Robust Clustering by Evolutionary Computation

Wolfgang von der Gablentz
Fraunhofer IPK
Pascalstr. 8-9, 10587 Berlin, Germany
wolfgang.gablentz@ipk.fhg.de

Mario Köppen
Fraunhofer IPK
Pascalstr. 8-9, 10587 Berlin, Germany
mario.koeppen@ipk.fhg.de

Evgenia Dimitriadou
Institut für Statistik TU Wien
Wiedener Hauptstr. 8-10,A-1040 Wien, Austria
evgenia.dimitriadou@ci.tuwien.ac.at

Abstract

In this paper we present a scheme driven by evolutionary computation to overcome the problem of comparing clustering results. The clustering results are achieved by qualitatively different clustering algorithms, which produce different partitionings. Our scheme helps us to overcome these problems of algorithms by generating clustering ones and selecting the best evaluated to evolve in another generation until the whole procedure reaches a robust result.

1 Introduction

Partitioning a given population of individuals into ”similarity” groups has many applications in science and business. Many clustering algorithms have been suggested and used in the literature to partition data into clusters (see [1],[2],[3],[4],[5],[6],[7],[8],[9],[10],[11],[12],[13],[14], [15],[16],[17]). When partitioning individuals into plausible subgroups, due to the origin of the data sets and also to the algorithms themselves, some main problems for this task are encountered. The true structure (ground truth), especially the number and shapes of the clusters, remains unknown. Different cluster algorithms and even multiple replications of the same algorithm result in different solutions due to random initializations and stochastic learning methods. Moreover, there is no clear indication which of the different solutions of the replications of an algorithm is the best one. Different clustering algorithms applied on a specific data set result in different results (see [18]). The quality criteria commonly used from the clustering methods are not compatible for different clustering approaches, i.e. from a good value of such a criterion it can not be decided, whether the clustering was the more suitable one for a given kind of data. Any attempt to fuse a clustering result for an uninspected data set simply has to reject one method and choose another. However, this selection can not be done by using the ”native” quality measures of the clustering methods. Perturbations or shuffling of the data points in a data set have been also introduced to overcome the problem of instabilities and of wrong partitioning results. In this way artificial instabilities are, applied in a data set by perturbing its points. The idea is that clusterings which does’nt change when the points in the datasets are slightly modified will be the adequate ones. Moreover, this method can influence the clustering algorithm to create more different cluster structures of the ones normally achieved by the algorithms and it also provides more robust results that explain adequately the structure of the original set (for more details see [19]). In this paper we introduce a method which produces different clustering results following a scheme driven by evolutionary computation. This scheme takes advantage of the polyphony of the clustering results, produced by the several algorithms due to the referred problems, by maintaining and using them as a point of comparison. The best results generated continue their evolution in a next step generation until we reach a robust result. This paper is organized as follows. In section 2 we describe how instabilities and a wrong

clustering result caused by an improper choice of clustering algorithm can be solved. The section 3 explains the use of evolutionary computation for the proposed task. In section 4 we explain the algorithms and datasets used in deeper detail. Section 5 descibes the results of our experiments, which leads to the conclusions descibed in section 6.

2 Robust Clustering

The basic assumption on robust clustering, which guides to the framework presented in this paper, is as follows: When a clustering method is inappropriate for the given data set, it will give more differing clustering results when the initial conditions or the given data values are randomly modified, than for an appropriate clustering method. Assume a (numbered) data set $d = (d_1, d_2, \ldots, d_n)$, which has to be separated into two clusters C_1 and C_2. Further assume a set of $M_1, M_2, \ldots, M_k$ clustering algorithms (e.g. k-means, neural gas,...), the $M_1(c_i), M_2(c_i), \ldots, M_k(c_i), 1 \leq i \leq 2$, results of which, when applied to d, will depend on an initial configuration of the c_i centers, and a set V of l random modifications of d, e.g. moving the data point by a small amount in a random direction. Then, when a clustering algorithm $M_p, 1 \leq p \leq k$, is applied to the modified data set $V_j(d)$, it will assign each data point in d to either C_1 or C_2. This can be represented by a clustering-bitstring. For example, if there are four data points, the bitstring 0110 describes that data point d_1 and d_4 have been assigned to cluster C_1 and data points d_2 and d_3 have been assigned to cluster C_2. Applying all k clustering algorithms to all l modifications of the data set d will give $k \cdot l$ bitstrings. Under the assumption given above, for appropriate clustering methods, there should be more similar bitstrings than for inappropriate ones. In other words: the common schemata of all $k \cdot l$ bitstrings hint on a more fitted clustering. In search is a method for selecting common schemata. Genetic algoritms are well-known optimization-methods for such a purpose.

