Clustering by a genetic algorithm with biased mutation operator

Benjamin Auffarth

***Abstract*—In this paper we propose a genetic algorithm that partitions data into a given number of clusters. The algorithm can use any cluster validity function as fitness function. Cluster validity is used as a criterion for cross-over operations. The cluster assignment for each point is accompanied by a temperature and points with low confidence are preferentially mutated. We present results applying this genetic algorithm to several UCI machine learning data sets and using several objective cluster validity functions for optimization. It is shown that given an appropriate criterion function, the algorithm is able to converge on good cluster partitions within few generations. Our main contributions are: 1. to present a genetic algorithm that is fast and able to converge on meaningful clusters for real-world data sets, 2. to define and compare several cluster validity criteria.**

I. Introduction

Clustering is a fundamental problem in pattern recognition. It refers to visualization techniques that group data into subsets (clusters), according to a distance measure. Cluster analysis (CA) partitions points of a data set into groups, so that data points within a group are more similar to each other than to points in different groups.

The use of clustering technique constitutes often a first step in a data mining process to reveal natural structures and identify patterns in the data. Clustering is applied in many disciplines and plays an important role in a broad range of applications that include data mining. Applications of clustering usually deal with large datasets and data with many attributes, where simplification or concise summaries can provide an idea of the structure of the data.

Formally, given a data set of m dimensions and n points, $D \in \mathbb{R}^{n,m} = \{d_1, \ldots, d_n\}$, clustering is the process of dividing the points up into k groups (clusters) based on a distance (or similarity) measure, $\mathrm{dist}(d_i, d_j)$. Often, the number of clusters k is given as a pre-defined parameter to the optimization problem. For crisp solutions, a membership matrix $U \in \{1, k\}^n$ defines the attributed cluster membership for each point. For fuzzy solutions, the membership for each point would be defined in terms of probability for each cluster, $U \in [0, 1]^{n,k}$.

Many algorithms have been developed to tackle clustering problems in a variety of application domains, including the hierarchical agglomerative clustering algorithm [1], k–means [2], and self–organizing maps [3]. The most popular algorithms are probably the fuzzy c-means [4] and the k-means algorithms. All of these clustering algorithms rely on Euclidean distances from cluster centroids as criterion function. Therefore they are limited to detecting spherical clusters and do not work well with non–Gaussian data [5, 6, c.f.].

In the case of k-means, the criterion function is as follows:

$$\underset{S}{\operatorname{argmin}} \sum_{i=1}^{k} \sum_{x_j \in S_i} \| x_j - \mu_i \|^2 \tag{1}$$

where $S_1, S_2, \ldots, S_k$ are partitioned point sets and μ_i is the mean of points in S_i.

A. Distance Measures

Other distance measures have been proposed, however, have not found a wide application for reasons of computational efficiency or robustness. For example it has been proposed to use the Mahalanobis distance for clustering, e.g. Yih et˜al. [7] and Liu et˜al. [8] presented a fuzzy c-means based on Mahalanobis distances.

Other solutions to produce nonlinear separating hypersurfaces between clusters involve kernel density estimation, non-parametric regression, and spectral methods [9]. For example, Gaffney and Smyth [10] proposed a clustering method for univariate data based on probability estimates from a mixture of non-parametric regression models and Wang [11] presented a small simulation study with clustering using non–parametric kernel regression. These solutions however can be sensitive to the choice of bandwidth parameters and lack robustness for a broad set of problems. In spectral clustering [12, 13], data are preprocessed to extract eigenvalues of the Laplacian of the distance matrix before clustering is performed. These methods find, however, little application to real-world data because of their high computational costs.

Information entropy is a measure of the uncertainty associated with a distribution. The Maximum Entropy Principle [14] and Renyi's entropy [15] have been proposed as information-theoretic distance measures between centroids and data points. Butte and Kohane [16] computed entropy between gene pairs and used thresholding to build clusters. A problem underlying the computation of entropy measures is the estimation of probability density, especially with data that have few data points compared to number of attributes (dimensions).

Most clustering algorithms rely on distances from centroids because of the faster computation as compared to measures that take into account complete linkage of points.

B. Genetic Algorithms for Clustering

Evolutionary algorithms [17] are optimization algorithms that use mechanisms inspired by biological evolution such as inheritance, mutation, selection, and crossover. At each iteration the fitness of a population of candidate solutions is computed and this determines their selection. A genetic

Institute for Bioengineering of Catalonia and with the Department of Electronical Engineering, University of Barcelona, Barcelona Science Park, C/Baldiri Reixac 4-6 (torre I), 08028 BCN, Spain Contact: phone: +34 9340 31118; email: bauffarth@el.ub.es.

algorithm (GA), proposed by Holland [18], is a search heuristic, mimicking the process of natural evolution, used for optimization and search problems. Genetic algorithms belong to the class of evolutionary algorithms in that they use operations from evolutionary algorithms and extend evolutionary algorithms by encoding candidate solutions as strings, called chromosomes). GAs have the following phases:

- Initialization: Generate an initial population of K candidates and compute fitness.
- Selection: For each generation, select μ_K candidates based on fitness to serve as parents.
- Crossover: Pair parents randomly and perform crossover to generate offspring.
- Mutation: Mutate offspring.
- Replacement: Replace parents by offspring and start over with selection.

