Maximin Initialization for Cluster Analysis

Richard J. Hathaway[1], James C. Bezdek[2], and Jacalyn M. Huband[2]

[1] Department of Mathematical Sciences, Georgia Southern University
Statesboro, GA 30460, USA
r.hathaway@ieee.org
[2] Computer Science Department, University of West Florida
Pensacola, FL 32514, USA
jbezdek@uwf.edu, jhuband@uwf.edu

Abstract. Most iterative clustering algorithms require a *good* initialization to achieve accurate results. A new initialization procedure for all such algorithms is given that is exact when the data contain compact, separated clusters. Our examples use c-means clustering.

1 Introduction

In this note we introduce, analyze and demonstrate a new initialization procedure, called the *maximin initialization* (MMI) algorithm, which is applicable to any clustering method that requires an initial guess for a partition of the data. The core of the proposed initialization strategy appeared as one part of the progressive sampling scheme used in the eNERF (extended non-Euclidean relational fuzzy c-means) algorithm of Bezdek et al. [1] for clustering large relational data sets. MMI is also used in the sVAT (scalable visual assessment of tendency) scheme of Hathaway et al. [2], which produces image displays of large unlabeled data sets. However, neither of these papers discussed the use of MMI in the present context as a standalone tool for initialization of clustering algorithms. In a nutshell, MMI systematically identifies objects that are distributed throughout the data, and uses the identified objects to inexpensively generate an initial partition of the entire data set. We will prove that the MMI partition is exact if the data set consists of compact and separated clusters in the sense of Dunn [3].

Clustering or cluster analysis is the problem of partitioning a set of objects $O = \{o_1,\ldots, o_n\}$ into c self-similar subsets based on available data and some well-defined measure of (cluster) similarity. When each object in O is represented by a (column) vector $\mathbf{x}$ in $\Re^s$, the set $X = \{\mathbf{x}_1,\ldots,\mathbf{x}_n\} \subset \Re^s$ is called an *object data* representation of O. The k^{th} component of the i^{th} feature vector (x_{ki}) is the value of the k^{th} feature (e.g., height, weight, length, etc.) of the i^{th} object. It is in this data space that practitioners sometimes seek geometrical descriptors (often called prototypes) of the clusters. Alternatively, when each *pair* of objects in O is represented by a relationship, then we have *relational data*. The most common case of relational data is when we have (an n×n matrix of) dissimilarity data, say $D = [d_{ij}]$, where d_{ij} is the pair wise dissimilarity (usually a distance) $dis(o_i,o_j)$ between objects o_i and o_j, for $1 \le i, j \le n$. More

J.F. Martínez-Trinidad et al. (Eds.): CIARP 2006, LNCS 4225, pp. 14 – 26, 2006.

generally, the relational data can be a matrix of similarities based on a variety of measures (Borg and Lingoes [4]; Kendall and Gibbons [5]).

One of the better-known families of clustering models that must be initialized by a partition of the data is the family of c-means models. There are *hard* (Ball and Hall [6]), *fuzzy* (Bezdek [7]) and *possibilistic* (Krishnapuram and Keller [8]) c-means models and algorithms for object data (HCM, FCM, PCM), and corresponding duals of each of these for relational data (Hathaway et al., [9]). The new initialization procedure can be used with all versions of c-means. We use only HCM and FCM in this note, so clustering is done on object data.

A partitioning of the data (or objects) into c clusters is represented by a c·n partition matrix U. The sets of (nondegenerate) fuzzy and hard c-partitions of n objects are denoted by M_{fcn} and M_{hcn}:

$$M_{hcn} = \left\{ U \in M_{fcn} \mid u_{ik} \in \{0,1\} \right\}; \text{ and} \tag{1}$$

$$M_{hcn} = \left\{ U \in M_{fcn} \mid u_{ik} \in \{0,1\} \right\}. \tag{2}$$

The element u_{ik} of a partition matrix U represents the degree or extent to which object o_k (or datum $\mathbf{x}_k$) belongs to cluster i. The crucial difference between the two sets in (1) is that fuzzy partitions, which were first used by Ruspini [10], allow memberships in [0,1], so that (partial) membership of a datum can be shared between clusters, while hard partitions require membership values to be 0 or 1, so each datum is unequivocally placed into one and only one of the c clusters.

