An Efficient Algorithm for l_1 Fuzzy c-Means and Its Termination

Sadaaki Miyamoto* and Yudi Agusta**

November 15, 1995

ISE-TR-95-127

*Institute of Information Sciences and Electronics
University of Tsukuba
Ibaraki 305, Japan

**Graduate School of Engineering
University of Tokushima
Tokushima 770, Japan

An efficient algorithm for ℓ_1 fuzzy c-means and its termination

Sadaaki Miyamoto* and Yudi Agusta**
*Institute of Information Sciences and Electronics
University of Tsukuba, Ibaraki 305, Japan
**Graduate School of Engineering, University of Tokushima
Tokushima 770, Japan

Abstract

A fast algorithm for calculating cluster centers in the iteration procedure of the ℓ_1 fuzzy c-means clustering is proposed. The algorithm is a simple sequential search on the set of coordinates of data points. The complexity of calculation of each cluster center is the order of the number of data points except that the coordinates should be sorted before the iteration begins. The efficiency is comparable to the computation of a center in the ordinary Euclidean fuzzy c-means. Thus, the ℓ_1 fuzzy c-means algorithm is efficient and is applicable to large data sets. It is proved that the algorithm terminates after a finite number of iterations and the upper bound for the number of iterations is estimated. Numerical examples including a set of 10,000 data points are shown.

1 Introduction

The ℓ_1 space has been considered to be a natural space for statitical analysis in addition to Euclidean space, and recently this space has attracted researchers' interest (e.g., Devroye and Gyorfi, 1984; Nahorski, 1992).

In the case of fuzzy c-means (Dunn, 1974; Bezdek,1981), the ℓ_1 space based fuzzy c-means have also been considered in recent years by Jajuga (1991) and by Bobrowski and Bezdek (1991); the latter discusses fuzzy c-means in the ℓ_1 and and also ℓ_∞ spaces.

These two studies (Jajuga, 1991; Bobrowski and Bezdek, 1991) have difficulties in optimizing the fuzzy c-means model in the ℓ_1 and ℓ_∞ spaces. The general fuzzy c-means algorithm is an iterative procedure in which the step of determining grades of memberships and that of determining cluster centers are repeated. A unified formula can be used for the determination of grades for different distances, whereas the calculation of cluster centers strongly depends on a selected distance. Cluster centers in the inner product spaces are derived by a simple formula of weighted averages. For the ℓ_1 space, they have proposed iterative subprocedures for calculating cluster centers. The calculation of cluster centers in these studies requires much more computation than those in inner product spaces.

This paper reveals that, in the case of the ℓ_1 space, a small amount of computation is sufficient for calculating the cluster centers. Although no simple formula can be used, the computational complexity is low and comparable to that of fuzzy c-means for inner product spaces. Namely, each component of a cluster center is the minimizing element of a piecewise affine function. The component is calculated by a linear search on the derivative of the function, which is remarkably simple.

Convergence of an algorithm has frequently been referred to in literature of crisp and fuzzy clustering. The convergence does not imply that the result is good, however. For example, the crisp c-means algorithm is proved to be convergent (Anderberg, 1973, pp.165-166), but the result is not necessarily the optimal solution. In contrast, convergence in other fields of mathematical analysis more or less implies that the sequence approaches a desired solution. To avoid confusion, we use the term *termination* instead of convergence in order to simply mean that an algorithm is guaranteed to stop eventually, with no additional implication.

This paper presents a theorem about termination of the algorithm in the ℓ_1 space. Such a result has not been reported in foregoing studies. There have been two approaches for discussing termination of c-means algorithms: finiteness of combinations of data allocations to clusters is used in the crisp c-means (Anderberg, 1973, p.165); uniqueness of the optimal

solution in each step is used in Euclidean fuzzy c-means (Bezdek, 1981; Hathaway, Bezdek, Tucker, 1987). The present method uses both finiteness and the uniqueness: the finiteness for possible cluster centers and the uniqueness of the minimum of strictly convex functions.

Numerical examples are given to show that the algorithm actually works well on a large set of data with a small computation time. Indeed, an example includes 10,000 data points.

2 Cluster centers in the ℓ_1 fuzzy c-means

The problem herein is that n objects, each of which is represented by a p-dimensional real vector $x_k = (x_{k1}, ..., x_{kp}) \in \mathbf{R}^p$, $k = 1, ..., n$, should be divided into c fuzzy clusters. Namely, the grade u_{ik}, $1 \le i \le c$, $1 \le k \le n$, by which the object k belongs to the cluster i should be determined.