Genetic algorithms have been applied to clustering problems before; see Sheikh et˜al. [19] for a survey. Genetic algorithms have been used to overcome local minima, however the associated computational costs have been prohibitive to broad application.

Scheunders [20] demonstrated that a hybrid of k–means and genetic algorithm depends less on initial conditions. Krishna et˜al. [21] used the k–means clustering algorithm as a cross-over operation. Maulik and Bandyopadhyay [22] applied genetic algorithms to find cluster centers. This was improved on by Lu et˜al. [23], who removed overhead from the algorithm. Maulik and Bandyopadhyay [22] applied genetic algorithms for selection of cluster centers and evaluated fitness by Euclidean distances of points to these centers. Laszlo and Mukherjee [24] also based their algorithm on cluster centers.

We start off by presenting our genetic algorithm for clustering, then we show several criterion functions[1] that we tried out, and then we present results by clustering real–world data sets from the UCI machine learning repository. We conclude with a discussion of applicability of our methods, limitations, and suggest possible avenues for future research.

II. Methods

A. A Genetic Clustering Algorithm with Local Annealing

In the proposed algorithm, the searching capability of genetic algorithms is exploited to find appropriate cluster partitions in the feature space so that a fitness metric of the resulting clusters is optimized. The advantage of this technique is the generality of the admitted fitness function. We made the choice to not rely on the concept of a cluster centroid (prototype). This was meant to allow the clustering to find even very complex clusters, as opposed to Gaussian or hyper–spheroidal shapes preferred by more conventional algorithms (see introduction).

Our genetic algorithm has the following characteristics:

- as fitness function any internal cluster validity measure can be used,
- it incorporates a measure of intra–cluster distance in the cross–over operation (as opposed to just fitness),
- it uses temperature to find appropriate genes (points) for mutation,
- it adjusts the mutation rate self–adaptively in order to keep variance of fitness between iterations within a certain range,
- it keeps a small ($M = 20$) population of candidate solutions and employs elitist recombination.

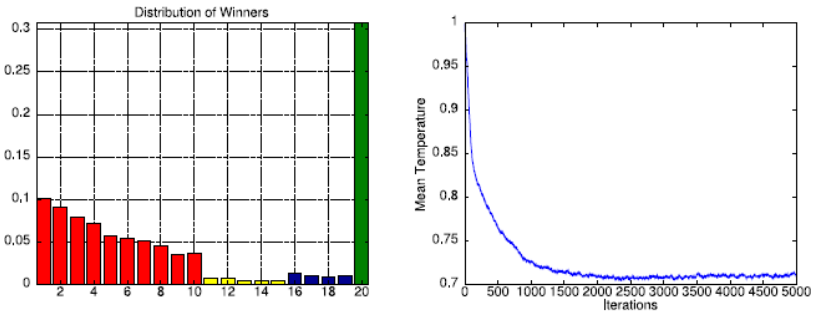
Algorithm 1 shows a pseudo–code to explain the algorithm.

Input: D, k
Output: U, M
// Parameter definitions:
maxiteration $\leftarrow 5000$, $K \leftarrow 20$, minchange $\leftarrow 0.0001$, ;
$\alpha \leftarrow 0.1$, $i \leftarrow 0$, ;
// The mutation matrix of size $n \times k$ is set to equal values and normalized to row sum 1:
$M \leftarrow 1^{n,k} \times 1/k$;
// Algorithm starts:
$U \leftarrow \text{newgeneration}(M, K, \alpha)$;
while $i <$ maxiteration *and not* stop **do**
  $(F, intra)^K \leftarrow \text{fitness}(D, U)$;
  if $i > 0$ **then**
    $\text{learningrateupdate}(F, F_{\text{old}}, \alpha)$
  else
  end
  // Select the fittest individual:
  $w_1 \leftarrow \text{argmin}_i F^i$;
  // Select the individual with the best single cluster:
  $(w_2, \text{wc}) \leftarrow \text{argmin}_{i,j} \text{intra}^{i,j}$;
  // Select the individual that is best for a random cluster:
  $\text{rc} \leftarrow \text{ceil}(\text{rand} \times k)$;
  $w_3 \leftarrow \text{argmin}_i \text{intra}^{i,\text{rc}}$;
  $U1 \leftarrow U^{w_1}$;
  $U2 \leftarrow \text{crossover}(U1, U^{w_2}, \text{wc})$;
  $U3 \leftarrow \text{crossover}(U1, U^{w_3}, \text{rc})$;
  $M \leftarrow \text{mutationbiasupdate}(F, intra, M)$;
  $U \leftarrow \text{mutate}(U1, K/2, M, \alpha) \cup \text{mutate}(U2, K/4, M, \alpha) \cup \text{mutate}(U3, K/4, M, \alpha)$;
  $i \leftarrow i + 1$;
end

