Convex-Set-Based Fuzzy Clustering

Il Hong Suh, *Member, IEEE*, Jae-Hyun Kim, and Frank Chung-Hoon Rhee, *Member, IEEE*

***Abstract*— Prototype-based methods are commonly used in cluster analysis and the results may be highly dependent on the prototype used. In this paper, we propose a two-level fuzzy clustering method that involves adaptively expanding and merging convex polytopes, where the convex polytopes are considered as a "flexible" prototype. Therefore, the dependency on the use of a specified prototype can be eliminated. Also, the proposed method makes it possible to effectively represent an arbitrarily distributed data set without *a priori* knowledge of the number of clusters in the data set. In the first level of our proposed method, each cluster is represented by a convex polytope which is described by its set of vertices. Specifically, nonlinear membership functions are utilized to determine whether an input pattern creates a new cluster or whether an existing cluster should be modified. In the second level, the expandable clusters that are selected by an intercluster distance measure are merged to improve clustering efficiency and to reduce the order dependency of the incoming input patterns. Several experimental results are given to show the validity of our method.**

***Index Terms*—Convex sets, FCM, fuzzy clustering, PCM.**

I. Introduction

CLUSTER analysis has been extensively studied in various fields of engineering [1]–[8] and the clustering methods used may be considered as either hard (crisp) or fuzzy depending on whether a pattern data belongs exclusively to a single cluster or to all clusters to different degrees. Thus, in hard clustering, a membership value of zero or one is assigned to each pattern data (feature vector), whereas in fuzzy clustering, a value between zero and one is assigned to each pattern by a membership function. In general, fuzzy clustering methods may be considered to be superior to that of its hard counterparts since they can represent the relationship between the input pattern data and clusters more naturally [2].

Many of the fuzzy clustering methods [9]–[21] are based on the fuzzy C-means (FCM) algorithm proposed by Bezdek [2] and are developed under the probabilistic constraint that the sum of memberships of a data point across all the clusters must be equal to unity. This constraint on the memberships is given to avoid the trivial solution of all memberships being equal to zero and it often gives meaningful results in applications where it is appropriate to interpret memberships as probabilities or degrees of sharing [2]. However, since memberships generated by this constraint are relative numbers, they may not be suitable for applications in which the memberships are supposed to represent typicality [20], [22], [23]. Specifically, FCM-based algorithms may suffer from some difficulties [20] given as follows: 1) some input data may have different membership values depending upon their locations in the input space even though they are equidistance from a cluster center and 2) when an input pattern is considered as a noise point (even though such a noise point is located far away from all the clusters), the data is assigned to all clusters with a membership value of $1/n$, where n denotes the number of clusters. To overcome the problem in 2), a number of algorithms such as the robust fuzzy clustering algorithm (RFCA) [13] and possibilistic C-means (PCM) [20] have been proposed, where the sum of membership values for all the clusters need not be unity; that is, a membership value of each input pattern in one cluster is determined without considering the distances between the input pattern and the other remaining clusters. But, in many FCM-based clustering algorithms, including PCM, the shape of each cluster is usually fixed to a single prototype such as a point, line, hyperellipsoid, or quadric shell, and the problem is to determine the prototype parameters such as center and radius [17]–[21]. Thus, an amorphous cluster cannot be effectively represented by applying FCM-based algorithms. As an example, consider a tear-shaped cluster $\boldsymbol{A}$ shown in Fig. 1, where the contours of the same membership values are displayed. Then, from Fig. 1(b), it is observed that if a single-point prototype is used, the membership value of input pattern $\boldsymbol{a}$ that is located outside of cluster $\boldsymbol{A}$ is considered to be the same as that of pattern $\boldsymbol{b}$, which is located inside the cluster. This is due to the assumption that cluster $\boldsymbol{A}$ can be represented by a single prototype. The problem can be partially remedied by using three-point prototypes as shown in Fig. 1(c). However, the number of clusters is usually unknown *a priori* for representing any given data set. Thus, a clustering technique that can obtain an arbitrarily shaped cluster as depicted in Fig. 1(d) is highly desired.

The fuzzy min–max clustering neural network (FMMCNN) [24]–[26] and fuzzy adaptive resonance theory (ART) [27]–[29] algorithms may be considered as candidates for such a clustering technique since they possess the ability to dynamically assign an adjustable prototype (i.e., a hyperbox) for an input data set. But as in the case of the FMMCNN method, since the shape of the prototype is constrained to a hyperbox, an arbitrarily shaped cluster may be ineffectively represented by a group of hyperboxes. For example, Fig. 2 shows that a data set may be compactly

Manuscript received September 5, 1996; revised September 30, 1998. This paper was supported in part by a 1995 Special Research Fund from the University Research Institute, Korea Research Foundation.

I. H. Suh and J.-H. Kim are with the Intelligent Control and Robotics Laboratory, Department of Electronic Engineering, Hanyang University, Ansan, Korea.

F. C.-H. Rhee is with the Computational Vision and Fuzzy Systems Laboratory, Department of Electronic Engineering, Hanyang University, Ansan, Korea.

Publisher Item Identifier S 1063-6706(99)04935-8.

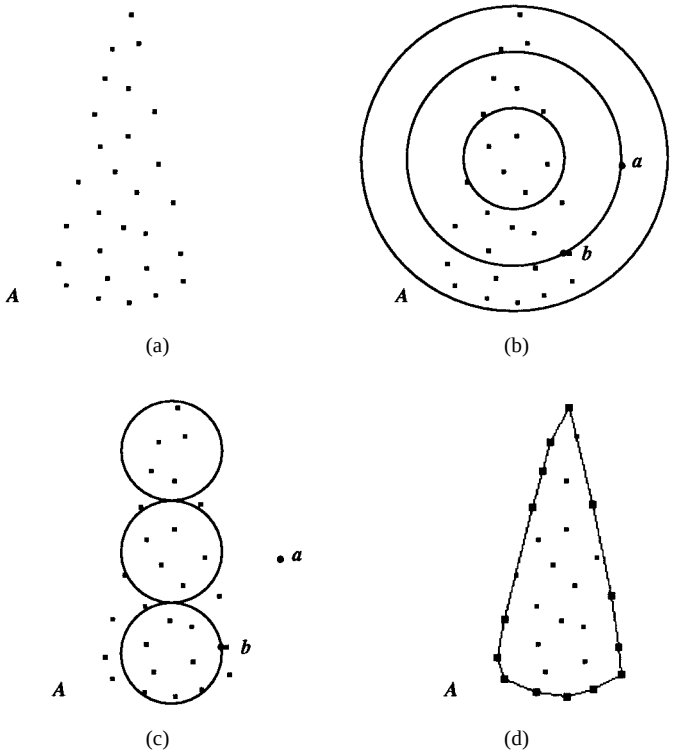


Fig. 1. An example of assigning various membership values to a tear-shaped input data set. (a) The original data set. (b) The data set represented by a single point prototype. (c) The data set represented by three point prototypes. (d) The data set represented a tear-shaped prototype.

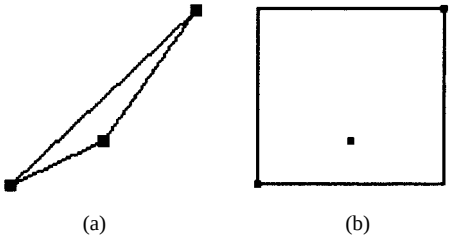


Fig. 2. A simple data set represented by (a) a triangular-shaped prototype and (b) a hyperbox.

fitted when using a triangular-shaped prototype rather than a hyperbox. However, triangular-shaped prototypes may not be appropriate for yet other amorphous data sets. Therefore, the use of a prespecified prototype may not be sufficient in representing an arbitrarily distributed data set.

Furthermore, there exists another important problem that needs to be addressed in the FMMCNN clustering process mentioned above. That is the expansion-contraction procedure, which was used to resolve the undesirable case of overlapping hyperboxes that was present in the original fuzzy ART algorithm. Hence, the procedure eliminated the problem of allowing an input pattern to have full membership in more than one cluster. However, the expansion-contraction procedure should be reconsidered in the clustering process, since it may lead to a fluctuating (unstable) membership

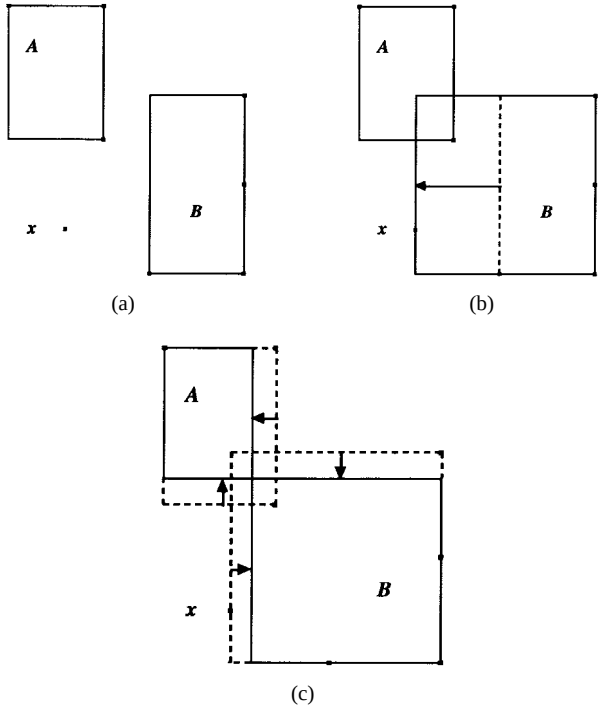


Fig. 3. Fluctuating (unstable) membership assignment due to a contraction process. (a) Initial state. (b) Results after explanation. (c) Results after contraction.

assignment of previous well memorized input patterns. As an example, consider the case shown in Fig. 3, where there are two clusters A and B. For an input pattern x, let us say cluster B provides the highest degree of membership and, therefore, expands to the input position. As soon as cluster B detects an overlap with cluster A, both clusters undergo the contraction process, which results in a fluctuation of membership assignment for pattern x as well as other previous input patterns. This is shown in Fig. 3(c).

To solve the problems mentioned above, we proposed a fuzzy clustering method that involved adaptively growing convex polytopes [30]. In a refined version of [30], we perform clustering in two sequential levels: 1) initial clusters are approximately obtained by fitting convex polytopes to the input data and 2) the initial clusters are combined, if necessary, by using a fuzzy convex cluster merging algorithm, where an expandability measure is defined by employing membership functions. In Level 1), we design a clustering algorithm based on adaptively fitting convex polytopes to a given data set in order to obtain initial cluster boundaries. In the algorithm, each cluster is represented by the vertices of a convex polytope and an input pattern located in the interior of a cluster is considered to have a membership value of one. For an input pattern that is located outside of all the present clusters, a new cluster is either created or a new vertex is created for one of its neighboring clusters with the possibility of some of the existing vertices becoming candidates for replacement. Here nonlinear membership functions are utilized. Specifically, a membership value of an input pattern to each existing cluster is assigned according to a distance measure for the input

pattern and each cluster. Then the cluster to which the input pattern maximally belongs is possibly reshaped by two types of modification. One is the addition of a new vertex with the existing ones and the other is the addition of a new vertex replacing some existing ones. In Level 2), cluster merging is performed by allowing a pair of clusters to merge under the constraint that the expanded cluster volume does not exceed a threshold size and the merged cluster does not overlap with any other cluster.