For each object k, the grades of membership should satisfy the conditions of a fuzzy partition:

$$\sum_{i=1}^{c} u_{ik} = 1, \quad 1 \le k \le n; \qquad 0 \le u_{ik} \le 1, \quad 1 \le i \le c, \quad 1 \le k \le n. \tag{1}$$

The formulation by Bezdek (1981) is by optimization of the objective function

$$J(U, v) = \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^m d(x_k, v_i)$$

in which $d(x, v)$ is a measure of dissimilarity between x and v, m is a real parameter such that $m > 1$, v_i is the center of the fuzzy cluster i, and $U = (u_{ik})$ and $v = (v_1, ..., v_c) \in \mathbf{R}^{cp}$.

Bezdek (1981) takes $d(x, v)$ to be the square of any inner product induced distance. Here we assume that d is the ℓ_1 norm:

$$d(x_k, v_i) = \|x_k - v_i\|_1 = \sum_{j=1}^{p} |x_{kj} - v_{ij}|$$

where $v_i = (v_{i1}, ..., v_{ip})$. Namely, the objective function is

$$J(U, v) = \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^m \|x_k - v_i\|_1 \tag{2}$$

for which the constraints are shown in (1). In other words, the admissible region

$$M = \{(u_{ik}) \mid \sum_{i=1}^{c} u_{ik} = 1,\ 1 \le k \le n;\ 0 \le u_{ik} \le 1,\ 1 \le i \le c,\ 1 \le k \le n\}$$

is used. The present formulation is called the ℓ_1 fuzzy c-means or ℓ_1 FCM here. This formulation is not new: Jajuga (1991), and Bobrowski and Bezdek (1991) proposed ℓ_1 FCM and methods of calculating cluster centers v_i. Here we show a better method of calculating the centers and the termination of the algorithm.

It is well-known that the direct optimization of J by (U, v)

$$\min_{U \in M, v \in \mathbf{R}^{cp}} J(U, v)$$

is difficult. A two stage iteration algorithm (called alternating optimization) is often used.

A General FCM Algorithm (Bezdek, 1981)

(a) Initialize $U^{(0)}$; Set $s = 0$.

(b) Calculate cluster centers $v^{(s)} = (v_1^{(s)}, ..., v_c^{(s)})$ that minimize $J(U^{(s)}, \cdot)$:

$$J(U^{(s)}, v^{(s)}) = \min_{v \in \mathbf{R}^{cp}} J(U^{(s)}, v). \tag{3}$$

(c) Update U: calculate $U^{(s+1)}$ that minimize $J(\cdot, v^{(s)})$:

$$J(U^{(s+1)}, v^{(s)}) = \min_{U \in M} J(U, v^{(s)}). \tag{4}$$

(d) Check stopping criterion using a given $\epsilon > 0$ and a suitable matrix norm: if $\|U^{(s+1)} - U^{(s)}\| < \epsilon$, then stop; otherwise $s = s + 1$ and go to (b).

This general procedure can also be used for ℓ_1 FCM. Since no concrete methods for calculating $U^{(s)}$ and $v^{(s)}$ are described above, we consider them in the following.

In general, calculation of $U^{(s+1)}$ does not depend on a particular choice of a norm. It is well-known that u_{ik} is easily derived by using Lagrange multipliers (Bezdek, 1981). Namely, for x_k such that $x_k \neq v_i$, $i = 1, ..., c$, and $m > 1$,

$$u_{ik} = \frac{1}{(\sum_{j=1}^{c} \frac{\|x_k - v_i\|_1}{\|x_k - v_j\|_1})^{\frac{1}{m-1}}}. \tag{5}$$

For x_k such that there exists v_i that satisfies $x_k = v_i$, let $W_k = \{i | x_k = v_i\}$ and take an arbitrary $u_{ik} \in [0,1]$, $i = 1, ..., c$, such that

$$\sum_{i \in W_k} u_{ik} = 1; \qquad u_{ik} = 0, \quad i \notin W_k \tag{6}$$

In the latter case, the solution u_{ik} for a given x_k is not unique if and only if W_k includes more than one element.

On the other hand, calculation of the cluster centers is not simple except the case of the inner product spaces where

$$v_i = \frac{\sum_{k=1}^{n} (u_{ik})^m x_k}{\sum_{k=1}^{n} (u_{ik})^m}.$$

Bobrowski and Bezdek (1991) propose a basis exchange algorithm for cluster centers in ℓ_1 and ℓ_∞ FCM; Jajuga (1991) considers an iterative algorithm for ℓ_1 FCM.

