Clustering with a Genetically Optimized Approach

L.O. Hall[1], B. Ozyurt[1], J.C. Bezdek[2]

[1]Department of Computer Science and Engineering
University of South Florida
Tampa, Fl. 33620

[2]Department of Computer Science
University of West Florida
Pensacola, Fl. 32514

List of Figures

Abstract

This paper describes a genetically guided approach to optimizing the hard (J_1) and fuzzy (J_m) c-means functionals used in cluster analysis. Our experiments show that a genetic algorithm ameliorates the difficulty of choosing an initialization for the c-means clustering algorithms. Experiments use six data sets, including the Iris data, magnetic resonance and color images. The genetic algorithm approach is generally able to find the lowest known J_m value or a J_m associated with a partition very similar to that associated with the lowest J_m value. On data sets with several local extrema, the GA approach always avoids the less desirable solutions. Degenerate partitions are always avoided by the GA approach, which provides an effective method for optimizing clustering models whose objective function can be represented in terms of cluster centers. The time cost of genetic guided clustering is shown to make a series random initializations of fuzzy/hard c-means, where the partition associated with the lowest J_m value is chosen, an effective competitor for many clustering domains.

Keywords: Genetic algorithms, clustering, fuzzy, c-means.

1 Introduction

Unsupervised learning is useful in exploratory data analysis, image segmentation and, with some added class knowledge, may be used for classification as well. Here we present a *genetically guided algorithm* (GGA) approach to optimization of certain clustering models. This approach can be directly applied to any clustering model which can be represented as a functional dependent upon a set of cluster centers (or point prototypes). The approach can be further generalized for models that require parameters other than the cluster centers.

In this paper the *fuzzy and hard c-means* (FCM/HCM respectively) functionals, J_m and J_1, are used as fitness functions [6, 17]. This allows us to compare performance of the GGA with the conventional FCM/HCM algorithms and examine GGA optimization performance with similar but different objective functions. It allows comparison with other GA work on clustering [1, 34, 10, 30]. Clustering algorithms such as FCM which use calculus-based optimization methods can be trapped by local extrema in the process of optimizing the clustering criterion. They are also very sensitive to initialization. Other conventional optimization methods, such as the Nelder-Mead Simplex method [18, 27] can be used to optimize J_m. Hathaway and Bezdek offer that this is probably a good option for only 10 or fewer unknowns and a few hundred data points [27]. Faster techniques based on Newton's method and its variants, such as those described in [19], can be applied to form partitions with the use of appropriately differentiable distance metrics. However, these methods have significant sensitivity to the chosen initialization.

The GGA presented here attempts to achieve both avoidance of local extrema and minimal sensitivity to initialization. The cluster centers can be represented as real values or encoded as binary strings. We report on both representations in the paper, though we focus upon the binary representation. The overall conclusions of the paper are independent of the representation. The binary cluster center representation is in gray codes. Tournament selection, two-point crossover, and binary mutation comprise the rest of the algorithm. Crossover is applied to each cluster center to quickly move towards an extremum, thereby minimizing the required number of generations.

This paper provides answers to the following questions. Can a GA approach to clustering find extrema that may not be found with the iterative approach to minimizing the c-means functionals? Can a GA find the same extremum that an iterative version of FCM/HCM would? Or does FCM/HCM need to be run using the final cluster centers found by the GA as an initialization? Will the GA find the best or nearly best final partitions for a given data set, i.e. those partitions associated with the lowest J_m values?

In Section 2, we review the FCM/HCM algorithms with which comparisons will be made. In Section 3 the genetic guided clustering approach is presented. Section 4 contains a description of how the various parameters for GA clustering can be set. Section 5 discusses the six data sets employed in this work. Section 6 details the experiments that were performed and contains results from the experiments. Section 7 discusses time considerations in doing the clustering and how a real-valued representation may be used in genetically guided clustering to partially address time issues. Section 8 compares our results with other work in generating evolutionary computation solutions to the clustering problem. Lastly, Section 9 summarizes our results.

2 Clustering with HCM and FCM

Consider a set of n vectors $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\}$ to be clustered into c groups of like data. Each $\mathbf{x}_i \in \Re^s$ is a feature vector consisting of s real-valued measurements describing the features of the object represented by $\mathbf{x}_i$. The features could be length, width, color, etc. Hard or fuzzy clusters of the objects can be represented by a hard/fuzzy membership matrix called a hard/fuzzy partition. The set of all $c \times n$ non-degenerate hard (or crisp) partition matrices is denoted by M_{cn} and defined as

$$M_{cn} = \{U \in \Re^{c \times n} | \sum_{i=1}^{c} U_{ik} = 1, 0 < \sum_{k=1}^{n} U_{ik} < n, \ and \\ U_{ik} \in \{0,1\}\,; 1 \le i \le c; 1 \le k \le n\}. \quad (1)$$

The set of all $c \times n$ non-degenerate constrained fuzzy partition matrices is denoted by M_{fcn} and defined as

$$M_{fcn} = \{U \in \Re^{c \times n} | \sum_{i=1}^{c} U_{ik} = 1, 0 < \sum_{k=1}^{n} U_{ik} < n, \ and \\ U_{ik} \in [0,1] \, ; 1 \le i \le c; 1 \le k \le n\}. \tag{2}$$

The clustering criterion used to define good clusters for hard c-means partitions is the HCM function:

$$J_1(U,V) = \sum_{i=1}^{c} \sum_{k=1}^{n} (U_{ik}) D_{ik}^2(\mathbf{v}_i, \mathbf{x}_k), \qquad where \tag{3}$$

where $U \in M_{cn}$ is a hard partition matrix;
$V = [\mathbf{v}_1, \dots, \mathbf{v}_c]$ is a matrix of prototype parameters (cluster centers) $\mathbf{v_i} \in \Re^{\mathbf{s}} \ \forall \mathbf{i}$; and
$D_{ik}(\mathbf{v}_i, \mathbf{x}_k)$ is a measure of the distance from $\mathbf{x}_k$ to the i^{th} cluster prototype. The *Euclidean distance* metric is used for all HCM results reported here. Good cluster structure in X is taken as a (U,V) minimizer of (2). Typically, optimal (U,V) pairs are sought using an alternating optimization scheme of the type generally described in [17, 6].

The clustering criterion used to define good clusters for fuzzy c-means partitions is the FCM function:

$$J_m(U,V) = \sum_{i=1}^{c} \sum_{k=1}^{n} (U_{ik})^m D_{ik}^2(\mathbf{v}_i, \mathbf{x}_k), \qquad where \tag{4}$$

where $U \in M_{fcn}$ is a fuzzy partition matrix;
$m \in [1, \infty)$ is the weighting exponent on each fuzzy membership;
$V = [\mathbf{v}_1, \dots, \mathbf{v}_c]$ is a matrix of prototype parameters (cluster centers) $\mathbf{v_i} \in \Re^{\mathbf{s}} \ \forall \mathbf{i}$; and
$D_{ik}(\mathbf{v}_i, \mathbf{x}_k)$ is a measure of the distance from $\mathbf{x}_k$ to the i^{th} cluster prototype. The *Euclidean distance* metric and *diagonal distance* metric [6] are used for all FCM results reported here, with $m = 2$. The larger m is, the fuzzier the partition. Good cluster structure in X is taken as a (U,V) minimizer of (2). Typically, optimal (U,V) pairs are sought using an alternating optimization scheme of the type generally described in [6].

3 Genetically guided clustering

In any generation, element i of the population is V_i, a $c \times s$ matrix of cluster centers in FCM/HCM notation. The initial population of size P is constructed by random assignment of real numbers to each of the s features of the c cluster centers. The initial values are constrained to be in the range (determined from the data set) of the feature to which they are assigned, but are otherwise random.

Since only the V's will be used within the GA it is necessary to reformulate the objective functions (3) and (4) for optimization. For HCM each data vector is assigned to the nearest cluster via some distance metric (Euclidean here). Given the way assignments to clusters are made, it is follows that

$$R_1(V) = \sum_{k=1}^{n} min\{D_{1k}, D_{2k}, \cdots, D_{ck}\} \tag{5}$$

is an equivalent reformulation of J_1 which eliminates U.

