In Proc. 5th European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, 1997, 1683–1688

SELECTION OF CLUSTER PROTOTYPES FROM DATA BY A GENETIC ALGORITHM

Ludmila I. Kuncheva[*], James C. Bezdek[†]
Department of Computer Science, University of West Florida
Pensacola, FL 32514, USA
e-mails: lucy@bgcict.acad.bg, jbezdek@dcsuwf.dcsnod.uwf.edu

Abstract

We focus on the *selection* of cluster prototypes for hard c-partitioning of unlabeled data. The benefit of selection is twofold: (1) The cluster prototypes are physically plausible data items that can be submitted to an expert or end-user for inspection or verification. (2) The search set for selection is significantly smaller than that for partitions. Combinatorial-type search is infeasible for extracted prototypes, but is possible when they are selected. We use a *genetic algorithm* (GA) to select prototypes with respect to the classical within-group square distance criterion J_1. Results on two data sets are used to illustrate the expediency of the GA compared with hard c-means and random selection.

1 Introduction

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a set of unlabeled feature vectors that represent n objects, $\mathbf{x}_i \in \Re^d$. Clustering seeks a partition of X into c "homogeneous" groups (clusters) $C = \{C_1, \ldots, C_c\}$, where the number c ($2 \leq c < n$) is either prespecified or subject to evaluation. Let $U = [u_{ij}]$ be a $c \times n$ matrix with element u_{ij} denoting the membership of $\mathbf{x}_j$ in C_i. U is an equivalent characterization of C when (i) $u_{ij} \in [0, 1]$, $\forall i, j$; and (ii) $\sum_{i=1}^{c} u_{ij} = 1$, $\forall j$. We consider the problem of finding a partition U and a set of point cluster prototypes $V = \{\mathbf{v}_1, \ldots, \mathbf{v}_c\} \subset \Re^d$ for X, minimizing the criterion function

$$J_m(U, V) = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^m \, ||\mathbf{x}_j - \mathbf{v}_i||^2, \tag{1}$$

where $||.||$ is any norm on $\Re^d$, and m is an user-specified parameter, $m \geq 1$.

When searching for an optimal partition there are two sets of parameters involved. For inner product norms, expressions for $U = \phi(V)$ and $V = \psi(U)$ are derived through the first-order necessary conditions on (U, V). The correctness of algorithms based on optimizing a reformulated version of (1) for either U or V is guaranteed by the reformulation theorem in Hathaway and Bezdek [12]. Picard iteration between successive estimates of U and V is called *alternating optimization* (AO) and is the basis of the "hill-climbing" (local search) clustering algorithms like *Hard c-means* (HCM) and *Fuzzy c-means* (FCM). These algorithms can be trapped in undesirable local extrema when initialized poorly. The usual approach to avoiding this is to run the procedure from different initializations, hoping that some runs will lead to the global minimum of J_m.

Random-search approaches are a good alternative to hill-climbing techniques because they can avoid local extrema. When the search space is discrete (e.g., the set of all possible hard partitions U of X), algorithms of this type appear to be an expedient option. In this group are the clustering methods based on simulated annealing [18] and tabu-search [1]. Still, *genetic algorithms* (GA) are the most appealing choice from the random-search group. GA based clustering differs by the encoding scheme, genetic operators, reproduction strategy, etc. Some recent publications in this group are [2, 6, 7, 9, 17, 19, 15, 8]. In all of these studies, either U or V is evolved, the other being derived by the expression $U = \phi(V)$ or $V = \psi(U)$.

[*]On leave from CLBME, Bulgarian Academy of Sciences, Research supported by the NRC COBASE program.
[†]Research supported by ONR Grant # N 00014-96-1-0642

In this paper we propose a GA for optimization of J_1 by selection of prototypes $P \subset X$. From P we calculate U so that $J_1(U,V) = J_1(U(P), \psi(U(P))$. We show that the search set for P is much smaller than those used in the previous studies cited above, and the results are better than using HCM-AO to solve (1). Section 2 describes our GA. Results with two data sets are presented in Section 3.