For the derivation of a far simpler algorithm for ℓ_1 FCM, notice that

$$\begin{aligned} J(U,v) &= \sum_{i=1}^{c} \sum_{k=1}^{n} (u_{ik})^m \|x_k - v_i\|_1 \\ &= \sum_{i=1}^{c} \sum_{j=1}^{p} \sum_{k=1}^{n} (u_{ik})^m |x_{kj} - v_{ij}|. \end{aligned}$$

We put

$$F_{ij}(w) = \sum_{k=1}^{n} (u_{ik})^m |x_{kj} - w|$$

as a function of a real variable w. Then

$$J(U,v) = \sum_{i=1}^{c} \sum_{j=1}^{p} F_{ij}(v_{ij})$$

in which U does not represent variables but parameters.

In determining cluster centers, each $F_{ij}(w)$, $1 \le i \le c$, $1 \le j \le p$, should be minimized with respect to the real variable w without any constraint.

It is easily seen that the following properties are valid. The proofs are omitted.

(A) $F_{ij}(w)$ is a convex, piecewise affine function.

(B) The intersection between the set $X_j = \{x_{1j}, ..., x_{nj}\}$ and the set of the solutions of

$$\min_{w \in \mathbf{R}} F_{ij}(w) \tag{7}$$

is not empty. In other words, at least one of the j-th coordinates of the points $x_1, ..., x_n$ is the optimal solution. In particular when the solution of (7) is unique, it is included in X_j.

In view of the property (B), we limit ourselves to the minimization problem

$$\min_{w \in X_j} F_{ij}(w) \tag{8}$$

instead of (7). The reason for this limitation is to simplify the description. Indeed, when the solutions of (8) are found, the solution set of (7) becomes obvious: the smallest interval in which the solutions of (8) are included. Thus the limitation to (8) is harmless.

Since no simple formula for cluster centers in ℓ_1 FCM seems to be available, an efficient algorithm of search for the solution of (8) is considered using the above properties. Two ideas are used in the following algorithm: ordering of $\{x_{kj}\}$ and derivative of F_{ij}. We assume that when $\{x_{1j}, ..., x_{nj}\}$ is ordered, first subscripts are changed using a permutation function $q_j(k)$, $k = 1, ..., n$, that is, $x_{q_j(1)j} \leq x_{q_j(2)j} \leq ... \leq x_{q_j(n)j}$. Using $\{x_{q_j(k)j}\}$,

$$F_{ij}(w) = \sum_{k=1}^{n} (u_{iq_j(k)})^m |w - x_{q_j(k)j}|. \tag{9}$$

Although $F_{ij}(w)$ is not differentiable on $\mathbf{R}$, we extend the derivative of $F_{ij}(w)$ on $\{x_{q_j(k)j}\}$:

$$dF_{ij}^{+}(w) = \sum_{k=1}^{n} (u_{iq_j(k)})^m sign^{+}(w - x_{q_j(k)j}) \tag{10}$$

where

$$sign^{+}(z) = \begin{cases} 1 & (z \geq 0), \\ -1 & (z < 0). \end{cases}$$

Thus, $dF_{ij}^{+}(w)$ is a step function which is right continuous and monotone nondecreasing in view of its convexity and piecewise affine property. Now, it is easy to see that the minimizing element for (8) is one of $x_{q_j(k)j}$ at which $dF_{ij}^{+}(w)$ changes its sign. More precisely, $x_{q_j(t)j}$ is

the optimal solution of (8) if and only if $dF_{ij}^{+}(w) < 0$ for $w < x_{q_j(t)j}$ and $dF_{ij}^{+}(w) \geq 0$ for $w \geq x_{q_j(t)j}$.

Let $w = x_{q_j(r)j}$, then

$$dF_{ij}^{+}(x_{q_j(r)j}) = \sum_{k=1}^{r}(u_{iq_j(k)})^m - \sum_{k=r+1}^{n}(u_{iq_j(k)})^m$$

These observations lead us to the next algorithm.

```
begin
  S := -Σ_{k=1}^{n} (u_ik)^m;
  r := 0;
  while ( S < 0 )do begin
    r := r + 1;
    S := S + 2(u_{iq_j(r)})^m
  end;
  output v_ij = x_{q_j(r)j} as the j-th coordinate of
  the cluster center v_i
end.
```

It is easy to see that this algorithm correctly calculates one coordinate of the cluster center: the solution of (8).

This algorithm is a simple linear search on nodes of the piecewise affine function. It is very efficient, since additions, conditional branches, and the calculation of $(u_{ik})^m$ from (u_{ik}) of $\mathcal{O}(n)$ should be processed. Furthermore, $\mathcal{O}(n)$ is optimal. To see this, it is sufficient to note that examination of all u_{ik}, $1 \leq k \leq n$, is necessary in order to calculate a coordinate of a cluster center, since by modifying the value of u_{ik} arbitrarily, any element of x_{kj}, $1 \leq k \leq n$, can be the solution. The lower bound is thus $\mathcal{O}(n)$ and hence the complexity of the above algorithm is optimal.