In order to work only with V's in FCM, equation (4) can be rewritten by substitution for U using the first order necessary condition for U [6]. Specifically, for $m > 1$ as long as $D_{jk}(\mathbf{v}_j, \mathbf{x}_k) > 0 \quad \forall\, j, k$, we can substitute

$$U_{ik} = 1/\sum_{j=1}^{c} \left(\frac{D_{ik}(\mathbf{v}_i, \mathbf{x}_k)}{D_{jk}(\mathbf{v}_j, \mathbf{x}_k)} \right)^{2/(m-1)} \quad for\ 1 \le i \le c; \quad 1 \le k \le n. \tag{6}$$

into (2), resulting in the reformulated FCM functional

$$R_m(V) = \sum_{k=1}^{n} \left(\sum_{i=1}^{c} D_{ik}^{1/(1-m)} \right)^{1-m} \tag{7}$$

Hathaway and Bezdek [27] have shown that local (V) minimizers of R_m and (U) at (3) produce local minimizers of J_m, and conversely, the V part of local minimizers of J_m yields local minimizers of R_m. Our approach will be to optimize R_m with a GGA. The reformulation theorem given in [27] provides theoretical justification for this technique.

The genetic guided algorithm is shown in Figure 1. It consists of selecting parents for reproduction, performing crossover with the parents, and applying mutation to the bits of the children. We use a binary gray code representation in which any two adjacent numbers are one bit different. This encoding in some cases yields faster convergence and improved performance over a straightforward binary encoding [2]. Recently, Fogel and Ghozeil [22] have shown that all bijective representations are theoretically equivalent; however, there may be a most efficient representation for a particular problem. In our experiments, gray coding was more effective than simple binary encoding [26]. For all experiments reported here an elitist strategy [3] of passing the two fittest population members to the next generation was used. This guarantees that the fitness never declines from one generation to the next, which is often a desirable property in clustering and is useful in tracking population improvement.

In the process of experimenting with hard c-means we occasionally observed GGA getting caught in an extremum associated with a degenerate partition (i.e a partition with one or more empty rows meaning that fewer than c clusters were obtained in the final partition). A row i is considered empty if $u_{ik} \leq 0.00001, \forall k$. To minimize the chance of the GA becoming stuck at a degenerate partition, we use the following heuristic.

- If a partition, defined by its cluster centers $\mathbf{V}$, has b clusters with no feature vectors assigned to them and the objective function for the partition evaluates to a value Tot, the new value will be $b \times Tot$.

This results in the modified objective function:

$$R'_m(V) = R_m(V) + b \times R_m(V), \tag{8}$$

where $b \in [0, c]$ is the number of empty clusters. Since the clustering goal is to minimize the objective function, the above heuristic penalizes degenerate partitions by increasing their objective function value. This makes them less likely to be chosen for reproduction and less likely to survive to the next generation in any form. The above heuristic is applied in all experiments using the HCM objective function.

The following subsections describe our approach to selection, crossover, and mutation. In each case our choices are motivated by the desire to apply genetically guided clustering to image data sets, which are usually large. So, small populations and minimal generations are preferred.

3.1 Selection

The selection mechanism is k-fold tournament selection [25]. By default all results reported here use k=2 unless otherwise stated. From two randomly chosen population members, the more fit is chosen to be one parent. This process is repeated with the two new competitors chosen from the entire population to find the second parent. The two population members selected are then used in the crossover operation. All parental pairs for crossover are selected in this way.

3.2 Crossover

Each feature v_{ij} of a cluster center is a real number. In order to do crossover the feature value is converted into an integer which captures three decimal places of precision. The integer is then represented as a binary string. For example, if the largest value for a feature is 7.999 then it will be converted to 7999 which may be represented by a 13 bit number.

GGA.1 For HCM/FCM: Choose m, c, and D_{ik}.

GGA.2 Randomly initialize P sets of c cluster centers. Constrain the initial values to be within the space defined by the vectors to be clustered.

GGA.3 Calculate R_m by Eqn. (5) or (7) for each population member. Apply (8).

GGA.4 Convert population members to binary gray codes.

GGA.5 For i = 1 to number of generations Do

- Use k-fold tournament selection (default $k = 2$) to select P/2 parent pairs for reproduction.
- Do two point crossover and bitwise mutation on each feature of the parent pairs.
- Calculate R_m by Eqn. (5) or (7) for each population member. Apply (8).
- Create the new generation of size P from the 2 best members of the previous generation and the best children that resulted from crossover and mutation.

GGA.6 Provide the cluster centers for the terminal population member with the best (smallest) R'_m value and report R'_m. Note that $R'_m = R_m$ for all non-degenerate partitions.

Figure 1: The GGA algorithm

The binary representation of a V matrix with c clusters and s features each represented by 13 bits requires $c \times s \times 13$ bits (for this example).

Two-point crossover [24] is done on each of the c cluster centers of the mating parents generating two offspring. It is necessary to do c crossovers to enable each cluster center to move independently of the others. Adjusting each cluster center every generation minimizes the number of generations needed. This is important because each generation requires significant time in the calculation of the R_m value of each population member.

3.3 Mutation

After every crossover, each bit of the children is considered for mutation with a mutation probability p_m. Mutation consists of flipping the value of the chosen bit from 1 to 0 or vice versa.

4 Setting Parameters and related issues

The genetic guided approach to clustering has more parameters to set than FCM or HCM. There is the mutation rate, the crossover probability, the stopping criterion (generations, population diversity, other), the order of tournament selection, and the population size. How should these parameters be chosen?

Generally, we have found that a crossover rate of 90-100% offers best results. The mutation rate is important and can be estimated with the following equation [3, 33]:

$$p_m \approx 1.75/(P \times \sqrt{(}bits)) \tag{9}$$

where P is the population size and *bits* is the number of bits for an individual population member. Generally, we used a slightly higher mutation rate than determined by (9). We found that a population size of at least 30 is needed for acceptable results, and better performance can often be obtained by using a larger population. Generally, we found a population of 75 to be an acceptable upper limit on population size for the GGA approach.

The order of tournament selection controls how quickly a population is "taken over" by dominant individuals in the population. A higher order of tournament selection leads to faster convergence to a local extremum. However, to provide adequate search in the case of many extrema, the tournament order needs to be low (e.g. 2); otherwise, a "poor" local extremum that is found early may become the optimized value of the GGA. In our experiments, we found it useful to increase the order of tournament selection (to 10) for the last n (usually n=50) generations to push the GGA towards a partition associated with a local extremum of FCM/HCM. This approach enabled us to avoid the necessity of running HCM or FCM on the final GGA cluster centers to improve the GGA solution, and it also used fewer generations for most domains. See [2] for a good overview of takeover times (in terms of generations) for different orders of selection on different population sizes. For example, the takeover time for a population of size 30 and order 10 is 2.01 generations vs. 6.67 generations for order 2.

In domains where clustering will be applied many times (e.g. images of the same region), a subset of the data to be clustered can be used to determine useful crossover and mutation rates. When little improvement occurs for the best individual of the population in early generations, the mutation rate is usually too high (or low) and can be adjusted to increase the rate of convergence for the sample set of data. After a "good" mutation rate is found crossover can be varied to determine whether better extrema can be found in the sample set of data for an equivalent stopping criteria.

Our stopping criterion for GGA was the number of generations. There are several other possibilities for determining when to stop. Options include examining the standard deviation of the population members' fitness values, or stopping when the current standard deviation reaches some percentage of the original population standard deviation. Alternatively, a hard limit can be used for termination on the value of the standard deviation. The GA can also be halted after the fitness value of the best population member has not changed for n generations, with $n = 20$ or $n = 30$ being a reasonable choice for our approach. This places a premium on finding extrema quickly.

A final variable that can be manipulated is the set of cluster centers used in the initial population. They can be created by, for example, running FCM or HCM for q iterations on a random initialization. We experimented with this approach for FCM in the single feature and Iris domains. However, the results with a 1 iteration initialization were within a standard deviation of those obtained with random population initialization. Further, the number of generations required to find a local extremum was not reduced. The GA does not use the initial population in the same way as FCM/HCM, so these results are not surprising.

4.1 Automatically setting crossover and mutation rates

There have been various suggestions for automatically setting crossover and mutation rates [15, 21, 38]. We have experimented with a technique proposed by Srinivas and Patnaik in [36]. In their approach each population member has its own crossover and mutation probability. Note that in clustering the maximally fit individual has the minimum $R_m(V)$ value, so in the notation below f_max corresponds to $max_V(1/R_m(V))$. The crossover and mutation probabilities are calculated as follows.