2 GA for selection of cluster prototypes from X

We consider a hard c-partition of X, i.e.

$$u_{ij} = \begin{cases} 1, & \mathbf{x}_j \in C_i \\ 0, & \text{otherwise} \end{cases}; \quad \sum_{i=1}^{c} u_{ij} = 1, \;\; j = 1, \ldots, n; \;\; \sum_{j=1}^{n} u_{ij} > 0, \;\; i = 1, \ldots, c. \tag{2}$$

For the set of all hard c-partitions of X we use the notation M_{hcn}. We use the classical within-group sum of squared errors [10] (or HCM criterion)

$$J(U) = \frac{1}{2} \sum_{i=1}^{c} \frac{1}{n_i} \sum_{j=1}^{n} \sum_{k=1}^{n} u_{ij} \; u_{ik} \;\; ||\mathbf{x}_j - \mathbf{x}_k||^2. \tag{3}$$

where n_i is the number of objects in cluster C_i. This criterion is equivalent to J_1 if the cluster centers $\{\mathbf{v}_i\}$ are calculated as the sample means (the function $\psi(.)$ fixed by the first-order necessary conditions for extrema of J_1),

$$\mathbf{v}_i = \frac{\sum_{j=1}^{n} \; u_{ij} \; \mathbf{x}_j}{\sum_{j=1}^{n} \; u_{ij}} \tag{4}$$

Instead of using alternating optimization, we minimize directly J_1 in the form (3) by a global search procedure. We search for prototypes $P = \{\mathbf{p}_1, \ldots \mathbf{p}_c\}$ by *selecting* them from the data, i.e., $P \subset X$. The partition U is derived from $\{\mathbf{p}_i\}$ with the nearest prototype rule, i.e. $\mathbf{x}_j$ is assigned to the cluster C_i if

$$||\mathbf{x}_j - \mathbf{p}_i|| = \min_{k=1,\ldots,c} ||\mathbf{x}_j - \mathbf{p}_k||, \;\; \text{(ties resolved arbitrarily)}. \tag{5}$$

Note that each P defines a c-partition U *and* a set of geometrical centers V calculated as at (4). Thus, the necessary conditions connecting U and V are not abandoned but are just "hidden" in the form of the criterion we use.

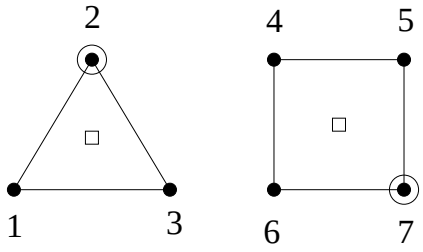

Figure 1: An example of constituting $(U(P), \psi(U(P))$ from P

Figure 1 shows seven 2-D points denoted $X = \{1, \ldots, 7\}$, partitioned into $\{C_1, C_2\}$ with respect to prototypes $P = \{2, 7\}$. The cluster centers V are depicted by '□'. The partition U shown by the links is optimal with respect to J_1. We emphasize that every pair of points $\{a, b\}$, where $a \in \{1, 2, 3\}$ and $b \in \{4, 5, 6, 7\}$ is an optimal choice for P because it leads to this same optimal partition.

To distinguish P from V we will call P *selected* point prototypes, and V *extracted* point prototypes. What do we expect to gain by selecting P?

• *Interpretability.* Selected cluster prototypes are physically interpretable while extracted centers cannot be. In some problems, especially in medical diagnostics, it is essential to retrieve the prototypes and to submit them to a domain-expert (or end-user) for verification or inspection. In this case physical feasibility is crucial. An example: suppose the ith feature is binary, e.g. "sex", coded by 0 and 1. An extracting algorithm can easily converge to a prototype whose ith value is, e.g., 0.7, which can hardly be regarded as physically plausible. When the dimensionality d is high, the construction of plausible prototypes by extraction becomes more unlikely because complex dependencies between features will be lost during the extraction process.

• *Search efficiency.* The search set for selecting c prototypes from n data points is much smaller than the set of all c-partitions of X ($|M_{hcn}|$). The size of the selection set is the number of combinations of of n elements taken c at a time:

$$S_s(n,c) = \left(\begin{array}{c} n \\ c \end{array} \right),$$

while for extraction, the number of feasible partitions is [13]

$$S_e(n,c) = |M_{hcn}| = \frac{1}{c!} \sum_{i=1}^{c} (-1)^{(c-i)} \left(\begin{array}{c} c \\ i \end{array} \right) (i)^n.$$

For example, with $c = 2$, the number of possible sets P from X in Figure 1 is 21 while for V it is 63. For $c = 3$, these numbers are 35 and 301, respectively. This demonstrates the combinatorial explosion of extracting prototypes even in very small problems. Encoding the problem in terms of a genetic model is the most important step in the GA approach. Selection of a subset of a given set is perhaps one of the problems that is most amenable to the GA approach because the natural encoding is to make every chromosome a binary string of length n [14]. We represent $P \subset X$ by a chromosome whose i-th bit is one if $\mathbf{x}_i \in P$, and zero, otherwise. The fitness function to be minimized is

$$\mathcal{F}(P) = J_1\big(U(P), \psi(U(P))\big) + \alpha \left(|P| - c\right)^2. \tag{6}$$