Algorithm 1: The genetic algorithm with biased mutation and local cross–over (simplified description). Input parameters are data D and number of clusters k. Parameters within learningrateupdata() are $\alpha_{\min} \leftarrow 0.01$, $\alpha_{\max} \leftarrow 0.5$ and $\alpha_{\text{inc}} \leftarrow 0.05$. The stop criterion is explained in the text.

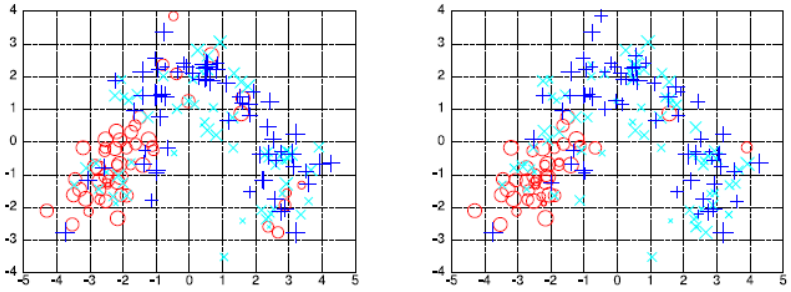
Given data D and k the desired number of clusters the

[1]Following common use of terminology in genetic algorithm literature, we use the terms criterion function and fitness function interchangeably throughout the article



(a) Distribution of Winners

(b) Mean Temperature



(c) Temperature after 1 Iteration

(d) Temperature after 40 Iterations

Fig.˜1: Graphs show characteristics of the genetic algorithm: **a** displays the (probability) distribution of winners from 1000 iterations. **b** shows how mean temperature goes down over many iterations. Subfigures **c** and **d** show PCA plots of the wine data set with dominant cluster assignment indicated in color and marker shape and local temperature indicated in marker size.

algorithm returns a cluster assignment for each point in a membership matrix $U \in \{1, k\}^n$.

The population consists of K individual candidate solutions, which are crisp cluster partitions in form of membership matrices $U^K = \{U_1^n, \ldots, U_K^n\}$, where n are the number of points in our dataset and $U_i^j \in [1, k]$. K is the number of individuals or chromosomes for each generation. Before the first iteration one random parent is initialized and K children are created from random permutations.

At each iteration the winner U1 was determined as the individual from U^K that obtained the highest fitness according to a given cluster validity function. This individual was then permuted to yield $K/2$ individuals. For the rest of $K/2$ individuals in the next generation we used the following procedure: for each cluster assignment, we evaluated the individual with the lowest intra-cluster distance (or inversely, the highest intra-cluster similarity). We took the individual with the best value for intra-cluster distance, cross-combined it with the overall winner, and permuted the result to yield $K/4$ individuals. We then randomly took one of the individuals with low intra-cluster distance to cross-combine with the overall winner and again permuted to obtain $K/4$ individuals.

For permutations, the matrix M was taken as a mutation bias for each point. M regulates mutation probability for each point in the dataset. At each iteration this matrix $M \in [0, 1]^{n,k}$ is updated according to success and failure of cluster assignment in the previous generation, giving each point its proportional share.

A high variability in M between entries for a particular point $\{M^{i,1}, \ldots, M^{i,k}\}$ indicates that cluster assignment is more certain, while a low variability means that different cluster assignments are equally successful. Formally we calculated the temperature of a vector m of M corresponding to a point, with k elements as

$$T = 1 - \frac{\sigma}{\sqrt{k}} \tag{2}$$

At the onset, point mutation biases are initialized to a matrix $1^{n,k} \times 1/k$, which means that for any point, mutations to any cluster assignment are assumed equally probable. Over time, M should converge so that the arguments of the maximum for each point reflect the cluster assignment. In this way, the adaption of M is a simulated annealing. The temperature of each point is indicative of the algorithms confidence in its assignment, similar to a fuzzy cluster membership matrix.

In fig.˜1a the distribution of winners in a sample trial with the wine data set over 1000 iterations can be seen with $K = 20$. 20 corresponds to the winner of the last generation which was kept over. The mutation rate was increased when the fitness function could not distinguish between candidates and decreased when the previous winner was best again.