Following [36] let f_{max} be the maximum fitness in a population, $\bar{f}$ the average fitness in a population, f the fitness of an individual child about to have mutation applied to it and f' be the larger of two fitness values of individuals about to have crossover applied to them. Then the probability of crossover p_c is given by:

$$\begin{aligned} p_c &= k_1(f_{max} - f')/(f_{max} - \bar{f}), \quad f' \geq \bar{f}, & (10) \\ p_c &= k_3, \quad f' < \bar{f} & (11) \end{aligned}$$

The probability of mutation is given by:

$$\begin{aligned} p_m &= k_2(f_{max} - f)/(f_{max} - \bar{f}), \quad f \geq \bar{f}, & (12) \\ p_m &= k_4, \quad f < \bar{f} & (13) \end{aligned}$$

where, $k_1, k_2, k_3, k_4 \leq 1.0$.

Srinivas and Patnaik showed good results in [36] using: $k_1 = k_3 = 1, k_2 = k_4 = 0.5$. These settings cause solutions with low fitness to be disrupted by mutation. The most fit individual does not have mutation or crossover applied to it by the above equations. To prevent premature convergence, a default minimum mutation rate of 0.005 is used for all population members. The above settings were used here.

5 Data set descriptions

Six data sets were used for the experiments reported in the next section. Two artificial data sets are useful for this study because they each have multiple local extrema for J_m.

The two-dimensional touching clusters data (25 points), Bezdek [5], was used by Babu and Murty [1] and is shown in Figure 2. The other artificial data having multiple local extrema is a single feature data set consisting of the output y of the non-linear equation $y = (1 + x_1^{-2} + x_2^{-1.5})^2$, where $1 \leq x_1, x_2 \leq 5$. This equation is taken from [37], where 6 classes were found to provide a useful grouping in the development of fuzzy rules to model the output of the equation over a set of 50 distinct values. The same 50 y values were used in this study.

The other two small data sets are the Iris data and multiple sclerosis data [29]. Iris consists of 150 patterns belonging to 3 species of Iris, with 50 patterns in each class. Each of the patterns is described by 4 real-valued features: petal-length, petal-width, sepal-length, and sepal-width.

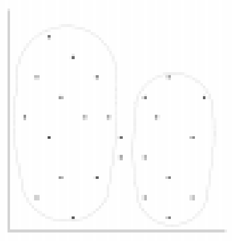


Figure 2: Touching clusters data with artificial boundaries.

The multiple sclerosis (MS) data consists of 2 classes each described by 5 measurements (features) of a patient. The classes are MS (29 examples) and non-MS (69 examples). The first feature is age, and measurements connected with two different visual stimuli (S1 and S2) provide the other 4 features. These four features are the sums and absolute differences of the responses of the stimuli as observed in the left and right eyes, respectively. Letting L stand for left eye response and R stand for right eye response, these features are $(S1L + S2L)$, $(|S1L - S2L|)$, $(S2L + S2R)$, and $(|S2L - S2R|)$.

Clustering is often applied to image segmentation [8]. Images are real-world domains of significant complexity in terms of number of items to be clustered and number of classes. We use two images in the experiments discussed in Section 6.

The color Lena image was used in [34] to show that a simple hybrid GA using real values could be used to skip some local extrema when hard c-means was applied to cluster the middle 128×128 subset of the color image (available at http://vision.ce.pusan.ac.kr among other sites). The goal in [34] was color quantization, so the features used for clustering were the RGB space values with 8 bits/color. The values in each of the R, G and B planes were linearly stretched to fill the range $([0, 255])$. In the comparative experiment that we undertook the Lena sub-image was quantized from 24 bit color (8 bits/color) into 16 total colors, so clustering is done with 16 classes (c=16). That is, we process $n = 16,384$ vectors of $s = 3$ features (R, G and B), clustered into $c = 16$ classes.

Our magnetic resonance image consists of a 5mm thick slice with 3 features, T1, T2 and PD weighted images. The raw intensity values at each spatial location make 3D pixel vectors for clustering. This image consists of a significant number of data $(n = 22,320)$ to be clustered, the most in our study. Air, bone, fat and other extra-cranial tissue were removed from the original 256×256 images before clustering. Clustering is into 10 classes (c=10) with gray matter, white matter and cerebro-spinal fluid the classes of interest. More clusters than classes were used to provide clusters that are more likely to contain only one tissue type or are more homogeneous and to be consistent with [4].

6 Experiments and results

In all experiments the stopping criterion for FCM was a maximal squared difference in membership values of two successive U matrices of $\epsilon = 0.001$ (i.e. $max_{ik}\{(u_{ik}^{new} - u_{ik}^{old})^2\} < 0.001$). The stopping criterion for HCM was an iteration for which no data object changed classes. The initial V's for FCM and HCM were created randomly within the feature space. For each feature the initial values were randomly chosen to be in the range (determined from the data set) of the feature.

The two image domains require significant computation time for GGA and conventional clustering. Only the FCM algorithm was applied to the MRI image, which has been used in work on brain tissue quantization [4]. HCM was the only clustering algorithm applied

Table 1: FCM results with the Euclidean norm and J_2.

Data set	# of runs	Average Values		std. dev.	Lowest J_2 value
		iter.	J_2		
touching clusters	1000	11	216.140	0.81325	215.439
single feature	2000	26	0.926	0.099	0.731
Iris	5000	19	60.576	0	60.576
multiple sclerosis	3000	20	65766.453	0	65766.453

to the Lena sub-image since our intent was to compare GGA with the GCMA algorithm of [34], which uses HCM.

We will examine the results of the FCM experiments first because, for the studied domains we found fewer extrema, so presentation of the results is easier. For all FCM trials $m = 2$.

In doing experiments, we are primarily interested in domains that have multiple local extrema for a given clustering algorithm. For FCM, using the Euclidean distance metric, we can find only one extremum in the Iris and MS domains. The touching clusters data has just 2 extrema with the larger found less than 1% of the time. The single feature data set was found to have 11 extrema with the Euclidean norm and 10 extrema with the diagonal norm. In both cases only 3 of the extrema occur more than 10% of the time. However, the fact the best extremum is only the third most likely makes this domain an interesting test for genetic guided clustering.

Table 1 lists number of trials, the average J_2 values and the standard deviations found by FCM for each of the non-image domains using the Euclidean norm. Table 2 shows the extrema found by FCM for each of the 4 small data sets. The diagonal norm is used on the single feature data set with the average of the extrema obtained shown in Table 3 and the exact extrema values shown in Table 4. The diagonal norm is used to illustrate that the GGA approach is also effective with norms other than Euclidean.

Although GGA is applied to R_m in (7), we will discuss GGA outputs in terms of J_m values as computed by (4) in order to facilitate clear comparisons with FCM. Table 5 shows the GGA results in terms of J_2 values, number of generations, population size and mutation rate. All results are averages over 50 trials. Our GGA approach always finds the best known local extremum for these domains given a large enough population, sufficient generations, and a mutation rate high enough to enable the necessary search. This stands in contrast with two previous approaches, which both required subsequent iteration of FCM on the best GA result to find the best FCM partition (equivalent J_m values) [1, 10].

Table 2: FCM extrema found with Euclidean norm for 4 domains. Value is the extremum value and Count is the number of trials on which it occurred.

	Iris		MS		Touching Clusters		single feature	
Extrema	Value	Count	Value	Count	Value	Count	Value	Count
1	60.576	5000	65766.453	3000	215.439	994	0.731	355
2					332.32	6	0.893	889
3							0.949	10
4							1.016	1
5							1.050	720
6							1.068	3
7							1.069	5
8							1.104	3
9							1.488	12
10							1.535	1
11							1.689	1

Table 3: FCM results with the diagonal norm.

Data set	# of runs	Average Values			Lowest J_2 value
		iter.	J_2	std. dev.	
single feature	2000	23	0.8165	0.165	0.686

In the single feature domain, as the mutation rate is lowered the GA finds the second lowest extremum (0.893) 5 times, which is indicated by a higher average J_2 in Table 5. Our experiments have indicated that low mutation rates work well (less than 5%) with less than (1%) being acceptable for the data sets with few extrema. For these small data sets, the GGA relatively quickly finds the same best extremum that FCM does in most cases. There is occasionally some slight roundoff error for the MS domain. So, for FCM a GA can skip local extrema and will terminate at a best FCM extremum in almost all cases.