The hard and fuzzy c-means algorithms arise by (approximately) minimizing a member of the family of functionals

$$J_m(U,V) = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{ik}^m \, d_{ik}^2, \tag{3}$$

where: $d_{ik}^2 = [d(\mathbf{x}_k, \mathbf{v}_i)]^2$ is the distance from $\mathbf{x}_k$ to $\mathbf{v}_i$ in any inner product induced norm, $n = |X|$, $m \in [1, +\infty)$ is a user- defined fuzzification constant, c is the number of clusters assumed, U is in M_{fcn} (m > 1) or M_{hcn} (m = 1), $V = [\mathbf{v}_1,\ldots,\mathbf{v}_c]$ is a matrix whose columns are c prototypes in $\Re^s$, and $d(\mathbf{v}_i, \mathbf{x}_k)$ measures the distance between data point $\mathbf{x}_k$ and prototype $\mathbf{v}_i$ in any inner product induced norm metric. Zeroing the Lagrangian of J_m results in the following first order necessary conditions (m = 1 for HCM, m > 1 for FCM):

$$\text{HCM V Update}: \; v_i = \left(\sum_{k=1}^{n} u_{ik} \mathbf{x}_k \right) \Big/ \left(\sum_{k=1}^{n} u_{ik} \right) \quad \forall \, i \tag{4}$$

$$\text{HCM U Update}: \; u_{ik} = \left\{ \begin{matrix} 1; & d_{ik} < d_{jk} \;\; \forall \, j \neq i \\ 0; & \text{otherwise} \end{matrix} \right\} \forall \; i,k \tag{5}$$

$$\text{FCM V Update}: \quad v_i = \left(\sum_{k=1}^{n} u_{ik}^m \mathbf{x}_k \right) \Big/ \left(\sum_{k=1}^{n} u_{ik}^m \right) \quad \forall\, i \tag{6}$$

$$\text{FCM U Update}: \quad u_{ik} = \left(\sum_{j=1}^{c} \left(d_{ik} / d_{jk} \right)^{\frac{2}{(m-1)}} \right)^{-1} \quad \forall \quad i,k \tag{7}$$

The iteration can be initialized using either a partition U or matrix of prototypes V. For example, for either algorithm, a current U is used to update the prior set of prototypes $V = [\mathbf{v}_1,...,\mathbf{v}_c]$, which are in turn used to calculate a new partition U, and then successive estimates (of either set of variables) are compared to a termination threshold. The theory of this *alternating optimization* (AO) procedure is given in Bezdek and Hathaway [11]. Many authors have considered the sensitivity of AO to its initialization. Bezdek et al. [12] contains an extensive discussion of this issue for the c-means algorithms. Our current contribution to this ongoing body of research is a new initialization scheme that has some theoretical substance - viz., the MMI algorithms produces an initial guess for U that is exact when X (or D) contains c compact, separated clusters in the sense of Dunn [3]. In the experiments conducted in Section 3, initialization is always done using an MMI partition $U \in M_{hcn}$.

2 The Maximin Initialization Algorithm

The core of MMI involves selecting c *distinguished objects* $o_{m_1}, o_{m_2}, ..., o_{m_c}$ that are distributed throughout the set of objects $O = \{o_1,...,o_n\}$, relative to the available measure of dissimilarity between pairs of objects in O. If a relational dissimilarity matrix D is available, then this is used directly to measure dissimilarity of pairs of objects. When using object data set $X = \{\mathbf{x}_1,...,\mathbf{x}_n\} \subset \Re^s$, we pick a metric $d(\cdot\,,\cdot)$ on $\Re^s \times \Re^s$ and use $d(\mathbf{x}_j, \mathbf{x}_k)$ as a measure of dissimilarity between o_j and o_k. The first distinguished object selected is simply $o_{m_1} = o_1$. The second chosen (o_{m_2}) is the object in O *most dissimilar* to o_{m_1}. All subsequent choices of distinguished objects involve picking the object with the largest minimum dissimilarity to all of the previously selected objects. This selection of distinguished objects is formally described in Step 1 of the statement of the MMI. The second step of the algorithm computes the (hard) partition that corresponds to grouping each object into the same class as its nearest distinguished object. For the case of object data, this amounts to doing (2b) with $\mathbf{v}_i = \mathbf{x}_{m_i}$, for i = 1,...,c. The MMI algorithm follows.

MMI: Maximin Initialization Algorithm

Choose: The number of clusters c, $1 < c < n$; and, $\Re^s \times \Re^s$ if the available data are object data X, an inner product induced metric $d(\cdot\,,\cdot)$ on.

Input: Object data $X = \{\mathbf{x}_1,...,\mathbf{x}_n\} \subset \Re^s$ or dissimilarity data $D = [d_{ij}] \subset \Re^{n \times n}$.

Step 1 *Select the indices m_1, ..., m_c of the c distinguished objects.*
Select $m_1 = 1$ (Remark: this is the "***object seed***";$m_1 = 1$ is arbitrary)
(If the data are X, calculate dissimilarities $\{ d_{1,k} \}$, k = 1 to n.)
Initialize the minimum distance array $\boldsymbol{\delta}^1 = [\delta_1^1,...,\delta_n^1] = [d_{1,1}, ..., d_{1,n}]$
For t = 2,…, c:

$$\text{Update } \boldsymbol{\delta}^t \leftarrow [\min\{ \delta_1^{t-1}, d_{m_{t-1},1} \}, ..., \min\{ \delta_n^{t-1}, d_{m_{t-1},n} \}]$$

$$\text{Select } m_t \in \arg\max_{1 \le j \le n} \{\delta_j^t\}$$

(If the data are X, calculate dissimilarities $\{d_{m_t,k}\}$, k = 1 to n.)
Next t

