpha \times N_k + \lambda_t}{2d^2(\mathbf{x}_t, \beta_k)} = 1,$$

or

$$\alpha \sum_{k=1}^{C} \frac{N_k}{d^2(\mathbf{x}_t, \beta_k)} + \lambda_t \sum_{k=1}^{C} \frac{1}{2d^2(\mathbf{x}_t, \beta_k)} = 1. \qquad (6)$$

Solving equation (6) for λ_t, we obtain

$$\lambda_t = \frac{1 - \alpha \sum_{k=1}^{C} [N_k / d^2(\mathbf{x}_t, \beta_k)]}{\sum_{k=1}^{C} [1/d^2(\mathbf{x}_t, \beta_k)]}. \qquad (7)$$

Substituting equation (7) in equation (4), we obtain the following update equation for the membership of feature point $\mathbf{x}_t$ in cluster β_s:

$$u_{st} = \alpha \frac{N_s}{d^2(\mathbf{x}_t, \beta_s)} + \frac{1 - \alpha \sum_{k=1}^{C} [N_k / d^2(\mathbf{x}_t, \beta_k)]}{\sum_{k=1}^{C} [d^2(\mathbf{x}_t, \beta_s)/d^2(\mathbf{x}_t, \beta_k)]},$$

or rearranging the terms, we obtain

$$u_{st} = u_{st}^{\mathrm{FCM}} + u_{st}^{\mathrm{Bias}}, \qquad (8)$$

where

$$u_{st}^{\mathrm{FCM}} = \frac{[1/d^2(\mathbf{x}_t, \beta_s)]}{\sum_{k=1}^{C} [1/d^2(\mathbf{x}_t, \beta_k)]}, \qquad (9)$$

and

$$u_{st}^{\mathrm{Bias}} = \frac{\alpha}{d^2(\mathbf{x}_t, \beta_s)} (N_s - \bar{N}_t). \qquad (10)$$

In equation (10), $\bar{N}_t$ is defined as

$$\bar{N}_t = \frac{\sum_{k=1}^{C} [1/d^2(\mathbf{x}_t, \beta_k)] N_k}{\sum_{k=1}^{C} [1/d^2(\mathbf{x}_t, \beta_k)]},$$

which is simply a weighted average of the cluster cardinalities, where the weight of each cluster reflects its proximity to the feature point $\mathbf{x}_t$ in question.

As seen from equation (9), u_{st}^{FCM} in equation (8), is the well-known membership term in the FCM algorithm which takes into account only the relative distances of the feature point to all clusters. The second component in equation (8), u_{st}^{Bias}, is a signed bias term which depends on the difference between the cardinality of the cluster of interest, and the weighted average of cardinalities from the point of view of feature point $\mathbf{x}_t$ [see equation (10)]. For clusters with cardinality higher than average, the bias term is positive, thus appreciating the membership value. On the other hand, for low cardinality clusters, the bias term is negative, thus depreciating the membership value. Moreover, this bias term is also inversely proportional to the distance of feature point $\mathbf{x}_t$ to the cluster of interest β_s, which serves as an amplification factor. For example, the membership values of data points in spurious (low cardinality) clusters are depreciated heavily when their distances to such clusters are low. This leads to a gradual reduction of the cardinality of spurious clusters. When the cardinality of a cluster drops below a threshold, we discard the cluster, and update the number of clusters. Since the initial partition has an overspecified number of clusters, each cluster is approximated by many small clusters in the beginning. As the algorithm proceeds, the second term in equation (1) causes each cluster to expand and include as many points as possible. At the same time, the constraint in equation (2) causes adjacent clusters to compete. As a result, only a few clusters will survive, while others will shrink and eventually become extinct.

It should be noted that when a feature point $\mathbf{x}_j$ is close to only one cluster (say cluster i), and far from other clusters, we have

$$N_i \approx \bar{N}_j, \quad \text{or} \quad u_{ij}^{\mathrm{Bias}} \approx 0.$$

In this case the membership value, u_{ij}, is independent of the cluster cardinalities, and reduces to u_{ij}^{FCM}. In other words, if a point is close to only one cluster, it will have high membership value in this cluster and no competition is involved. On the other hand, if a point is close to many clusters, these clusters will compete for this point based on cardinality. This encourages the formation of larger clusters, i.e. it promotes agglomeration.

The choice of α in equation (1) is important in the CA algorithm since it reflects the importance of the second term relative to the first term. If α is too small, the second term will be neglected and the number of clusters will not be reduced. If α is too large, the first term will be neglected, and all points will be lumped into just one cluster. The value of α should be chosen such that both terms are of the same order of magnitude. As the dimensionality of feature space increases, the first term becomes larger since more components contribute to the value of the distance. Thus, to make the algorithm independent of the distance measure, α should be proportional to the ratio of the two terms. That is

$$\alpha \propto \frac{\sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^2 d^2(\mathbf{x}_j, \beta_i)}{\sum_{i=1}^{C} [\sum_{j=1}^{C} u_{ij}]^2}.$$

Intuitively, it is better to start with a large value of α so that the second term of the objective function dominates. This will cause the number of clusters to dwindle rapidly.

The value of α can be decreased slowly in each iteration to help the CA algorithm to seek an appropriate partition with a number of clusters that is close to the "optimum" in the first few iterations. As α decreases, the algorithm will strive to refine the initial approximate partition by emphasizing the first term in the objective function.

In all examples described in this paper, we choose α to be

$$\alpha(k) = \eta(k)\frac{\sum_{i=1}^{C}\sum_{j=1}^{N}(u_{ij})^2 d^2(\mathbf{x}_j, \beta_i)}{\sum_{i=1}^{C}[\sum_{j=1}^{N} u_{ij}]^2}. \quad (11)$$

In equation (11), α and η are functions of the iteration number k. Note that α is not confined to [0,1] in general. A good choice for η is the exponential decay defined by

$$\eta(k) = \eta_0 \exp(-k/\tau), \quad (12)$$

where η_0 is the initial value and τ the time constant. (Here time is measured in iterations k.) In all the experiments presented in this paper, we choose $\alpha(k)$ in iteration k according to equations (11) and (12), with η_0=5 and τ=10.