This point is further illustrated by the results shown in Table 6 for the single feature data set with the diagonal norm. While there exist a number of extrema at which FCM gets stuck, the GGA does not get caught at all for a population of size 30 and a mutation rate within 0.05 of 5%. For a smaller population the GA can get trapped at the 0.893 extremum and gets trapped more often as the mutation rate is lowered. If the data set has many extrema, small populations converge too quickly to an encountered extremum without doing enough search to find better ones. As the mutation rate is lowered, the search pressure is reduced, leading to a situation wherein which a higher local extremum will not be escaped.

Table 4: FCM extrema for single feature data with Diagonal norm.

Extrema	Value	Count
1	0.686	1241
2	0.838	427
3	0.892	1
4	0.986	90
5	1.003	1
6	1.004	4
7	1.093	2
8	1.180	1
9	1.398	231
10	1.452	2

Table 5: GGA: Results with the Euclidean norm.

Data set	Pop. size	Gens.	Mut. Rate	Average Values		Lowest J_2
				J_2	st. dv.	value found
single feat.	30	500	0.046	0.731	0	0.731
single feat.	30	500	0.030	0.747	0.049	0.731
single feat.	20	500	0.046	0.734	0.023	0.731
touch. clus.	20	300	0.017	215.439	0	215.439
touch. clus.	20	300	0.005	215.439	0	215.439
Iris	30	300	0.008	60.58	0	60.58
multiple scl.	30	300	0.007	65766.469	0.014	65766.453
multiple scl.	30	300	0.009	65766.484	0.028	65766.453

6.1 MRI Results

Over 50 trials of FCM with $c = 10$, four extrema of the MR image were observed. They are shown in Table 7 along with the number of times each occurred. The average was J_2 = 69,289,175 with a standard deviation of 994,931. To reduce the time the GGA approach takes to run for the MR image, it was run in stages on random subsamples of the image pixels. This approach has been found [12] to be effective in speeding up FCM with no loss in the quality of the final partition. Eight subgroups of data of sizes $1004, 2008, 3013, 4017, 5022, 6026, 7030, 22320$ were used. The number of generations for each of the subgroups are 40, 30, 30, 20, 20, 20, 40, and 300 respectively for a total of 500 generations.

A population of size 30 was used for the MR image with $c = 10$ and 30 random trials were performed. To ensure that an extremum was discovered, the final set of cluster centers was used to initialize FCM, which then ran until it stopped with $\epsilon <= 0.001$ as before. The average after the application of FCM was $J_m = 69,002,432$ with standard deviation

Table 6: GGA: Results with the Diagonal norm.

Data set	Pop. size	Gens.	Mut. Rate	Average Values		Lowest J_2 value found
				J_2	st. dv.	
single feat.	30	500	0.046	0.686	0	0.686
single feat.	30	500	0.03	0.702	0.046	0.686
single feat.	20	500	0.046	0.689	0.021	0.686
single feat.	20	500	0.033	0.702	0.046	0.686
single feat.	20	500	0.005	0.723	0.065	0.686

Table 7: Magnetic resonance image extrema with 50 random FCM initializations and after initializing FCM with 30 GGA runs.

Value	FCM: Number of Occurrences	FCM $\circ$ GGA: Number of Occurrences
68,485,264	19	14
68,485,272	11	8
70,363,736	0	4
70,485,536	19	4
70,675,592	1	

of 758,508. The lowest raw value found by the GGA (i.e. before applying FCM) was $J_2 = 70,154,440$.

An experiment was run with a data size of 7% for 150 generations and 14% for 300 generations. The population size was 50 and the resulting cluster centers were used to initialize FCM, which then ran until it stopped with $\epsilon <= 0.001$. This scheme is indicated in Table 7 by FCM $\circ$ GGA (GGA followed by FCM). The average J_2 value was $69,202,458$ with standard deviation $959,368$ which is a small improvement over random initialization with FCM. For this set of data, the GGA found one of two extrema with the higher one resulting in initializations for FCM that resulted in one of the poorer partitions. A larger population should improve performance in this case.

In the MRI domain, effective GA clustering requires larger populations and more generations to generally provide the best results. In applying the GGA to the MRI domain it was noted that the best J_2 value is still decreasing at the end of the generations even with the reduced size data set. There is a direct correlation between the final J_2 value for the reduced data set and the final J_2 value after applying FCM. High J_2 values result in the poorer final J_2 values.

6.2 HCM experiments

In this subsection, the performance of our GGA using the HCM objective function on the Iris, single feature and MS domains will be discussed. The touching clusters domain will not be discussed with the HCM objective function as it was included for comparison with earlier work done with the FCM objective function [1].

Table 8 shows the J_1 values of the local extrema (and some degenerate partitions) found for many random initializations of HCM in the Iris, MS and single feature domains. Only the 4 most likely to occur extrema for the single feature domain are shown as it has 96 local extrema and degenerate partitions which are displayed in the histogram in Figure 3. There are 58 single feature extrema, which occur in the J_1 value range $[1, 3]$ more than 10 times over 20,000 trials. Several of the hard partitions discovered during these experiments were degenerate, i.e. consisted of some clusters that have no data elements assigned to them. The single feature extremum at $J_1 = 53.236$ is a degenerate partition of 1 cluster; at $J_1 = 8.9$ and $J_1 = 6.8630$ lie degenerate partitions of 3 clusters; at $J_1 = 5.3$ is a degenerate partition of 4 clusters; and finally, at about $J_1 = 5.2$ there are 3 slightly different degenerate partitions of 5 clusters.

Clustering the Iris data can result in 2 degenerate crisp partitions, of 1 cluster ($J_1 = 680.824$) and 2 clusters ($J_1 = 152.309$) which occur in more than 18% of the 9000 trials. The best two partitions of the Iris data differ by 1 feature vector and have 17 misclassifications (as resubstitution errors of the crisp nearest prototype classifier) and 16 misclassifications, respectively at $J_1 = 78.945$ and $J_1 = 78.941$. Clustering the MS data with HCM can result in a degenerate partition as well as a partition with a high $J_1 = 162,345.28$ value that contains 45 resubstitution errors on the 98 feature vectors. The other two non-degenerate MS partitions found by the application of HCM differ by 1 feature vector (with 17 and 16 misclassifications at $J_1 = 82494.57$ and $J_1 = 82572.96$ respectively). The MS domain provides an example in which the lowest extremum of J_1 does **not** result in a partition which best groups the classes in the data.

The GGA was applied 50 times with different random initial populations to each of the data sets used in Table 8. Again, for clarity we report J_1 values even though the equivalent R_1 functional was optimized. Table 9 provides a summary of the results by data set, population size, mutation rate, average J_1 value found, standard deviation and lowest J_1 value found.

The GGA applied to the single feature data set finds two extrema, $J_1 = 0.935$ most of the time and $J_1 = 1.189$ the rest of the time. The 1.189 extremum is found 2, 3, 2 and 0 times respectively for the three single feature entries shown in Table 9. The GGA never

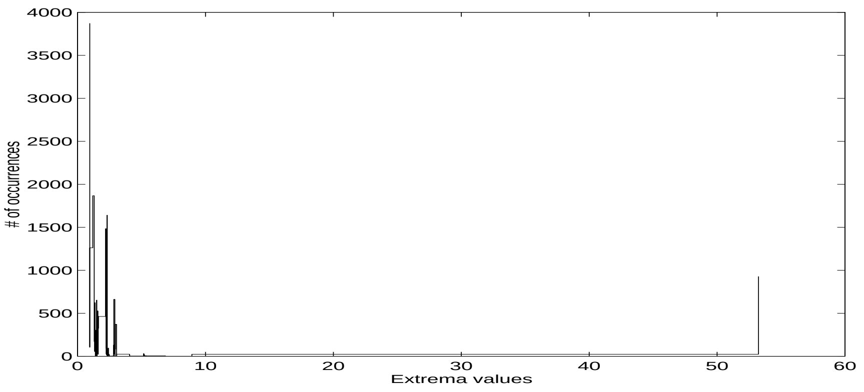


Figure 3: The frequency plot of J_1 values of 91 HCM extrema and the 5 degenerate partitions for the single feature domain (found over 20,000 trials).