3 Scheme Abstract

The proposed scheme generates clustering-bitstrings by clustering the l modifications with all clustering algorithm in use. Then these clustering-bitstrings are evaluated, compared through a fitness function. The best ones continue their evolution to the next generation. The final clustering bitstrings have evolved towards the stable schemata, which provide us a robust partition of the data points in the set.

4 Description of Experiment

4.1 Clustering Approaches

While studying the famous problem of *intertwined spirals* we achieved in former experiments (see [20], for more details on the problem and its definition, see [21]) the result that the submethod *single-linkage* of the class of agglomerative Clustering Algorithms is suitable to cluster them (due to some limitations). That's why we now use results of some of these algorithms to compare their suitability for a special kind of clustering problem.

4.1.1 Agglomerative Clustering

The basic scheme of the Agglomerative Clustering, which stands for a class of algorithms, fuses in each step one pair of clusters, consisting of only one datapoint at the beginning. The selection of the pair to fuse is based on some distance-criterion, thats why an adjacency matrix is used. Suppose a set of data $E=\{e_1, e_2, \ldots, e_i\}$:

1. start with the partitioning G , where $g_i = \{e_i\}$
2. search the two groups of data with the smallest distance between them among all possible pairs e.g. g_p und g_q mit $d_{pq} = \min d_{ij}$, $i \neq j$
3. fuse the groups g_p and g_q into the new group g_{qnew}. $g_{qnew} = g_p \cup g_q$
4. change the q-th row and column of the adjacency matrix by recomputing all distances between the new group g_{qnew} and the other groups and remove the p th row and column.
5. stop if a given criterion is reached, otherwise go back to the second step

4.1.2 Specification of a single Algorithm

The single algorithm is specified according to the manner, in which the distances are computed. It is usual to compute the distance of two clusters based on their centerpoints, what leads to the submethod *average-linked* . Another submethod computes the distance of two clusters using one abitrary element of each cluster. Two clusters will possess the smallest distance, if

one of their pairs of elements has a smaller distance than every possible pairs of elements between all the other pairs of clusters. This submethod, which is called *single-linkage* is, according to former experiments (see [20]) suitable to fuse datapoints located on an arbitrary bended line. This means, that this algorithm is suitable to cluster the famous problem of intertwined spirals unless the amount of datapoints is too great, so that the adjacency matrix which all Agglomarative Clustering algorithm depend on, will not be capable by computer memory (RAM).

4.2 Datasets

According to the results of our former work we use as one original set of data the intertwined spirals. They are produced by the method "mlbench.spirals" of the R language. The R Language is an open source reimplementation of the S Language which is meant to be a *programming enviroment for dataanalysis and graphics* (can be received at [22]). So, our former results lead to the assumption that this clustering problem could be solved by one of the algorithms in use.
The second original set of data is also implemented in R and is nearly suitable to be clustered by the *averagelinkage* submethod.

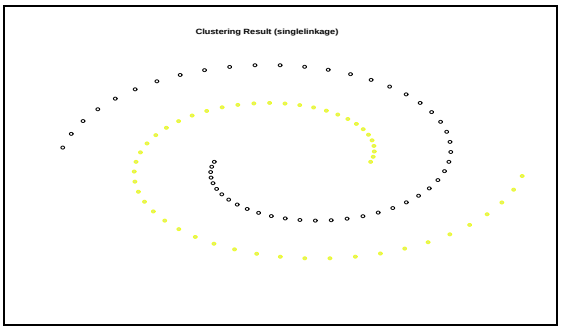

Figure 1: Intertwined spirals clustered by the singlelinkage algorithm.