The second (penalty) term is weighted by the coefficient $\alpha > 0$ to force the algorithm to terminate at the desired number of clusters. Briefly, our GA goes through the the following steps:

1. *Initialization.* A set of N_{pop} randomly generated chromosomes is the initial "population set" $\Pi = \{P_1, \ldots, P_{N_{pop}}\}$. Each bit in a chromosome takes the value 1 with a prespecified probability P_{ini}. GA parameters are initialized: maximal number of generations M_{gen}; mutation probability P_m; the weighting coefficient α for the penalty term of (6). The chromosomes in Π are then evaluated by the fitness function.
2. *Forming the mating set* $\mathbf{M}$. In the current implementation $\mathbf{M}$ coincides with Π.
3. *Crossover.* Parent couples are randomly selected from $\mathbf{M}$. Every couple produces two offspring chromosomes by exchanging parts of their codes. Here we adopted uniform crossover, as recommended in [3, 4, 17, 19]. The parent chromosomes swap their i-th bits with a certain probability (we set it to 0.5 here), and i goes from 1 to n. The offspring chromosomes constitute the set $\mathbf{O}$.
4. *Mutation.* Each bit of each offspring chromosome alternates (mutates) with a predefined probability (mutation rate P_m). All elements of $\mathbf{O}$ are then evaluated by the fitness function.
5. *Recombination.* Π and $\mathbf{O}$ are pooled and the best N_{pop} individuals survive, i.e. they stand as the new Π (elitist strategy).
6. Steps (2) to (5) are executed M_{gen} times.

Experiments with two data sets are presented in the next section. The parameters of the GA were set up ad hoc and were not tuned to make the search more efficient because we found the results satisfactory.The parameters were:

- N_{pop} was either 10 or 20.
- P_{ini} varied between 0.01 and 0.1 depending on n and the desired number of clusters c. For example, with $n = 303$, and $c = 3$, we used the value 0.01, which initializes Π with approximately 3 clusters in a chromosome.
- M_{gen} was 500 in all experiments.
- P_m was 0.015.

- α was set to a specific value for each GA run. This value was chosen so that the "punishment" term was large enough to force the GA to terminate at c prototypes.

3 Experimental results

Comparison of clustering algorithms lacks a rigorous methodology because one mathematical model cannot incorporate various approaches [13]. Thus, the usual means for comparing various methods is via numerical examples with common data sets. Our experiments show that the proposed technique competes well with HCM and *random search* (RS). The RS encoding is the same as for the GA with these differences: no evolutionary operators are applied; and all the chromosomes are generated at random with exactly c 1's each (therefore $|P| = c$, so (6) reduces to $J_1(U(P), \psi(U(P)))$.