Table 8: J_1 values associated with partitions found by applying HCM with the Euclidean norm for the Iris, MS, and single feature domains.

	Iris		MS		single	
Partition	J_1 Value	Count	J_1 Value	Count	J_1 Value	Count
1	78.941	2857	82494.57	6802	0.935	3781
2	78.945	3929	82527.961	2259	1.189	1867
3	142.852	18	162345.281	10	2.1810	1484
4	142.859	151	178135.359	629	2.3040	1641
5	142.879	34				
6	143.218	1				
7	143.454	307				
8	145.279	70				
9	152.369	1049				
10	680.824	584				

found degenerate partitions in this domain or any partition, except the two best. The best results were with populations of size 50.

The GGA applied to the Iris data also finds the two partitions with lowest J_1 values in each of the 50 runs. It finds the best partition 28, 25, 26, 31 and 32 times respectively for the Table 9 entries from top to bottom. The last two results were obtained with a crossover rate of 90%, which enabled a slight improvement in terms of the number of times the second best partition was found. The GGA applied to the MS data set finds the best two results only with 7, 6, 0 and 1 instances of the second best result, respectively for the 4 results shown in Table 9.

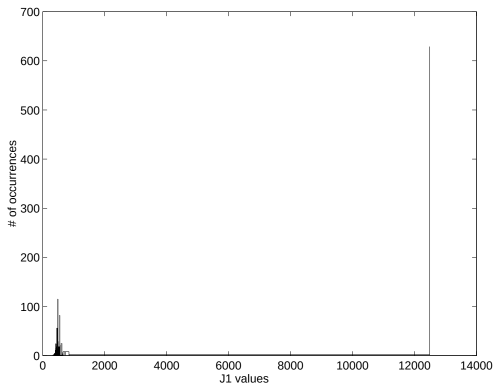
Table 9: GGA: Results for 50 trials with hard c means and the Euclidean Norm. Best results are in bold for each data set.

Data set	Pop. size	Gens.	Mut. Rate	Average Values		Lowest J_1 value found
				J_1	st. dv.	
single feat.	30	550	0.046	0.947	0.050	0.935
single feat.	30	550	0.030	0.951	0.060	0.935
single feat.	50	550	0.020	0.945	0.050	0.935
single feat.	50	550	0.030	0.935	0.001	0.935
Iris	30	750	0.0015	78.943	0.002	78.941
Iris	30	750	0.003	78.943	0.002	78.941
Iris	50	550	0.0015	78.943	0.002	78.941
Iris	50	550	0.0015	78.942	0.002	78.941
Iris	75	400	0.0064	78.942	0.002	78.941
MS	50	700	.0113	82499.281	11.584	82494.570
MS	50	800	.0088	82498.578	10.843	82494.570
MS	50	3300	.0088	82498.578	0.003	82494.570
MS	75	800	.0088	82495.250	4.674	82494.570

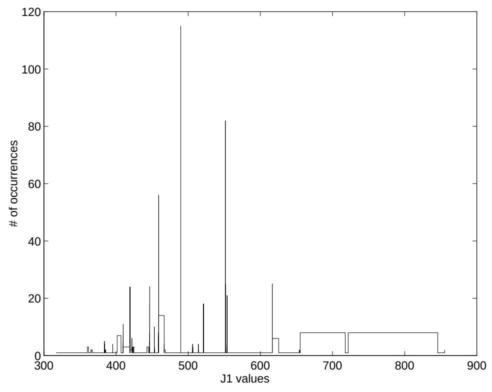
6.2.1 Lena image

A 128×128 subset of the Lena image was clustered 1800 times by HCM with random initializations for $c = 16$. Out of these only 1 crisp partition was non-degenerate (i.e. consisted of 16 clusters). There were 433 degenerate partitions found. The most numerous one (629 times) consisted of just one class. Figure 4 shows two histograms of the J_1 values associated with the final partitions. View (a) in Figure 4 plots all 1800 values of J_1 found by HCM, while view (b) accumulates the frequencies for all values in the $(1800 - 629) = 1171$ trials that do not include the $c = 1$ case. In [34] 5000 runs of HCM with random initializations were done on this data set and all of them apparently resulted in degenerate partitions [35]. To compare with the earlier results on this data set, we report the MSE rather than the raw values from the HCM objective function (MSE = $J_1/16384$).

In order to obtain some non-degenerate partitions for comparison with our GA results, a different type of less random initialization of HCM was used. In this initialization scheme, each cluster center was initially assigned the values of a feature vector from the data set. The selection of feature vectors was random, but with the restriction that a feature vector could be selected only once, which forces each cluster center to be distinct (assuming no two feature vectors are the same). For this approach with over 1200 trials, only 2 trials resulted in degenerate partitions! There were, however, a significant number of extrema found, 817. The most that any extremum was found was 38 times, $J_1 = 316.87$, followed by 31 times for $J_1 = 321.936$. Figure 5 shows a plot of all the J_1 values found for this set of partitions. There are 106 extrema in the 313 range, 79 in the 314 range and the rest range from 316 to 335. Figure 6 is a plot of all the J_1 values associated with partitions found for the 3000 initializations excluding those corresponding to $c = 1$.



(a)



(b)

Figure 4: (a) All of the J_1 values found by applying HCM to the Lena image with random initializations at $c = 16$ and (b) without 629 $c = 1$ degenerate partition instances.

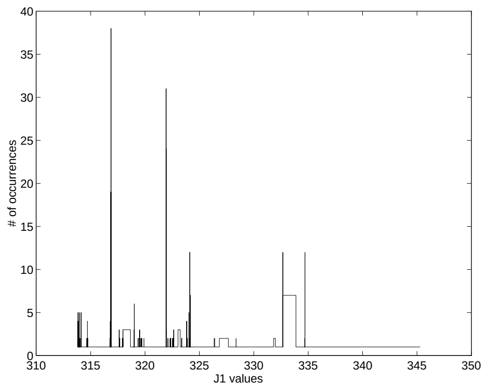


Figure 5: A frequency plot of all 1200 J_1 values found by applying HCM to the Lena image with each cluster center initialized to a distinct feature vector in the data set with $c = 16$.

Applying the GGA approach to the Lena domain is computationally expensive. To enable more tractable experiments, we use sub-sampling to see if the GA approach can provide an initialization for HCM that results in a partition with one of the lower extremal values. The GGA with population size 30 was applied to a randomly chosen subset of the original data (consisting of 7% for 150 generations and then 14% for the next 1850 generations). The mutation rate was 0.004. At the end of 2000 generations with order 10 tournament selection the cluster centers are still changing, indicating that an extrema

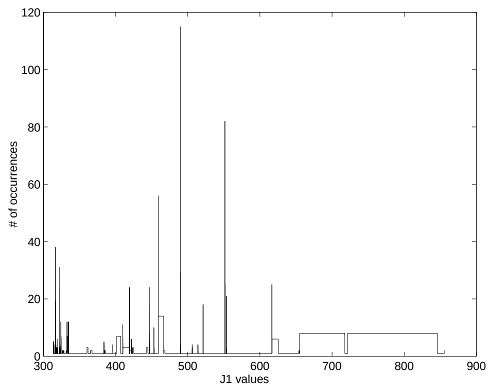


Figure 6: A frequency plot of all 3000 J_1 values associated with partitions found by applying HCM to the Lena image when the degenerate $c = 1$ value is left out.

has not been found. The best cluster centers from the final GA generation were used to initialize HCM, which was then run until no changes occurred in the partition (no feature vectors changed classes for 1 iteration). The GGA results are averaged over 30 trials with different random initializations. The average mean squared error (MSE) and the standard deviation are reported in Table 10. In Table 10 we exclude the single partition ($c = 1$) result from the random initialization scheme because its inclusion obscures the results by making the standard deviation larger than the average.

Table 10: Comparison of GGA and HCM on the Lena sub-image, $c = 16$.

Algorithm	Average MSE	Standard Deviation
HCM Best init.	320.73	4.3203
HCM Random init.	475.2704	56.9159
HCM Overall	397.939	81.1104
Best HCM $\circ$ GGA P=75	316.826	2.0598
GGA on full Data, P=50	317.245	3.44
GGA P=30	318.758	4.632

An experiment with a larger population (75) was also tried, with 7% of the data randomly chosen and used for 150 generations and then 14% for the next 2050 generations. The mutation rate was 0.008. With a larger search space we expected the larger population to yield better results. Order 2 tournament selection was used for the first 2150 generations and then order 10 was used for the last 50 (to drive the algorithm towards an extremum). The results improved to an average of 316.8268 and a standard deviation of 2.0598. This indicates that a relatively small random subsample can be used to get good initializations for HCM in a large domain.