4.3 Search of Schemata

Evolutionary algorithms are a family of computer models based on the mechanics of natural selection and natural genetics. Among them are genetic algorithms (GA) [23] and genetic programming (GP) [24]. Genetic algorithms were introduced and investigated by John Holland [23]. Later, they became popular by the book of

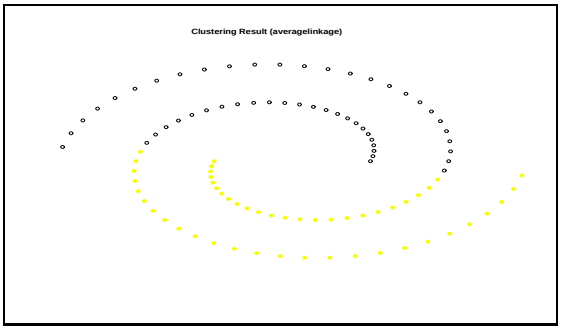

Figure 2: Intertwined spirals clustered by the averagelinkage algorithm.

David Goldberg [25]. Also, consider the GA tutorial of David Whitley [26] as a very good introduction to the field.

GAs and GPs are typically used for optimization problems. An optimization problem is given by a mapping $F : X \to Y$. The task is to find an element $x \in X$ for which $y = f(x), y \in Y$ is optimal in some sense. Genetic algorithms encodes a potential solution on a simple chromosome-like data structure, and apply genetic operators such as crossover or mutation to these structures. Then, the potential solution is decoded to the value x in the *search space* X, and $y = f(x)$ is computed. The obtained value y is considered as a quality measure, i.e. the *fitness* for this data structure. Some genetic operators, such as the mating selection, are under control of these fitness values, some other, like the mutation, are not related to fitness at all.

An implementation of a GA begins with a population of "chromosomes" (generation 1). For standard GA, each chromosome (also referred to as individual) is represented as a bitstring of a fixed length (e.g. `0101101` as a bitstring of length 7). Then, the genetic operators are applied onto all bitstrings iteratively in a fixed order, going from one generation to the next until a given goal (e.g. fitness value exceeds a given threshold or a predefined number of generations was completed) is met. Finally, the individual (or chromosome) with the best fitness value in the final generation is taken as the evolved solution of the optimization problem.

At first, $2m$ bitstrings are selected out of the k individuals of generation n for mating. Usually, this is done by fitness-proportionate selection, i.e., the relative probability for an individual to be selected is proportional to its fitness value. The better the fitness, the better is the chance to spread out its "genetic material" (i.e., some

of its bits) over the next generation.

Once the $2m$ individuals are chosen, they are paired. In the two bitstrings of each pair, a common splitting point is randomly selected, and a new bitstring is constructed by combining a half of the first bitstring with the other half of the other bitstring. Then, the new individuals are mutated, i.e. some of its bits are reversed with a given (usually small) probability. This gives the so-called *m children* of parent generation *n*.

Now, the fitness values of the children are evaluated by decoding them into *x* values and computing the $f(x)$. Some of the children might have a better fitness than its parents. From the *k* individuals of generation *n* and the *m* children, the best *k* individuals constitute the next generation $(n+1)$.

While randomized, GAs are no simple random walks. For the standard GA, John Holland has derived the well-known Schemata Theorem, which models a GA by means of the so-called schematas (or similarity templates). A schema is an incomplete bitstring in the sense that it contains unspecified bits. An example for a schema is `10*110`, which leaves position 3 unspecified. `101110` is a realization of this schema. Generation *n* contains each possible schema to some extent. It can be said, that such a schema is tested by the GA, or that trials are allocated to it by the GA. Now, one measure for a schema is the average fitness of all of its realizations. A second measure is the ratio of this avarage to the "average average" of all schemata present in the generation *n*, i.e. its *above-averageness*. The Schemata Theorem relates the rate of a schema within a population with this measure. It says, that the rate of a schema within a population grows exponentially with its above-averageness. The most important point here is that all schematas are tested in parallel.

Strongly related to the application of a GA is the encoding problem. In general, GAs are applied to highly non-linear, complex problems, where it is hard to find a model which provides an approach to the solution. In these applications, they are the most simple approach. However, a GA is not guaranteed to find the global optimum of a problem. It only ensures, by the Schemata Theorem, to find better solutions than the random initialized ones. GAs find evolved solutions.