Thus, the calculation of cluster centers in ℓ_1 FCM is simple and does not require much computation time, except that the ordering of $\{x_{kj}\}$ for each coordinate j is necessary. Notice that the ordering is performed only once before the iteration of FCM. Further ordering is unnecessary during the iteration.

Thus, the computational complexity for calculating a v in the FCM iteration is $\mathcal{O}(npc)$; the complexity of the ordering before the iteration is $\mathcal{O}(np \log n)$; mostly p and c are small.

Notice that such estimation of the complexity is difficult in the former algorithms (Jajuga, 1991; Bobrowski and Bezdek, 1991).

3 Termination of ℓ_1 fuzzy c-means algorithm

We prove a theorem of termination of our ℓ_1 FCM algorithm which is a modified version of the algorithm in the previous section.

Two modifications are necessary. First, although the solution $U^{(s)}$ is not unique in general, it is *made* to be unique by a simple trick. As we have noted earlier, $x_k = v_i$ for more than one v_i is not impossible: it may occur that $x_k = v_{i_1} = ... = v_{i_h}$ for $W_k = \{i_1, ..., i_h\}$, $i_1 < ... < i_h$, in (6). In such a case we select a strategy of allocating unity to the first cluster:

$$u_{i_1 k} = 1, \tag{11}$$

$$u_{jk} = 0, \quad j \neq i_1, \tag{12}$$

in order to make the solution $U^{(s)}$ unique.

Second, in the calculation of $v^{(s)}$, we take $v^{(s)} = v^{(s-1)}$ whenever

$$J(U^{(s)}, v^{(s)}) = \min_{v \in \mathbf{R}^{cp}} J(U^{(s)}, v) = J(U^{(s)}, v^{(s-1)}). \tag{13}$$

In other words, if the value of the objective function is not improved, the procedure outputs the previous center.

In the following algorithm CC, the input is the ordered sequence $\{x_{q_j(k)j}\}$ and also the previous center $v_i^{(s-1)} = (v_{i1}^{(s-1)}, ..., v_{ip}^{(s-1)})$; the output is the new center $v_i = (v_{i1}, ..., v_{ip})$. In general the center is not unique, and moreover when (13) is satisfied, the previous $v_i^{(s-1)}$ is one of the minimizing elements. In this case the algorithm outputs $v_i = v_i^{(s-1)}$. The algorithm uses a set W which stores all minimizing elements for (8). Notice that all minimizing elements are contiguous in view of the convexity of $F_{ij}(w)$. Moreover, if $W = \{x_{q_j(r)j}, x_{q_j(r+1)j}, ..., x_{q_j(r+t)j}\}$, then

$$dF_{ij}^{+}(x_{q_j(r)j}) = ... = dF_{ij}^{+}(x_{q_j(r+t-1)j}) = 0, \quad t \geq 1$$

Algorithm CC
begin
  $S := -\sum_{k=1}^{n}(u_{ik})^m$;
  $r := 0$;
  while ($S < 0$)do begin
    $r := r + 1$;
    $S := S + 2(u_{iq_j(r)})^m$
  end;
  $W := \emptyset$;
  while ($S = 0$)do begin
    $W := W \cup \{x_{q_j(r)j}\}$;
    $r := r + 1$;
    $S := S + 2(u_{iq_j(r)})^m$
  end;
  $W := W \cup \{x_{q_j(r)j}\}$;
  if $v_{ij}^{(s-1)} \in W$ then $v_{ij} := v_{ij}^{(s-1)}$
    else take an arbitrary element $z \in W$ and $v_{ij} := z$;
  output v_{ij}
end.

The upper bound of the number of iterations can be estimated. For this purpose the following set is useful:

$$Z = \{y = (y_1, ..., y_p) \mid y_j \in \{x_{1j}, ..., x_{nj}\}, \quad j = 1, ..., p\}.$$

Let Z^c be the Cartesian product of Z: $Z^c = \overbrace{Z \times Z \times ... \times Z}^{c}$. The number of elements in Z^c is denoted by $|Z^c|$.

Now we have

Theorem. For any $\epsilon > 0$ used in FCM, the FCM algorithm with (5), (11), and (12) for calculating u_{ik} and CC for calculating v_i terminates after a finite number of iterations. The upper bound of the number of iterations is given by $|Z^c| + 1$. During the iteration, the value of the objective function is monotone nonincreasing:

$$J(U^{(s)}, v^{(s)}) \geq J(U^{(s+1)}, v^{(s)}) \geq J(U^{(s+1)}, v^{(s+1)}), \quad s = 0, 1, 2, ... \tag{14}$$

The proof consists of a few steps.

