Document Clustering into an unknown number of clusters using a Genetic Algorithm

A. Casillas[1], M. T. González de Lena[2], and R. Martínez[2]

[1] Dpt. Electricidad y Electrónica
Universidad del País Vasco
arantza@we.lc.ehu.es
[2] Dpt. Informática, Estadística y Telemática
Universidad Rey Juan Carlos
{m.t.gonzalez,r.martinez}@escet.urjc.es

Abstract. We present a genetic algorithm that deals with document clustering. This algorithm calculates an approximation of the optimum k value, and solves the best grouping of the documents into these k clusters. We have evaluated this algorithm with sets of documents that are the output of a query in a search engine. The experiments show that, most of the times, our genetic algorithm obtains better values of the fitness function than the well known Calinski and Harabasz stopping rule, and takes less time.

1 Introduction

Clustering involves dividing a set of n objects into a specified number of clusters k, so that objects are similar to other objects in the same cluster, and different from objects in other clusters. Although clustering algorithms can work with objects of different kinds, we have focused on documents.

Several clustering approaches assume that the appropriate value of k is known. However, there may be quite a few situations in which it is not possible to know that appropriate number of clusters, or even an approximation. For instance, if we want to divide into clusters a set of documents that are the result of a query to a search engine, the value of k can change for each set of documents that result from an interaction with the engine.

In this work, we have dealt with the problem of clustering a set of documents without prior evidence on the appropriate number of clusters. Our aim is to provide an approximation of an appropriate value of k, with an acceptable computational cost, for a small number of documents.

There are various approaches for obtaining the optimum number, k, of clusters. We have focused on global stopping rules. These rules evaluate a measure, $C(k)$, of the goodness of the partition into k clusters, and calculate the value of k for which $C(k)$ is optimal. This $C(k)$ measure is usually based on the within-cluster and between-cluster similarity or distance. A drawback of many of these rules is that $C(1)$ is not defined [Gordon 99], so the set of data is always assumed to be partitioned, and this is one of our assumptions. Another general drawback

of stopping rules is the high computational cost of calculating the best value of k, even with a moderate number of documents, n.

Many different stopping rules have been proposed. In [Milligan & Cooper 85] a comparative study of 30 stopping rules is presented. That study was the result of a simulation experiment carried out with artificial data sets containing nonoverlapping clusters. Although the results of that experiment depend on the cluster generating strategy, the study is useful in identifying which stopping rules perform better. One of the rules which performed best in that experiment is Calinski and Harabasz's rule. This is the stopping rule which we have selected.

The Calinski and Harabasz [Calinski & Harabasz 74] stopping rule calculates an informal indicator of the best number of clusters. On this rule we have designed and implemented a Genetic Algorithm (GA) that finds an approximation of the k optimal value, with significantly lower computation time than the former in most cases.

There are earlier works that apply GA and evolutionary programming to clustering. Some of them deal with clustering a set of objects by assuming that the appropriate value of k is known ([Estivill-Castro & Murray 98], [Chu et al. 02], [Murthy & Chowdhury 96], [Mertz & Zell 02], [Lucasius et al. 93]). However, in [Sarkar et al. 97] an evolutionary programming-based clustering algorithm is proposed that groups a set of data into an optimum number of clusters. It is based on the well known K-means algorithm. They use two objective functions that are minimized simultaneously: one gives the optimum number of clusters, whereas the other leads to proper identification of each cluster's centroids. In [Makagonov et al. 02] discusses other heuristics to split the dendrite in an optimal way without fixing the number of clusters.

In our approach, only one objective function is maximized, so we calculate at the same time both aspects of the solution: an approximation to the optimum k value, and the best grouping of the objects into these k clusters.

The remainder of the paper is organized as follow: Section 2 describes the Calinski and Harabasz stopping rule. Section 3 presents the genetic algorithm. Section 4 describes the experiments and their results. Section 5 includes the main conclusions and suggestions for future work.