It should be noted that, depending on the value of α the membership values, u_{ij} may not be confined to [0, 1]. In fact, u_{ij} can become negative if N_i is very small and point $\mathbf{x}_j$ is close to other dense clusters (i.e. $\bar{N}_j$ is high). This generally implies that cluster β_i is spurious. In this case, it is safe to set u_{ij} to zero to indicate that feature vector $\mathbf{x}_j$ is atypical of cluster β_i. It is also possible for u_{ij} to become larger than 1 if N_i is very large and feature point $\mathbf{x}_j$ is close to other spurious clusters (i.e. $\bar{N}_j$ is low). In this case it is safe to clip u_{ij} to 1 to indicate that feature vector $\mathbf{x}_j$ is typical of cluster, β_i.

Minimization of equation (1) with respect to the prototypes to derive the update equation for $\mathbf{B} = (\beta_1, \ldots, \beta_C)$ varies according to the choice of the prototypes and the distance measure. Each choice leads to a different algorithm. Since the second term in equation (1) does not depend on the prototypes and the distance measure, the parameter update equations in the CA algorithm are the same as those in traditional objective function based fuzzy clustering algorithms that minimize the sum of squared distances.

The CA algorithm is summarized below

The Competitive Agglomeration Algorithm

```
Fix the maximum number of clusters $C$=$C_{\max}$;
Initialize iteration counter $k$=0;
Initialize the fuzzy $C$ partition $\mathbf{U}^{(0)}$;
Compute the initial cardinalities $N_i$ for $1 \le i \le C$ by using equation (5);
Repeat
    Compute $d^2(\mathbf{x}_j, \beta_i)$ for $1 \le i \le C$, $1 \le j \le N$;
    Update $\alpha(k)$ by using equations (11 and 12);
    Update the partition matrix $\mathbf{U}^{(k)}$ by using equation (8);
    Compute the cardinality $N_i$ for $1 \le i \le C$ by using equation (5);
    If ($N_i < \epsilon_1$) discard cluster $\beta_i$;
    Update the number of clusters $C$;
    Update the prototype parameters;
    $k$=$k$+1;
Until (prototype parameters stabilize).
```

3. EXPERIMENTAL RESULTS

In this section we illustrate the effectiveness of the proposed unsupervised approach for the special cases of spherical, ellipsoidal, linear, and shell clusters. We use examples involving both real and synthetic images. In all examples shown in this section, a cluster β_i was discarded if its cardinality (N_i) is less than 5. As mentioned in Section 2, the initial value of η (η_0) was set to 5 and the time constant τ was set to 10. The initial partition was obtained by running the FCM algorithm[7] (with the Euclidean distance measure and m=2) in the first five iterations. In the next five iterations, the distance measure is changed depending on the type of clusters being sought. These distance measures are discussed in Sections 3.1–3.4. In our experience, the CA algorithm is independent of the initial number of clusters $C_{\max}$ as long as $C_{\max}$ is chosen to be much larger than the expected number of clusters. Choosing a large value of $C_{\max}$ allows the clusters to be approximated by many prototypes at the beginning. In all examples, we tried many different values for $C_{\max}$ ranging from 15 to 30, and in all cases, the algorithm converged to the same final partition. If $C_{\max}$ is chosen to be comparable to the actual number of clusters, the chances of converging to local minima increase. In this paper we display the intermediate partitions of the algorithm when $C_{\max}$ is chosen to be 20. We now discuss the distance measures and prototype parameter update equations for various types of clusters.

3.1. Detecting spherical clusters

The Euclidean distance given by

$$d^2(\mathbf{x}_j, \beta_i) = d^2_{Eij} = \|\mathbf{x}_j - \mathbf{c}_i\|^2, \quad (13)$$

can be used in the CA algorithm to seek spherical clusters. In equation (13), the prototypes $\mathbf{c}_i$ are the cluster centers. It can be easily shown that the necessary condition for the minimization of equation (1) with respect to $\mathbf{B}$ yields the following equation for updating the centers:

$$\mathbf{c}_i = \frac{\sum_{j=1}^{N}(u_{ij})^2 \mathbf{x}_j}{\sum_{j=1}^{N}(u_{ij})^2}. \quad (14)$$

In Fig. 1 we illustrate the performance of the CA algorithm when the distance measure used is d^2_{Eij}. Figure 1(a) is a synthetic data set generated using a two-dimensional Gaussian random number generator. This data set consists of four clusters of various sizes and densities. Figure 1(b) shows the prototype parameters of the initial partition. The prototypes' (centers) locations are shown as "+" symbols superimposed on the data set. As can be seen in this figure, large clusters are split into many clusters and small clusters are split into fewer number of