Lemma 1. The objective function is monotone nonincreasing, i.e., (14) holds. Moreover, if

$$J(U^{(s)}, v^{(s-1)}) = J(U^{(s)}, v^{(s)}),$$

for some $s \geq 1$, then $v^{(s)} = v^{(s-1)}$.

The proof is straightforward, since

$$\begin{aligned} J(U^{(s+1)}, v^{(s)}) &= \min_{U \in M} J(U, v^{(s)}) \leq J(U^{(s)}, v^{(s)}) \\ J(U^{(s)}, v^{(s)}) &= \min_{v \in \mathbf{R}^{cp}} J(U^{(s)}, v) \leq J(U^{(s)}, v^{(s-1)}). \end{aligned}$$

Moreover algorithm CC guarantees $v^{(s)} = v^{(s-1)}$ whenever $J(U^{(s)}, v^{(s)}) = J(U^{(s)}, v^{(s-1)})$.

Lemma 2. The solution $U^{(s+1)}$ of

$$J(U^{(s+1)}, v^{(s)}) = \min_{U \in M} J(U, v^{(s)}) \tag{15}$$

is unique. Therefore if

$$J(U^{(s+1)}, v^{(s)}) = J(U^{(s)}, v^{(s)}),$$

then $U^{(s+1)} = U^{(s)}$.

For the proof, let C and D be subsets of the set of integers $\{1, 2, ..., n\}$ such that $C \cup D = \{1, 2, ..., n\}$: for any $k \in C$, $x_k \neq v_i$ for all $i = 1, ..., c$; for any $k \in D$, there exists some $i \in \{1, .., c\}$ such that $x_k = v_i$. Then for an arbitrary v,

$$J(U, v) = \sum_{i=1}^{c} \sum_{k \in C} (u_{ik})^m \|x_k - v_i\|_1 + \sum_{i=1}^{c} \sum_{k \in D} (u_{ik})^m \|x_k - v_i\|_1 \tag{16}$$

The objective function is thus divided into two parts: each part can be independently optimized with respect to u_{ik} regardless of the other. For the first part, u_{ik} given by (5) is the unique optimal solution, since the function is strictly convex with respect to u_{ik} (see, e.g., Rockafellar, 1970, p.263). For the second part, we select the strategy (11) and (12) for obtaining the unique minimizing solution (cf. Hathaway, Bezdek, Tucker, 1987).

(Proof of the theorem) Since algorithm CC outputs one of $\{x_{1j}, ..., x_{nj}\}$ as the j-th coordinate of v_i, $v_i^{(s)} \in Z$ and hence $v^{(s)} \in Z^c$. If $J(U^{(s)}, v^{(s)}) = J(U^{(s)}, v^{(s-1)})$, $v^{(s)} = v^{(s-1)}$ by Lemma 1. In this case

$$J(U^{(s+1)}, v^{(s)}) = \min_{U \in M} J(U, v^{(s)}) = \min_{U \in M} J(U, v^{(s-1)}) = J(U^{(s)}, v^{(s-1)}) = J(U^{(s)}, v^{(s)})$$

which means $U^{(s+1)} = U^{(s)}$ by Lemma 2. Hence $\|U^{(s+1)} - U^{(s)}\| < \epsilon$ and FCM stops.

If $v^{(s)} \neq v^{(s-1)}$, then $J(U^{(s)}, v^{(s)}) < J(U^{(s)}, v^{(s-1)})$ by Lemma 1. In this case $v^{(r)} \neq v^{(s-1)}$ for all $r \geq s$. Assume the contrary, that there exists $r \geq s$ such that $v^{(r)} = v^{(s-1)}$. Since the value of the objective function is monotone nonincreasing,

$$J(U^{(s)}, v^{(s-1)}) > J(U^{(s)}, v^{(s)}) \geq ... \geq J(U^{(r)}, v^{(r)}) = J(U^{(r)}, v^{(s-1)}).$$

On the other hand,

$$J(U^{(r)}, v^{(s-1)}) \geq \min_{U \in M} J(U, v^{(s-1)}) = J(U^{(s)}, v^{(s-1)}).$$

Hence we have a contradiction.

Thus, while $J(U, v)$ is strictly monotone decreasing, $v^{(s)}$ visits different points of Z^c, and when the optimal value of J is not improved, the algorithm stops. Since Z^c is a finite set and except when $v^{(s)} = v^{(s-1)}$, $v^{(s)}$ cannot visit the same point of Z^c more than once. Therefore the algorithm terminates after a finite number, i.e., at most the number of elements in Z^c plus one, of iterations. Thus the theorem is proved.

This theorem guarantees termination of our ℓ_1 FCM algorithm and the algorithm is reasonable since the value of the objective function is monotone decreasing. The present theorem is the first result that gives termination of an ℓ_1 FCM algorithm. Bobrowski and Bezdek (1991) show only the convergence of a subprocedure of ℓ_1 FCM.