Figure˜1b shows how mean temperature declines over 5000 iterations. In this case the cross–overs were not successful, so that temperature stagnates. Figures 1c and 1d show local temperatures in a PCA plot of the wine data set. Colors

and marker shape indicate the dominant cluster assignment according to the mutation matrix. The temperature is indicated by the size of the points. Figure˜1c shows temperature at the onset after random initialization of M. It can be seen that temperature is broadly distributed in the sense that it is not specific for certain points or clusters. In figure˜1d which shows temperature after 40 iterations it can be seen that cluster 3 (points in blue with plus signs) and cluster 2 (points in cyan with crosses) are strongly overlapping. This results in high temperatures for both clusters.

For the purpose of this bias mutation operator, mutations had to be kept small so that cluster assignment would not radically shift from one zone to another. Mutation rate $alpha$ is regulated between $\alpha_{\min}$ and $\alpha_{\max}$. If the previous winner comes up a second time as winner, this could mean that mutations were too strong and α is down-regulated. If, on the other hand, changes are so minuscule that the chromosomes cannot be distinguished by the fitness function (in our implementation this could be the case when the winner is 1), α is up-regulated. We took the same value, $\mathrm{alphaInc}$ for linear increments and decrements.

In each permutation, $a \times n$ points are pseudorandomly sampled using temperature as weight vector so that points with higher temperature are more likely to be taken. These sampled points are then randomly assigned to a cluster,

We concluded in pre–trials (not reported in this thesis) that the biased mutation operator gives a significant speed up for optimization. We used as stopping criterion (`stop`), a threshold `minchange` which is compared to changes to the mutation bias operator M. If the changes per point of M was not reached twice in a row this threshold, the algorithm was terminated. Otherwise the algorithm iterated until `maxiteration`.

B. Cluster Validity Measures

Assignment to clusters relies on a distance measure, in the case of genetic algorithms, the criterion function of the optimization is called the fitness function. In our algorithm, we can plug in many different cluster validation functions. In this subsection, we present some measures which we tested with our algorithm.

As a general guideline, these measures should favor minimal differences between points within clusters (intra–cluster distance) and maximal differences between points of different clusters (inter–cluster distance).

The distance measure can be any distance function or goodness-of-fit function. A function that measures validity of partitions based on the structure of data in the clusters are called internal cluster validity measures. An overview over some measures especially proposed for internal cluster validation can be found in the review by Pfitzner et˜al. [25]. They define desiderata for internal cluster validity functions, which include

- being able to work with different distributions,
- to be robust to outliers,
- being robust or invariant to scaling.

Internal validity measures that received special mention in their article include information-theoretic measures such as Lopez and Rajski's measures [26, 27], and several normalized mutual information measures [28, 29, 30, 31].

Other popular validity measures include Davies–Bouldin [32], Calinski–Harabasz [33], Hartigan [34], Krzanowski–Lai [35], and Silhouette [36].

C. Euclidean distance

The fuzzy c-means and k-means algorithms use the Euclidean distance from centroids as criterion function. The distance of a point x to centroid i, $c_i = \{c_{i,1}, \ldots, c_{i,m}\}$ (m *features* or *dimensions*) is defined as follows:

$$d_{\mathrm{Euclidean}}(c_i, x) = \sqrt{\sum_j^m (c_{i,j} - x_j)^2} \tag{3}$$

where c_i, the centroid, is adapted at each iteration to the center of gravity of cluster i. Points should be assigned to a cluster such that distances to its cluster centroids are be minimal, while distances to centroids of other clusters are maximal.

D. Global and local Mahalanobis distances

The Mahalanobis distance [37] is a distance measure capable of dealing with hyper–elipsoidal distances, as opposed to hyper–spherical distances (e.g. Euclidean). The point–distance from cluster centroids would be defined as:

$$d_{\mathrm{Mahalanobis}}(c_i, x) = \sqrt{(x - c_i)^T S^{-1} (x - c)} \tag{4}$$

where S is the covariance matrix usually corresponds to D, however there have been also experiments with taking cluster covariances. It has been shown experimentally that taking local and global results can make a big difference [38].

We implemented the Mahalanobis distance for our algorithms in two variants, with global and local (cluster–specific) covariances, which we will refer to as *global* and *local Mahalanobis*.

The computation of the inverse of the covariance matrix presented a problem with few data points, because it easily becomes singular or near singular. This was especially true for the local version, where the covariance is computed from points of each cluster. The inverse of the covariance matrix could take extreme values. To prevent this from happening, a restriction was implemented, so that clusters had to have at least the size $\frac{(k-1)}{k} * \frac{n}{k}$, where k is the number of clusters and n the number of points in the dataset. Smaller clusters were heavily penalized. This restriction was implemented for all used cluster validity measures in order to have them comparable.