As mentioned above, levels 1) and 2) are sequentially performed for each incoming input pattern in only one pass. Thus, our clustering method may be considered as an on-line process. In summary, our proposed clustering method has the following appealing attributes: 1) the need for a fixed prototype is eliminated (i.e., a given data set can be compactly fitted by arbitrary-shaped convex clusters); 2) *a priori* knowledge of the number of clusters in the data set is not necessary; and 3) clustering is performed on a given data set in only one pass through the data set and it can process new additional input data without having to recluster the previous input data set (i.e., it possesses the on-line property).

The remainder of this paper is organized as follows. In Section II, we introduce some of the convex properties that are used in describing a convex cluster. In Section III, we present our fuzzy convex clustering method in detail, which consists of the fuzzy convex cluster expansion and the fuzzy convex cluster merging algorithms. In Section IV, we provide several numerical examples showing the validity of our proposed method. Finally, Section V gives the summary and conclusions.

II. Cluster Approximation Using Convex Polytopes

A convex cluster can be represented by the vertices of a convex polytope and it is considered to have full membership within its boundary. Hence, if an input pattern is located in the interior of a convex cluster, then the membership value of the input pattern to the cluster is equal to one. Otherwise, the convex cluster becomes a candidate for modification. For this, it should be determined whether the input pattern is located outside of the cluster and whether the convex property can still be preserved after its modification. In this section, we describe several important techniques that involve the convex properties that are used in the fuzzy clustering algorithm that will be explained later in Section III.

A. Determining the Inclusion of a Pattern Vector Within a Convex Set

In order to determine whether an input pattern lies within a convex set, it is necessary to discuss the existence of a hyperplane that separates the pattern from the set. This is based on the following two definitions and two theorems.

Let $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_m$ denote a set of m vectors in $\boldsymbol{R}^n$. Then a convex combination of $\boldsymbol{a}_i$ vectors may be defined as follows [31].

Definition 1: A convex combination of vectors $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_m$ in $\boldsymbol{R}^n$ is a linear combination $w_1\boldsymbol{a}_1 + \cdots + w_m\boldsymbol{a}_m$ subject to the constraint $w_1 + \cdots + w_m = 1$, where $0 \leq w_i \leq 1$ for $i = 1, \cdots, m$.

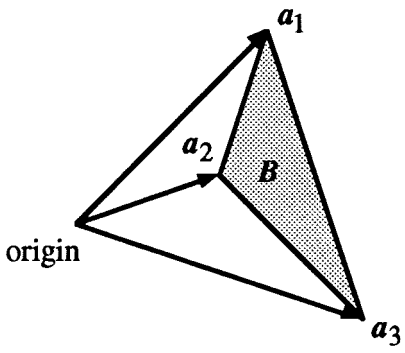


Fig. 4. An $n - D$ convex polytope represented by three vertices.

Now, a convex polytope and the existence of a hyperplane that separates an input pattern $\boldsymbol{x}$ from a convex polytope can be described by Definition 2 and Theorem 1, respectively.

Definition 2: A convex polytope of the set of vectors $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_m$ in $\boldsymbol{R}^n$ is the set of all convex combinations of these vectors.

As an illustration, Fig. 4 shows an example of a $n - D$ convex polytope which is represented by three vertices.

Theorem 1 (Eidelheit Separation Theorem [32]): Let $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ be convex sets in $\boldsymbol{R}^n$ such that $\boldsymbol{C_1}$ has interior points and $\boldsymbol{C_2}$ contains no interior point of $\boldsymbol{C_1}$. Then there exists a hyperplane $\boldsymbol{H}$ that separates $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$.

Now, from the above two definitions and Theorem 1, Theorem 2 can be derived as follows.

Theorem 2: Suppose that a set $\boldsymbol{B} \subseteq \boldsymbol{R}^n$ is a convex polytope with m vertices. If a pattern vector $\boldsymbol{x}$ is located outside of $\boldsymbol{B}$, then there always exists a hyperplane separating $\boldsymbol{x}$ from $\boldsymbol{B}$.

Proof: Let convex polytope $\boldsymbol{B}$ and pattern $\boldsymbol{x}$ be substituted with convex set $\boldsymbol{C_1}$ and $\boldsymbol{C_2}$, respectively. By Theorem 1, if pattern $\boldsymbol{x}$ is located outside of $\boldsymbol{B}$, then $\boldsymbol{x}$ contains no interior point of $\boldsymbol{B}$. Hence, there always exists a hyperplane $\boldsymbol{H}$ that separates $\boldsymbol{x}$ from $\boldsymbol{B}$.

To show how to detect such a separating hyperplane, consider an input pattern $\boldsymbol{x}$ that is located outside of a convex polytope k which has m vertices. Now, consider an expansion vector to be the normalized vector from a vertex of the convex polytope to the pattern $\boldsymbol{x}$. Then the expansion vector $\boldsymbol{e}_i^k$ for the ith vertex of the convex polytope k can be given by

$$\boldsymbol{e}_i^k = \frac{\boldsymbol{x} - \boldsymbol{a}_i^k}{||\boldsymbol{x} - \boldsymbol{a}_i^k||} \tag{1}$$

where $\boldsymbol{a}_i^k$ denotes the ith vertex for the convex polytope k. If we arbitrarily select the expansion vector $\boldsymbol{e}_j^k$ as a reference vector and perform vector addition with all the other remaining expansion vectors, then the resultant vector after normalization

$\boldsymbol{f}_i^k$ can be given as

$$\boldsymbol{f}_i^k = \frac{\boldsymbol{e}_i^k + \boldsymbol{e}_j^k}{||\boldsymbol{e}_i^k + \boldsymbol{e}_j^k||}, \qquad \text{for all } i, \qquad i \neq j. \tag{2}$$

Now, it can be easily verified that the angle between $\{\boldsymbol{e}_j^k, \boldsymbol{f}_i^k\}$ becomes contracted in half with respect to the angle between $\{\boldsymbol{e}_j^k, \boldsymbol{e}_i^k\}$. By Theorem 2, since we consider that the input pattern $\boldsymbol{x}$ is located outside of convex polytope k, there exists a hyperplane separating $\boldsymbol{x}$ from k. Furthermore, since a hyperplane is considered to be the boundary of half-spaces, all the expansion vectors must lie in the same half-space for input pattern $\boldsymbol{x}$ to be considered to be located outside of convex polytope k. For this, the angles between the arbitrary selected reference vector $\boldsymbol{e}_j^k$ and all the other remaining expansion vectors must all be less than 180° and the angles in the addition space must all be less than 90°. If this is the case, the resultant vectors all lie in the contracted space which span the space within 90° and the dot product of all pairs of $\boldsymbol{f}_i^k$ and $\boldsymbol{f}_j^k, i \neq j$ will always be greater than zero. Therefore, we can say that the input pattern $\boldsymbol{x}$ lies outside of a convex polytope if all the vectors from the vertices of the convex polytope to the input pattern lie in the same half-space. (It is to be noted that for vectors that lie on the boundary of the half-space, then the pattern $\boldsymbol{x}$ lies on or inside the convex polytope.) Hence, the existence of a hyperplane that separates the input pattern $\boldsymbol{x}$ from the convex polytope may be detected.

It can be easily verified that for the case of a pattern $\boldsymbol{x}$ that lies inside of a convex polytope there exist no hyperplane that separates $\boldsymbol{x}$ from the convex polytope. In other words, referring back to the discussion above, if for a case where the dot product of some pair $\boldsymbol{f}_i^k$ and $\boldsymbol{f}_j^k$ is less than or equal to zero, then the vectors are said to lie outside of the contracted space which spans the space within 90°. In this case, all the vectors from the vertices of a convex polytope to the input pattern do not lie in the same half-space. Hence, there exists no hyperplane that can separate the input pattern from the convex polytope. Therefore, pattern vector $\boldsymbol{x}$ must lie inside the convex polytope. In conclusion, if we can detect the existence of a hyperplane that separates a pattern vector $\boldsymbol{x}$ from a convex polytope, then we can determine whether the pattern $\boldsymbol{x}$ lies in the interior or exterior of the convex polytope.

As an example to illustrate how to detect a separating hyperplane, consider an input pattern $\boldsymbol{x}$ that is located outside of a convex polytope B that has three vertices [see Fig. 5(a)]. From the figure, it clearly shows that there exists a hyperplane that separates $\boldsymbol{x}$ from $\boldsymbol{B}$. Now, consider unit vectors $\boldsymbol{e}_1, \boldsymbol{e}_2$, and $\boldsymbol{e}_3$ to be the normalized vectors from the vertices $\boldsymbol{a}_1, \boldsymbol{a}_2$, and $\boldsymbol{a}_3$ of $\boldsymbol{B}$ to the pattern $\boldsymbol{x}$, respectively. Fig. 5(b) shows that the unit vectors $\boldsymbol{e}_1, \boldsymbol{e}_2$, and $\boldsymbol{e}_3$ all lie in the same half-space. If we arbitrarily select $\boldsymbol{e}_1$ as a reference vector and perform vector addition with each of the other unit vectors (i.e., $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$), then the resultant vectors after normalization are say, $\boldsymbol{f}_2$ and $\boldsymbol{f}_3$, respectively. Then, the angles between $\{\boldsymbol{e}_1, \boldsymbol{f}_2\}$ and $\{\boldsymbol{e}_1, \boldsymbol{f}_3\}$ are contracted in half with respect to the angles between $\{\boldsymbol{e}_1, \boldsymbol{e}_2\}$ and $\{\boldsymbol{e}_1, \boldsymbol{e}_3\}$, respectively.