4 Numerical examples

Two examples are considered. Throughout these examples we take the parameter $m = 2$, the number of clusters $c = 2$, and the dimension $p = 2$. In each example, a region is

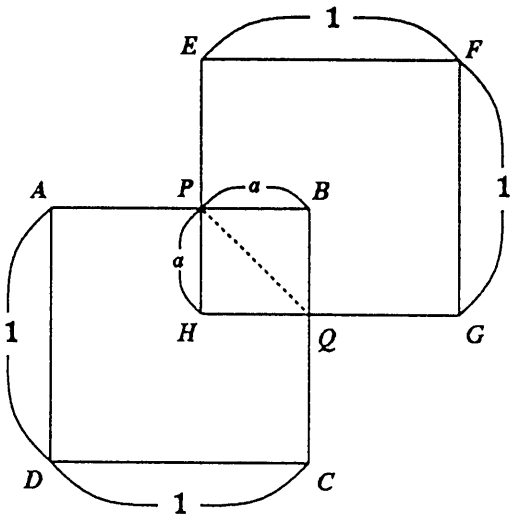

Figure 1: Two squares $ABCD$ and $EFGH$ with the intersection $PBQH$ used for Example 1.

considered and data points are scattered over the region.

The region in Example 1 is as follows. Two squares $ABCD$ and $EFGH$ of unit size with the intersecting square $PBQH$ are considered, as shown in Figure 1. The square $PBQH$ has edge length a $(0 < a < 1)$. Data points have been scattered over the area surrounded by $APEFGQCDA$ using the uniformly distributed random numbers. Two different cases of $1,000$ data points (called Example 1.1 hereafter) and $10,000$ data points (called Example 1.2) are considered.

In Example 2, the region in Example 1 is rotated by $-45°$ so that the line connecting D, H, B, and F becomes horizontal (Figure 2). Two cases of $n = 1,000$ (Example 2.1) and $n = 10,000$ (Example 2.2) of the randomly scattered data are considered likewise.

The initial value for the grade $u_{1k}^{(0)}$ for each x_k is generated by the pseudo random number uniformly distributed over $[0,1]$; $u_{2k}^{(0)} = 1 - u_{1k}^{(0)}$ to form a fuzzy partition. Ten trials with different initial grades $U^{(0)}$ have been carried out for each case of the two examples.

Fuzzy clusters are transformed into crisp clusters using the α - cut of $\alpha = 0.5$. Then, a measure of *misclassification* is introduced for a quantitative evaluation of the results. Namely, when a data point that is in the left and lower side of the broken line segment

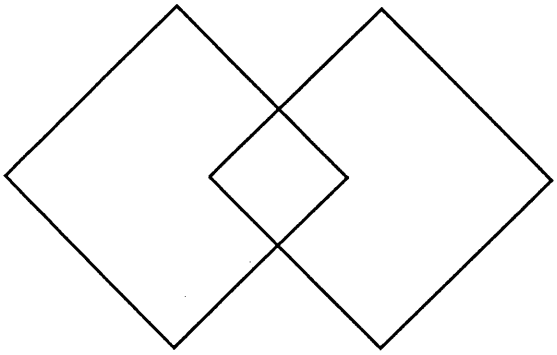

Figure 2: The rotated region for Example 2.

PQ in Figure 1 is classified into the same class as the north east cluster, i.e., the one to which data in the area surrounded by $BPEFGQB$ belong, the former data point is called *misclassified.* In the same way, when a data point that is in the right and upper side of the broken segment PQ in Figure 1 is classified into the same class as the south west cluster, i.e., the one to which data in the area surrounded by $HPADCQH$ belong, the data point is also called *misclassified.* The misclassification in Example 2 is defined in the same way. Notice that the term of *misclassification* is used for convenience of the description: it is a tentative measure of appropriateness of a classification used only for these two examples.

Tables 1–4 show the number of successes, the average number of misclassified data, the average and maximum number of iterations, and the average CPU time (sec) throughout the ten trials for three values of the parameter a: $a = 0.1$, 0.2, 0.3 in the respective cases. Moreover these tables compare results by the ℓ_1 c-means and Euclidean c-means.

The number eight, for example, of successes means that eight trials out of the ten have produced good results, while the other two have led to unacceptable classifications of large numbers of misclassified data. The average number of misclassifications has been calculated from the successful trials: if the eight trials are successful, the data of the other two trials are not used for the calculation. For example, the number 0.6 in the row for $a = 0.1$ in

Table 1: The number of successes out of ten trials, the average number of misclassifications, and the average/maximum number of iterations, and CPU time (for one cycle of calculating a pair $(v^{(s)}, U^{(s+1)})$ in the main loop), for $a = 0.1$, 0.2, 0.3 in Example 1.1 ($n = 1,000$) by the ℓ_1 and Euclidean c-means.