E. SVD entropy

Alter et˜al. [39] proposed an entropy measure based on the distribution of eigenvalues. This entropy has found application in different areas including feature filtering [40]. However, we are not aware of any previous application to clustering. The main idea of the entropy criterion is to find

clusters with points that have a low entropy and, at the same time, clusters that have a high entropy when joined with other clusters.

Following the definition by Varshavsky et˜al. [40], formally, if s_j denotes the singular values of the matrix A, s_2 are the eigenvalues of the $n \times n$ matrix AA^t. The normalized relative values are given as:

$$V_j = \frac{s_j^2}{\sum_k s_k^2} \tag{5}$$

and the dataset entropy as:

$$H = -\frac{1}{\log n} \sum_{j=1}^{n} V_j \log V_j \tag{6}$$

This entropy takes values in the range $[0, 1]$. $H = 0$ stand for an ultra-ordered dataset that can be explained by a single eigenvector and $H = 1$ corresponds to a disordered matrix with a uniformly distributed spectrum. We used this as our intra–cluster distance.

As a measure of distance between two clusters, the inter–cluster distance, we used this SVD entropy:

$$H_d = H_{\text{all}} - (H_1 + H_2) \tag{7}$$

where H_1 and H_2 correspond to the SVD entropies of clusters C_1 and C_2 and H_{all} to the entropy of the combined cluster $C_1 \cup C_2$.

We combined intra- and inter-cluster validity thus obtained linearly and refer to this as the *SVD Entropy Cluster-Validity Index* which we use as a fitness function.

F. Cluster Validation

For validation of our clustering method we use an external validity measure that compares coincidence of clusters found with our method with correct cluster assignments. For this purpose we use the Jaccard index which measure similarity of partitions. It is based on the Rand index [41] which compares two hard partitions R and Q of some data.

$$\text{Rand} = \frac{a+d}{a+b+c+d} \text{where,} \tag{8}$$

- a denotes the number of point pairs belonging to same partition in R as well as in Q.
- b the number of point pairs belonging to the same cluster in R but to different in Q.
- c the number of point pairs belonging to different clusters in R but to same clusters in Q.
- d the number of point pairs belonging to different clusters in R and different clusters in Q.

The term d can cause problems by becoming big, biasing the index. The Jaccard coefficient˜[5] leaves out d with the motivation that point pairs which are neither in the same cluster in R nor in Q are insignificant for consistency between R and Q. Denoeud et˜al. [42] showed experimentally that the Jaccard index is approximately equally efficient with other measures based on the Rand index, while showing lower variance. Formally, the Jaccard index is defined as:

$$\text{Jaccard} = \frac{a}{a+b+c} \text{where,} \tag{9}$$

We use the Jaccard index to quantify the correctness of results from the genetic clustering algorithm by comparing the correct cluster assignment and the obtained clusters from the algorithm. A score of 1 would mean that all points were correctly assigned.

Data	FCM	GA-Eucl	GA-MaG	GA-MaL	GA-Ent
Breast	0.5219	0.4035	0.6564	**0.6750**	0.4008
Wine	0.4120	0.4160	0.4261	**0.5741**	0.3565
Ionosphere	0.4314	0.4176	0.4544	**0.5606**	0.4040
Iris	0.6959	**0.6997**	**0.6997**	0.5779	0.4677

TABLE˜I: **Comparison of clustering Results**: Clustering results of the fuzzy c–means algorithm and the genetic algorithm based on four fitness functions. Units express the coincidences between correct partitions and found partitions (Jaccard Index). *FCM* stands for fuzzy c–means, *GA* for the genetic algorithm. As for the fitness functions employed for optimization in the genetic algorithm, *Eucl* stands as Euclidean distance, *MaG* for the Mahalanobis distance with global covariance, *MaL* for the Mahalanobis distance with local (i.e. cluster) covariance, and *Ent* for SVD entropy. The best results for each dataset are typeset in bold.