A second experiment with a population of size 50 that used all of the data for the last 300 generations resulted in $ave = 317.245$ and $std.dev = 3.4423$. Order 2 tournament selection was used until the last 50 generations, followed by order 10 for the final 50. The mutation rate was 0.004 and the random subsampling schedule was 7% of the data for the first 200 generations followed by 14% for the next 1200 generations, then 21% for 300 generations and the full data set for the last 300 generations (2000 total generations). This genetic clustering approach provided the best overall partitions and always yielded a reasonably good partition when HCM was initialized by the terminal GGA partition. This scheme always avoided degenerate partitions.

These results appear to agree with those in [34] where the population size was 500, the GA clustering scheme used real-values, HCM was initialized on each set of V values in the population, HCM was then run and finally crossover and mutation were applied. After applying the Scheunders algorithm for 10 generations, extrema in the low 300 range were found (as shown in a histogram). In [34] 2 experiments were run and all of the population members were plotted. Scheunders reported that 10% of the final population was in the lowest extremum which was called a global optimum at an MSE=310 (which was never found by HCM given a random initialization). The MSE found in [34] is slightly different than any found in our experiments although we attempted to exactly re-create the experiment. Scheunders was unclear on exactly how he created his data and we used the most likely approach [35].

6.3 Results with adaptive parameter settings

All experiments reported in [36] use a population size of 100 or greater, linear scaling and the stochastic remainder selection technique [3, 16] to get better results than fixed GA's on several domains. This relatively large population size will slow the clustering process, as does the requirement to calculate p_m, p_c for each individual involved in crossover and mutation. Hence, we experimented with smaller populations, but got poorer performance than with our fixed GGA's and poorer performance than with larger populations. Table 11 shows the results for 3 data sets using the adaptive genetic algorithm approach applied to the HCM functional with order 2 tournament selection. These results are generally not quite as good as the best of those obtained with the GGA approach reported in Table 9.

Table 11: Adaptive GA results with HCM over 50 trials.

Data set	Population size	Generations	Extrema, # Found ()	Extrema, # Found ()	Ave.	Std. Dev.
single	50	350	0.935, (43)	1.189, (7)	0.971	0.088
single	100	350	0.935, (44)	1.189, (6)	0.966	0.083
single	100	5000	0.935, (46)	1.189, (4)	0.955	0.037
Iris	50	500	78.941, (22)	78.945 ,(28)	78.943	0.002
Iris	100	800	78.941, (31)	78.945, (19)	78.942	0.002
Iris	100	2000	78.941, (33)	78.945, (17)	78.942	0.002
MS	50	350	82494.57, (38)	82527.96, (12)	82502.578	14.261
MS	100	350	82494.57, (44)	82527.96, (6)	82498.562	10.851
MS	100	4000	82494.57, (50)		82494.578	0.004

However, in all cases where the smallest extrema is not found, the second lowest extrema is found.

These results are representative of those found with the adaptive GA and FCM, where for the single feature data (the only small domain with more than 2 extrema for FCM) the lowest extremum was found 41/50 times in 350 generations and the second best was found the other 9 times. For the other small domains the best J_2 value was always found with the adaptive GA for populations as small as 30 and 600 generations. However, in those domains there has only been 1 extremum found in all but 1 case (2 for touching clusters, but the larger one less than 1% of the time).

The adaptive GA approach experimented with here provides a useful genetic clustering algorithm that gets good results. However, the use of non-adaptive mutation and crossover probabilities leads to better final partitions in some cases for both FCM/HCM.

7 Time considerations

The GA approach to clustering requires time proportional to the number of bits per feature, population size, number of features, number of feature vectors and number of clusters. An increase in any of these parameters results in a linear increase in time per generation for the GA clustering approach.

The dominant cost is that of calculating the fitness value for each population member for each generation. This cost is a function of s, n, and c, i.e. the number of features, number of objects, and number of classes to cluster the data into. The number of bits needed to represent features has a lesser effect on the time per generation. GA speed could

Table 12: CPU Time in seconds for different approaches to clustering the Iris data. Time is per generation or iteration, respectively. The population size of the GA's is 50.

GGAHCM Binary	GGAHCM Real	HCM	FCM	GGAFCM Binary	GGAFCM Real
0.035	0.0146	0.00101	0.00253	0.0313	0.0202

be much improved by a parallel implementation where the P fitness values are calculated in parallel for each generation.

The Iris data set is a good concrete example of the times required by the GA approach to clustering as opposed to FCM or HCM. Table 12 shows the time per generation or iteration in seconds for GGA on the HCM functional (GGAHCM), GGA applied to the FCM functional (GGAFCM), FCM and HCM on the Iris data. The GA population is size 50. The times are CPU times on a SUN Ultra-Sparc 1 running the Solaris 2.6 operating system. Times for real-valued representations are included and will be discussed in the next section. HCM is about twice as fast as FCM. The GA approach is about 15 to 20 times slower than HCM/FCM for a generation vs. iteration. The maximum number of iterations for HCM/FCM to terminate on the Iris data is 20, while the GA will need close to 300 generations. Hence, GGA clustering can take up to 2 orders of magnitude longer than FCM or HCM in some cases.

The adaptive GA approach is approximately 2.5 times slower than the non-adaptive GA version. This is due to the need to calculate fitness for every child before mutation is applied as well as recalculation of the fitness value after mutation, and the fact that the probabilities of mutation and crossover must be calculated at each application of the operators.

The GGAFCM applied to subsamples of the MRI data (7% of the data for 150 generations and 14% of the data for 300 generations) with a population of size 30, takes an average time per generation of 4.2 seconds on an unloaded Ultra-sparc 1 or 15.75 hours to run 30 trials. For the GGAHCM applied to the Lena data for 7% of the data for 150 generations and 14% of the data for 2050 generations with a population of size 30 an average of 0.47 seconds per generation or 8.6 hours for 30 trials is required.

7.1 Improving performance

The GGA approach generally finds a very good partition if it uses a population of the appropriate size (i.e. large enough). It can be caught by extrema that are close to a better solution (i.e. partitions may differ by only a few data assignments to clusters). To escape "nearby" extrema a higher mutation rate can be applied as a population begins to converge. At the same time a slight lowering of the crossover rate will cause just some cluster centers to be modified [32] by crossover. Another approach would be to perform local search on a final partition using a validity guided clustering algorithm [4] or some other approach which attempts to split and merge clusters to optimize a validity metric.

One of the many other approaches to crossover, mutation, selection, and choice of members for the next generation [16, 20, 31, 3] may provide better performance in some domains. Experiments with mutation as the main operator in optimizing the cluster center [26, 11] provided worse partitions than those reported here.

It is possible that a real-valued encoding [16, 34] might provide results that are better than those reported here, and the real-valued GGA approach will be faster per generation without the need to encode/decode cluster centers. In fact, a real-valued approach with each population member represented as a list of real-numbers was implemented by us. Blended crossover [20] and non-uniform mutation [31] were used as the respective crossover and mutation functions. A different blend point was chosen for each cluster. In applications to the Iris and MS domains, optimizing the HCM functional, the time was reduced by a factor of greater than 2 per generation. The time gain was only about 30% for the FCM GGA with a real-valued representation as shown in Table 12. The GGA with a real-valued representation requires more generations than the GGA with a binary representation which offsets the per generation time gains. The optimization performance was approximately as good as the gray valued GGA in either domain in our limited tests on populations of size 50. For the Iris data, the best result (100% crossover and probability of mutation $p_m = 0.067$, alpha for blended crossover of 0.5 and b=5 for the non-uniform mutation) was 31/50 trials (2000 generations) at the lowest found extremum and the other 19 trials at the second best J_1 extremum. For the MS data the best result was 49/50 trials at the lowest extremum and the other at the second lowest extremum with the same parameters as above except $p_m = 0.056$ for 1000 generations.