Example 1.1 ($n = 1,000$) by the ℓ_1 c-means				
a	successes	misclassifications	iterations(average/max)	CPU time(sec)
0.1	8	0.6	8.6/10	0.0751
0.2	9	4.4	9.0/10	0.0755
0.3	7	6.1	10.6/12	0.0743
Example 1.1 ($n = 1,000$) by Euclidean c-means				
a	successes	misclassifications	iterations(average/max)	CPU time(sec)
0.1	10	1.1	11.2/13	0.0817
0.2	10	3.7	12.8/14	0.0820
0.3	10	4.2	13.5/16	0.0820

Table 1 has been obtained from $(5 + 0 + 0 + 0 + 0 + 0 + 0 + 0)/8$, which means that one trial has been with five misclassfied data points and other trials have been without any misclassification.

The CPU time is for one cycle of calculating a $v^{(s)}$ and a $U^{(s+1)}$ in the main loop of FCM. The total CPU time needed until the termination is, for example, $8.6 \times 0.0751 \cong 0.646$ on average for $a = 0.1$ by ℓ_1 c-means in Example 1.1.

Comparison of the results by the ℓ_1 c-means and Euclidean c-means leads to the following observations.

(a) The computation for one cycle by ℓ_1 FCM is faster than Euclidean FCM in all 12 cases.

(b) The number of iterations by ℓ_1 FCM is less than that by Euclidean FCM in every case.

(c) From (a) and (b), the total computation by ℓ_1 FCM is faster than Euclidean FCM in all cases.

(d) Seeing the numbers of the misclassifications of the total 12 cases, we find that ℓ_1 FCM produces better results in four cases, while Euclidean FCM is better in the other eight

Table 2: The number of successes out of ten trials, the average number of misclassifications, and the average/maximum number of iterations, and CPU time for $a = 0.1, 0.2, 0.3$ in Example 1.2 ($n = 10,000$) by the ℓ_1 and Euclidean c-means.

Example 1.2 ($n = 10,000$) by the ℓ_1 c-means				
a	successes	misclassifications	iterations(average/max)	CPU time(sec)
0.1	7	5.4	10.3/11	0.758
0.2	9	10.8	12.1/13	0.752
0.3	10	19.8	12.0/13	0.755
Example 1.2 ($n = 10,000$) by Euclidean c-means				
a	successes	misclassifications	iterations(average/max)	CPU time(sec)
0.1	10	3.3	11.5/14	0.813
0.2	10	6.0	12.5/15	0.812
0.3	10	24.4	12.9/14	0.812

Table 3: The number of successes out of ten trials, the average number of misclassifications, and the average/maximum number of iterations, and CPU time for $a = 0.1, 0.2, 0.3$ in Example 2.1 ($n = 1,000$) by the ℓ_1 and Euclidean c-means.

Example 2.1 ($n = 1,000$) by the ℓ_1 c-means				
a	successes	misclassifications	iterations(average/max)	CPU time(sec)
0.1	7	2.0	6.3/7	0.0796
0.2	9	2.0	6.3/7	0.0784
0.3	10	5.6	7.2/8	0.0759
Example 2.1 ($n = 1,000$) by Euclidean c-means				
a	successes	misclassifications	iterations(average/max)	CPU time(sec)
0.1	10	1.4	12.7/14	0.0811
0.2	10	0.1	13.5/15	0.0814
0.3	10	3.4	15.4/18	0.0812

Table 4: The number of successes out of ten trials, the average number of misclassifications, and the average/maximum number of iterations, and CPU time for $a = 0.1$, 0.2, 0.3 in Example 2.2 ($n = 10,000$) by the ℓ_1 and Euclidean c-means.

Example 2.2 ($n = 10,000$) by the ℓ_1 c-means				
a	successes	misclassifications	iterations(average/max)	CPU time(sec)
0.1	9	7.0	7.8/8	0.777
0.2	10	12.3	8.1/9	0.773
0.3	9	12.3	8.7/10	0.772
Example 2.2 ($n = 10,000$) by Euclidean c-means				
a	successes	misclassifications	iterations(average/max)	CPU time(sec)
0.1	10	4.6	12.6/14	0.807
0.2	10	17.6	13.7/15	0.809
0.3	10	26.0	14.9/18	0.811

cases.

(e) Seeing the number of successes for all 120 trials, we observe that ℓ_1 FCM has failed 16 times, while Euclidean FCM has succeeded in all trials.