Step 2 *Cluster each object in $\{o_1, ..., o_n\}$ with its nearest distinguished object.*
Clear the initialization matrix array $U = [u_{ik}] \subset \Re^{cn}$: $u_{ik} = 0 \ \forall$ i,k
For k = 1,..., n:

$$\text{Select } i \in \arg\min_{1 \le j \le c} \{d_{m_j,k}\} \text{ and then set } u_{ik} = 1$$

Next k

Output: A crisp c×n initialization partition U in M_{hcn}

If initialization (of the c-means algorithms) by a set of c prototypes rather than by a partition is desired, then it is only necessary to execute Step 1 and terminate with $\mathbf{v}_i = \mathbf{x}_{m_i}$, for i = 1,...,c. The name *maximin initialization* follows by noticing that the loop in Step 1 can be "unrolled" to show that m_t is selected as an index in {1,...,c} that maximizes the minimum of the distances to the previously selected distinguished objects; i.e., m_t is a solution of

$$\arg\max_{1 \le j \le n} \{\min\{d_{m1,j}, d_{m2,j}, \ldots, d_{mt-1,j}\}\} \tag{8}$$

Any tie breaking strategy can be used if arg max of Step 1 or arg min of Step 2 does not specify a unique index. The crisp clusters found at Step 2 of MMI are produced by labeling each of the (n-c) remaining objects with the nearest prototype (1-np) rule. Seen in this light, Step 2 of MMI is just a crisp 1-np classifier for the objects in the data that are not selected in Step 1. Finally, notice that MMI requires the user to specify a value for c, the number of clusters that a subsequent clustering algorithm will seek in the data. MMI produces its initialization for any $1 < c < n$, and does not offer an uninformed user any means for inferring a "best" choice for this important parameter. This important problem - the *cluster validity* problem - is addressed, for example, by the sVAT algorithm of Hathaway et al. [2]. In this note we simply pick and use (the "correct") value of c for various examples to illustrate how MMI then finds a good initialization for subsequent clustering. In practice, the data should be submitted to an algorithm such as sVAT before using MMI, so that initialization is done at a reasonable value of c.

MMI has about the same computational cost as one iteration of HCM, and is a bit cheaper for relational data D than for object data X. For object data, Step 1 requires

calculation of cn distances in $\Re^s$, which is $\boldsymbol{O}(scn)$. Then MMI performs [(c-1)n] min operations involving 2 elements each, and (c-1) max operations involving n elements each. In Step 2, we must do n min operations involving c elements each, which is exactly the cost of performing (2b) one time. Overall then, MMI is $\boldsymbol{O}(scn)$ for object data. For relational data, we don't have to calculate the cn distances in $\Re^s$ because the dissimilarities are already available as elements of the input data set D. Thus, MMI is $\boldsymbol{O}(cn)$ when its input is relational data set D.

The main theoretical result for MMI is that if the clusters are well separated, then the initialization is *exact*. To formalize this, we recall the notion of compact and separated clusters defined by Dunn [3]. For a set of objects $O = \{o_1,...,o_N\}$ with corresponding relational dissimilarity data D, we say that a partitioning $O^{(1)}, O^{(2)}, ..., O^{(c)}$ of O is *compact and separated* (CS) relative to D if each of the possible intra-cluster distances is strictly less than each of the possible inter-cluster ones. When the data has this property, we say simply that "O can be partitioned into c CS clusters". The main result (Theorem 1) is that if the data consists of c CS clusters, and MMI is applied to it with c as the specified number of distinguished objects, then the initial partition produced by MMI will correctly partition O into these c CS clusters. Based on this property - perfect initialization in the compact and separated case - we expect MMI to provide good initializations in most (non-CS) cases. We will investigate this expectation empirically in Section 3.

Theorem 1. Suppose that the set of objects $O = \{o_1,...,o_N\}$, represented by either an object data set X (with metric $d(\cdot\,,\cdot)$ on $\Re^s$) or a relational dissimilarity matrix D, can be partitioned into $c \geq 1$ CS clusters. Then the crisp MMI c-partition of X (or D) partitions O into its c CS clusters.