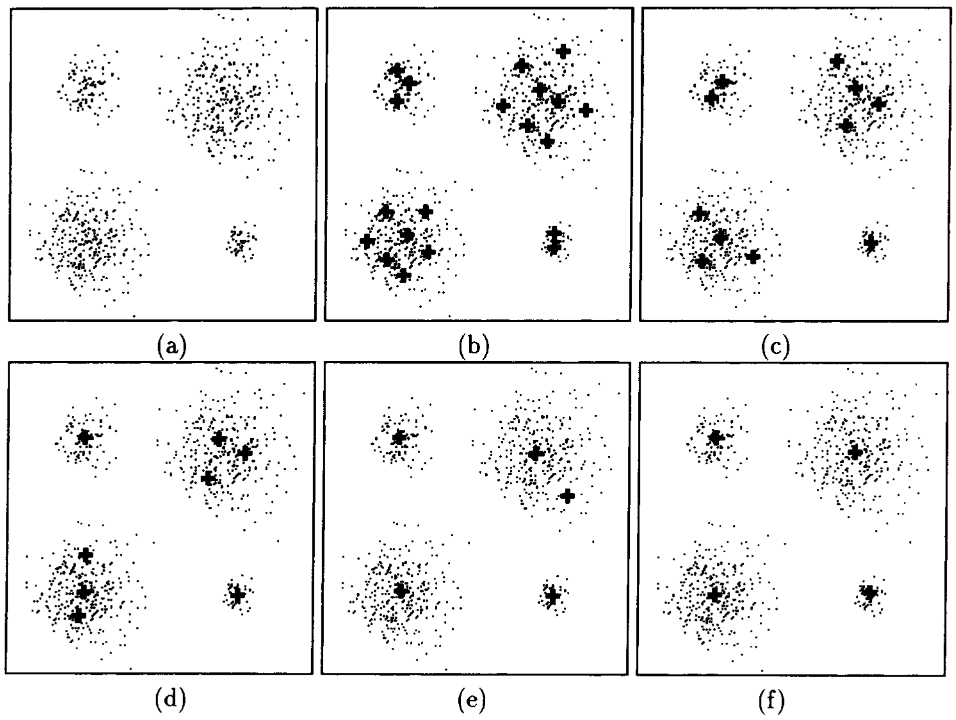


Fig. 1. Intermediate results of the CA algorithm on a data set with spherical clusters. (a) Original image, (b) prototypes used for initialization. Results at the end of (c) two iterations, (d) three iterations, (e) five iterations, and (f) 10 iterations (convergence).

clusters. Figure 1(c) shows the result after running two iterations of the CA algorithm. The number of clusters is reduced to 11 after nine empty clusters are discarded. After the third iteration, another three empty clusters are discarded as shown in Fig. 1(d). Figure 1(e) shows the result after five iterations, where the number of clusters is reduced to five. After a total of seven iterations, the number of clusters becomes four and after three more iterations, the algorithm converges. The final result is shown in Fig. 1(f).

3.2. Detecting ellipsoidal clusters

To detect ellipsoidal clusters, we use the following distance measure proposed by Sebestyen and Roemder[8] [and later independently derived by Gustafson and Kessel[9]]

$$d^2(\mathbf{x}_j,\beta_i) = d^2_{Cij} = |\mathbf{C}_i|^{1/n}(\mathbf{x}_j - \mathbf{c}_i)^{\mathrm{T}}\mathbf{C}_i^{-1}(\mathbf{x}_j - \mathbf{c}_i), \quad (15)$$

where $\mathbf{c}_i$ is the center of cluster β_i, and $\mathbf{C}_i$ is its covariance matrix. Since the second term in equation (1) does not depend explicitly on the prototype parameters, β_i=($\mathbf{c}_i$,$\mathbf{C}_i$), the necessary conditions for the minimization of equation (1) with respect to **B** are the same as those in the Gustafson–Kessel[9] (G–K) algorithm (with m=2), i.e. the centers are updated as in equation (14), and the covariance matrices are updated using

$$\mathbf{C}_i = \frac{\sum_{j=1}^{N}(u_{ij})^2(\mathbf{x}_j - \mathbf{c}_i)(\mathbf{x}_j - \mathbf{c}_i)^{\mathrm{T}}}{\sum_{j=1}^{N}(u_{ij})^2}. \quad (16)$$

In Fig. 2 we illustrate the performance of the CA algorithm when the distance measure used is d^2_{Cij}. Fig. 2(a) shows a synthetic data set containing six Gaussian clusters of various sizes, shapes, and orientations. The total number of points in this data set is 1100. Fig. 2(b) shows the initial prototypes. As before, the "+" signs indicate the cluster centers, and the ellipses shown in the figure enclose points having a Mahalanobis distance less than 4. After performing two iterations of the CA algorithm, three empty clusters are discarded. The prototypes of the remaining 17 clusters are shown in Fig. 2(c). The number of clusters is reduced to 10 after three iterations and to seven after five iterations. These intermediate results are shown in Fig. 2(d) and (e) respectively. After a total of seven iterations, the number of clusters is reduced to six. As the algorithm proceeds for five more iterations, the number of clusters remains six, but the prototypes become more representative of the data. The final result is shown in Fig. 2(f). The total CPU time for this experiment was 1 min 11 s on a Sun SPARC LX.

To compare our approach with the traditional approach of obtaining the optimal number of clusters, we ran a second experiment on the same data set in which the G–K algorithm was applied repeatedly to every value of C between 1 and 20. The average partition density[10] was used to evaluate the validity of the partition and for each value of C. Figure 3 shows the graph of the average partition density versus the number of clusters. As can be seen, the optimal partition (i.e. the partition that maximizes the chosen validity measure) consists of six