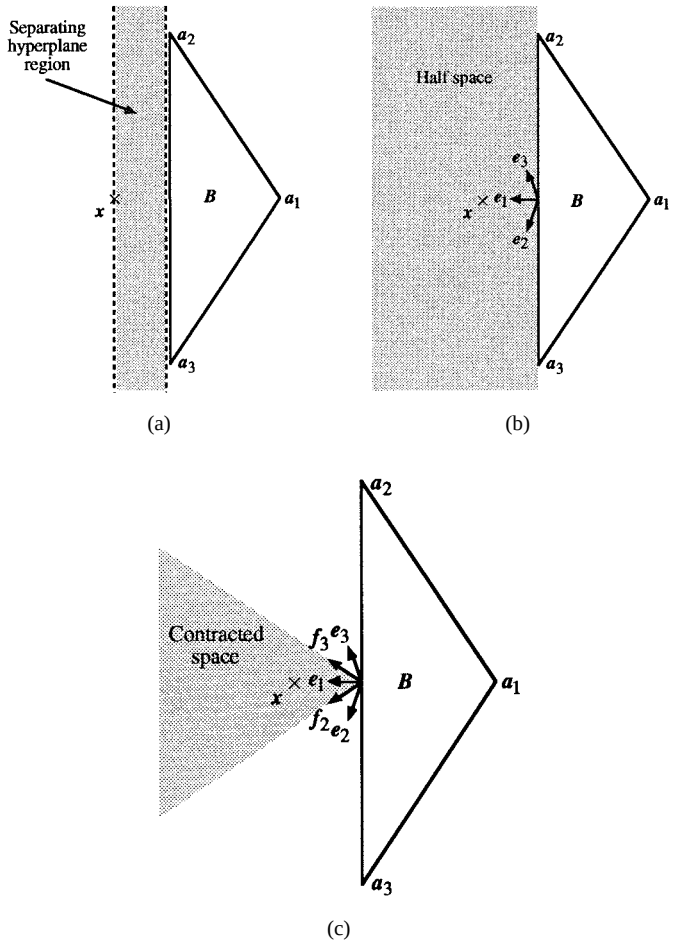


Fig. 5. An example that illustrates the detection of a separating hyperplane for an input pattern $\boldsymbol{x}$ that is located outside of a convex polytope $\boldsymbol{B}$. (a) The input pattern $\boldsymbol{x}$ is shown outside of the convex polytope $\boldsymbol{B}$, which has three vertices. (b) The normalized vectors $\boldsymbol{e}_1, \boldsymbol{e}_2$, and $\boldsymbol{e}_3$ obtained from the vertices $\boldsymbol{a}_1, \boldsymbol{a}_2$, and $\boldsymbol{a}_3$ of $\boldsymbol{B}$ to the pattern $\boldsymbol{x}$, respectively, are shown to lie in the same half-space. (c) The resultant unit vectors $\boldsymbol{f}_2$ and $\boldsymbol{f}_3$ are shown to lie in the contracted space resulting after addition of vectors $\{\boldsymbol{e}_1, \boldsymbol{e}_2\}$ and $\{\boldsymbol{e}_1, \boldsymbol{e}_3\}$, which spans the space within 90°.

This is shown in Fig. 5(c). Now, since the angle between an arbitrary reference vector and any other vector within the same half-space is less than 180°, the angle in the additional space results in an angle less than 90°. Furthermore, since vectors $\boldsymbol{f}_2$ and $\boldsymbol{f}_3$ both lie in the contracted space which spans the space within 90°, the dot product of $\boldsymbol{f}_2 \cdot \boldsymbol{f}_3$ will always be greater than zero. Therefore, we can say that the input pattern $\boldsymbol{x}$ lies outside of convex polytope $\boldsymbol{B}$ if all the vectors from the vertices of the convex polytope to the input pattern lie in the same half-space. Hence, there exists a hyperplane that separates the input pattern from the convex polytope.

In summary, if the dot product of all pairs of $\boldsymbol{f}_i^k$ and $\boldsymbol{f}_j^k$, $i \neq j$ is greater than zero, then pattern $\boldsymbol{x}$ is considered to be located outside of the convex polytope. Hence, there exists a hyperplane that separates pattern $\boldsymbol{x}$ from the convex polytope. Otherwise, pattern $\boldsymbol{x}$ is determined to be located in the interior of the convex polytope. If we consider a cluster to be represented by a convex polytope, then the detection of

a separating hyperplane may be summarized by the algorithm that follows:

Separating Hyperplane Detection (SHD)

Algorithm

Given an input pattern $\boldsymbol{x}$;
FOR all vertex vectors in kth cluster **DO**
Obtain expansion unit vectors $\boldsymbol{e}_i^k$ using (1);
END FOR
Arbitrarily select expansion vector $\boldsymbol{e}_j^k$, as a reference vector among m expansion unit vectors;
FOR all remaining expansion unit vectors **DO**
Compute resultant unit vector $\boldsymbol{f}_i^k$ using (2);
END FOR
Set interior = FALSE
(The input pattern is initially assumed to lie outside the kth cluster);
Set resultant unit vector pair number $l = 0$;
WHILE (interior = FALSE **AND** $l \neq$ total number of resultant unit vector pairs)
Increment l;
IF $\boldsymbol{f}_i^k \cdot \boldsymbol{f}_j^k \leq 0$ **THEN**
Set interior = TRUE
(The input pattern is in the interior side of the kth cluster);
END IF
END WHILE.

B. Convexity Preservation of a Convex Set After Modification

In Section II-A, we proposed an algorithm that is used to determine whether there exists a hyperplane that separates an input pattern $\boldsymbol{x}$ from a convex cluster. In doing so, we can determine whether the input pattern $\boldsymbol{x}$ lies inside or outside of the convex cluster. In this section, we describe a method for determining the type of cluster modification for the addition of an input pattern $\boldsymbol{x}$. This can be achieved by testing the convexity of the convex cluster with the addition of the input pattern $\boldsymbol{x}$. The method becomes very simple when it is used with the separating hyperplane detection (SHD) algorithm. The convexity test algorithm is summarized as follows:

Convexity Test (CT) Algorithm

Set convexity = TRUE;
Construct an arbitrary cluster $\boldsymbol{C}_i$ with all vertices except for $\boldsymbol{a}_i^i$ (the arbitrary cluster need not be convex);
Apply the **SHD** algorithm with $\boldsymbol{a}_i^k$ and vertices of $\boldsymbol{C}_i$;
IF interior = TRUE **THEN**
Set convexity = FALSE;
END IF.

As mentioned above, by using the CT algorithm we may develop a method to determine how a cluster is modified for an incoming input vector. This is achieved by considering input pattern $\boldsymbol{x}$ as an additional vertex of a convex cluster and determining if the vertex due to pattern $\boldsymbol{x}$ can be separated by a hyperplane from the other remaining vertices. If this can be achieved, then either all the vertices of the cluster including pattern $\boldsymbol{x}$ can be considered as a convex cluster or there exists at least one interior vertex when considering pattern $\boldsymbol{x}$ as a new vertex for the modified cluster. It is to be noted that cluster modification is not performed if pattern $\boldsymbol{x}$ is an interior vertex. Hence, there exists two types of cluster modification. One is the addition of a new vertex with the existing ones and the other is the removal of some existing ones. Each type is used for the expansion of a convex cluster. This is illustrated in Fig. 6 and summarized as follows:

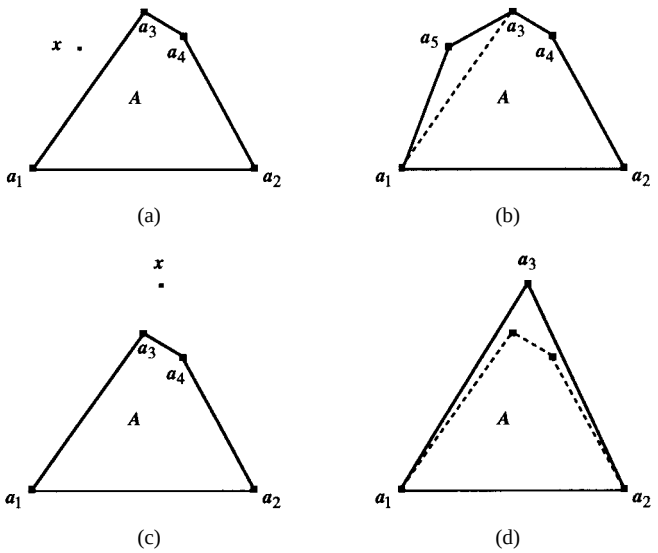


Fig. 6. Cluster modification involving the addition of a new vertex with the existing ones. [(a) → (b)]. With the removal of some existing ones [(c) → (d)].

Convex Cluster Modification (CCM) Algorithm

Construct a new cluster with input pattern $\boldsymbol{x}$ as a new vertex and all vertices of kth cluster
Apply the **CT** algorithm with input pattern $\boldsymbol{x}$ and the kth cluster;
IF convexity = FALSE **THEN**
Discard vertex due to $\boldsymbol{x}$ from kth cluster;
ELSE
FOR all remaining vertices of kth cluster **DO**
Apply the **CT** algorithm;
IF convexity = FALSE **THEN**
Store vertex $\boldsymbol{a}_i^k$ as a vertex to be removed;
END IF
END FOR
Discard all vertices that are stored to be removed;
END IF.

C. Intracluster Expandability Measure of a Convex Set

It is obvious that it is not desirable for a cluster to expand continuously without reason. Therefore, as can be seen in most clustering algorithms, we need a termination condition for the cluster modification process mentioned above. To obtain such a termination condition, a method for obtaining the volume of a cluster is needed. For this, we define the volume of a cluster as an intracluster expandability measure, which is used to determine whether a cluster should be modified or not. We discuss two methods in obtaining this measure.

The first of the two methods involves counting the number of volume cells that are located in the interior of the convex cluster. This is achieved by selecting a reasonable small axis interval I for each dimension of the cluster and counting the number of interior volume cells such that the error between the actual and estimated volume is minimal. The number of interior cells may be easily obtained by using the SHD algorithm. In this case, the intracluster expandability measure $\varepsilon_{\text{intra}}(\boldsymbol{A}^k)$ for a convex cluster $\boldsymbol{A}^k$ in $\boldsymbol{R}^n$, can be given by

$$\varepsilon_{\text{intra}}(\boldsymbol{A}^k) = \sum_{l_1=1}^{N_1} \cdots \sum_{l_n=1}^{N_n} m(l_1, \cdots, l_n) \cdot \prod_{r=1}^{n} I_r \qquad (3)$$

where $I_r = \max_{i \neq j} \{|a_{ri}^k - a_{rj}^k|\}/N_r$ is the cell length for the rth dimension, N_r is the number of intervals, and $m(\cdot)$ is an $n - D$ binary valued matrix.

In (3), the value of $m(\cdot)$ is one for an interior cluster cell and zero otherwise. This method of calculating the volume may be satisfactory for problems of low dimensionality, but can become very time consuming for problems of high dimensionality.

As an alternative for describing the cluster volume, the average of the Euclidian distance between the averaged vertex vector $\boldsymbol{c}_k$ and all the vertex vectors in the cluster may be used. In this case, the intracluster expandability measure $\varepsilon_{\text{intra}}(\boldsymbol{A}^k)$ can be given by

$$\varepsilon_{\text{intra}}(\boldsymbol{A}^k) = \frac{1}{m} \sum_{i=1}^{m} ||\boldsymbol{a}_i^k - \boldsymbol{c}_k||^2 \qquad (4)$$

where

$$\boldsymbol{c}_k = \frac{1}{m} \sum_{i=1}^{m} \boldsymbol{a}_i^k.$$

It can be seen from (4) that the cluster volume can be obtained much faster than the first method mentioned above, especially for high-dimensional problems. We can now use the intracluster expandability measures mentioned above to set a termination condition for the cluster expansion process that is described in the following section.

III. The Fuzzy Convex-Clustering Algorithm

In many fuzzy clustering algorithms, a membership function is used to effectively assign an input pattern to a cluster. Here, a membership function is used to describe the degree to which a pattern belongs to a cluster. We now introduce a two-step procedure for assigning membership values.

First, the membership value for an input pattern $\boldsymbol{x}$ to a prototypical member x_0 is given by

$$\mu^k(\boldsymbol{x}) = \frac{1}{1 + \dfrac{d(\boldsymbol{x}, x_0)^{\gamma(x_0)}}{\lambda}} \qquad (5)$$

where $d(\boldsymbol{x}, x_0)$ is defined as a distance measure for an input pattern $\boldsymbol{x}$ to the prototypical member x_0, λ is the distance from x_0 at which $\mu^k(\boldsymbol{x}) = 0.5$ (bandwidth) and $\gamma(x_0)(\gamma(x_0) > 0)$ represents the rate of decay measure (fuzzification or sharpness function). It is to be noted that the membership function in (5) is similar to the one suggested by Zimmermann *et al*. [33], which is used to model vague concepts.