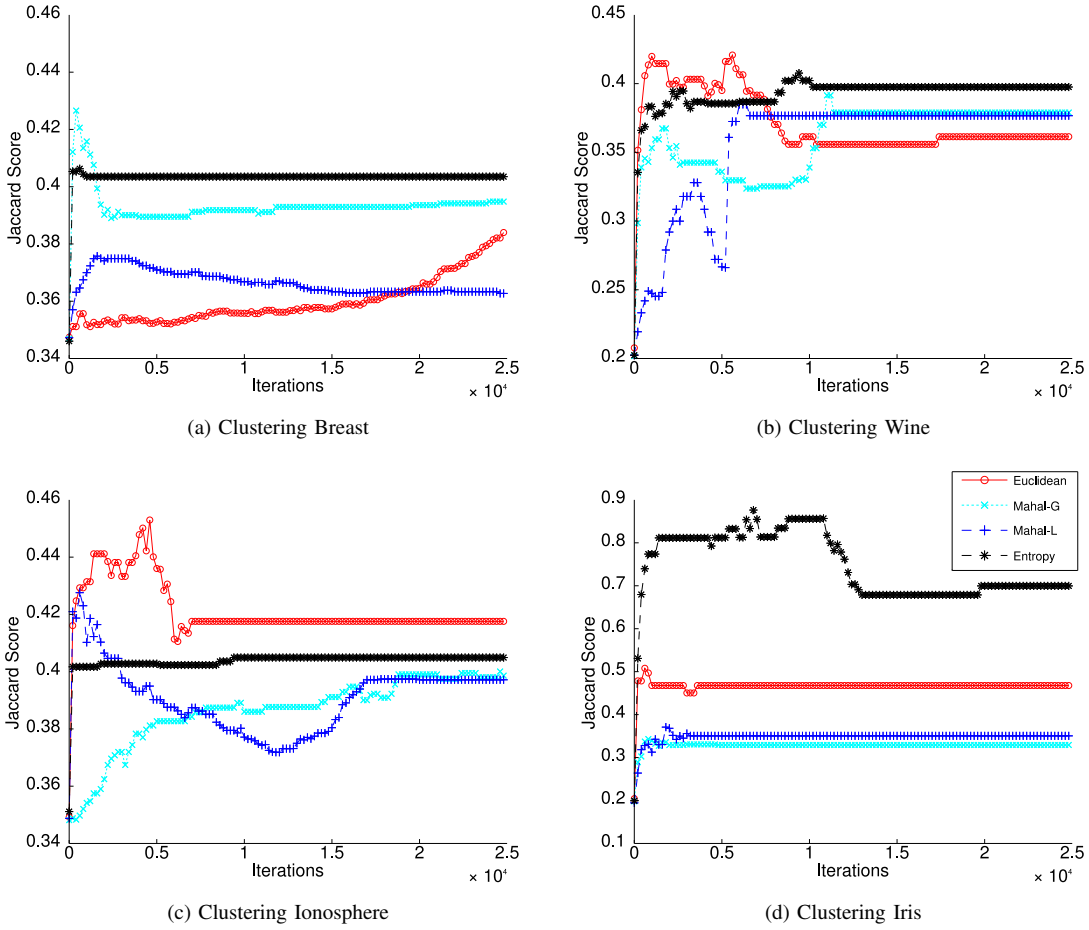
III. Results

We applied our algorithm to several data sets from the UCI machine repository [43]. These were the wine, iris, ionosphere, and breast data sets. We randomly initialized the candidates, set K to 20, and set the maximum iterations for our algorithm to 5000. The clustering of all data sets was completed within several minutes on an off-the-shelf desktop office computer

We compared our results to results from the fuzzy c-means algorithm, which next to k-means is probably the most popular algorithms for clustering. Fuzzy c-means [4] is an improved version of k–means. Particularly it is more robust to outliers and overlap than k-means (e.g.˜44).

Fig.˜2 shows how the genetic algorithm starting from random initialization optimizes according to a fitness function. The ordinate axis shows the goodness of a partition in terms of the Jaccard index. The abscissa shows the number of iterations of the genetic algorithm. K was set to 20. The parameter set values are shown in algorithm 1.

Table˜I compares clustering performance by fuzzy cmeans clustering (FCM) and the genetic algorithm (GA) based on four different fitness functions, Euclidean distance (Eucl), Global Mahalanobis (MaG), Local Mahalanobis (MaL), and SVD Entropy (Ent). Values displayed correspond to averages (medians) over 10 runs. The genetic algorithm was run for 5000 iterations.



(a) Clustering Breast

(b) Clustering Wine

(c) Clustering Ionosphere

(d) Clustering Iris

Fig.˜2: Optimization behavior of the genetic algorithm with different fitness functions

IV. Discussion

A genetic algorithm for clustering clustering has been has been proposed and tested here. We presented results of our algorithm using measures that relied on information theoretic measures on complete clusters (SVD entropy), two variants of Mahalanobis distance with global and local covariance, respectively, and the Euclidean distance. Results for real-life datasets have been presented and compared to partitions obtained by fuzzy c–means. The dataset comprises many different data distributions. Clusters have different overlap, different dimensionality, and number of points.

We think that the results show that some of the problems are very difficult, where popular off–the–shelf algorithms such as the fuzzy c-means do not achieve good results. We conjecture that the unsatisfactory results for fuzzy c–means reflect in part the fact that Euclidean distances from the centroids are not very sensitive and might even be inappropriate for some datasets. We think that our results underline the importance of finding a fitness function that is both sensitive to small changes in partitions and works for the particular dataset [c.f. 5].