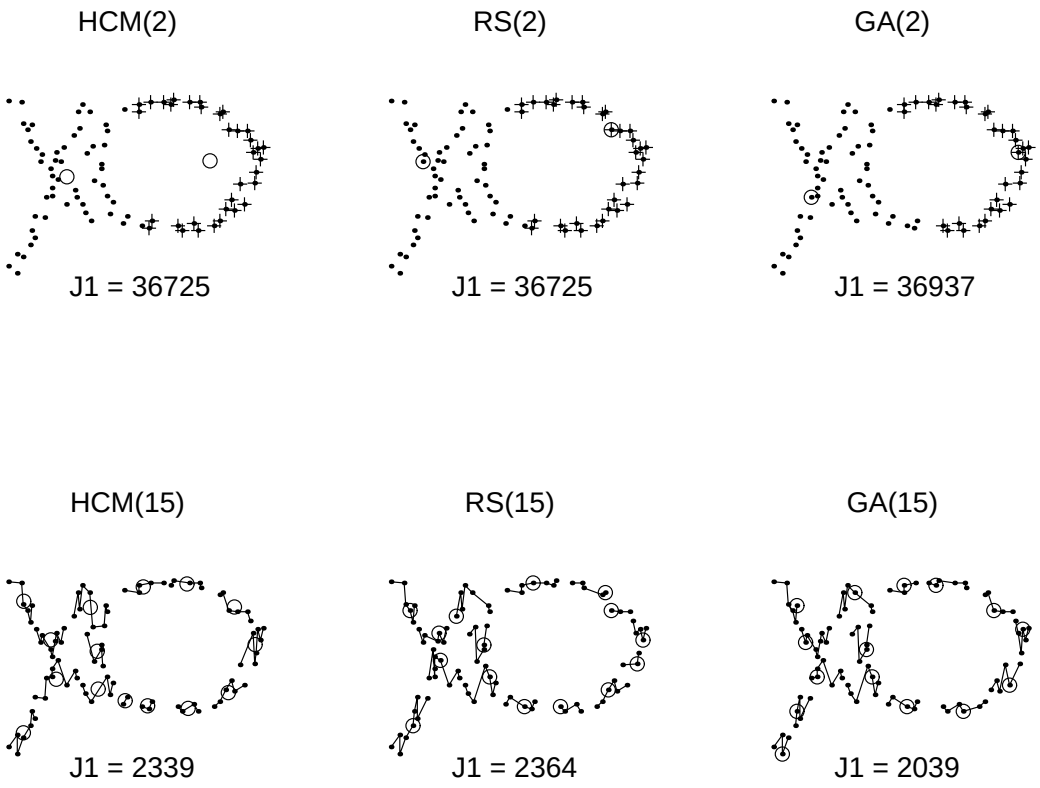

Figure 2: Clustering of Goldfish data

For each experiment HCM was run 100 times. The number of evaluations of the criterion function RS was either 5010 or 10020. Each of the GA results below is from a single run. We choose the optimal P as the one giving the *lowest* value of J_1 reached by the respective techniques.

First, the small artificial 2D data set "Goldfish" from [5] (p. 200) was used to illustrate the performance of the three algorithms in terms of J_1 values. The results for the three competing techniques are presented in Figure 2 for $c = 2$ and 15. Points assigned to the same cluster are joined (or marked for c=2), and the cluster prototypes are circles (extracted as V) or circled (selected as P). With 15 clusters, the J_1 value reached by the GA is the best one.

Finally, a problem from cardiology was considered using the benchmark data set called "heartc" from the database PROBEN1, ftp address: `ftp://ftp.ira.uka.de/pub/neuron/proben1.tar.gz`. A detailed description of the database and of this set in particular can be found in the Technical Report by Prechelt[16]. The data set has been supplied by Dr. Robert Detrano, V.A. Medical Center, Long Beach and Cleveland Clinic Foundation. The problem is to estimate whether or not a patient has ischemic heart disease, i.e.

whether one or more of the four coronary vessels is narrowed by more than 50 %. For clustering we pooled all the patient records, disregarding the class labels.

The data consists of 303 patient records with mixed variables: continuous-valued, categorical, and binary, representing a patient's history, laboratory test results, clinical observations, measurements, etc. Each patient record is encoded as a 20-dimensional vector in $[0,1]^{20}$ and represents 11 initial features (# 0 .. # 10 from the original data set). The augmentation to 20 is needed to encode the categorical variables. Features # 11 and # 12 of the original data were excluded from consideration because of missing values.

We used two sets as X: the entire set of 303 records, and the first 100 records. The minimal values of J_1 are plotted versus c in Figure 3.

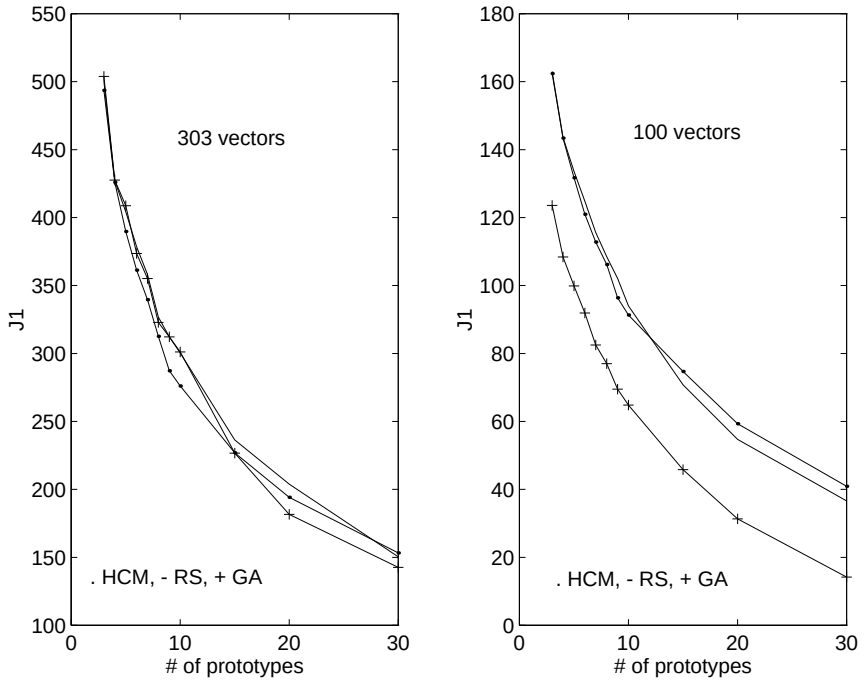

Figure 3: Results with "heartc" data

Clustering of the 303-element X using HCM, RS, and GA produced similar results. With the 100-element set, however, GA clearly outperformed the other techniques.