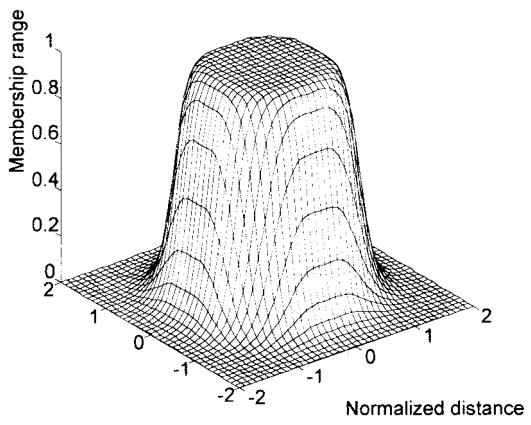
In (5), we consider a convex cluster $\boldsymbol{A}^k$ to be the prototypical member. In doing so, the distance measure $d(\boldsymbol{x}, x_0)$ and $\gamma(x_0)$ in (5) may be given by

$$d(\boldsymbol{x}, x_0) = \text{dist}\,(\boldsymbol{x}, \boldsymbol{A}^k)$$

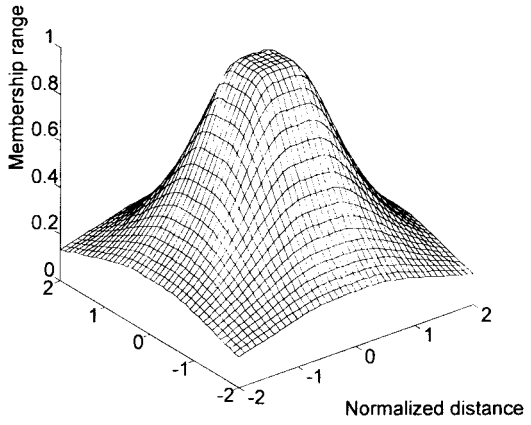
and

$$\gamma(x_0) = \frac{N_k}{1 + \dfrac{\varepsilon_{\text{intra}}(\boldsymbol{A}^k)}{c_0}} \qquad (6)$$

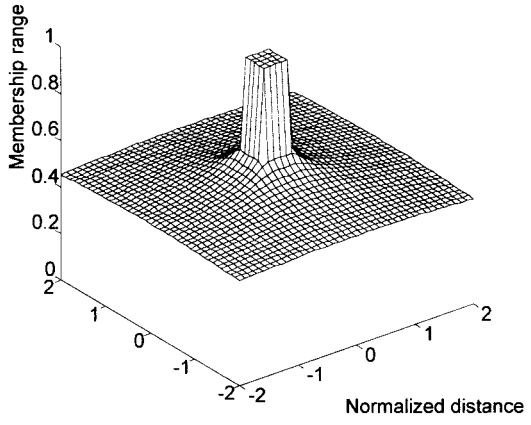
respectively. In (6), $\text{dist}\,(\boldsymbol{x}, \boldsymbol{A}^k)$ denotes the minimum distance from input pattern $\boldsymbol{x}$ to the convex cluster $\boldsymbol{A}^k$, N_k is the number of input patterns located on or in the interior of $\boldsymbol{A}^k$, and c_0 $(0 < c_0 \leq 1)$ is a weighting factor for the cluster volume measure $\varepsilon_{\text{intra}}(\boldsymbol{A}^k)$. The value of c_0 contributes to the fuzziness of the shape of the membership function in (5). When $c_0 \to 1$, the membership function is maximally hard and when $c_0 \to 0$, the memberships are maximally fuzzy. Since, $\gamma(x_0)$ takes into account the number of samples as well as the cluster volume, its value can be considered as the cluster density for the cluster $\boldsymbol{A}_k$. Thus, the membership assignment for an input pattern to a cluster depends on not only the distance measure $d(\boldsymbol{x}, x_0)$ but also the cluster density. It is to be noted that there exists a classical method to obtain $\text{dist}\,(\boldsymbol{x}, \boldsymbol{A}^k)$ that involves nonlinear equations and a Gram matrix [32]. It has been shown that the solutions to the equations represent a good approximation of $\text{dist}\,(\boldsymbol{x}, \boldsymbol{A}^k)$, but the solutions can not be easily obtained due to the nonlinearities. Thus, we consider a more practical method in approximating $\text{dist}\,(\boldsymbol{x}, \boldsymbol{A}^k)$. This involves the use of the first method for calculating the cluster volume that was mentioned in Section II-C. Recall that the method involved counting



(a)



(b)



(c)

Fig. 7. Membership function plots for a 2-D square-shaped cluster for various fuzzification function values ($\gamma(x_0)$) using a normalized distance (i.e., $\lambda = 1$). (a) $\gamma(x_0) = 10$. (b) $\gamma(x_0) = 2$. (c) $\gamma(x_0) = 0.2$.

the number of interior cells in a cluster. Therefore, to obtain $\mathrm{dist}(\boldsymbol{x}, \boldsymbol{A}^k)$ we can simply take the Euclidean distance of the closest interior cell to the input pattern $\boldsymbol{x}$. Fig. 7 shows membership function plots for a two-dimensional (2-D) square-shaped cluster for various values of $\gamma(x_0)$ using a normalized distance (i.e., $\lambda = 1$). As can be seen in the figure, for large values of $\gamma(x_0)$ the membership function becomes hard, and for small values, the memberships become maximally fuzzy.

Using the results above, the membership value of an input pattern $\boldsymbol{x}$ for the kth cluster can be given as

$$m(k) = \begin{cases} 1, & \text{if input pattern } \boldsymbol{x} \text{ lies on or inside the } k\text{th cluster} \\ \mu^k(\boldsymbol{x}), & \text{otherwise.} \end{cases} \tag{7}$$

Now, to determine the most expandable cluster in which input pattern $\boldsymbol{x}$ maximally belongs, it is necessary to obtain the largest membership value among all of the expandable clusters. That is

$$k^* = \arg\{\max_k \{\mu^k(\boldsymbol{x})\}\} \tag{8}$$

where k^* denotes the most expandable cluster in which input pattern $\boldsymbol{x}$ maximally belongs.

By using the membership assignment mentioned above, clustering is performed as follows. For an incoming input pattern, the SHD algorithm is utilized to determine if the pattern lies inside any of the current clusters. If so, the pattern belongs to one of the clusters and, therefore, cluster expansion is not performed for that particular pattern. (In other words, a cluster is considered expandable if and only if the input pattern is located outside of all the current clusters.) Otherwise, the most expandable cluster is identified as k^*; that is, the cluster which has the highest membership value for the input pattern among all the expandable clusters. Now, if $\mu^{k^*}(\boldsymbol{x})$ exceeds a prespecified membership threshold value θ, then the convex cluster modification (CCM) algorithm in Section II is applied to expand cluster k^*. Next, the size of the expanded cluster is examined to see if it lies within a maximum allowable cluster size β. Finally, the expansion of cluster k^* is examined to see if it causes an overlap with some neighboring clusters. To detect such a situation, an overlapping case can be simply determined by examining whether any of the neighboring cluster vertices lie inside of the expanded cluster k^*. In this case, the SHD algorithm can also be utilized here.

If any one of the above constraints is not satisfied, then the expandable cluster is not expanded for the given input pattern and the next expandable cluster becomes a candidate for expansion. This process is repeated for all the remaining expandable clusters until a cluster for expansion is found. If none exists then pattern $\boldsymbol{x}$ becomes a new point cluster. We summarize our proposed fuzzy convex cluster expansion (FCCE) algorithm as follows:

Fuzzy Convex Cluster Expansion (FCCE) Algorithm (LEVEL I)

Set all necessary initial parameters $(\lambda, e_0, \theta, \beta)$;
Given an input pattern $\boldsymbol{x}$;
FOR all clusters **DO**
  Apply **SHD** algorithm;
END FOR
IF interior = FALSE **THEN**
  Set expandable cluster number $i = 0$;
  FOR all clusters **DO**
    Obtain $\gamma(x_0)$ using (6);
    Obtain $\mu^k(\boldsymbol{x})$ using (5);
    IF $\mu^k(\boldsymbol{x}) >$ threshold value θ **THEN**
      Increment i;
      Sort cluster indexes in decreasing order according to corresponding $\mu^k(\boldsymbol{x})$ values;
    END IF
END FOR
Set expand = FALSE;
Set expandable cluster index $i = 0$;
WHILE (expand = FALSE **AND** $i \neq$ total number of expandable clusters)
  Increment i;
  Expand cluster i using **CCM** algorithm;
  Test following conditions for each cluster;
    1) Determine if size of expanded cluster $< \beta$ using (3) or (4);
    2) Determine if no overlap is detected between the expanded cluster and the other current clusters using **SHD** algorithm;
  IF the above two conditions are satisfied **THEN**
    Set expand = TRUE;
    k^* = expanded cluster i;
  END IF
END WHILE
IF expand = FALSE **THEN**
  Create a new point cluster;
END IF
END IF.

After applying the FCCE algorithm, the results may give an undesirable number of clusters. This may be due to a small limit on the maximum allowable cluster size parameter (i.e., β) used in the algorithm. However, in order to reduce the order dependency on the incoming pattern data, the expansion of a cluster should be limited to a reasonably small size (i.e., the value of β should be kept small to avoid undesirable cluster growth). Therefore, to improve such a result, we now propose a fuzzy convex cluster merging algorithm to partially remedy the situation. For this we define an intercluster expandability measure as

$$\varepsilon_{\text{inter}}(\boldsymbol{A}^j, \boldsymbol{A}^k) = \varepsilon_{\text{intra}}(\boldsymbol{A}^j \oplus \boldsymbol{A}^k) - \Delta(\boldsymbol{A}^j, \boldsymbol{A}^k) \cdot [\varepsilon_{\text{intra}}(\boldsymbol{A}^j) + \varepsilon_{\text{intra}}(\boldsymbol{A}^k)] \tag{9}$$

where $\oplus$ denotes a convex cluster merging operation and $\Delta(\boldsymbol{A}^j, \boldsymbol{A}^k)$ is a membership function given as

$$\Delta(\boldsymbol{A}^j, \boldsymbol{A}^k) = \frac{1}{1 + \min\limits_{i,l} \{||\boldsymbol{a}_i^k - \boldsymbol{a}_i^j||\}}. \tag{10}$$

It is to be noted that the $\oplus$ operation in (9) creates a minimum convex cluster which encloses all the vertices of the clusters to be merged.