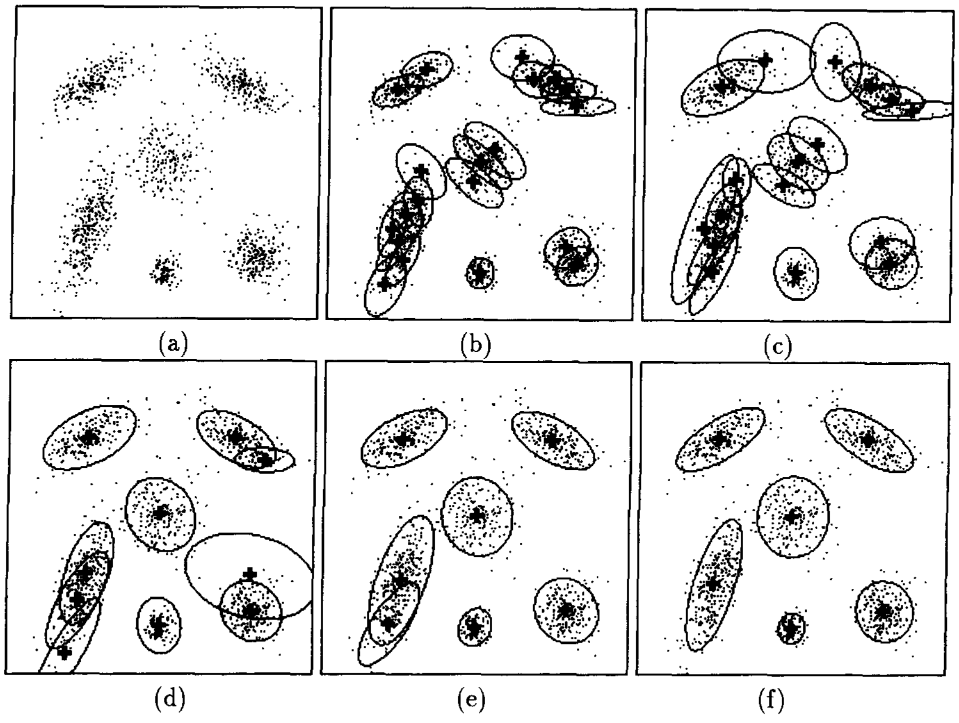


Fig. 2. Intermediate results of the CA algorithm on a data set with ellipsoidal clusters. (a) Original image, (b) prototypes used for initialization. Results at the end of (c) two iterations, (d) three iterations, (e) five iterations, and (f) 12 iterations (convergence).

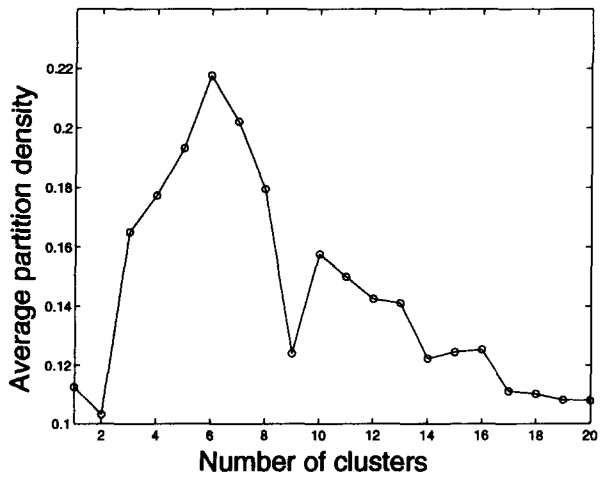


Fig. 3. Average partition density versus the number of clusters.

clusters. The CPU time required to perform this experiment was 25 min 10 s. Thus the computational advantage of the CA approach is obvious.

3.3. *Detecting linear clusters*

To detect clusters that resemble lines or planes, we use a generalization of the distance measure proposed by Bezdek *et al.*[3] and Davé.[4] The generalized distance is given by

$$d^2(\mathbf{x}_j, \beta_i) = d_{Lij}^2 = \sum_{k=1}^{n} \nu_{ik}[(\mathbf{x}_j - \mathbf{c}_i) \cdot \mathbf{e}_{ik}]^2, \quad (17)$$

where $\mathbf{e}_{ik}$ is the kth unit eigenvector of the covariance matrix $\mathbf{C}_i$. The eigenvectors are assumed to be arranged in descending order of the corresponding eigenvalues. The value of ν_{ik} in equation (17) is chosen dynamically, i.e.

$$\nu_{ik} = \frac{\lambda_{in}}{\lambda_{ik}},$$

where ν_{ik} is the kth eigenvalue of the covariance matrix $\mathbf{C}_i$. The resulting algorithm, uses the distance measure defined in equation (17), and updates the centers, and the covariance matrices 12 using equations (14) and (16) respectively.

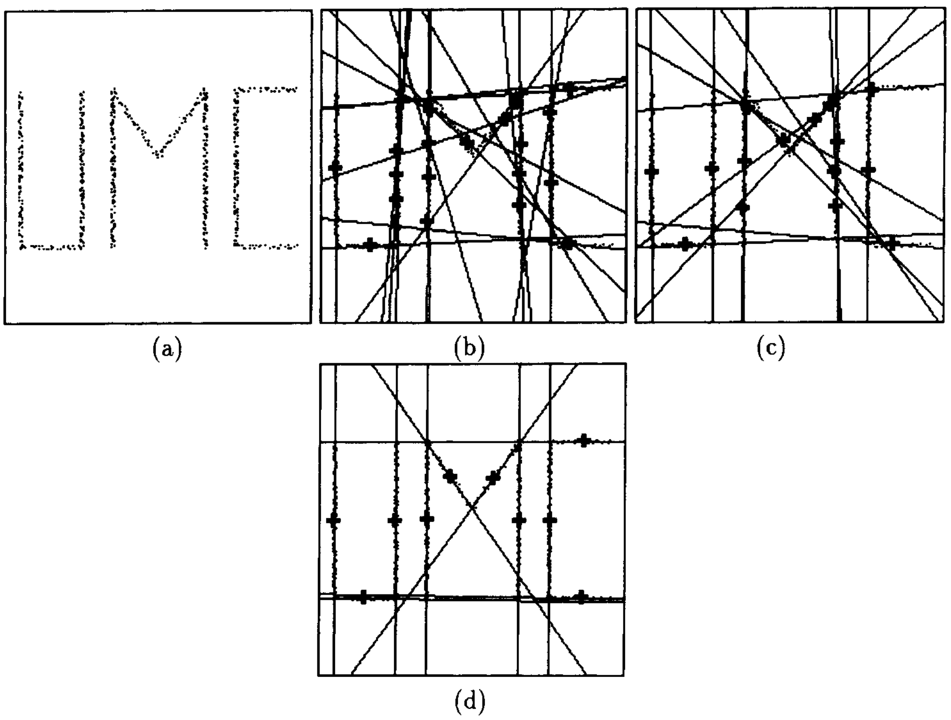


Fig. 4. Intermediate results of the CA algorithm on a data set with linear clusters. (a) Original image, (b) initial prototypes, (c) results after two iterations, and (d) final results.