Proof. We denote the dissimilarity between objects o_j and o_k by $d_{jk} = dis(o_j,o_k)$, understanding that d_{jk} either comes directly from the matrix D if relational data is available or is calculated as $d_{jk} = d(x_k,x_j)$ if object data is available. Also, we denote the c CS clusters of $O = \{o_1,...,o_n\}$ by $O^{(1)}, O^{(2)}, ..., O^{(c)}$, and when convenient, we indicate the cluster of a datum or object by a superscript in parentheses; e.g., $o_7^{(2)}$ indicates that the seventh object is in the second CS cluster. First we prove that Step 1 of MMI selects exactly one distinguished object from each of the c CS clusters. The result is trivially true for c = 1. Now, suppose we can partition $O_n = \{o_1,...,o_n\}$ into $c \geq 2$ CS clusters $O^{(1)}, O^{(2)}, ..., O^{(c)}$. Since the clusters are compact and separated, it is true that for $1 \leq i \neq h \leq c$ and applicable k, p, j, we have

$$d_{kp} = dis(o_k^{(i)},o_p^{(i)}) < dis(o_k^{(i)},o_j^{(h)}) = d_{kj} \,. \tag{9}$$

We first select object o_1, and without loss of generality, assume that it belongs to $O^{(1)}$. Then the initial search array $\boldsymbol{\delta}^1$ is defined (either using elements of D or $d(\cdot\,,\cdot)$ on corresponding object vectors) as $\boldsymbol{\delta}^1 = [dis(o_1,o_1), \ ... \ , dis(o_1,o_n)] = [d_{11},...,d_{1n}]$. Then applying (9) with i = 1, we see that the maximum element in $\boldsymbol{\delta}^1$ (and therefore the choice of the second distinguished feature) must correspond to an object in $O^{(2)}, ...,$

$O^{(c)}$ (but not in $O^{(1)}$). This completes the proof that each CS cluster is represented by exactly one distinguished object for c = 2, and we now continue for the case of $c \geq 3$. Suppose the max occurs in the second entry of $\boldsymbol{\delta}^1$ so that the next object selected is o_2; and further suppose that o_2 belongs to CS cluster $O^{(2)}$. The updated search array $\boldsymbol{\delta}^2$ is:

$$\boldsymbol{\delta}^2 = [\min\{\delta_1^1, d_{21}\}, \ldots, \min\{\delta_n^1, d_{2n}\}].$$

Suppose that a maximum entry is found in the third component of $\boldsymbol{\delta}^2$ and that o_3 is the third object selected ($m_3 = 3$). We will prove that o_3 cannot belong to $O^{(1)}$ or $O^{(2)}$ by contradiction. So, assume that o_3 *does* belong to either $O^{(1)}$ or $O^{(2)}$, say $O^{(1)}$. Selection of o_3 implies that, for all j = 1 to n:

$$\min\{\delta_3^1, d_{23}\} = \min\{dis(o_1, o_3), dis(o_2, o_3)\} \geq \min\{dis(o_1, o_j), dis(o_2, o_j)\} \tag{10}$$

But (10) implies that, for j = 1 to n:

$$dis(o_1, o_3) \geq \min\{dis(o_1, o_j), dis(o_2, o_j)\} \tag{11}$$

Now, let $j \geq 4$ be any index such that o_j is in neither $O^{(1)}$ nor $O^{(2)}$. (At least one such j exists since $c \geq 3$ and for k = 1, 2, 3 we have $o_k \in O^{(1)} \cup O^{(2)}$.) Without loss of generality we suppose that j = 4 satisfies (7) with $o_4 \in O^{(3)}$, and that $dis(o_1,o_4) \leq dis(o_2,o_4)$). Then (11) gives $dis(o_1,o_3) \geq dis(o_1,o_4)$,where $o_1 \in O^{(1)}$, $o_3 \in O^{(1)}$, and $o_4 \in O^{(3)}$; but this contradicts (9) for i = k = 1, p = h = 3, and j = 4. Thus, the third object chosen cannot be in a previously represented cluster. We can now repeat this argument for the 4th, 5th, …, up to the cth choice. This establishes our claim that each CS cluster is represented by (i.e., contains) exactly one of the distinguished objects $o_{m_1}, o_{m_2}, \ldots, o_{m_c}$. Finally, since Step 1 of MMI has selected one distinguished object from each of the c CS clusters, (5) implies that for i = 1,...,c, each object in cluster $O^{(i)}$ is grouped with object o_{m_i} in Step 2. Thus, our construction produces a crisp c-partition of $O = \{o_1,\ldots,o_n\}$ into its c compact and separated clusters. ■

Theorem 1 guarantees that MMI produces an initial crisp c-partition corresponding to c compact and separated clusters whenever c CS clusters exist in the data. We assert that this provides an excellent initialization for (*any*) partitioning algorithm that initializes at U in M_{hcn}. Indeed, when X (or D) has c CS clusters, the MMI c-partition of it is part of a necessary pair for J_1 at (3, m=1) - i.e., it is part of an HCM solution. The experiments we present in the next section investigate whether MMI also produces useful initializations when the clusters are *not* w¹ell separated.