To summarize the above results, we find that the ℓ_1 c-means is faster than Euclidean c-means. However, the ℓ_1 method has sometimes failed to produce an appropriate result. As to the misclassifications, Euclidean FCM is a little better, but there is no remarkable difference between the two methods.

We have analyzed the 16 cases of the failure of the ℓ_1 method, and found that in 14 cases the iteration stopped at $s = 1$, i.e., after $(v^{(1)}, U^{(2)})$ had been calculated. For the other two cases, one stopped at $s = 2$; the other at $s = 3$. Thus, the failure to produce an appropriate result occurred when the iteration terminated too early. A simple technique for improving the algorithm is to incorporate an empirical rule into the ℓ_1 algorithm, whereby if an early termination is detected, the calculation starts again with renewed initial membership values.

This failure has not been caused by algorithm CC, since the calculation by CC always

produces the optimal solution for the cluster center without any approximation. In other words, one cannot theoretically expect a better result by replacing CC by any other procedure for calculating the cluster centers. (This does not mean, however, that we are unable to improve the algorithm by using heuristic or *ad hoc* rules.)

5 Conclusions

We have derived a simple method of calculating cluster centers in the ℓ_1 fuzzy c-means algorithm. The computational complexity of the present method has been shown to be optimal. A theorem of termination for the whole iterative procedure of ℓ_1 FCM has been proved for a modification of this algorithm. We have tested the present algorithm using a large number of data points. Computation time is sufficiently small and the classification results are satisfactory.

We assert that the present algorithm is superior to the foregoing ones (Bobrowski and Bezdek, 1991; Jajuga, 1991) due to the following reasons:

(i) The present algorithm gives an exact solution, i.e., no approximation or iteration is used, whereas iterative procedures have been used in the foregoing studies.

(ii) The algorithm always produces the optimal solution for the cluster center.

(iii) The computational complexity is optimal, while it is difficult to estimate the complexity of the other algorithms.

Numerical results have shown that the ℓ_1 method has been faster than Euclidean FCM. There has been no great difference between these two methods with respect to the misclassifications. The major drawback of the ℓ_1 method is that sometimes it has failed to give an appropriate result. The method requires to be improved by incorporating empirical rules to avoid an early termination. Anyway, however, the examples have shown that the present ℓ_1 algorithm can compete with Euclidean c-means.

Possible future studies include theoretical investigations such as improvement of termina-

tion of an ℓ_1 FCM algorithm. Small modifications of CC might lead to a better theoretical result, and strategies other than (11) and (12) may be adopted. It seems difficult to improve greatly algorithm CC, since the complexity is optimal.

The present termination theorem is superior to that in standard fuzzy c-means in the sense that an upper bound on the number of iterations is estimated for ℓ_1 FCM, while it is possible that only a subsequence converges and the whole sequence does not converge in the inner product case.

The present method of proving termination of an algorithm can be applied to fuzzy c-means based on other distances. The key issues are the finiteness of the set of possible cluster centers (Bezdek, Bobrowski, 1990) and the uniqueness of the grades.

Thus, we have shown that ℓ_1 FCM is a practical method. Problems in real applications should therefore be studied using this method.

References

Anderberg, M. R., (1973) Cluster Analysis for Applications, New York, Academic Press.

Bezdek, J. C., (1981) Pattern Recognition with Fuzzy Objective Function Algorithms, New York, Plenum.

Bezdek, J. C. and Bobrowski, L., (1990) Matching "boxy" data: clustering with the Minkowski norms, Proc. of 3rd International IPMU Conference., ed. B. Bouchon and R. Yager, 295-297.

Bobrowski, L. and Bezdek, J. C., (1991) c-means clustering with the ℓ_1 and ℓ_∞ norms, IEEE Transactions on Systems, Man, and Cybern., 21, 3, 545-554.

Devroye, L. and Gyorfi, L., (1984) Nonparametric Density Estimation: The L_1 View, New York, Wiley.

Dunn, J. C., (1974) A fuzzy relative of the ISODATA process and its use in detecting compact, well-separated clusters, J. Cybern., 3, 32-57.

Hathaway, R. J., Bezdek, J. C., and Tucker, W. T., (1987) An improved convergence

theory for the fuzzy c-means clustering algorithms, in The Analysis of Fuzzy Information, J. C. Bezdek, Ed., Boca Raton, Florida, CRC Press, 123-131.

Jajuga, K., (1991) L_1-norm based fuzzy clustering, Fuzzy Sets and Systems, 39, 43-50.

Nahorski, Z., (1992) Regression analysis: a perspective, in Fuzzy Regression Analysis, J. Kacprzyk, M. Fedrizzi, Eds., Warsaw, Omnitech Press, 3-13.

Rockafellar, T., (1970) Convex Analysis, Princeton, New Jersey, Princeton University Press.