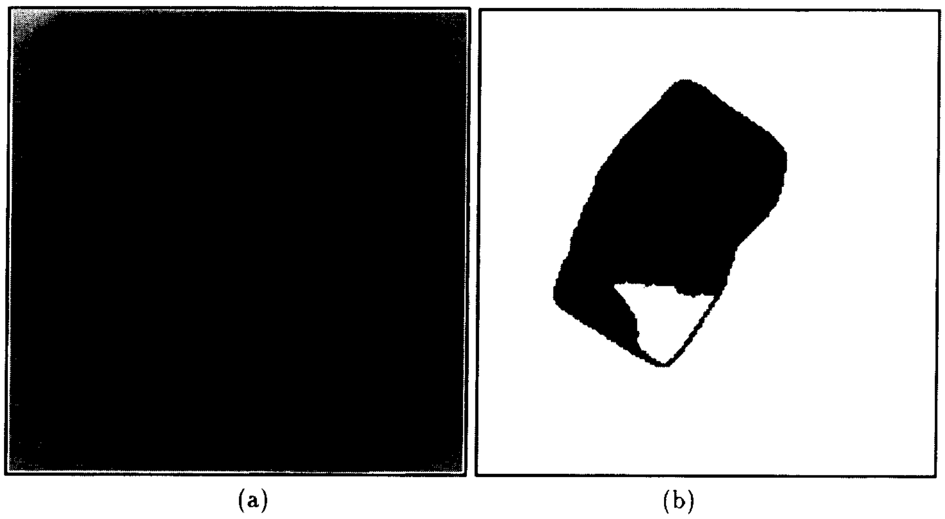


Fig. 5. Planar surface fitting using CA algorithm. (a) Original range image of a block and (b) planar surfaces obtained.

In Figs 4–6, we use d_{Lij}^2 as defined in equation (17) in the CA algorithm, and show that it can be used to find the optimal number of lines/planes in a given data set. Figure 4(a) shows an image consisting of 10 line segments. This data set consists of 605 points. Figure 4(b) shows the initial prototypes superimposed on the data set. After running the CA algorithm for two iterations, the number of clusters is cut to 15 as shown in Fig. 4(c) (some of the prototypes are almost identical and appear as just one prototype in the figure). After a total of five iterations, the number of clusters is reduced to the "optimal" number and the algorithm converges at the end of the seventh iteration. The final result is shown in Fig. 4(d). This experiment took only 45 s of CPU time. In comparison, estimating the same optimal number of linear clusters using the Compatible Cluster Merging algorithm(12) (starting with $C_{\max}=20$) required about 4 min.

Figure 5(a) shows a real 200 × 200 range image of a block obtained from the ERIM. The results of the CA algorithm are displayed in Fig. 5(b) which consists of the

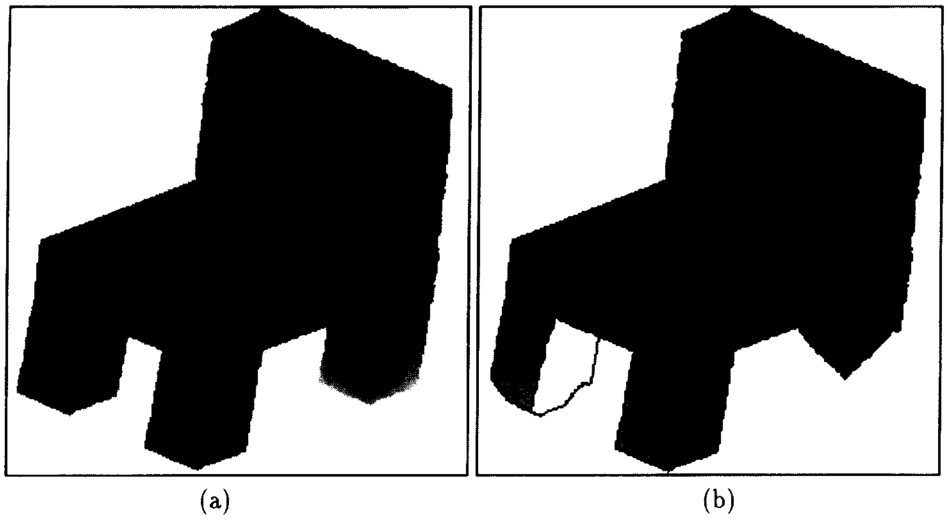


Fig. 6. Planar surface fitting using CA algorithm. (a) Original range image of a chair and (b) planar surfaces obtained.

correct number of planar surfaces. Each surface is displayed with a different gray value. A sampling rate of two in the x- and y-directions was used to reduce computations. It took the CA algorithm about 20 iterations to find the correct number of planar surfaces.

Figure 6(a) shows a real 252 × 263 range image of a chair obtained from the University of Southern California. The results of the CA algorithm are displayed in Fig. 6(b). A sampling rate of three in the x- and y-directions was used in this example. The CA algorithm converged in about 30 iterations.

3.4. *Detecting quadric shell clusters*

To detect shell clusters (i.e. clusters with no interior points), we use quadric curves for prototypes and measure the distances to the curves rather than to the cluster centers. The equation of a quadric curve is given by

$$\mathbf{x}_j^{\mathrm{T}}\mathbf{A}_i\mathbf{x}_j + \mathbf{x}_j^{\mathrm{T}}\mathbf{v}_i + b_i = 0,$$

where $\mathbf{A}_i$ is a symmetric matrix. The prototype β_i consists of $(\mathbf{A}_i,\mathbf{v}_i,b_i)$ and the distance from a point $\mathbf{x}_j$ to a prototype