3 Numerical Examples

In this section we test the MMI algorithm with HCM and FCM. The data sets we choose are samples drawn from mixtures of c = 4 normal distributions in either $\Re^2$ or $\Re^{10}$. Thus, the data sets used for each figure and simulation in this section nominally have c = 4 clusters (or components); and all are of size n = 1000. The

parameter we use to control the amount of overlap between the four clusters is the (common) variance σ^2 of the component normals. We call the two types of data sets used **DIAGONAL** data and **SQUARE** data, terms chosen to (roughly) refer to the arrangement of clusters. The DIAGONAL data sets are samples from a mixture distribution of 4 normal distributions with cluster centers arranged along a diagonal line. The specific component parameters in the case of s = 2 (i.e., $X \subset \Re^2$) are:

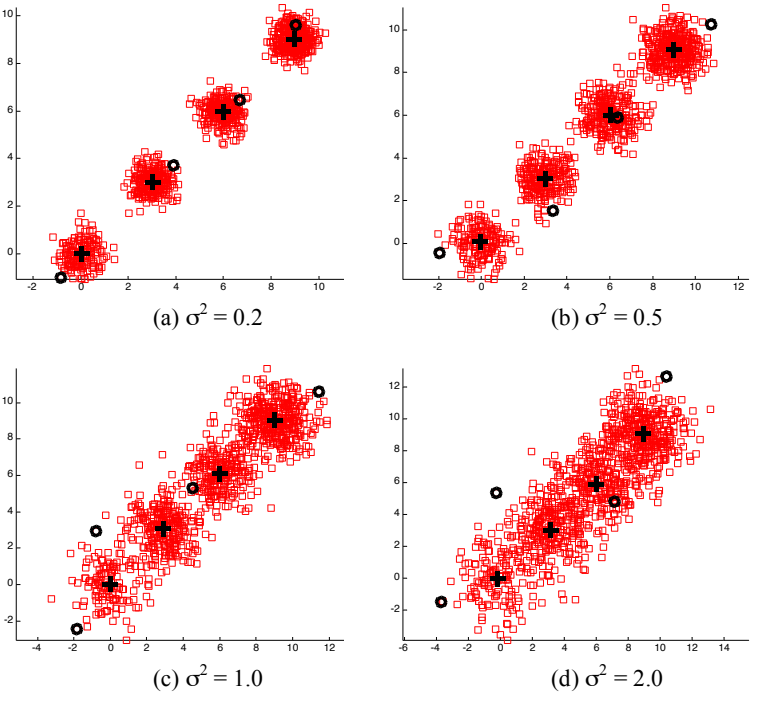
$$\text{mixing proportions : } \alpha_1 = 0.15, \alpha_2 = 0.25, \alpha_3 = 0.25 \text{ and } \alpha_4 = 0.35; \tag{12}$$

$$\text{means : } \boldsymbol{\mu}_1 = [0\ 0]^T, \boldsymbol{\mu}_2 = [3\ 3]^T, \boldsymbol{\mu}_3 = [6\ 6]^T \text{ and } \boldsymbol{\mu}_4 = [9\ 9]^T; \text{ and} \tag{13}$$

$$\text{covariance matrices : } \Sigma_1 = \Sigma_2 = \Sigma_3 = \Sigma_4 = \sigma^2 I_{2\times 2} \tag{14}$$

where $I_{2\times 2}$ is the 2x2 identity matrix and the positive number σ^2 is varied according to the experiment. The SQUARE data distribution is so named because the clusters form the corners of a square configuration, and its component parameters are given by (12), (14), and:

$$\text{means : } \boldsymbol{\mu}_1 = [0\ 0]^T, \boldsymbol{\mu}_2 = [6\ 0]^T, \boldsymbol{\mu}_3 = [0\ 6]^T \text{ and } \boldsymbol{\mu}_4 = [6\ 6]^T. \tag{15}$$



(a) $\sigma^2 = 0.2$ (b) $\sigma^2 = 0.5$

(c) $\sigma^2 = 1.0$ (d) $\sigma^2 = 2.0$

Fig. 1. Typical DIAGONAL data; means (+) and MMI selected data (o)

Figures 1 and 2 are scatter plots of typical samples from the two dimensional mixtures in $\Re^2$ (n = 1000) for various values of σ^2. The symbol (**o**) represents the c= 4 MMI-selected distinguished objects (i.e., object data in this case).