It was shown that — given a fitness function which is appropriate for the data — the algorithm can converge to good solutions. We conclude that the results from clustering using our genetic algorithm were competitive with results from fuzzy c–means. Our good results reflect in part the use of proper fitness functions, the application of which was made possible by our genetic algorithm. We tested the algorithm with Euclidean distances, Mahalanobis distances, but we could have used other validity measures, as well. Results with the SVD Entropy measure were generally disappointing, performing worse than the fuzzy c–means algorithm in our tests. We think that both Mahalanobis distances (local and global versions) gave very promising results. The local version returned at least satisfactory results for all data sets, achieving the top result for Breast, while the global version performed well for Iris and Breast.

In practice, the algorithm often converged fast. We think that the self–adaptation of the mutation rate helped greatly to find solutions. We conjecture that this is because it helps to maintain a certain level of correlation of fitness between candidates [c.f. 45]. We think that another reason for this good convergence is the cross–over operation, where we applied a goodness value for each cluster. We are not aware of a previous application of intra–cluster validity to the

crossover operation in genetic algorithms.

We think that the self–adaptation of the mutation rate helped greatly to find solutions. We conjecture that this is because it helps to maintain a certain level of correlation of fitness [c.f. 45].

Preliminary experiments have been performed where this genetic algorithm was extended to run without fixing the parameter k in advance. The inter–cluster distance was used to merge or split clusters. Further, by extensions of the mutation routine new clusters could appear or old clusters could disappear.

We want to emphasize that results from our algorithms were achieved with a global set of parameters. We assume that parameter optimization could yield better results. The algorithm did not converge for some combinations of data sets and fitness functions which shows that there is still work to do and therefore we see total results as preliminary. There is still testing to be done on more datasets, however we think that results presented here are promising. Source code of scripts in MATLAB can be made available upon request.

Acknowledgment

The author is supported by a grant from the federal state government of Catalonia (formació de personal investigador, FI). He wants to thank the anonymous reviewers, who made very useful comments which helped to improve this article.

References

[1] J˜Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[2] J˜MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium*, 233(233):281–297, 1967.

[3] T˜Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982. ISSN 0340-1200.

[4] J.˜C Bezdek, R˜Ehrlich, and W˜Full. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984. ISSN 00983004.

[5] A.˜K Jain and R.˜C Dubes. *Algorithms for Clustering Data*, volume 355 of *Prentice Hall Advanced Reference Series*. Prentice Hall, 1988. ISBN 013022278X.

[6] S.˜J Roberts. Parametric and non-parametric unsupervised cluster analysis. *Pattern Recognition*, 30(2):261–272, February 1997. ISSN 00313203.

[7] J˜Yih, Y˜Lin, H˜Liu, and C˜Yih. FCM & FPCM Algorithm Based on Unsupervised Mahalanobis Distances with Better Initial Values and Separable Criterion. In *ACS'08 Proceedings of the 8th conference on Applied computer scince*, volume˜3, pages 9–18, 2008.

[8] H.-c Liu, J.-m Yih, D.-b Wu, and S.-w Liu. Fuzzy possibility c-mean clustering algorithms based on complete mahalanobis distances. In *2008 International Conference on Wavelet Analysis and Pattern Recognition*, pages 50–55. IEEE, August 2008. ISBN 978-1-4244-2238-8.

[9] M˜Filippone, F˜Camastra, F˜Masulli, and S˜Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, January 2008. ISSN 00313203.

[10] S˜Gaffney and P˜Smyth. Trajectory clustering with mixtures of regression models. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '99*, pages 63–72, New York, New York, USA, 1999. ACM Press. ISBN 1581131437.

[11] N˜Wang. Marginal nonparametric kernel regression accounting for within-subject correlation. *Biometrika*, 90(1):43–52, March 2003. ISSN 0006-3444.

[12] U˜Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, August 2007. ISSN 0960-3174.

[13] B˜Auffarth. Spectral Graph Clustering. Technical report, Universitat de Barcelona, 2007.

[14] G˜Beni. A least biased fuzzy clustering method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):954–960, 1994. ISSN 01628828.

[15] R˜Jenssen, K˜Hild, D˜Erdogmus, J˜Principe, and T˜Eltoft. Clustering using Renyi's entropy. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume˜1, pages 523–528. IEEE, 2003. ISBN 0-7803-7898-9.

[16] a.˜J Butte and I.˜S Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 418–29, January 2000. ISSN 1793-5091.

[17] D.˜B Fogel. *Evolutionary computation: toward a new philosophy of machine intelligence*. IEEE Press, 1995. ISBN 0780310381.

[18] J˜Holland. Genetic algorithms. *Scientific American*, 267 (July):66—-72, 1992.