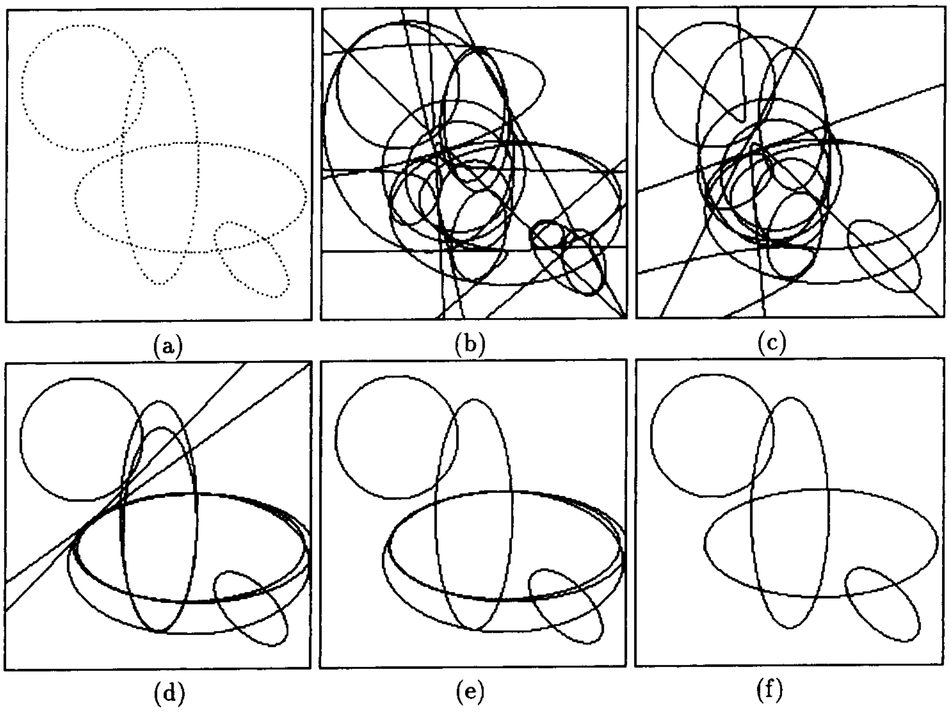


Fig. 7. Intermediate results of the CA algorithm on a data set with shell clusters. (a) Original image, (b) prototypes used for initialization. Results at the end of (c) one iteration, (d) seven iterations, (e) 10 iterations, and (f) 12 iterations (convergence).

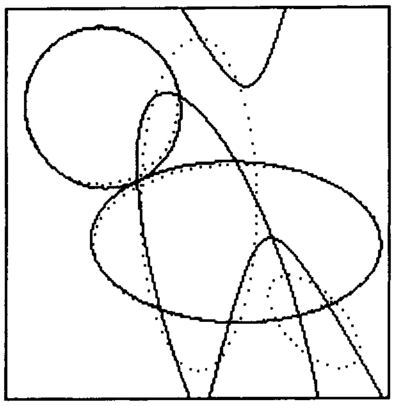


Fig. 8. Results of the FCQS algorithm with C=4.

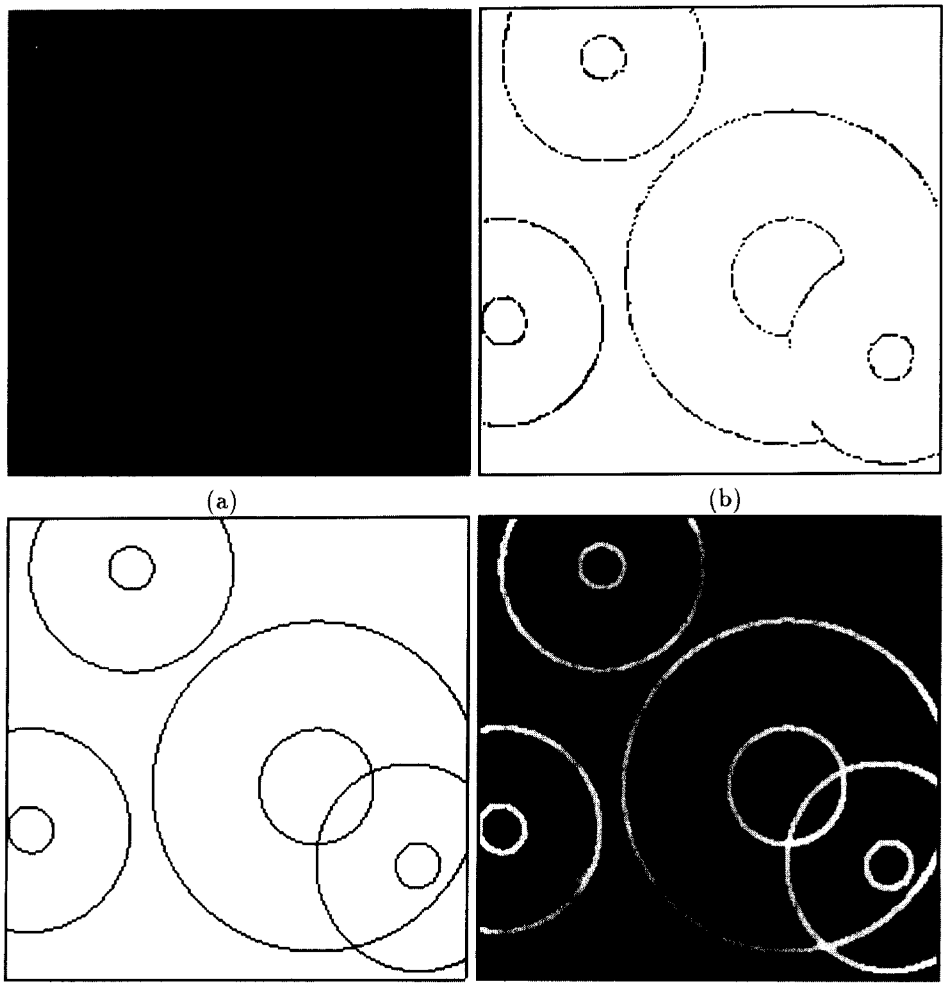
β_i is defined as[6,11]

$$d^2_{Pij} = \min \|\mathbf{x}_j - \mathbf{z}\|^2 \text{ such that } (\mathbf{z}^T\mathbf{A}_i\mathbf{z} + \mathbf{z}^T\mathbf{v}_i + b_i) = 0. \quad (18)$$

For the sake of brevity, we do not include the parameter update equations. The reader can refer to the papers by Krishnapuram *et al.*[6,11] for more details.