2 The Calinski and Harabasz Stopping Rule

In [Calinski & Harabasz 74] a method for identifying clusters of points in a multidimensional Euclidean space is presented. An informal indicator of the best number of clusters k is also calculated.

The method supposes there are n individuals with values of the same v variables for each individual. These individuals can be represented by n points in a v-dimensional Euclidean space. An $n \times n$ distance matrix is then calculated. Next, the method needs to calculate the Minimum Spanning Tree (MST), so that the enormous number of possible partitions of a set of points is reduced to those which are obtainable by splitting the MST.

This tree is then partitioned by removing some of its edges. If we want to divide the n points into k clusters, $k-1$ edges have to be removed. For each possible partition, the within-cluster sum of squared distances about the centroids ($WGSS$) is computed. In order to calculate the optimal value of k, first $k = 2$ is taken, then $k = 3$, and so on. For each value of k, the best partition is calculated with the minimum $WGSS$ and the Variance Ratio Criterion (VRC):

$$VRC = \frac{\frac{BGSS}{k-1}}{\frac{WGSS}{n-k}},$$

where $BGSS$ is the total between-cluster sum of squared distances.

The authors suggest using VRC as an informal indicator for the best value of k. They also suggest the computation of VRC for $k = 2, 3, \ldots$ choosing the value on k for which the VRC has an absolute or local maximum. The computation can be stopped when the first local maximum is reached.

Although of working with the minimum spanning tree instead of the whole graph reduces the number of partitions to be examined, this number, $\binom{n-1}{k-1}$, is high enough to use this method with even moderate value of n.

Our genetic algorithm uses the variance ratio criterion of Calinski and Harabasz's stopping rule as its fitness function.

3 The Genetic Algorithm

Genetic Algorithms were developed by John Holland at the University of Michigan. They are search algorithms based on the mechanics of natural selection and natural genetics [Holland 75]. The algorithm begins with an initial solutions population of our problem. This population is generated randomly. Each one of these solutions must be evaluated by means of a fitness function; the result of this evaluation is a measure of individual adaptation. The individuals with the best adaptation measure have more chances of reproducing and generating new individuals. Each individual (chromosome) is represented by a set of parameters (genes).

The GA uses two methods for generating new individuals: Crossover and Mutation. In the Crossover method, two parents (individuals of the current population) are selected in order to generate two offspring which are added to the next generation. That selection is carried out using the fitness function. The Mutation method guarantees that all the search space has a nonzero probability of being explored. Once the next population has been generated, by means of Crossover, Mutation or both, it has to be evaluated, and it then replaces the earlier population. This process is repeated a finite number of times with the aim of obtaining the global optimum of the problem.

A current description of GA can be found in [Goldberg 02] and [Michalewicz 96].

3.1 Population Representation

Our Genetic Algorithm is based on the Calinski and Harabasz Stopping Rule. We have used a chromosome (individual) description that allows us to represent two different points: the value of k, and which edges of the MST have to be eliminated. A vector with $n-1$ binary elements can deal with both points. The $n-1$ elements represent the $n-1$ edges of the MST. A vector element with value "0" means that this edge remains, whereas a vector element with value "1" means that this edge is eliminated. The number of elements with value "1" represents the value of $k-1$.

For instance, with 5 documents the MST will have 5 nodes and 4 edges. One chromosome could be the vector $(0,0,1,1)$, where $k=3$ and the edges removed are the third and fourth ones. We have selected this representation because it is straightforward and permits us to create valid chromosomes.

3.2 Solution Space

The solution space (search space) in the Calinski and Harabasz rule is $\binom{n-1}{k-1}$, where $k=2,\ldots,n-1$, and n is the number of documents. This formula has the maximum value when $k=\frac{n-1}{2}+1$. We have adapted our GA to search for an optimal solution in this specific search space. The probability of generating a chromosome with a value of k near that maximum is greater than with a value of k distant from that maximum.