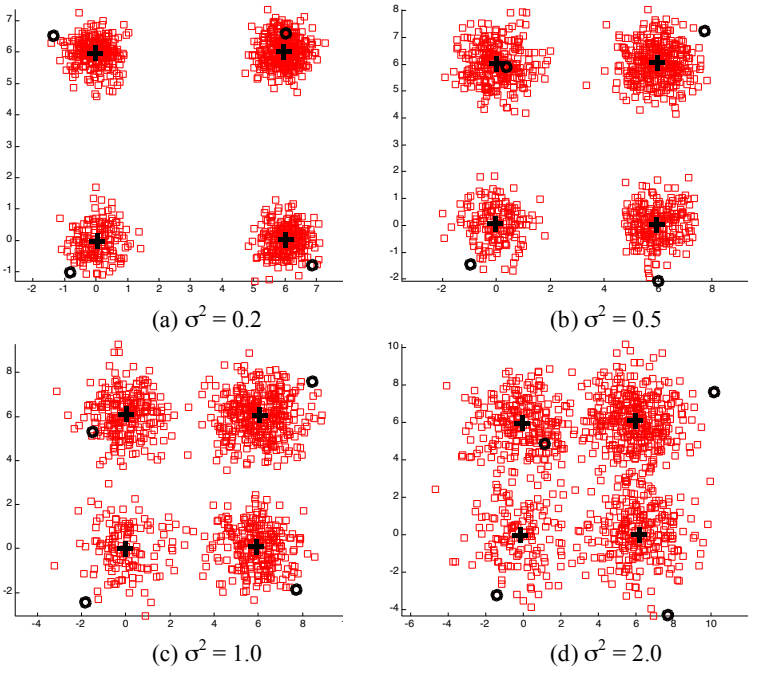


Fig. 2. Typical SQUARE data; means (**+**) and MMI selected data (**o**)

As expected, the distinguished data (**o**) selected by the MMI algorithm are distributed throughout the data sets in the figures and are clearly present in all four clusters in the more separated cases such as those of Figures 1(a) and 2(a). The 2-dimensional simulations that are reported in this section use data sets distributed like those in the figures. Also performed are simulations using 10-dimensional data sets whose first two coordinates are distributed like those in the figures while the 3rd through 10th coordinates are normally distributed with mean 0. Specifically, the mean parameters for the 10-dimensional versions of the DIAGONAL and SQUARE data are:

$$\boldsymbol{\mu}_1 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T;\ \boldsymbol{\mu}_2 = [3\ 3\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T;\ \text{(DIAGONAL)} \tag{16}$$
$$\boldsymbol{\mu}_3 = [6\ 6\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T;\ \boldsymbol{\mu}_4 = [9\ 9\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T.$$

$$\boldsymbol{\mu}_1 = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T;\ \boldsymbol{\mu}_2 = [6\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T;\ \text{(SQUARE)} \tag{17}$$
$$\boldsymbol{\mu}_3 = [0\ 6\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]^T;\ \boldsymbol{\mu}_4 = [6\ 6\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0].$$

The component covariance matrices for the 10-dimensional distributions all equal $\sigma^2 I_{10\times10}$, and the mixing proportions in (12) are unchanged. Extension from 2 to 10 dimensions in this way allows us to examine the effect of dimensionality without making the clustering problem substantially easier (or harder) since the separation between components is essentially unchanged and depends only on the first 2 coordinates.

The purpose of our simulations is to investigate the effectiveness of MMI as an initializer of HCM and FCM. We want to compare HCM (and FCM) results obtained using MMI initialization to those obtained using the true class labels as the point of initialization. Toward this end, the true labels are tabulated and represented as a crisp c×n partition (in this case 4×1000) during the generation of each normal mixture data set. We chose this structuring of the tests because we want to know whether or not the MMI works well compared to an "optimal" initialization (i.e., the true labels); not merely whether or not it compares relatively well to some other existing initialization scheme.

The measurement recorded in Tables 1 and 2 is referred to as DIF(✶), which corresponds to the difference between HCM and FCM partitions of the input data obtained from starting the iteration through equations (4)-(5) or (6)-(7) using the MMI and true initializations. We denote the terminal fuzzy partitions obtained using FCM with the two initializations as U_{MMI}^{FCM} and U_{TRUE}^{FCM}; and we similarly denote the HCM results by U_{MMI}^{HCM} and U_{TRUE}^{HCM}. We define DIF(h) and DIF (f) to measure the percentage difference between the crisp (h) or fuzzy (f) clusters obtained using the two initializations. For example, if 45 of n = 1000 data are grouped by HCM into different clusters using the two initializations, then DIF(h) = 4.5%. The percentage can be computed in the crisp case as:

$$DIF(h) = 50 * \sum_{i=1}^{c}\sum_{j=1}^{n}\left|(u_{TRUE}^{HCM})_{ij} - (u_{MIA}^{HCM})_{ij}\right| \Big/ n \ . \tag{18}$$

To define the analog of DIF(h) for FCM, we must "harden" the terminal fuzzy partitions obtained by starting FCM at the MMI and True partitions. This amounts to replacing the maximum entry in each column of U by a 1, and replacing all (c – 1) other entries with 0's (this is just Bayes rule when U is a partition of posterior probabilities). We denote the hardening of a partition U by H(U) and define DIF(f) for the FCM results as:

$$DIF(f) = 50 * \sum_{i=1}^{c}\sum_{j=1}^{n}\left|(H(u_{TRUE}^{FCM}))_{ij} - (H(u_{MIA}^{FCM}))_{ij}\right| \Big/ n \ . \tag{19}$$

Next, we describe the simulations and the types of entries that appear in Tables 1 and 2. All experiments were done using MATLAB on a PC, with m = 2 in equations (3), (6) and (7) for FCM. The iterations for HCM and FCM were terminated as soon as the maximum change in the (cn) membership values for successive U iterates became less than or equal to 0.00001. Table 1 gives the results for the DIAGONAL data and Table 2 contains the SQUARE data results. Each row of the two tables corresponds to samples from a mixture specified by the component variance σ^2 and dimension s.