4.4 Fitness Function

The fitness function $y = f(x), x \in X$ in search, has to keep track of the difference of the tested indiviual compared to all other (original) individuals. Because all individuals are given as bitstrings, the Hamming distance, which keeps track of inverted bitstrings will be the right measure. So we define the fitness function $y(b)$ as follows:

$$y(b) = \frac{1}{m}\sum_{i=1}^{m} \min\left[\sum_{j=1}^{n}|x_{i,j} - b_j|, \sum_{j=1}^{n}|(1-x_{i,j}) - b_j|\right]$$

where:

$n =$ length of the Bitstring

$m =$ number of orginals (clusterstrings)

$x_g \in X$

so that:

$x_{g,v}$ is the *v*th bit of *g*th original clustering-string

The alternating measure of the fitness function reflects the issue that different clustering approaches may decide differently for assigning class 1 or 2. Hence, the more suitable schema should be more similar to either the cluster string or its inverted form.

5 Experimental Results

We started our experiment so that 28 original clustering-strings with a length of 100 bit were computed. As parameterization of the Genetic Algorithm we decided to chose

- as the stop criteria 300 generations
- 30 parents in each generation
- 70 children in each generation
- one-point crossover
- mutation:
  - probability = 0.8
  - mutation methods: swapping of two random bits, invertation of a random bit

The highest fitness $1/y(b)$ we achieved was $1/y(b_{best}) = 0.0393256$. This bitstring b_{best} was not equal to any original clustering-string, but had a very low distance (Hamming distance) to the first of the originals. That's why we decided to name the clustering-bitstring with the minimal Hamming distance compared

with the individual reaching the highest fitness (b_{best} where $y(b_{best}) = \min(y(b))\forall b \in X$) the result of the procedure. The first original clustering-string, which represents, according to our results, the most appropriate clustering, is the one shown in figure 1, which shows obviously the correct partitioning of the intertwined spirals problem. After some more trials we got even more results leading again to the first original clustering-bitstring with even higher fitnesses, i.e. $1/y(b_{best}) = 0.04096$ with only four different bits inside the compared strings.

6 Conclusions

In this paper we have presented a scheme driven by evolutionary computation to overcome the problem of comparing results achieved by qualitatively different clustering algorithms. We have produced a couple of noisy copies of a given two-class clustering problem. Because it was a two-class problems, which means that we were clustering all data into two clusters, the clustering results could be represented as binary bitstrings, so they were compatible to the format genetic algorithms work on. Taking the whole lot of clustering results as input to the genetic algorithm we assumed to let it find the scheme of the right clustering for the presented problem. At the end we achieved reproducable result, for many runs of the genetic algorithm were leading to the same original clustering-bitstring which of course points to the most appropriate clustering algorithm. It is, according to our results, possible to find the appropriate clustering for a given problem, but it is also possible to identify the most suitable clustering algorithm for an unknown dataset. Last but not least it seems to be feasible, to classify clustering problems in comparison to their appropriate clustering algorithm. For future work we are going to apply the proposed approach to real data, including also more than two classes. Also, the study of the application of other evolutionary algorithms, like Cultural Algorithms (see [27]), are a topic of ongoing resaerch.

References

[1] Peng-Yeng Yin and Ling-Hwei Chen. A new non-iterative approach for clustering. *Pattern Recognition Letters*, 15:125–133, 1994.

[2] D. Chaudhuri, B. B. Chaudhuri, and C. A. Murthy. A new split-and-merge clustering technique. *Pattern Recognition Letters*, 13:399–409, 1992.

[3] Torbjorn Eltoft and Rui J. P. DeFigueiredo. A new neural network for cluster-detection-and-labeling. *IEEE Transactions on Neural Networks*, 9(5):1021–1034, September 1998.

[4] C. L. Begovich and V. E. Kane. Estimating the number of groups and group membership using simulation cluster analysis. *Pattern Recognition*, 15(4):335–342, 1982.

[5] Donald E. Brown, Christopher L. Huntley, and Paul J. Garvey. Clustering of homogeneous subsets. *Pattern Recognition Letters*, 12:401–408, 1991.