3.3 Selection

The selection operator mimics the selection concept of natural genetic systems: the best chromosome survives. The probability of selection of chromosome is directly proportional to the fitness value (VRC formula for us). The chromosomes with the highest VRC values have more chances of reproducing and generating new chromosomes.

3.4 Crossover

Once two parents are selected, two offspring are generated. These offspring will receive information from both parents. The classical crossover method uses only a crossing point chosen at random. This crossing point marks the position where the vector will be cut in order to exchange information between the parents. For example, these two parents:

$$P_a = 0101$$

$$P_b = 1100$$

and a crossing point $c=2$ will generate the following offspring:

$$O_a = 0100$$

$$O_b = 1101$$

where O_a receives the first two values from P_a and the second ones from P_b, whereas the O_b offspring receives the first two values from P_b and the second ones from P_a.

We use a number of crossing points chosen at random (every number has the same probability).

3.5 Mutation

Before evaluating each next generation mutation is applied, so that each chromosome is subjected to a low probability of change or mutation. In order to guarantee that all the search space can be explored, our GA uses a mutation probability of 0.008%. In future experiments we will try other values.

3.6 Stopping criterion

There is no stopping criterion in the relevant literature which ensures the convergence of a GA to an optimal solution. We have used the two most usual criteria. Our GA stops when:

- After a number x of iterations, the best chromosome does not change. We have fixed $x = 3$.
- The maximum number of generations is reached. We chose this number to be n, the number of documents.

4 Experiments

For the experiments we implemented both algorithms: the Calinski and Harabasz stopping rule and our GA.

We used a collection of 14,000 news items from a Spanish newspaper. The documents that are the input of the algorithms are those resulting from a query with a search engine concerning that collection of news items. In this framework, no evidence of an appropriate number for k is known.

We are interested in calculating a "good" value for k in less time than the Calinski and Harabasz stopping rule. The goodness of k value will be represented by means of the VRC value; the bigger VRC, the better k, so that the value of VRC is the measure of the quality of the clustering solution.

We used four sets of documents (the output of four queries) containing 10, 12, 31, and 100 documents respectively. We use the same document representation as the search engine: the lemmas of each document filtered with stoplists. This is the input of the clustering algorithms. First, the pattern or profile matrix of the documents and then the distance matrix are computed.

The GA works with an initial population of $n \times 10$ chromosomes, where n is the number of documents. The maximum number of generations is n.

In Table 1 the results of the experiments can be seen in terms of VRC value, k value, and time. The results for time in Table 1 of the GA refer to the average time after running the algorithm 10 times. These experiments were carried out on a Pentium IV of 2 GHZ with 512 MB of RAM.

The Calinski and Harabasz stopping rule performs well in relation to time when the k value is close to 2. However, for other values of k (see the experiment with 31 documents in Table 1) the time is unacceptable. Our GA always works better with regard to the VRC value expected in that experiment.

	Genetic Algorithm			**Calin. and Harab. Alg.**		
Num. Doc	k **value**	**Average Time**	**VRC value**	k **value**	**Time**	**VRC value**
10	8 (70%)	9324μs	3260	4	4707μs	407
	9 (20%)	10814μs	1426			
	7 (10%)	8132μs	620			
12	8 (50%)	17528μs	6264	4	14255μs	511
	10 (40%)	18552μs	2436			
	9 (10%)	12248μs	4111			
31	16 (40%)	341118μs	16087	10	5864s	37536
	17 (40%)	68045μs	14076			
	18 (10%)	1 s	12301			
	15 (10%)	221949μs	18385			
100	43 (30%)	10s	1685	2	1s	77
	41 (20%)	31s	2073			
	40 (10%)	64s	2414			
	42 (10%)	7s	1756			
	44 (10%)	11s	2043			
	45 (10%)	10s	1961			
	48 (10%)	16s	1736			

Table 1. Results of Genetic Algorithm implementation

5 Conclusions

We propose a genetic algorithm that calculates an approximation of the optimum k value, and finds the best grouping of the objects into these k clusters.