The other component parameters are given in the appropriate parts of equations (12-17). For each row, 1000 samples each of size n = 1000 were generated. For each sample, both HCM (results in columns 2-4) and FCM (results in columns 5-7) were initialized using both the true component labels and the MMI initialization. Columns 2 and 5 give the percentage of the 1000 trials for which DIF(✶) = 0; i.e., the percent of trials for which the terminal c-means partitions (hardened in the case of FCM) produced by the MMI initializations are exactly the same as those produced by initialization with the true class labels. Columns 3 and 6 give the average DIF(✶) over the 1000 trials, while columns 4 and 7 give the worst DIF(✶) that occurred for any single trial.

Table 1. 1000 trials for DIAGONAL data : n = 1000

		DIF(h) for HCM			DIF(f) for FCM		
s	σ^2	% Trials where DIF(h) = 0	Ave. DIF(h)	Worst DIF(h)	% Trials where DIF(f) = 0	Ave. DIF(f)	Worst DIF(f)
2	0.2	99.9%	0.0%	0.1%	100%	0%	0%
2	0.5	94.5%	0.3%	35.3%	100%	0%	0%
2	1.0	69.4%	0.1%	39.4%	100%	0%	0%
2	2.0	42.1%	0.8%	45.4%	99.5%	0.0%	0.1%
10	0.2	99.9%	0.0%	33.5%	100%	0%	0%
10	0.5	89.8%	0.7%	34.5%	100%	0%	0%
10	1.0	43.6%	1.8%	42.7%	99.4%	0.0%	0.1%
10	2.0	13.5%	3.1%	49.4%	99.7%	0.0%	0.8%

All clustering algorithms assign numerical labels to their clusters, and these algorithmic labels may or may not correspond to the "true" labels of the input data. Thus, algorithmic cluster 1 might correspond to input cluster 2, and so on. To solve this correspondence problem in the actual computation of DIF(✶), all permutations of the rows of one of the partitions are considered, so that the recorded DIF(✶) is based on the permutation that gives the smallest possible value of (18). This ensures that the calculated disagreement between partitions actually measures a difference in the grouping of the data among clusters, and not simply a difference in the (arbitrary) numerical label assigned to each cluster by HCM or FCM. For c = 4, this amounts to trying 4! = 24 different permutations. This factorial growth in the calculation of DIF is one reason we limited our experiments to the modest value of c = 4. An entry of 0% in either table means *exactly* 0%, while an entry of 0.0% indicates a *rounded* positive number.

A strong implication of the values in Tables 1 and 2 is that MMI works much better for FCM than it does for HCM. This is *not* surprising, since it is well known that initialization sensitivity is more of a problem for the hard c-means algorithm (Bezdek et al., [12]). The performance of MMI with FCM is consistently good throughout the

experiments, although the 10-dimensional cases of the SQUARE data with highest overlap (σ^2 = 1.0 and 2.0) resulted in several values of DIF(f) above 20%. The

Table 2. 1000 trials for SQUARE data : n = 1000

		DIF(h) for HCM			DIF(f) for FCM		
s	σ^2	% Trials where DIF(h) = 0	Ave. DIF(h)	Worst DIF(h)	% Trials where DIF(f) = 0	Ave. DIF(f)	Worst DIF(f)
2	0.2	100%	0%	0%	100%	0%	0%
2	0.5	100%	0%	0%	100%	0%	0%
2	1.0	94.2%	0.0%	0.2%	100%	0%	0%
2	2.0	64.0%	0.1%	0.7%	99.7%	0.0%	0.1%
10	0.2	100%	0%	0%	100%	0%	0%
10	0.5	99.8%	0.0%	0.1%	100%	0%	0%
10	1.0	88.9%	0.6%	37.9%	98.8%	0.2%	20.5%
10	2.0	32.8%	1.5%	36.5%	96.4%	0.2%	22.3%

consistently high percentage of cases for which DIF(f) = 0 indicates that with a very high probability, MMI initialization produces the same FCM result as initialization at the true class labels. The average DIF(f) values in both tables show that the average differences between the MMI and true label results are very small: zero for all choices of σ^2 except at σ^2 =1.0 and 2.0 for the 10 dimensional data, and just 0.2% for these two cases.

MMI initialization of HCM worked adequately for very well separated problems (σ^2= 0.2), but even in this case there was a Worst DIF(h) value of 33.5% for s=10 dimensions in Table 1. As σ^2 increases, the cluster separation decreases, and it becomes more and more difficult to obtain the same HCM result using the MMI initialization as that obtained by HCM using the true class labels for initialization. For example, in the worst HCM case, (σ^2 = 2.0 and s = 10 in Table 1), agreement was reached only 135 times in 1000 trials. This is not so much an indictment of MMI initialization as it is an indication that there are a large number of local trap states for minima of J_1 when the data have so much overlap. Even so, the average values of DIF(h) for HCM never got worse than 3.1% for either data set.