Using (9), cluster merging is achieved by allowing a pair of clusters to merge under the constraint that its intercluster expandability measure lies within a maximally allowable merging cluster size $\Delta\beta$ and the merged pair of clusters do not overlap with any other cluster. As in the FCCE algorithm, the SHD algorithm can also be utilized here. It is to be noted that there may exist more than one pair of clusters that satisfy the above conditions. In this case, the pair that has the smallest intercluster expandability value is the candidate to be merged. The value of $\Delta\beta$ is usually set slightly greater than or equal to β. The fuzzy convex cluster merging algorithm is summarized as follows:

Fuzzy Convex Cluster Merging (FCCM) Algorithm (LEVEL II)

Set an initial value for $\Delta\beta$;
Set $\delta = \Delta\beta$;
FOR all pairs of clusters $(\boldsymbol{A}^j, \boldsymbol{A}^k)$ **DO**
  Find $\varepsilon_{\text{intra}}(\boldsymbol{A}^j \oplus \boldsymbol{A}^k), \varepsilon_{\text{intra}}(\boldsymbol{A}^j)$, and $\varepsilon_{\text{intra}}(\boldsymbol{A}^k)$ using (3) or (4);
  Find $\Delta(\boldsymbol{A}^j, \boldsymbol{A}^k)$ and $\varepsilon_{\text{inter}}(\boldsymbol{A}^j, \boldsymbol{A}^k)$ using (10) and (9), respectively;
  IF $\varepsilon_{\text{inter}}(\boldsymbol{A}^j, \boldsymbol{A}^k) < \delta$ **AND** $\boldsymbol{A}^j \oplus \boldsymbol{A}^k$ does not overlap with any other cluster **THEN**
    Replace δ by $\varepsilon_{\text{inter}}(\boldsymbol{A}^j, \boldsymbol{A}^k)$;
    Set $j^* = j$ and $k^* = k$;
  END IF
END FOR
IF $\delta < \Delta\beta$ **THEN**
  Perform $\boldsymbol{A}^{j^*} \oplus \boldsymbol{A}^{k^*}$;
  FOR each vertex of cluster $\boldsymbol{A}^{j^*} \oplus \boldsymbol{A}^{k^*}$ **DO**
    Apply **SHD** algorithm to eliminate all possible interior vertices;
  END FOR
END IF.

Now, for a complete clustering procedure, the FCCE and FCCM algorithms mentioned above are performed sequentially for each incoming input pattern. Initially, a point cluster for the first input pattern $\boldsymbol{x}$ is created. Then, the FCCE and FCCM algorithms are applied to the remaining input patterns to obtain a final clustering result; that is, clustering is performed on a given data set in only one pass through the

TABLE I
THE PARAMETER VALUES USED IN THE FCCE ALGORITHM

Parameter	Value
Bandwidth (λ)	0.10
Weighting factor (e_0)	0.10
Membership threshold value (θ)	0.80
Maximum allowable cluster size (β)	0.01

data set. Thus, our clustering method can be considered as an on-line process. Finally, we summarize our proposed fuzzy clustering algorithm as follows:

Fuzzy Convex Clustering (FCC) Algorithm

Create an initial point cluster for first input pattern $\boldsymbol{x}$;
FOR remaining input patterns **DO**
  Apply **FCCE** algorithm;
  Apply **FCCM** algorithm;
END FOR.

IV. EXPERIMENTAL RESULTS

In this section, we present some examples that illustrates our proposed clustering method. We first present two simple examples showing how the FCCE algorithm operates. We then present more realistic examples involving our two-level fuzzy clustering algorithm, and compare the results with others. In all of the following examples, the parameter values used in the FCCE algorithm are shown in Table I unless specified.

A. Example 1: Seven Point 2-D Data Problem

As our first example, Fig. 8 shows a demonstration of the FCCE algorithm, where in a 2-D feature space, seven input pattern vectors are presented sequentially. First, input 1 is presented and it becomes a vertex for cluster $\boldsymbol{A}^1$, say $\boldsymbol{a}_1^1$, since there exists no other patterns. Next, input vector 2 arrives and the closest cluster $\boldsymbol{A}^1$ is not allowed for adaptation (expansion) since input 2 is considered relatively far from $\boldsymbol{A}^1$; that is, the membership value relating input 2 and $\boldsymbol{A}^1$ is below a certain fixed threshold value. Thus, input 2 becomes a vertex of another cluster $\boldsymbol{A}^2$, namely $\boldsymbol{a}_1^2$. Input 3 arrives and the closest cluster $\boldsymbol{A}^2$ is allowed for expansion since input 3 is considered relatively close to $\boldsymbol{A}^2$ and does not violate the convex property if expanded and the expanded cluster is within a maximum allowable cluster size. Hence, input 3 becomes cluster vertex $\boldsymbol{a}_2^2$. Same result occurs for input 4 and, therefore, it becomes $\boldsymbol{a}_3^2$. Next, the closest cluster to input vector 5 is $\boldsymbol{A}^2$ and for the same reasons explained above, cluster $\boldsymbol{A}^2$ becomes a candidate for expansion. By considering input 5 as a new cluster vertex for $\boldsymbol{A}^2$, $\boldsymbol{a}_1^2$ (due to input 2) now becomes located in the interior of the newly expanded cluster $\boldsymbol{A}^2$. This is detected by using the SHD algorithm. Hence, vertex $\boldsymbol{a}_1^2$ gets discarded as a vertex for $\boldsymbol{A}^2$ and input 5 becomes $\boldsymbol{a}_1^2$. Inputs 6 and 7 become vertices for cluster 1 for the same reasons mentioned above.

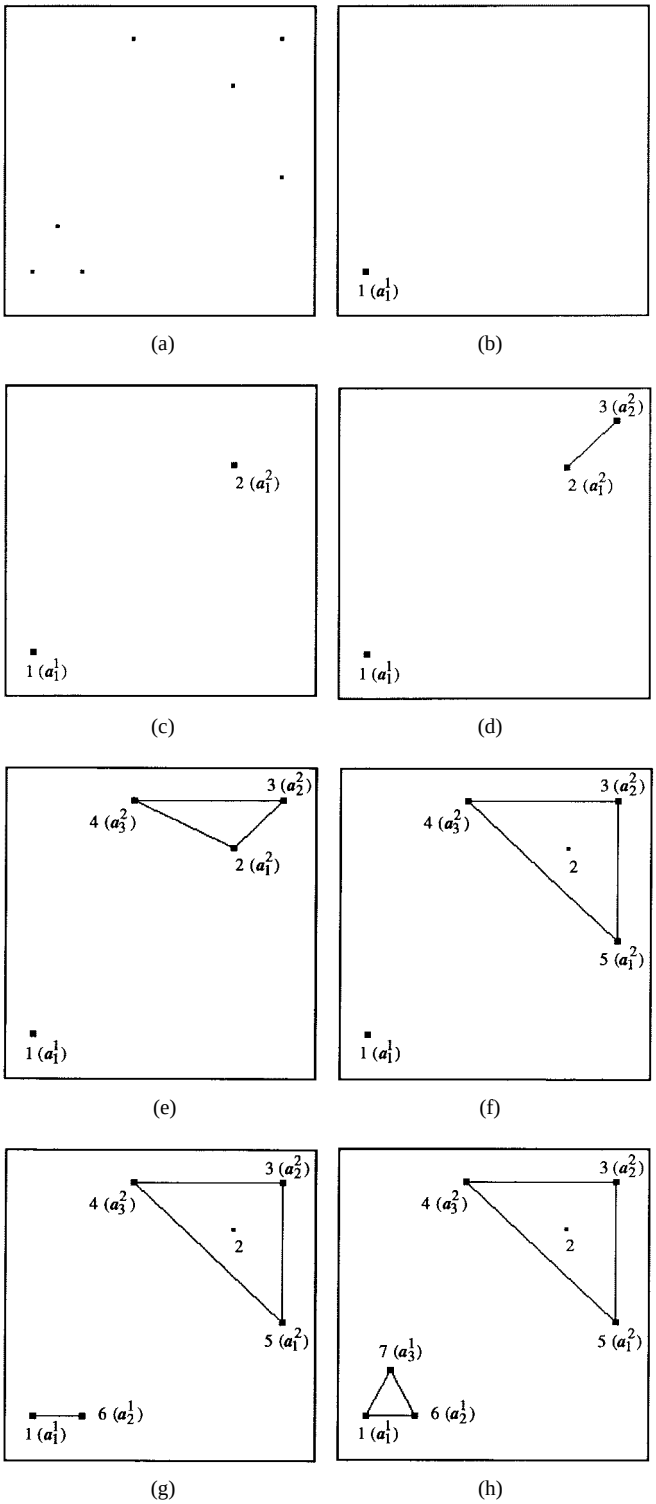


Fig. 8. A demonstration of the FCCE algorithm, where seven input patterns are presented sequentially [(b) → (h)].

B. Example 2: 24 Point 2-D Data Problem

Fig. 9 shows the results of an example that consists of 24 2-D data points. The data used was obtained from Simpson

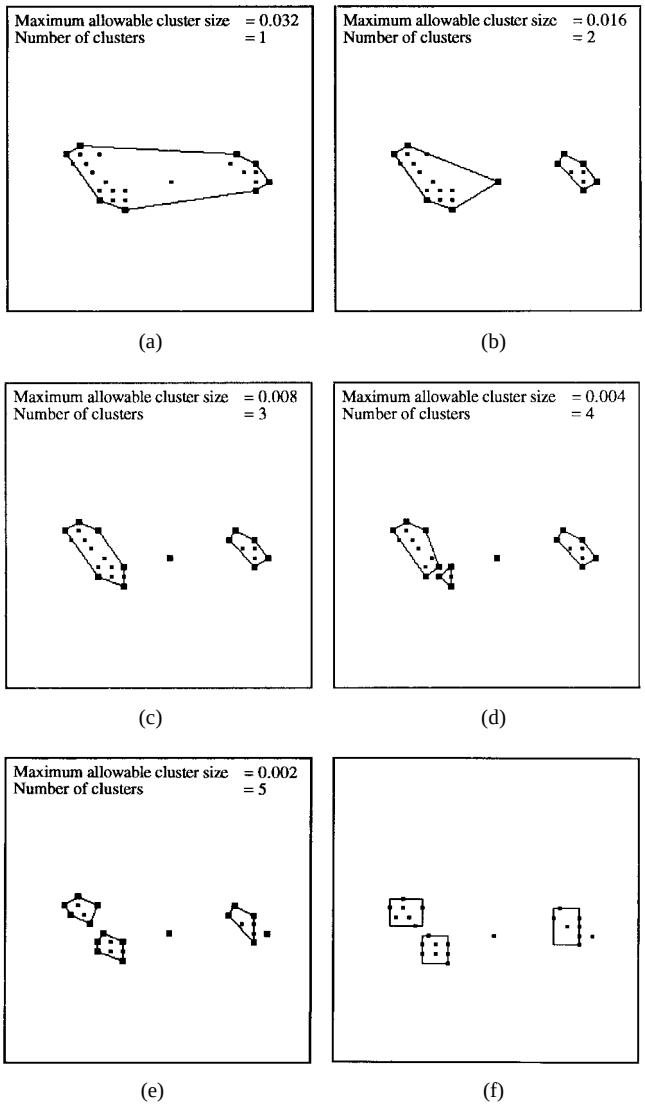


Fig. 9. Results of the FCCE algorithm using various maximum allowable cluster sizes ranging from 0.032 to 0.002 [(a) to (e)] and (f) the results when using the FMMCNN algorithm.