In Figs 7 and 8 we illustrate how our CA algorithm can be used to find the optimal number of shell clusters. Figure 7(a) shows a synthetic image consisting of three overlapping ellipses of various sizes and orientations and a circle. Figure 7(b) shows the 20 initial prototypes. In just one iteration, four clusters become empty and are discarded. The remaining 16 prototypes are shown in Fig. 7(c). As can be seen, some of these clusters are good and the others are spurious. The number of clusters is reduced to eight at the end of seven iterations, and to six after three more iterations. These intermediate results are shown in Fig. 7(d) and (e). Finally, after a total of 12 iterations, the algorithm converges to the optimal number of clusters. The final result is shown in Fig. 7(f).

Shell clustering is known to be very sensitive to initialization, especially when the number of clusters is large.[11] This sensitivity is illustrated in Fig. 8 where the FCQS algorithm[11] is applied with the correct num-



(a) (b) (c) (d)

Fig. 9. (a) Original image containing circular objects, (b) thinned edge image, (c) prototypes found by the CA algorithm, and (d) prototypes superimposed on the original image.

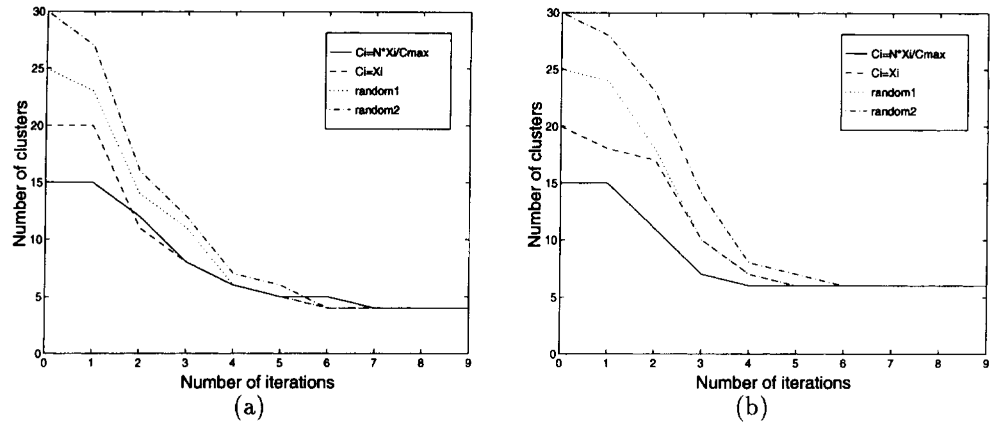


Fig. 10. Effect of initial number of clusters $C_{\max}$ on the final results (a) for the data set in Fig. 1(a) and (b) for the data set in Fig. 2(a).

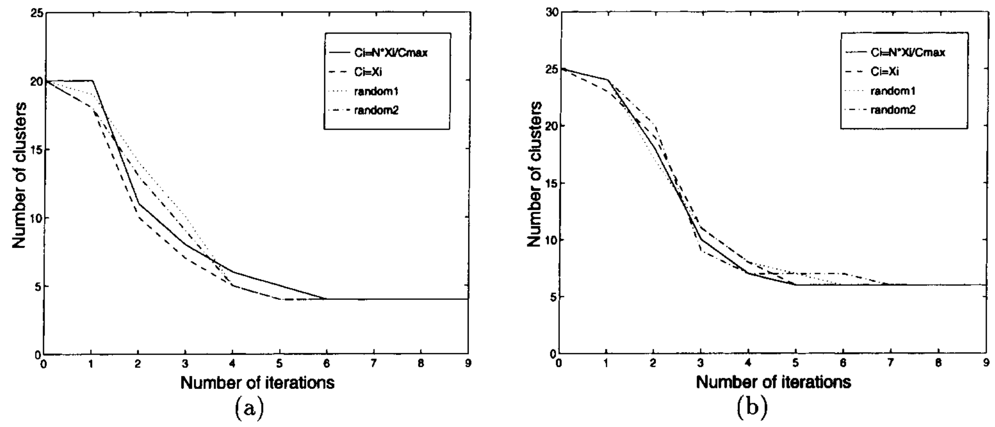


Fig. 11. Effect of the choice of the initial prototypes on the final results (a) for the data set in Fig. 1(a) and (b) for the data set in Fig. 2(a).

ber of clusters (C=4). As can be seen, the algorithm converges to a local minimum consisting of a circle, an ellipse, a parabola, and a hyperbola. Thus, the advantages of the CA algorithm with respect to local minima are obvious.

Figure 9(a) shows a real 200 × 200 image of circular objects. Figure 9(b) shows the thinned edge image that was used as input to the CA algorithm. Figure 9(c) shows the final prototypes found by the algorithm. It took the CA algorithm about 30 iterations to find the correct number of curves. The final prototypes are shown in Fig. 9(d) superimposed on the original image.

3.5. *Effect of the initial number of clusters*

To illustrate the independence of the CA algorithm from the choice of initial number of clusters $C_{\max}$, we present the results of the CA algorithm on two different data sets with various initial number of clusters each time. For the first data set shown in Fig. 1(a), we used the Euclidean distance as the distance measure, and we let $C_{\max}$ take the values 15, 20, 25, and 30. In all cases the algorithm converged to the same final partition [shown in Fig. 1(g)]. Figure 10(a) shows the reduction of the number of clusters in each iteration for the different values of $C_{\max}$. It can be noticed that the intermediate steps vary. However, in all cases it took the CA algorithm less than seven iterations to find the optimal number of clusters. Similar results for the data set shown in Fig. 2(a) with the distance measure defined in equation (15) were obtained. These results are shown in Fig. 10(b).

These experiments illustrate the insensitivity of the CA algorithm to the initial number of clusters, and the fact that the CA algorithm converges in a few iterations regardless of the initial number of clusters $C_{\max}$, as long as $C_{\max}$ is considerably higher than the expected number of clusters.