We tried using an approach of 100% blended crossover (alpha=0.5) with 2/3 probability and 100% non-uniform mutation with 1/3 probability [16]. For the Iris data, 41/50 partitions are found at the best known extrema and 9/50 at the next best in 1500 generations. For the MS data there are 50/50 at the lowest known extrema (or almost at the lowest known extrema) after 500 generations. This approach is slightly better than our binary approach for the Iris data. We also applied the (2/3, 1/3) real-valued GA approach to the reduced MRI data (7% for 200 generations and then 14% of the data for 400 generations) for a total of 650 generations with a population of size 50. FCM was then applied to the final partition produced by the GA for each of 30 trials. The results were an average $J_2 = 69,206,518$ with standard deviation of $964,694$. These results are insignificantly worse than those obtained with the binary GGA. In general, the results with the real-valued GGA do not provide any new clustering insights and are generally about the same as those using a binary representation which is predicted by the Fogel and Ghozeil paper [22].

8 Related work

In this section we provide further comparisons of our genetic guided hard/fuzzy c-means clustering results to other work on genetic guided clustering. In [10] a partition is treated as set of nodes and edges. Nodes (data elements) in the same class are connected by an edge. An ordered binary representation of the edges is used to represent the partition. For large data sets the strings will be quite long $((N \times (N-1))/2)$ where N is the number of data points. The system was compared against two greedy search algorithms from [28]. In limited tests on the British towns data it sometimes outperformed the greedy search algorithms. However, there is no evaluation of initialization for the greedy approach or comparison to FCM/HCM. In [1] an evolution strategies approach is used to perform clustering with the hard and fuzzy-c-means algorithms. They did not find final partitions with the evolution strategies clustering algorithm, but used the final partition as an initialization to hard/fuzzy c-means. They used the touching clusters data set, which is used here, as one example to illustrate their approach. In our work with an evolutionary strategies approach using just mutation and selection we could not generally find final partitions of FCM that were equivalent to those produced by alternating optimization [26, 11].

In [9] a genetic guided clustering system is run on two simple examples. A binary representation is used and the GA finds cluster centers close to optimal for a touching clusters data set and a well separated data set. No comparison is made to partitions produced by FCM/HCM nor are J_m values provided. In [30] there are no details of the evolutionary clustering algorithm given. It operates on cluster centers as ours does. Experiments with FCM on a simple data set are successful. Experiments in optimizing partitions for fuzzy c-shells [13] are less successful. The author notes the time complexity for evolutionary clustering is longer than an iterative approach, but does not quantify the difference. In [23] evolutionary programming is applied to find clusters in spatial data using a minimum description length evaluation function. This approach shows promise. Its data partitions are not compared with those generated by existing clustering algorithms.

Fuzzy clustering of noisy data using a GA approach is addressed in [7]. A binary integer representation of the cluster centers is used with c+1 cluster centers used in the case of c classes. The extra center is a noise cluster. The fitness function is the FCM functional, but it is not reformulated to remove the calculation of the U matrix as is done in this paper. The distance function is based on the work in [14]. Roulette wheel selection of the first parent coupled with random selection of the second, two point crossover on the whole string, and mutation are applied to a population of cluster centers. Elitism is implemented to keep the best member of a population. Each feature is represented in 8 bits. They claim that the binary representation outperformed a real-valued representation in this domain, but no results are given. The algorithm is tested on several synthetic examples. It finds cluster centers near those found by the noise clustering algorithm, called robust FCM [14] in the single case of no noise. It does not find acceptable cluster centers for the cases with noise. However, the algorithm is reported with a number of generations only 10 times the number of FCM or robust FCM iterations. In general, we found that more generations may be necessary for the GA to find equivalent cluster centers. They show the time for a GA generation to be approximately 1/10 that of a robust FCM iteration. This is partially because the robust FCM system does cluster validity (i.e. determines the correct number of clusters by applying a validity metric to the final partition for 2, 3, ..., c clusters). Robust FCM also has to calculate distance to an additional noise cluster. It is unclear how optimized the robust FCM code is. This approach might find better partitions faster by applying crossover to each feature of the cluster centers.

In summary, none of the previous work on genetic clustering shows that fuzzy partitions equivalent to those generated by alternating optimization methods can be reliably found. Our work does show this is true at least for small data sets. The work discussed here suggests that the GGA's may (with some further research in some cases) effectively optimize other types of clustering functionals besides FCM/HCM.

9 Summary and Discussion

A genetically guided approach to optimizing the hard and fuzzy c-means objective functions was described. The GGA algorithm was applied to four small data sets with two norm metrics and to two images (a magnetic resonance image of the brain and a color image of a scene). All experiments were carried out with at least 30 different initial populations to get a statistically meaningful average and standard deviation for the GGA clustering approach.

In contrast to earlier papers on GA clustering, we show that genetic guided clustering by itself can be used to provide the identical data partition that fuzzy c-means or hard c-means will when the latter are given the best possible initialization. The genetic clustering approach provides a framework for optimizing any clustering objective function that can be expressed in terms of a set of cluster centroids. The results shown here indicate that the GGA will always provide good partitions by settling in one of the most (in many cases the most) desirable extrema and never in an extremum representing a degenerate partition. It is ideal for testing an objective function for which no calculus based (or other approach) exists. If the data partitions or clusters produced by the genetic clustering approach are ”good”, faster approaches to optimizing the objective function can be developed.

In the introduction we promised answers to three questions about GA guided clustering. Here they are. In no case did the GGA find an extremum that was not found by FCM/HCM if given enough random initializations and we believe this will generally be true. The GGA approach can and does find the same extrema and associated partitions that FCM/HCM find. Further, the GGA will generally find one of the very best extrema and its associated partition if given enough generations and reasonable parameter settings. In domains with many extrema a larger population (between 75-100) will lead to the best partitions.

The GGA approach with the FCM functional finds the best local extremum for the Iris, MS and touching cluster data sets. It usually finds the best local extremum for the single feature data set. A subsampling configuration in which the full magnetic resonance image data set was used for the final 300 generations with a population of size 30 resulted in initializations to FCM that provided better final partitions on average than random initializations with FCM. Also on the MRI data set, we ran an experiment with a random subset (14%) of the data and a population of size 50 for 30 trials with FCM applied to the final clusters provided by the GGA. We found results that were about the same as randomly initializing FCM. It appears that the population needs to be larger in this computationally expensive domain.

For the three small data set experiments, HCM found more extrema than GGA, and GGA finds one of the best 2 extrema. In the majority of cases the best extremum is found by GGA (but not HCM) despite the fact that different extrema and their associated partitions can be very close. For example, the best two Iris and MS partitions differ by exactly 1 feature vector’s class assignment.

Applying HCM to the color image Lena subset 3000 times results in over 1000 local extrema, many of which represent degenerate partitions (i.e. partitions with less than the 16 clusters, representing colors, which were specified). The GGA approach *never* results in a degenerate partition. Due to the length of time required to test the GA 30 times, we used random subsamples in stages during clustering. Also, the GGA was run for a fixed number of generations rather than to convergence. HCM was then initialized with the final partition from the GA and run until it converged. The combined HCM $\circ$ GGA approach found low extrema even in tests with a small population size (30). A random subsample of size 14% (2294 feature vectors) also allowed for selection of an extremum and associated partition that was always among the best obtainable with the best approach to initializing HCM for this data set.

The exact choice of parameters may be determined on representative subsets of the data to be clustered. Alternatively, we have shown that an adaptive method [36, 32] of setting crossover and mutation rates provides partitions that are very good and is almost as consistent in finding these partitions over multiple trials as the best choice of crossover and mutation operators allows.

The real-valued approach to genetic guided clustering provides final partitions that are equivalent to those of the binary approach. The time cost is less per generation, but more generations are required resulting in no clear net time savings.

Overall, initialization has a significant effect on the final partition obtained by the iterative c-means clustering approaches discussed here. The GGA approach to clustering provides a viable way to avoid local extrema. However, it can take up to 2 orders of magnitude more time than FCM/HCM; in our experiments the GGA does take 2 orders of magnitude more time for all domains except MRI, where the average number of iterations is 296 and hence, the GGA approach takes just one order of magnitude greater time. However, it requires on average 114 iterations of FCM when initialized with the final GGA partition to find the final J_2 value. Hence, in the same time as the GGA one could on our experimental data sets, for example, try out 100 random initializations of FCM/HCM and use the partition associated with the lowest J_2/J_1 value. This approach will provide equivalent results to the GGA approach in the same amount of time for all domains except Lena. In fact, one could halve the time to find the best known J_2 or partition value by just using 50 random initializations for each domain. In the Lena domain, depending upon the initialization strategy, the GGA approach could provide a better final partition for the same amount of time.