4 Analysis and conclusions

We use a GA for optimizing criterion J_1 by *selecting* cluster prototypes P from X instead of successive iterations between U and V. By doing this we reduce the search set, keep the prototypes P physically plausible, and as the experiments show, even reach lower values of J_1.

In general, with increasing n HCM is believed to reach the global minimum of J_1 or at least a local solution that is not far from the global one [11]. Moreover, for large n the combinatorial search set becomes enormously large. This suggests that selection of prototypes can be more effective than HCM-AO up to a certain size data set. This will also depend on the feature space dimensionality d, and on how complex the cluster disposition is. The results with the "heartc" data show that with 303 cases all the techniques perform alike. GA search appeared significantly more effective than both HCM and RS with the set size restricted to 100. This hints that there might be be a "critical" cardinality of X up to which GA may outperform the other techniques.

Hill-climbing procedures like HCM-AO do more effective local search than random-based algorithms. Therefore an appealing idea is to combine both in one scheme, e.g., by evolving the initial conditions for HCM by a GA and completing the solution by HCM. Finally, the GA presented here is not specific to $J_1(U,V)$. The same GA procedure can be used with any clustering criterion where prototypes are points.

The only component that needs to be changed is the first term of the fitness function at (6).

References

[1] K.S. Al-Sultan, A tabu-search approach to the clustering problem, *Pattern Recognition* **28**, 1443-1451 (1995).

[2] K.S. Al-Sultan, M.M. Khan, Computational experience on four algorithms for the hard clustering problem, *Pattern Recognition Letters*, **17**, 295-308 (1996).

[3] D. Beasley, D.R. Bull, and R.R. Martin, An overview of genetic algorithms: Part1, fundamentals, *University Computing* **15**, 58-69 (1993).

[4] D. Beasley, D.R. Bull, and R.R. Martin, An overview of genetic algorithms: Part 2, research topics, *University Computing* **15**, 170-181 (1993).

[5] J.C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, NY (1981).

[6] J.C. Bezdek, S. Boggavarapu, L.O. Hall, and A. Bensaid, Genetic algorithm guided clustering, *Proc. Ist IEEE Conference on Evolutionary Computing*, Orlando, FL, 34-39 (1994).

[7] J.C. Bezdek and R.J. Hathaway, Optimization of fuzzy clustering criteria using genetic algorithms, *Proc. Ist IEEE Conference on Evolutionary Computing*, Orlando, FL, 589-594 (1994).

[8] K. Blekas and A Stafylopatis, real-coded genetic optimization of fuzzy clustering, *Proc. EUFIT'96*, Aachen, Germany, 461-465 (1996).

[9] B.P. Buckles, F.E. Petry, D. Prabhu, R. George, and R. Srikanth, Fuzzy clustering with genetic search, *Proc. Ist IEEE Conference on Evolutionary Computing*, Orlando, FL, 46-50 (1994).

[10] R.O. Duda, P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons (1973).

[11] J.A. Hartigan, *Clustering Algorithms*, John Wiley & Sons (1975).

[12] R.J. Hathaway, J.C. Bezdek, Optimization of clustering criteria by reformulation, *IEEE Transactions on Fuzzy Systems*, **3**, (2), 241-245 (1995).

[13] A.K. Jain and R.C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[14] L.I. Kuncheva, Editing for the k-nearest neighbors rule by a genetic algorithm, *Pattern Recognition Letters* **16**, 809-814 (1995).

[15] W.K. Ng, S. Choi, and C.V. Ravishankar, An evolutionary approach to vector quantizer design, *Proc. IEEE International Conference on Evolutionary Computation*, Perth, Australia, 406-411 (1995).

[16] L. Prechelt, PROBEN1 - A set of neural network benchmark problems and benchmarking rules, *Technical Report* # 21/94 (1994).

[17] C.M. Schulte, Genetic algorithms for prototype-based fuzzy clustering, *Proc. EUFIT'94*, Aachen, Germany, 913-921 (1994).

[18] S.Z. Selim and K. Al-sultan, A simulated annealing algorithm for the clustering problem, *Pattern Recognition* **24**, 1003-1008 (1991).

[19] T. Van Le, Evolutionary fuzzy clustering, *Proc. IEEE International Conference on Evolutionary Computation*, Perth, Australia, 753-758 (1995).