3.6. *Effect of the initial prototypes*

To illustrate the insensitivity of the CA algorithm to the initial prototypes, we present the results of the CA algorithm on the data sets shown in Figs 1(a) and 2(a) with $C_{\max} = 20$ and 25 respectively, and with different choices for the initial prototypes. The initial centers were selected in four different ways. In the first experiment, the ith center was chosen to be the $[(N/C_{\max}) \times i]$th data point. In the second experiment, the $C_{\max}$ initial centers were chosen to be the first $C_{\max}$ data points. In the final two experiments, the centers were chosen randomly. For the data set in Fig. 2(a), all covariance matrices were initialized to the identity matrix.

Figures 11(a) and 11(b) show the evolution of the algorithm versus the number of iterations for the data sets shown in Figs 1(a) and 2(a) respectively. In both cases the algorithm converged to the same optimal partition [shown in Figs 1(g) and 2(f)] regardless of its initialization.

4. CONCLUSION

In this paper, we have presented a new unsupervised and computationally attractive clustering algorithm. Our algorithm minimizes a fuzzy prototype-based objective function, and produces a sequence of partitions with a decreasing number of clusters. The initial partition has an overspecified number of clusters, and the final one has the optimal number of clusters. Constrained fuzzy memberships are used to force the clusters to compete for data points and reduce the number of clusters from one iteration to the next. Our algorithm uses a membership update equation that takes into account relative cluster densities, as well as the relative distances to the cluster prototypes. To elaborate, the CA membership consists of two components. The first component is the traditional FCM membership which takes into account only relative proximities between points and cluster prototypes. The second component is a signed bias term. This bias is negligible for a feature point that is close to just one cluster. This is the case of no competition. On the other hand, if a point is close to multiple clusters, then the bias will be positive for the clusters with the largest cardinalities, and negative for sparse ones. This bias term creates a competitive environment that promotes the survival of the largest clusters, and the extinction of spurious ones. Since the CA algorithm is formulated independently of the distance measure, it can be used to find the optimal number of clusters for the special cases of spherical, ellipsoidal, linear, and shell clusters. The proposed algorithm is also relatively insensitive to initialization and local minima effects when compared with traditional objective-function based clustering algorithms. This is due to its agglomerative property, i.e. each cluster is initially approximated by many prototypes. In all the examples shown, the CA algorithm converges in less than 30 iterations. This makes our algorithm computationally attractive.

REFERENCES

1. A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Prentice-Hall, Englewood Cliffs, New Jersey (1988).
2. R. Krishnapuram and J. Keller, Fuzzy and possibilistic clustering methods for computer vision, in *Neural and Fuzzy Systems*, S. Mitra, M. Gupta and W. Kraske, eds, 133–159 (1994).
3. J. C. Bezdek, C. Coray, R. Gunderson and J. Watson, Detection and characterization of cluster substructure: II. Fuzzy c-varieties and convex combination thereof, *SIAM J. Appl. Math* **40**, 358–372 (1981).
4. R. N. Davé, Use of the adaptive fuzzy clustering algorithm to detect lines in digital images, *Intell. Robots Comput. Vision VIII* **1192**, 600–611 (1989).
5. R. N. Davé, K. Bhaswan, Adaptive fuzzy C-shells clustering and detection of ellipses, *IEEE Trans. Neural Networks* **14**, 643–662 (1992).
6. R. Krishnapuram, H. Frigui and O. Nasraoui, Quadratic shell clustering algorithms and their applications, *Pattern Recognition Lett.* **14**, 545–552 (1993).
7. J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York (1981).
8. G. S. Sebestyen, *Decision-Making Process in Pattern Recognition*. Macmillan, New York (1962).
9. E. E. Gustafson and W. C. Kessel, Fuzzy clustering with a fuzzy covariance matrix, *IEEE CDC*, San Diego, California, pp. 761–766 (1979).
10. I. Gath and A. B. Geva, Unsupervised optimal fuzzy clustering, *IEEE Trans. Pattern Analysis Mach. Intell.* **11**, 773–781 (1989).
11. R. Krishnapuram, H. Frigui and O. Nasraoui, Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation—Part I, *IEEE Trans. Fuzzy Systems* **3**, 29–43 (1995).
12. R. Krishnapuram and C. P. Freg, Fitting an unknown number of lines and planes to image data through compatible cluster merging, *Pattern Recognition* **25**, 385–400 (1992).

About the Author—HICHEM FRIGUI received the B.S. degree in Electrical and Computer Engineering in 1990 (with honors), and the M.S. degree in Electrical Engineering in 1992, both from the University of Missouri, Columbia. From 1992 to 1994 he worked as a Software Engineer with IDEE, Tunis, where he participated in the development of banking software applications. He is currently pursuing the Ph.D. degree in Electrical Engineering at the University of Missouri, Columbia. His current research interests include pattern recognition, computer vision, neural networks, applications of fuzzy set theory and robust methods.

About the Author—RAGHU KRISHNAPURAM received his B.Tech. degree in Electrical Engineering from the Indian Institute of Technology, Bombay, in 1978. He was at Bush India Ltd., Bombay, as a Design Engineer for one year, and then later at Bharat Electronics Ltd., Bangalore, India, as a Deputy Engineer for three years. He obtained his M.S. degree in Electrical Engineering from Louisiana State University, Baton Rouge, in 1985 and his Ph.D. degree in Electrical and Computer Engineering from Carnegie Mellon University, Pittsburgh, in 1987. Dr Krishnapuram joined the University of Missouri, Columbia, in 1987. From January to December 1993, he was visiting the European Laboratory for Intelligent Techniques Engineering (ELITE), Aachen, Germany, as a Humboldt fellow. He is currently an Associate Professor in the Department of Computer Engineering and Computer Science, University of Missouri, Columbia. His research encompasses many aspects of computer vision, pattern recognition, fuzzy set theory, neural networks, and robust methods.