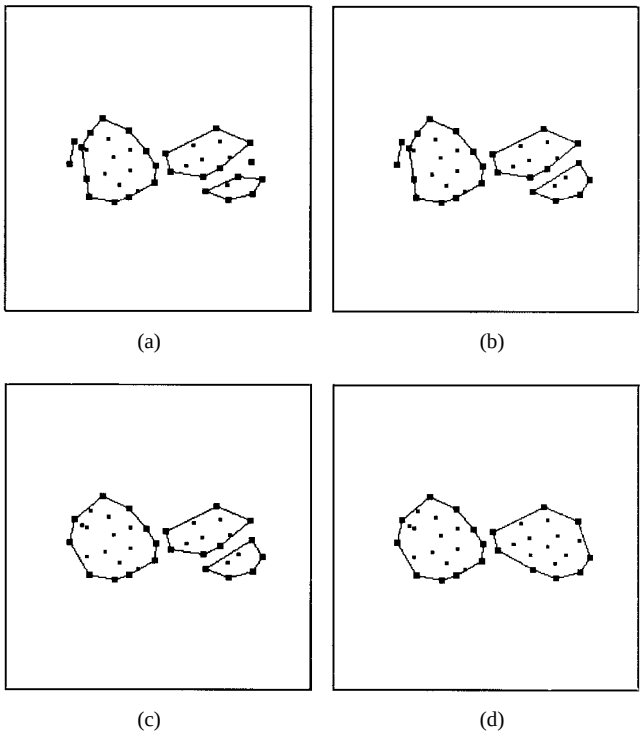


Fig. 10. Results of the FCCE algorithm using a maximum allowable cluster size. (a) $\beta = 0.01$. Results of the FCC algorithm using a maximum allowable cluster merging size. (b) $\Delta\beta = \beta$. (c) $\Delta\beta = 1.2\beta$. (d) $\Delta\beta = 1.5\beta$.

[26], where from this example it shows two common clustering dilemmas: 1) there exists a group of data points that may be considered as either one cluster or two and 2) there exists an outlier in the middle of the two primary groups of data. To illustrate the effect of the maximum allowable cluster size (β) has on the final cluster result, clustering was performed on the data using various β values ranging from 0.032 to 0.002. From the figure, results show that as the maximum allowable cluster size decreases, the number of clusters increase (i.e., a β equalling 0.032 results in one cluster and a β equalling 0.002 results in five clusters). The results show that the cluster assignment seems most appropriate for a β value of either 0.004 or 0.002 [i.e., Fig. 9(d) or (e)]. Furthermore, results show that our method can give slightly more fitted clusters than those of Simpson [26], which is shown in Fig. 9(f).

C. Example 3: Nagy and Lippmann Data Problem

Next, we present some results involving several data sets and show how the various parameter values used in the FCC algorithm influence the final clustering result. Fig. 10(a) shows the cluster results on a typical set of pattern data using the FCCE algorithm only. Here we used a β value of 0.01. Fig. 10(b)–(d) shows results involving the two-level FCC algorithm, where the values of the maximum allowable merging size $(\Delta\beta)$ are $\Delta\beta = \beta, \Delta\beta = 1.2\beta$, and $\Delta\beta = 1.5\beta$, respectively. As expected, as the value of the maximum allowable merging size increased, the number of clusters decreased. Fig. 11 shows clustering results on other sets of pattern data, where $\Delta\beta = 1.5\beta$. The data used are similar to those of Nagy [34] and Lippmann [35], in which the authors discuss some of the difficulties found in cluster analysis. The difficulties include nonspherical clusters [Fig. 11(a)], bridges between clusters [Fig. 11(b)], unequal cluster populations [Fig. 11(c)], and linearly nonseparable clusters [Fig. 11(d) and (e)]. From the figures, one can observe that our FCC algorithm works satisfactorily for these various types of data sets.

Now, to show the performance of our method for various values of the weighting factor (ε_0) for the cluster volume measure $\varepsilon_{\text{intra}}(\boldsymbol{A}^k)$, we used the data set shown in Fig. 11(e), where Fig. 12(a)–(c) shows the clustering results for $\varepsilon_0 = 0.1, 0.01$, and 0.001, respectively. As shown in the figures, results show that as the value of ε_0 decreases, the number of clusters increases. This is expected since by decreasing ϵ_0, the value for the rate of decay measure $\gamma(x_0)$ decreases. Hence, the memberships become more fuzzy (see Fig. 7) and reduces the opportunity for an expandable cluster to further expand.

To show the performance of our method with the addition of outlier input patterns, we simulated our method using the data set shown in Fig. 10, where Fig. 13(a)–(d) shows the clustering results for the data contaminated by 0, 25

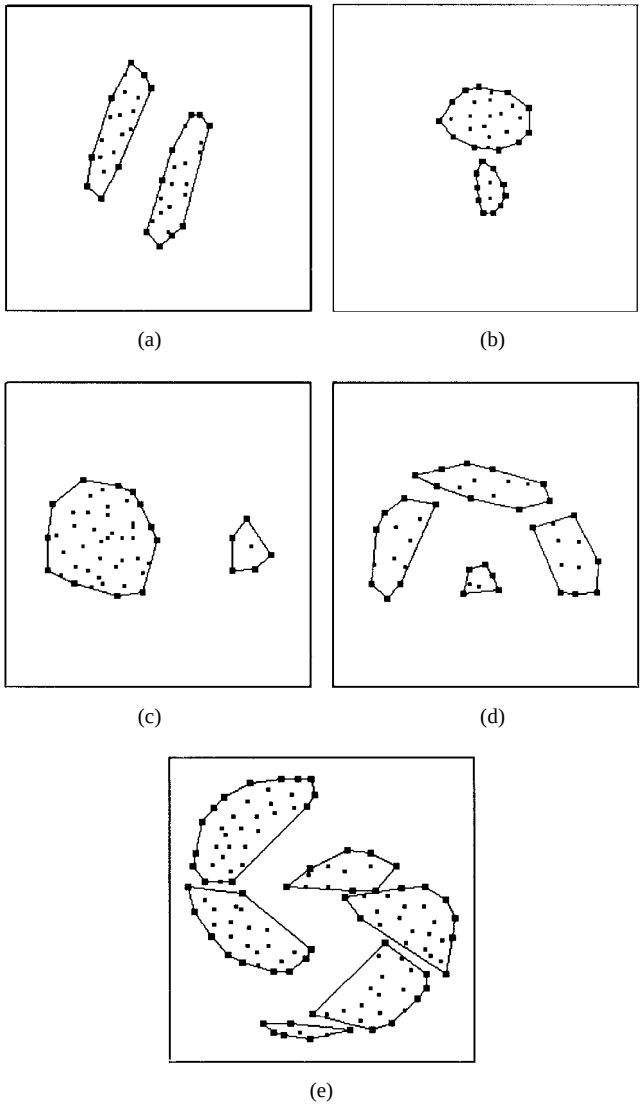


Fig. 11. The FCC results for various pattern data sets.

(ten data points), 50, and 100% randomly generated outlier data, respectively. As in the above examples, the maximum allowable merging size was set at $\Delta\beta = 1.5\beta$. Results show that our method can perform satisfactory in the presence of outliers.

Finally, as discussed in Section III, the order dependency on the incoming pattern data can be reduced by limiting the expansion of a cluster to a reasonably small size β and allowing clusters to merge within an allowable merging cluster size $\Delta\beta$. From experience, the value of $\Delta\beta$ is usually set slightly greater than or equal to β. However, if $\Delta\beta$ is set somewhat higher, the clustering result may greatly depend on the order of the input data. Fig. 14 shows the clustering results for randomly ordered input data sets. The maximum allowable merging size was set at $\Delta\beta = 2.5\beta$, where $\beta = 0.01$. As shown in Fig. 14(c), using a large $\Delta\beta$ value can give an undesirable clustering result.

D. Example 4: Iris Data Problem

As a frequently cited example, we show some results using the classical "iris data" problem [36]. This problem consists of

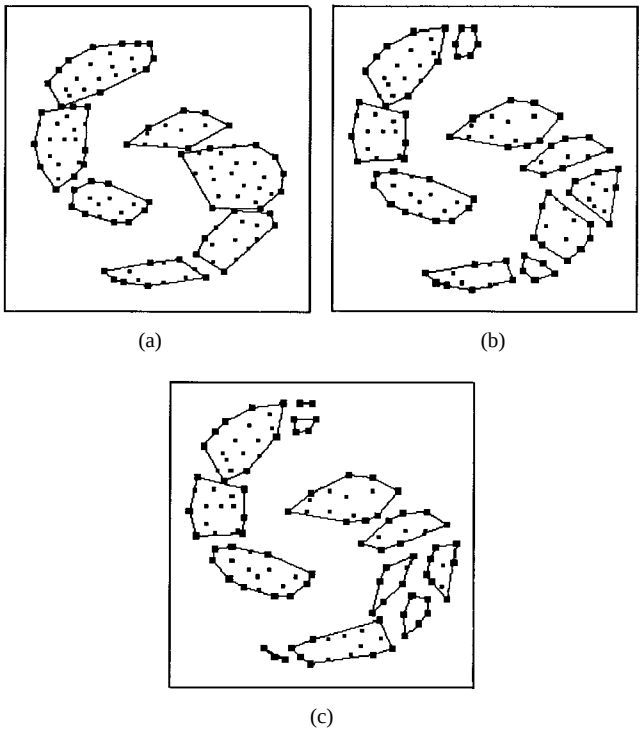


Fig. 12. Clustering results for various weighting factors (ε_0). (a) $\varepsilon_0 = 0.1$. (b) $\varepsilon_0 = 0.01$. (c) $\varepsilon_0 = 0.001$.

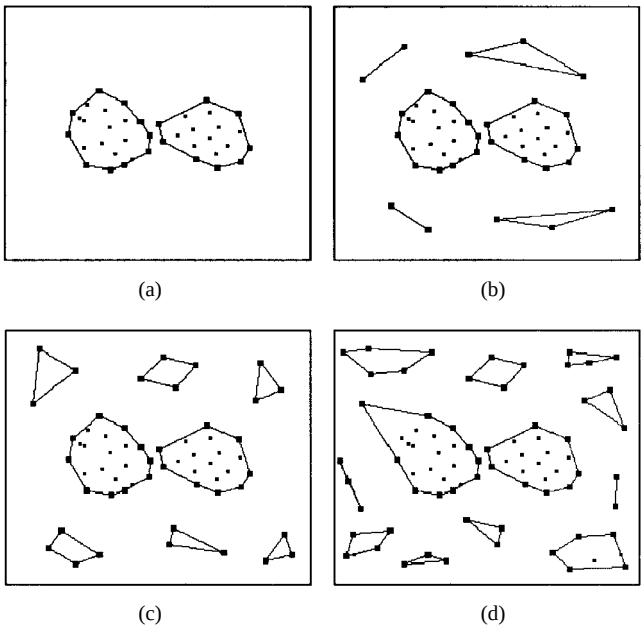
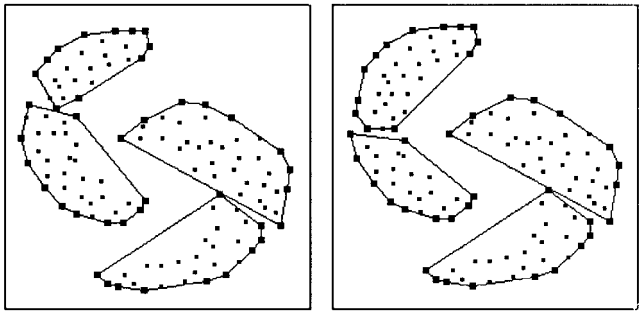
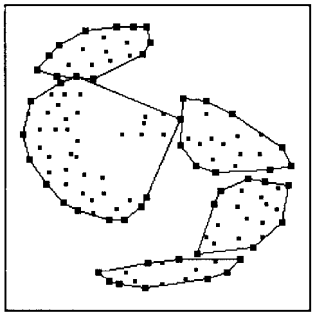


Fig. 13. Clustering results for a data set contaminated by (a) 0%, (b) 25%, (c) 50%, and (d) 100% randomly generated outliers.

three classes and four features (such as petal length) with 50 samples in each class. Each of the four features was mapped into the interval [0, 1] by dividing each feature of each pattern by eight. Features 1 and 2 are not very good because there is a lot of overlap in the feature values between the classes. However, features 3 and 4 are quite good for characterization of the classes. We used the "jack-knife" procedure, where 80% of the data from each class (i.e., 40 samples per class) was used



(a) (b)



(c)

Fig. 14. Clustering results for various randomly ordered input data for $\Delta\beta = 2.5\beta$, where $\beta = 0.01$.