The experiments show that, most of the times, the GA obtains much better values of VRC than the Calinski and Harabasz stopping rule and takes less time.

The Calinski and Harabasz stopping rule performs better in connection with time when the locally optimum k is a value close to 2. This is because this rule starts the exploration with $k = 2$, then 3, and so on, finishing when a local VRC maximum is found. However, the GA carries out a parallel search, where any value of k can be explored.

When the Calinski and Harabasz rule obtains better VRC value than our GA, it usually uses much more time (see the experiment with 31 documents in Table 1).

If there is evidence that an appropriate value of k is close to 2, then the Calinski and Harabasz stopping rule can calculate that value and the resulting k clusters in less time than the GA. But when the appropriate value of k is completely unknown, the GA performs better most of the time.

We have to test some points of the algorithm: (1) to try elitism, (2) to try other mutation methods, and (3) to carry out more experiments.

Acknowledgments

This research is supported by the Spanish Research Agency, project HERMES (TIC2000-0335-C03-03), and by the University of the Basque Country (9/UPV 00224.310-13566/2001).

References

[Calinski & Harabasz 74] Calinski, T., Harabasz, J.: "A Dendrite Method for Cluster Analysis". *Communications in Statistics*, 3(1), (1974) 1–27.

[Chu et al. 02] Chu S.C., Roddick J.F., Pan J.S.: "An Incremental Multi-Centroid, Multi-Run Sampling Scheme for k-medoids-based Algortihms-Extended Report". *Proceedings of the Third International Conference on Data Mining Methods and Databases, Data Mining III*, (2002), 553–562.

[Estivill-Castro & Murray 98] Estivill-Castro V., Murray A.T.: "Spatial Clustering for Data Mining with Genetic Algorithms". *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems, EIS-98*, (1998).

[Goldberg 02] Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley Longman, Inc. (2002).

[Gordon 99] Gordon, A. D.: *Classification*, Chapman & Hall/CRC, (1999).

[Holland 75] Holland, J. H.: *Adaptation in natural and artificial system*, Ann Arbor: The University of Michigan Press. (1975).

[Lucasius et al. 93] Lucasius C.B., Dane A.D., Kateman G.: "On k-medoid clustering of large data sets with the aid of Genetic Algorithm: background, feasibility and comparison". *Analytica Chimica Acta*, Elsevier Science Publishers B.V. 283(3), (1993) 647–669.

[Makagonov et al. 02] Makagonov, P., Alexandrov, M., Gelbukh, A.: "Selection of typical documents in a document flow". *Advances in Communications and Software Technologies*, WSEAS Press (2002) 197–202.

[Mertz & Zell 02] Merz P., Zell A.: "Clustering Gene Expresion Profiles with Memetic Algorithms". *Lecture Notes in Computer Science 2439*, Springer-Verlag Berlin (2002) 811–820.

[Michalewicz 96] Michalewicz, Z.: *Genetic algorithms + data structures = evolution programs*, Springer Comp. (1996).

[Milligan & Cooper 85] Milligan, G. W., Cooper, M.C.: "An Examination of Procedures for Determining the Number of Clusters in a Data Set". *Psychometrik*, 58(2), (1985) 159–179.

[Murthy & Chowdhury 96] Murthy C.A., Chowdhury N.: "In search of OptimalClusters Using Genetic Algorithms". *Pattern Recognition Letters*, 17(8), (1996), 825–832.
[Sarkar et al. 97] Sarkar, M., Yegnanarayana, B., Khemani, D.: "A clustering algorithm using an evolutionary programming-based approach". *Pattern Recognition Letters*, 18, (1997) 975–986.