[19] R.˜H Sheikh, M˜Raghuwanshi, and A.˜N Jaiswal. Genetic Algorithm Based Clustering: A Survey. *2008 First International Conference on Emerging Trends in Engineering and Technology*, 2(6):314–319, July 2008.

[20] P˜Scheunders. A genetic c-Means clustering algorithm applied to color image quantization. *Pattern Recognition*, 30(6):859–866, June 1997. ISSN 00313203.

[21] K˜Krishna, N˜Murty, and Others. Genetic K-means algorithm. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 29(3):433–439, 1999.

[22] U˜Maulik and S˜Bandyopadhyay. Genetic algorithm-based clustering technique. *Pattern recognition*, 33(9): 1455–1465, 2000.

[23] Y˜Lu, S˜Lu, F˜Fotouhi, Y˜Deng, and S.˜J Brown. FGKA : A Fast Genetic K-means Clustering Algorithm. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1–2, 2004.

[24] M˜Laszlo and S˜Mukherjee. A genetic algorithm that exchanges neighboring centers for k-means clustering. *Pattern Recognition Letters*, 28(16):2359–2366,

December 2007. ISSN 01678655.

[25] D˜Pfitzner, R˜Leibbrandt, and D˜Powers. Characterization and evaluation of similarity measures for pairs of clusterings. *Knowledge and Information Systems*, 19 (3):361–394, July 2009. ISSN 0219-1377.

[26] R.˜L.˜D Mántaras. A distance-based attribute selection measure for decision tree induction. *Machine Learning*, 6(1):81–92, 1991. ISSN 08856125.

[27] C˜Rajski. A metric space of discrete probability distributions. *Information and Control*, 4(4):371–377, December 1961. ISSN 00199958.

[28] F.˜M Malvestuto. Statistical Treatment of the Information Content of a Database. *Information Systems Journal*, 11(3):211–223, 1986.

[29] A˜Strehl and J˜Ghosh. Cluster EnsemblesA Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research*, 3(3):583–617, 2003. ISSN 15324435.

[30] A.˜L.˜N Fred and A.˜K Jain. Robust data clustering. *Engineering*, 2:II–128–II–133, 2003. ISSN 10636919.

[31] T.˜O Kvalseth. Entropy and correlation: Some comments. *Ieee Transactions On Systems Man And Cybernetics*, 17(3):517–519, 1987. ISSN 00189472.

[32] D.˜L Davies and D.˜W Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, April 1979. ISSN 0162-8828.

[33] T˜Calinski and J˜Harabasz. A dendrite method for cluster analysis. *Communications in Statistics Theory and Methods*, 3(1):1–27, 1974. ISSN 03610926.

[34] J.˜A Hartigan. Minimum mutation fits to a given tree. *Biometrics*, 29(1):53–65, 1973. ISSN 0006341X.

[35] W.˜J Krzanowski and Y.˜T Lai. A Criterion for Determining the Number of Groups in a Data Set Using Sum-of-Squares Clustering. *Biometrics*, 44(1):23–34, 1988. ISSN 0006341X.

[36] P˜Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(1):53–65, 1987. ISSN 03770427.

[37] P˜Mahalanobis. On the generalized distance in statistics. In *Proceedings of the National Institute of Science, Calcutta*, volume˜12, page˜49, 1936.

[38] L˜Lebart. Discrimination through the regularized nearest cluster method. In Y˜Dodge and J˜Whittaker, editors, *Computational Statistics*, volume˜1, pages 103–118. Physica Verlag, Vienna, 1992.

[39] O˜Alter, P.˜O Brown, and D˜Botstein. Singular value decomposition for genome-wide expression data processing and modeling. *Proceedings of the National Academy of Sciences of the United States of America*, 97(18):10101–10106, 2000.

[40] R˜Varshavsky, A˜Gottlieb, M˜Linial, and D˜Horn. Novel unsupervised feature filtering of biological data. *Bioinformatics (Oxford, England)*, 22(14):e507–13, July 2006. ISSN 1367-4811.

[41] W.˜M Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. ISSN 01621459.

[42] L˜Denoeud, H˜Garreta, and A˜Gu. Comparison of distance indices between partitions. In P˜L., editor, *Applied Stochastic Models and Data Analysis*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 21–28. Springer Berlin Heidelberg, 2005. ISBN 9783540344162.

[43] A˜Frank and A˜Asuncion. UCI Machine Learning Repository, 2010. URL `http://archive.ics.uci.edu/ml`.

[44] S.˜a Mingoti and J.˜O Lima. Comparing SOM neural network with Fuzzy c-means, K-means and traditional hierarchical clustering algorithms. *European Journal of Operational Research*, 174(3):1742–1759, November 2006. ISSN 03772217.

[45] L˜Altenberg. The Schema Theorem and Price s Theorem. In *Foundations of genetic algorithms*, volume˜3, pages 23–49. Citeseer, 1995.