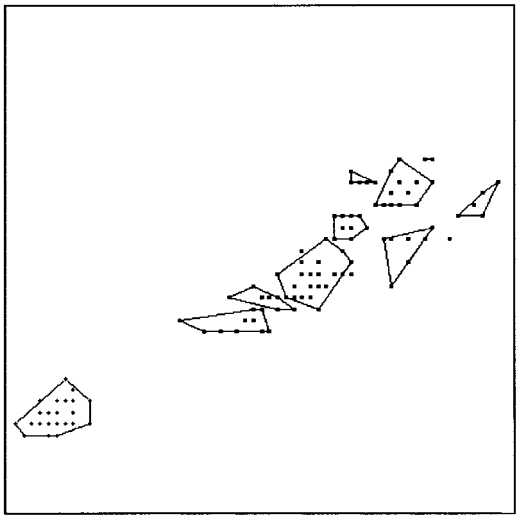


Fig. 15. The clustering results for the iris data using features 3 and 4.

for clustering and the remaining 20% for testing. This was repeated five times using different data sets for testing (i.e., every fifth sample in each class starting with sample number 1–5 was used) and the remainder for clustering. Fig. 15 shows clustering results using features 3 and 4 only. Results gave only three misclassifications with 13 clusters. It is to be noted that two misclassified patterns are overlapped. The parameter values for B and δB were both set at 0.01. The resulting confusion matrix is shown in Table II.

Next, we repeated the experiment using all four features by varying the maximum allowable cluster size parameter β. The results (i.e., confusion matrices) are shown in Fig. 16,

TABLE II
CONFUSION MATRIX FOR THE "IRIS DATA" USING FEATURES 3 AND 4

		Recognized class		
		1	2	3
True class	1	50	0	0
	2	0	47	3
	3	0	0	50

Maximum allowable cluster size (β) = 0.0200
Number of clusters = 7
Correct classification rate = 87.33%

		Recognized class		
		1	2	3
True class	1	50	0	0
	2	0	41	9
	3	0	10	40

(a)

Maximum allowable cluster size (β) = 0.0150
Number of clusters = 8
Correct classification rate = 88.67%

		Recognized class		
		1	2	3
True class	1	50	0	0
	2	0	42	8
	3	0	9	41

(b)

Maximum allowable cluster size (β) = 0.0125
Number of clusters = 10
Correct classification rate = 92.67%

		Recognized class		
		1	2	3
True class	1	50	0	0
	2	0	45	5
	3	0	6	44

(c)

Maximum allowable cluster size (β) = 0.0100
Number of clusters = 12
Correct classification rate = 95.33%

		Recognized class		
		1	2	3
True class	1	50	0	0
	2	0	47	3
	3	0	4	46

(d)

Fig. 16. Confusion matrices for various maximum allowable cluster sizes of the "iris data" using all four features. (a) $\beta = 0.0200$. (b) $\beta = 0.0150$. (c) $\beta = 0.0125$. (d) $\beta = 0.0100$.

where the range for B is 0.02 to 0.01. As before, the corresponding maximum allowable merging size was set at $\Delta\beta = \beta$. Fig. 16(d) shows the best performance when B is 0.01, which produced 12 clusters and a correct classification rate of 95.33%. In comparison with the FMMCNN algorithm, 14 clusters were produced and the correct classification resulted in a rate of 92.67%. This indicates that our method performs as well as the FMMCNN method.

E. Example 5: Pima Indians Diabetes Data Problem

As our final example, we show some results using the Pima Indians Diabetes database from the National Institute of Diabetes and Digestive and Kidney Diseases [37]. Several constraints were placed on the the selection of these instances that were obtained from a larger database. In particular, all the patients are females of at least 21 years old of Pima Indian heritage. The data consists of 768 instances each having eight

Maximum allowable cluster size (β) = 0.0200
Number of clusters = 18
Correct classification rate = 77.60%

		Recognized class 0	Recognized class 1
True class	0	395	105
	1	67	201

(a)

Maximum allowable cluster size (β) = 0.0150
Number of clusters = 22
Correct classification rate = 80.46%

		Recognized class 0	Recognized class 1
True class	0	410	90
	1	60	208

(b)

Maximum allowable cluster size (β) = 0.0125
Number of clusters = 25
Correct classification rate = 85.68%

		Recognized class 0	Recognized class 1
True class	0	432	68
	1	42	226

(c)

Maximum allowable cluster size (β) = 0.0100
Number of clusters = 27
Correct classification rate = 88.15%

		Recognized class 0	Recognized class 1
True class	0	441	59
	1	32	236

(d)

Fig. 17. Confusion matrices for various maximum allowable cluster sizes of the Pima Indians Diabetes data. (a) $\beta = 0.0200$. (b) $\beta = 0.0150$. (c) $\beta = 0.0125$. (d) $\beta = 0.0100$.

attributes where each attribute is numerically valued. Here, two classes are considered, where 500 instances for class 0 and 268 instances for class 1 are interpreted as "tested negative for diabetes" and "tested positive for diabetes," respectively. As in example 4, we used the "jack-knife" procedure, where 75% of the data from each class was used for clustering and the remaining 25% for testing. The clustering results are shown in Fig. 17, where the range for maximum allowable cluster size was varied from 0.02 to 0.01. The corresponding maximum allowable merging size was set at $\Delta\beta = \beta$. From the figure, Fig. 17(d) shows reasonable results that produced 27 clusters and a correct classification rate of 88.15%.

V. Summary and Conclusions

In this paper, we presented a two-level fuzzy clustering method (i.e., the FCC algorithm), which consisted of the FCCE and FCCM algorithms. In the FCCE algorithm, initial clusters were created by adaptively fitting convex polytopes to a given data set and each cluster was represented by the vertices of the convex polytope. Nonlinear membership functions were utilized to determine whether a new cluster would be created or how an existing cluster should be adjusted for cluster modification. In addition, a cluster merging algorithm (i.e., the FCCM algorithm) was also developed based on intra and intercluster expandability measures that were used to compute the volumes of the clusters. This was achieved by allowing a pair of clusters to merge under the constraint that its intercluster expandability measure lies within a maximally allowable merging cluster size ($\Delta\beta$) and the merged cluster would not overlap with any other neighboring clusters. This was developed to help reduce the sensitivity of the parameters (λ, e_0, θ, and β) used in the FCCE algorithm, and in turn further reduce the order dependency on the incoming pattern data.

Several numerical examples were illustrated to show the validity of our proposed methods. From the simulation results we can conclude that by using the FCC algorithm, cluster boundaries for an arbitrarily distributed data set can be satisfactorily obtained without *a priori* knowledge of the number of clusters in the data set. This is also true for the FMMCNN algorithm. Also, both the FCC and FMMCNN algorithms possess another desirable attribute; that is, they perform on-line clustering on a given data set as in others [38]–[40]. In addition, the FCC algorithm can be considered to have some advantages over the FMMCNN algorithm; that is, the FCC algorithm uses a more flexible prototype than the one used in the FMMCNN algorithm, namely convex polytopes versus hyperboxes. Therefore, a more "fitted" representation of the clusters can be obtained. Also, since our two-level clustering method performs clustering in both levels (i.e., clusters are expanded and merged), the order of the input patterns becomes less dependent. However, the FMMCNN algorithm is easier to implement than the FCC algorithm since it requires simple and fewer operations.

In regard to the computational complexity for the FCC algorithm, the order of magnitude for required computations can be obtained by adding the complexity for the FCCE and FCCM algorithms. In the case of the FCCE algorithm, the worst case occurs when all the input patterns become a vertex of a point cluster. Thus, in this case, each pattern vector must be compared with every vertex of all the point clusters. Hence, $N(N+1)/2$ comparisons will be needed for N input patterns. Therefore, the order of magnitude of computations for the FCCE algorithm can be considered as $O(N^2)$. In the case of the FCCM algorithm, $N(N-1)/2$ number of comparisons is needed for each input pattern that is considered as a cluster. Thus, the order of magnitude of computations for the FCCM algorithm can be considered as $O(N^2)$, as in the FCCE algorithm. Therefore, we can conclude that the order of magnitude for the FCC algorithm is $O(N^2)$.

As a final note, our method may be considered to be somewhat similar to the hierarchical clustering method of the agglomerative type [3], that is, in the sense that clusters are merged according to various strategies that are based on distance measures and criterion functions between the patterns of the training set. However, our method is different in the sense that each pattern in the data set is not initially considered as a separate cluster and is not an iterative method (i.e., our method gives a cluster result after only one pass though the data set.). Furthermore, the number of clusters are also assumed to be unknown in our method. As a possible comparison, Zahn's minimal spanning tree (MST) method does have a foundation in optimization and can be used for agglomerative clustering. In the method, all the input values

are provided *a priori* and $d_{\min}$, which is used to represent a cost function, is minimized for each input pattern relationship. However, since our clustering algorithm is performed on-line (i.e., all input patterns are not available *a priori*), it cannot be utilized for optimization as in the Zahn's MST method.

In conclusion, the simulation results suggest that our fuzzy convex clustering method works well on several data sets. However, to possibly improve our method, the following future works are currently under investigation.