[6] Lei Xu, Adam Krzyzak, and Erkki Oja. Rival penalized competitive learning for clustering analysis, rbf net, and curve detection. *IEEE Transactions on Neural Networks*, 4(4):636–649, 1993.

[7] Lei Xu. Bayesian ying-yang machine, clusterong and number of clusters. *Pattern Recognition Letters*, 18:1167–1178, 1997.

[8] G. Celeux and G. Soromenho. An entropy criterion for assessing the number of clusters in amixture model. *Journal of Classification*, 13:195–212, 1996.

[9] Christophe Biernacki, Gilles Celeux, and Gerard Govaert. Assessing a mixture model for clustering with the integrated classification likelihood. Rapport de recherche 3521, Theme 4, Unite de recherche INRIA Lorraine, http://www.inria.fr, 1997.

[10] Richard S. Wallace and Takeo Kanade. Finding natural clusters having minimum description length. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 1, pages 438–442, Los Alamitos, CA, USA, 1990. IEEE Comput. S. Press.

[11] A. Marazzi, P. Gamba, A. Mecocci, and A. Semboloni. Automatic selection of the number of clusters in multidimensional data problems. In *Proceedings of the International Conference on Image Processing*, volume 3, pages 631–634, NY, USA, 1996. IEEE.

[12] Shri Kant, T. L. Rao, and P. N. Sundaram. An automatic and stable clustering algorithm. *Pattern Recognotion Letters*, 15:543–549, 1994.

[13] G. H. Ball and D. J. Hall. Isodata, a novel method of data analysis and pattern classification. Technical report, Stanford Research Institute, Menlo Park, 1965.

[14] G. Carpenter and S. Grossberg. Adaptive resonance theory: Stable selforganization of neural recognition codes in response to arbitrary lists of input patterns. In *Proc. 8th Annu. Conf. Cognitive Sci. Soc.*, pages 45–62, 1986.

[15] R. J. P. DeFigueiredo. The oi, os, omni and osman networks as best approximations of nonlinear systems under training data constraints. In *Proc. IEEE Int. Symp. Circuits Syst.*, Seattle, WA, 1996.

[16] Yoseph Linde, Andrs Buzo, and Robert M. Gray. An algorithm for vector quantizer design. COM-28(1):84–95, January 1980.

[17] Thomas M. Martinetz, Stanislav G. Berkovich, and Klaus J. Schulten. "Neural-Gas" network for vector quantization and its application to time-series prediction. 4(4):558–569, July 1993.

[18] A. Weingessel E. Dimitriadou and K. Hornik. A voting-merging clustering algorithm. In Fuzzy-Neuro Systems '99, editor, *SFB "Adaptive Information Systems and Modeling in Economics and Management Science*, number Working Paper 31, 1999.

[19] A. Weingessel E. Dimitriadou and K. Hornik. Fuzzy voting in clustering. In Fuzzy-Neuro Systems '99, editor, *G. Brewka, R. Der S. Gottwald and A. Schierwagen*, pages 63–74, 1999.

[20] Wolfgang von der Gablentz and Mario Köppen. agglomerative single-linkage clustering and its capability for solving the interwtined spirals problem, a technical report. experimental results, Fraunhofer IPK, http://www.ipk.fhg.de, 2000.

[21] Marvin L. Minsky and Seymour A. Papert. *Perceptrons – Expanded Edition*. MIT Press, 1988.

[22] Cran. http://cran.at.r-projects.org.

[23] J. A. Holland. *Adaptation in natural and artificial systems*. MIT Press, Cambridge,MA, 1975.

[24] J. Koza. *Genetic programming — On the programming of computers by means of natural selection*. MIT Press, Cambridge, MA, 1992.

[25] D. E. Goldberg. *Genetic algorithms in search, optimization & machine learning*. Addison-Wesley,Reading, MA, 1989.

[26] D. Whitley. A genetic algorithm tutorial. In *Statistics and Computing*, 4, pages 65–85, 1994.

[27] R. G. Reynolds. An introduction tu cultural algorithm. In *In Proceedings of Evolutionary Programming*, EP-94. San Diego. CA, 1994.