The scatter plots of the data in Figures 1 and 2 visually suggest that the SQUARE data clusters are better separated than the DIAGONAL ones, and should therefore be easier for both HCM and FCM. Comparing the tables we see that our intuition is correct for HCM; MMI did significantly better for the SQUARE data than for DIAGONAL. But very surprisingly, the MMI-FCM combination did not have a significantly easier time with the SQUARE data, and in fact, it was this case that produced the only real problems for FCM (Worst DIF(f) values of 20.5% and 22.3%). Finally, we point out that increasing the data space dimensionality from 2 to 10 caused some increase in difficulty, typically more for HCM than FCM.

4 Discussion

The maximin initialization (MMI) algorithm was stated, analyzed and then demonstrated, using samples drawn from a variety of 4 component normal mixtures in two and ten dimensions. The computational cost of executing MMI is essentially equal to a single iteration of HCM or FCM in the case of object data ($O(scn)$), and even less in the case of relational data ($O(cn)$). Theorem 1 guarantees that the MMI c-partition corresponds to a compact and separated partitioning of the data whenever such a partitioning exists. MMI identifies c distinguished data (or objects) distributed throughout the data space, and then uses them as "seed" prototypes to build a 1-np partition of the remaining unlabeled data. Our simulations suggest that for a moderate number of clusters, FCM combines particularly well with MMI initialization to produce clustering results comparable to those obtainable when FCM is initialized with the "true" cluster labels.

On the basis of the simulation results, the theoretical implication of Theorem 1, and the inexpensive computational cost, we believe that MMI is a useful tool for generating initializations for FCM, and to a lesser extent, for HCM. Since HCM is notoriously sensitive to initialization, it is probably wise to initialize it from several starting points to make sure a "good" set of clusters is found. MMI can be used to generate multiple, different starting points. How? Just choose an "object seed" other than 1 for m_1 in Step 1 of MMI, and then change δ^1 accordingly. This change to MMI can be used over and over, to produce as many different initializations as desired.

We see several opportunities for future work. The presence of outliers in the data may create some problems for the MMI scheme, although this has yet to be tested. Perhaps a *trimmed maximin* selection, in the spirit of the more robust trimmed mean estimator of centrality, might offer an advantage in the case of data contaminated with outliers. Recently, much effort has been expended to "kernelize" classification and clustering methods. Can kernelized distances be introduced here in a way that causes the selection of better performing distinguished data (or objects)? Two other related clustering approaches that benefit from good initializations are the possibilistic c-means (PCM) algorithm and normal-mixture-based clustering using the expectation-maximization (EM) algorithm (McLachlan and Peel [13]). The EM approach is known to be somewhat sensitive to initialization and PCM produces coincident clusters from some initializations. MMI should stabilize this aspect of both algorithms, but this supposition has yet to be tested.

References

1. Bezdek, J.C., Hathaway, R.J., Huband, J.M., Leckie, C. and Kotagiri, R.: Approximate Clustering in Very Large Relational Data, in press, Inter. J. Intell. Systems (2005)
2. Hathaway, R.J., Bezdek, J.C. and Huband, J.M. 2005: Scalable Visual Assessment of Cluster Tendency for Large Data Sets, in press, Pattern Recognition (2005)
3. Dunn, J.C.:A Fuzzy Relative of the ISODATA Process and its Use in Detecting Compact Well-separated Clusters, J. Cybernetics 3 (1973), 32-57

4. Borg, I. and Lingoes, J.: Multidimensional Similarity Structure Analysis. Springer-Verlag, New York (1987)
5. Kendall, M. and Gibbons, J.D.: Rank Correlation Methods. Oxford University Press, New York (1990)
6. Ball, G. and Hall, D.: A Clustering Technique for Summarizing Multivariate Data, Behavioral Science 12(1967) 153-155.
7. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum, New York (1981)
8. Krishnapuram, R. and Keller, J.M.: A Possibilistic Approach to Clustering, IEEE Trans. On Fuzzy Systems 1 (1993), 98-110
9. Hathaway, R.J., Bezdek, J.C. and Davenport, J.: On Relational Data Versions of c-Means Algorithms, Pattern Recognition Letters 17 (1996), 607-612
10. Ruspini, E.: A New Approach to Clustering, Information and Control 15 (1969), 22-32
11. Bezdek, J. C. and Hathaway, R. J.: Convergence of Alternating Optimization, Neural, arallel and Scientific Computation, 11-4 (2003) 351-368
12. Bezdek, J.C., Keller, J.M., Krishnapuram, R. and Pal, N.R.: Fuzzy Models and Algorithms for Pattern Recognition and Image Processing, Kluwer, Norwell (1999)
13. McLachlan, G. and Peel, D.: Finite Mixture Models. John Wiley & Sons, New York (2000)