1) In our proposed method, all the necessary initial parameters (i.e., $\lambda, e_0, \theta, \beta$, and $\Delta\beta$) were chosen though experience. As an example, the maximum allowable cluster size parameter (β) was initially set as a small value in order to reduce the order dependency on the incoming data set. However, for algorithms that are performed on line, it is difficult to find an optimum (or suboptimum) parameter set for an arbitrary given data set. In the recent paper on robust clustering by Dave and Krishnapuram, the authors discuss many *resolution-parameter*-based techniques [41]. For example, the Ohashi algorithm requires the estimation of the parameter alpha (i.e., the parameter used in the modified objective function for prototype based clustering which involves the idea of a class of outliers), the noise clustermethod requires the specification of δ (i.e., the constant distance between the noise cluster and all the other data points) and possibilistic C-means requires the specification of η (i.e., the distance at which the membership value of a point in a cluster becomes 0.5). The authors mention ways to estimate the parameter values for those methods. However, the methods discussed deal with iterative based clustering. Since our method is performed on-line they do not apply to our method. Hence, we cannot estimate the parameter values *a priori*, since the information on the input patterns is assumed to be unknown. However, in some applications, where there is *a priori* information on the domain size for the input patterns, the maximum allowable cluster and maximally allowable merging cluster sizes β and $\Delta\beta$ may be chosen according to the desired maximum size for a cluster. This may be chosen to be as the product of a small percentage of the domain length in each dimension. As for the parameter λ (i.e., the distance at which the membership of a pattern to a cluster becomes 0.5), it may be chosen depending on the desired "bandwidth" of the membership distribution for each cluster. As in the case of selecting β, if the domain of the patterns is known *a priori*, λ may be chosen accordingly. The weighting factor parameter e_0 for the cluster volume measure $\varepsilon_{\text{intra}}(\boldsymbol{A}^k)$ may be chosen depending on the desired weight for each cluster volume. As for the prespecified membership threshold value θ, it may be chosen depending on the desired membership requirements for each cluster. However, if there is absolutely no prior information on the input data, the parameters used in our method must be chosen carefully. If this is the case, one way is to select the parameter values pessimistically; that is, to anticipate that the clusters are small in size and are not well separated. We are currently developing a method to find such a solution by incorporating neural networks.
2) Our proposed method assumes the number of clusters to be unknown, but it is expected that when the number of clusters are given *a priori*, the FCC algorithm is capable of making synergistic combinations [42] with other well-known methods such as the FCM and PCM algorithms in which each clusters vertex can be considered as inputs to those algorithms.
3) We are currently applying our method to computer vision and robotic applications.

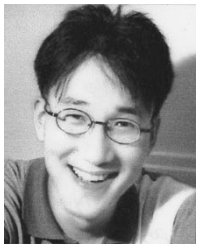
REFERENCES

[1] E. Backer and A. Jain, "A clustering performance measure based on fuzzy set decomposition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, pp. 66–74, 1981.
[2] J. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.
[3] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.
[4] J. Hartigan, *Clustering Algorithms*. New York: Wiley, 1975.
[5] A. Jain and R. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[6] J. Mao and A. Jain, "A self-organizing network for hyperellipsoidal clustering," *IEEE Trans. Neural Networks*, vol. 7, pp. 16–29, Jan. 1996.
[7] E. Ruspini, "A new approach to clustering," *Inform. Contr.*, vol. 15, no. 1, pp. 22–32, July 1969.
[8] J. Tou and R. Gonzalez, *Pattern Recognition Principles*. Reading, MA: Addison-Wesley, 1974.
[9] J. Bezdek and R. Hathaway, "Numerical convergence and interpretation of the fuzzy C-shells clustering algorithm," *IEEE Trans. Neural Networks*, vol. 3, pp. 787–793, Sept. 1992.
[10] R. Dave, "Use of the adaptive fuzzy clustering algorithm to detect lines in digital images," in *Proc. SPIE Conf. Intell. Robots Comput. Vision*, 1989, vol. 1192, no. 2, pp. 600–611.
[11] ———, "Fuzzy-shell clustering and applications to circle detection in digital images," *Int. J. Gen. Syst.*, vol. 16, pp. 343–355, 1990.
[12] ———, "New measures for evaluating fuzzy partitions induced through C-shells clustering," in *Proc. SPIE Conf. Intell. Robots Comput. Vision X: Algorithms Tech.*, Boston, MA, Nov. 1991, pp. 406–414.
[13] ———, "Robust fuzzy clustering algorithms," in *Proc. 2nd IEEE Int. Conf. Fuzzy Syst.*, San Francisco, CA, Mar.–Apr. 1993, vol. I, pp. 1281–1286.
[14] R. Dave and K. Bhaswan, "Adaptive C-shell clustering and detection of ellipses," *IEEE Trans. Neural Networks*, vol. 3, pp. 643–662, Sept. 1992.
[15] I. Gath and A. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, pp. 773–781, July 1989.
[16] E. Gustafson and W. Kessel, "Fuzzy clustering with a fuzzy covariance matrix," in *Proc. IEEE Conf. Decision Contr.*, San Diego, CA, 1979, pp. 761–766.
[17] R. Krishnapuram, H. Frigui, and O. Nasraoui, "The fuzzy shell clustering algorithms for boundary detection and pattern recognition," in *Proc. SPIE Conf. Intell. Robots Comput. Vision X: Algorithms Tech.*, Boston, MA, Nov. 1991, pp. 458–465.
[18] ———, "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation—Part I," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 29–43, Feb. 1995.
[19] ———, "Fuzzy and possibilistic shell clustering algorithms and their application to boundary detection and surface approximation—Part II," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 44–60, Feb. 1995.
[20] R. Krishnapuram and J. Keller, "A possibilistic approach to clustering," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 98–110, May 1993.
[21] R. Krishnapuram, O. Nasraoui, and H. Frigui, "The fuzzy C spherical shells algorithm: A new approach," *IEEE Trans. Neural Networks*, vol. 3, pp. 787–793, Sept. 1992.
[22] L. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets Syst.*, vol. 1, pp. 3–28, 1978.
[23] ———, "The theory of approximate reasoning," in *Machine Intelligence—Volume 9*, J. Hayes, D. Michie, and L. Mikulich, Eds. New York: Halstead, 1979, pp. 149–194.

[24] S. Abe and M. Lan, "A method for fuzzy rules extraction directly from numerical data and its application to pattern classification," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 18–28, Feb. 1995.
[25] P. Simpson, "Fuzzy min–max neural networks—Part 1: Classification," *IEEE Trans. Neural Networks*, vol. 3, pp. 776–786, Sept. 1992.
[26] _____, "Fuzzy min–max neural networks—Part 2: Clustering," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 32–45, May 1993.
[27] G. Carpenter, S. Grossberg, and D. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Networks*, vol. 4, pp. 4759–771, 1991.
[28] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds, and D. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. Neural Networks*, vol. 3, pp. 698–713, Sept. 1992.
[29] P. Simpson, "Fuzzy adaptive resonance theory," in *Southern Illinois Univ. Neuroeng. Workshop*, Carbondale, IL, Sept. 6–7, 1990.
[30] I. H. Suh, J. H. Kim, and F. C.-H. Rhee, "Fuzzy clustering involving convex polytopes," in *Proc. 1996 Int. Conf. Fuzzy Syst.*, New Orleans, LA, vol. III, pp. 2014–2019, Sept. 8–11, 1996.
[31] J. Goldberg, *Matrix Theory with Applications*. New York: McGraw-Hill, 1992.
[32] D. Luenberger, *Optimization by Vector Space Methods*. New York: Wiley, 1969.
[33] H. Zimmermann and P. Zysno, "Quantifying vagueness in decision models," *Eur. J. Operat. Res.*, vol. 22, pp. 148–158, 1985.
[34] G. Nagy, "State of the art in pattern recognition," *Proc. IEEE*, vol. 56, pp. 836–882, May 1968.
[35] R. Lippmann, "An introduction to computing with neural network," *IEEE Acoust., Speech, Signal Processing Mag.*, vol. 61, pp. 4–22, 1987.
[36] R. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen.*, vol. 7, pt. II, pp. 179–188, 1936.
[37] C. Merz and P. Murphy, UCI Repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, Univ. California, Dept. Inform. Comput. Sci., Irvine, CA.
[38] S. Newton and S. Mitra, "Self-organizing leader clustering in a neural network using a fuzzy learning rule," *SPIE Proc. Adaptive Signal Processing*, vol. 1565, July 1991.
[39] S. Newton, S. Pemmaraju, and S. Mitra, "Adaptive fuzzy leader clustering of complex data sets in pattern recognition," *IEEE Trans. Neural Networks*, vol. 3, pp. 794–800, Sept. 1992.
[40] J.-C. Lin and W.-H. Tsai, "Feature-preserving clustering of 2-D Data for two-class problems using analytical formulas: An automatic and fast approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, pp. 554–560, May 1994.
[41] R. Dave and R. Krishnapuram, "Robust clustering methods: A unified view," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 270–293, May 1997.
[42] H. Tagaki, "Fusion technology for fuzzy theory and neural network: Survey and future directions," in *Proc. Int. Conf. Fuzzy Logic Neural Networks*, Iizuka, Japan, July 1990, pp. 13–26.

Il Hong Suh (M'89) was born in Seoul, Korea. He received the B.S. degree in electronic engineering from Seoul National University, Korea, in 1977, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Seoul, in 1979 and 1982, respectively.

From 1982 to 1985, he was a Senior Research Engineer at the Technical Center of Daewoo Heavy Industries, Ltd., Inchon, Korea, where he was involved in the research on machine vision and the development of the Daewoo NOVA10 robot controller. From 1985 to 1986 he joined the Systems Control Laboratory of KAIST as a part-time Research Fellow for an automation-related Korea National Project. In 1987 he was a visiting Research Scientist at the Robotics Division, CRIM, at the University of Michigan, Ann Arbor. Since 1985, he has been with the Department of Electronic Engineering, Hanyang University, Korea, where he is currently a Professor. He has been serving as the Vice Dean of academic affairs at Hanyang University, Seoul, Korea, from 1996. His research interests include sensor-based control of robot manipulators, coordination of multiple robot arms, robust control, intelligent control involving fuzzy logic, and neural networks.



Jae-Hyun Kim received the B.S., M.S., and Ph.D. degrees in electronic engineering from Hanyang University, Seoul, Korea, in 1991, 1993, and 1998, respectively.

He is currently the Technical Director and Consultant for Right-Tek Systems Incorporated, Seoul, Korea. His research interests include intelligent fuzzy system, pattern recognition, and machine learning.

Frank Chung-Hoon Rhee (S'91–M'93) was born March 3, 1963, in Seoul, Korea. He received the B.S. degree in electrical engineering from the University of Southern California (USC), Los Angeles, in 1985, and the M.S. and Ph.D. degrees in electrical engineering from the University of Missouri, Columbia, in 1987 and 1993, respectively.

From 1990 to 1993, he was a Research Assistant in the Department of Electrical and Computer Engineering, University of Missouri, Columbia, where his research interests included applications of computer vision, pattern recognition, neural networks, and fuzzy set theory. From 1994 to 1995 he was a Senior Member of the Engineering Staff in the High-Speed Network Access Section at the Electronics and Telecommunications Research Institute (ETRI), Taejon, Korea, where his work involved applications of group communications. Since September 1995 he has been a faculty member in the Department of Electronic Engineering at Hanyang University, Seoul, Korea, where he is currently an Assistant Professor. His current research interests include applications of computer vision, pattern recognition, and all aspects of computational intelligence.

Dr. Rhee was a cochair for Image Processing and Pattern Recognition for the 1997 IEEE International Conference on Neural Networks, Houston, TX, June 1997. He is a member of the Korea Institute of Telematics and Electronics, the Korea Fuzzy Logic and Intelligent Systems Society, and the Acoustical Society of Korea.