The GGA approach never results in degenerate partitions which is a result of applying the penalty term in (8) to the objective function. Avoiding degenerate partitions is very useful in optimizing J_1 applied to data sets such as the Lena image where HCM will often find one of the degenerate partitions.

The GGA approach seems to have viability as a stand-alone optimization procedure for the hard and fuzzy c-means functionals only if its time cost can be reduced. However, even for the less time consuming real-valued GGA implementation the major cost is the evaluation of J_2 per individual. Orders of magnitude speed decreases for the GGA approach do not seem likely.

As noted above, a useful application of a modified GGA approach is as a framework for optimizing other functionals for partition generation, which can be compactly expressed, such as with a set of cluster centers, for which another optimization approach may not yet have been devised. The GGA produces good non-degenerate partitions which can be used to evaluate newly designed functionals.

Acknowledgements: This research was partially supported by the Whitaker Foundation and the National Cancer Institute (CA59 425-01). Amine Bensaid and Srinivas Boggavarapu contributed some code to the GA's used in the reported experiments.

References

[1] G.P. Babu and M.N. Murty, "Clustering with Evolution Strategies", *Pattern Recognition*, V. 27, No. 2, pp. 321-329, 1994.

[2] T. Back and F. Kursawe, "Evolutionary Algorithms for Fuzzy Logic: A Brief Overview", *Proc. of Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Vol. 2, pp. 659-664, 1994.

[3] T. Back, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, 1996.

[4] A. Bensaid, L.O. Hall, J. Bezdek, L.P. Clarke, M.L. Silbiger, J.A. Arrington, R.F. Murtagh, Validity-Guided (Re)Clustering for Image Segmentation, *IEEE Transactions on Fuzzy Systems*, V. 4, No. 2, May, pp. 112-123, 1996.

[5] J.C. Bezdek, "Cluster Validity with Fuzzy Sets", *Journal of Cybernetics*, V. 3, No. 3, pp. 58-73, 1974.

[6] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Functions*, Plenum, N.Y., 1981.

[7] M.A. Egan, M. Krishnamoorthy, and K. Rajan, "Comparative Study of a Genetic Fuzzy c-means Algorithm and a Validity Guided Fuzzy c-means Algorithm for Locating Clusters in Noisy Data", International Conference on Evolutionary Computation, pp. 440-445, 1998.

[8] J.C. Bezdek, L.O. Hall and L.P. Clarke, "Review of MR Image Segmentation Techniques using Pattern Recognition", *Medical Physics*, v. 20, No. 4, pp. 1033-1048, 1993.

[9] J.C. Bezdek and R.J. Hathaway, “Optimization of Fuzzy Clustering Criteria using Genetic Algorithms”, *Proc. First IEEE Conference on Evolutionary Computation*, Vol 2, pp. 589-594, 1994.

[10] J.N. Bhuyan, V.V. Raghavan and V.K. Elayavalli, “Genetic Algorithm for Clustering with an Ordered Representation”, *Proc. International Conference on Genetic Algorithms'91*, pp. 408-415, 1991.

[11] S. Boggaravapu, A genetic guided algorithm (GGA) for fuzzy clustering, M.S. Thesis, University of South Florida, Tampa, 1995.

[12] T.W. Cheng, D.B. Goldgof and L.O. Hall, *Fast Fuzzy Clustering*, *Fuzzy Sets and Systems*, V. 93, pp. 49-56, 1998.

[13] R.N. Dave, “ Fuzzy Shell Clustering and Application to Circle Detection in Digital Images”, *International Journal of General Systems*, V. 16, pp. 343-355, 1990.

[14] R. N. Dave, "Characterization and Detection of Noise in Clustering”, *Pattern Recognition Letters*, V. 12, pp. 657-664, 1991.

[15] L. Davis, “Adapting operator probabilities in genetic algorithms”, *Proc. Third Int. Genetic Algorithms Conf.*, pp. 61-69, 1989.

[16] L. Davis (ed.), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, N.Y., 1991.

[17] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, N.Y., 1973.

[18] J.E. Dennis, Jr. and D.J. Woods, “Optimization on microcomputers: The Nelder-Mead simplex algorithm”, in *New Computing Environments: Microcomputers in Large-Scale Computing.* A Wouk, Ed. Philadelphia: SIAM, pp. 116-122, 1987.

[19] J.E. Dennis, Jr., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Englewood Cliffs, NJ: Prentice-Hall, 1983.

[20] L.J. Eshelman and J.D. Schaffer, “Real-Coded Genetic Algorithms and Interval-Schemata”, in *Foundations of Genetic Algorithms-2*, Ed. L.D. Whitley, pp. 187-202, 1993.

[21] T.C. Fogarty, “Varying the probability of mutation and crossover in genetic algorithms”, *Proc. Third Int. Conf. Genetic Algorithms*, pp. 104-109, 1989.

[22] D.B. Fogel and A. Ghozeil, *A Note on Representations ad Variation Operators*, IEEE Transactions on Evolutionary Computation, V.1, No. 2, pp. 159-161, July, 1997.

[23] A. Ghozeil and D.B. Fogel, “Discovering Patterns in Spatial Data Using Evolutionary Programming”, Genetic Programming 1996: Proceedings of the first annual conference, J.R. Koza, D.E. Goldberg, D.B. Fogel, R.L. Riolo (eds.), MIT Press, Cambridge, MA., pp. 512-520.

[24] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[25] D.E. Goldberg and K. Deb, A Comparative Analysis of Selection Schemes used in GA’s, in *Foundations of Genetic Algorithms*, Ed. G.J.E. Rawlins, pp. 69-73, Morgan Kaufmann, 1991.

[26] L.O. Hall, J.C. Bezdek, S. Boggavarapu and A. Bensaid, “ Genetic Fuzzy Clustering”, *NAFIPS’94*, San Antonio, TX. pp. 411-415, 1994.

[27] R.J. Hathaway and J.C. Bezdek, “Optimization of Clustering Criteria by Reformulation”, *IEEE Transactions Fuzzy Systems*, 3(2), pp. 241-245, 1995.

[28] Ismail and Camel, Hybrid Search for Clustering in Hard C-Means, Pattern Recognition Journal, V. 22, No. 1.

[29] R.A. Johnson and D.W. Wichern, *Applied Multivariate Statistical Analysis, 3rd edition*, Prentice Hall, Englewood Cliffs, N.J., 1992.

[30] F. Klawonn, “Fuzzy Clustering with Evolutionary Algorithms”, *Proc. Seventh IFSA World Congress*, V. 2, pp. 312-323, 1997.

[31] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed., Springer-Verlag, N.Y., 1996.

[32] B. Ozyurt and L.O. Hall, “Scaling Genetically Guided Fuzzy Clustering”, *Proc. ISUMA-NAFIPS’95*, pp. 328-332, 1995.

[33] J.D. Schaffer, R.A. Caruna, L.J. Eshelman, and R. Das, “A study of control parameters affecting online performance of genetic algorithms for function optimization”, In Schaffer (ed.), *Proc. third Intl. Conf. on Genetic Algorithms*, pp. 51-60, 1989.

[34] P. Scheunders, A Genetic c-means Clustering Algorithm Applied to Color Image Quantization, *Pattern Recognition*, V. 30, No. 6, pp. 859-866, 1997.

[35] P. Scheunders, Personal Communication about Clustering the Lena Image, June, July 1997.

[36] M. Srinivas and L.M. Patnaik, Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms, *IEEE Transactions on Systems, Man and Cybernetics*, V. 24, No. 4, pp. 656-667, 1994.

[37] M. Sugeno and T. Yasukawa, "A Fuzzy Logic Based Approach to Qualitative Modeling", *IEEE Transactions on Fuzzy Systems*, V. 1, No. 1, pp. 7-31, 1993.

[38] D. Whitley and D. Starkweather, "Genitor-II: A Distributed genetic algorithm", *J. Expt. Theoretical Artificial Intelligence*, V. 2, pp. 189-214, 1990.

[39] D.H. Wolpert and W.G. Macready, *No Free Lunch Theorems for Optimization*, IEEE Transactions on Evolutionary Computation, V. 1, No. 1, pp. 67-81, April, 1997.