

A Parallel Genetic Algorithm for Pattern Recognition in Mixed Databases

Angel Kuri-Morales^(✉) and Javier Sagastuy-Breña

Instituto Tecnológico Autónomo de México, Río Hondo no. 1, 01000 Mexico D.F., Mexico
{akuri, jsagastu}@itam.mx

Abstract. Structured data bases may include both numerical and non-numerical attributes (categorical or CA). Databases which include CAs are called “mixed” databases (MD). Metric clustering algorithms are ineffectual when presented with MDs because, in such algorithms, the similarity between the objects is determined by measuring the differences between them, in accordance with some predefined metric. Nevertheless, the information contained in the CAs of MDs is fundamental to understand and identify the patterns therein. A practical alternative is to encode the instances of the CAs numerically. To do this we must consider the fact that there is a limited subset of codes which will preserve the patterns in the MD. To identify such pattern-preserving codes (PPC) we appeal to neural networks (NN) and genetic algorithms (GA). It is possible to identify a set of PPCs by trying out a bounded number of codes (the individuals of a GA’s population) and demanding the GA to identify the best individual. Such individual is the best practical PPC for the MD. The computational complexity of this task is considerable. To decrease processing time we appeal to multi-core architectures and the implementation of multiple threads in an algorithm called *ParCENG*. In this paper we discuss the method and establish experimental bounds on its parameters. This will allow us to tackle larger databases in much shorter execution times.

Keywords: Categorical databases · Neural networks · Genetic algorithms · Parallel computation

1 Introduction

Cluster Analysis is the name given to a diverse collection of techniques that can be used to classify objects in a structured database. The classification will depend upon the particular method used because it is possible to measure similarity and dissimilarity (distance between the objects in the DB) in a number of ways. Once having selected the distance measure we must choose the clustering algorithm. There are many methods available. Five classical ones are (a) Average Linkage Clustering, (b) Complete Linkage Clustering, (c) Single Linkage Clustering, (d) Within Groups Clustering, (e) Ward’s Method [1]. Alternative methods, based on computational intelligence, are (f) K-Means, (g) Fuzzy C-Means, (h) Self-Organizing Maps, (i) Fuzzy Learning Vector Quantization [2]. All of these methods have been designed to tackle the analysis of strictly numerical databases, i.e. those in which all the attributes are directly expressible as numbers.

If any of the attributes is non-numerical (i.e. categorical) none of the methods in the list is applicable. Clustering of categorical attributes (i.e., attributes whose domain is not numeric) is a difficult, yet important task: many fields, from statistics to psychology deal with categorical data. In spite of its importance, the task of categorical clustering has received relatively scant attention. Much of the published algorithms to cluster categorical data rely on the usage of a distance metric that captures the separation between two vectors of categorical attributes, such as the Jaccard coefficient [3]. An interesting alternative is explored in [4] where COOLCAT, a method which uses the notion of entropy to group records, is presented. It is based on information loss minimization. Another reason for the limited exploration of categorical clustering techniques is its inherent difficulty.

In [5] a different approach is taken by (a) Preserving the patterns embedded in the database and (b) Pinpointing the codes which preserve such patterns. These two steps result in the correct identification of a set of PPCs. The first issue implies the use of an algorithm which is certifiably capable of pattern identification. Multilayer perceptron networks (MLP) have been mathematically proven to do so [6]. On the other hand, we need a method which guarantees the correct identification of a set of codes. These stem from a very large ensemble and should minimize the approximation error implicit in the practical implementation of point (a) above. Genetic algorithms (GA) have been proven to always attain the global optimum of an arbitrary function [7] and, furthermore, a specific GA called the Eclectic GA (or EGA) has been shown to solve the optimization problem [8] efficiently. The resulting algorithm is called CENG (Categorical Encoding with Neural Networks and Genetic Algorithms) and its parallelized version ParCENG.

The rest of the paper is organized as follows. In Sect. 2 we briefly describe (a) Pseudo-binary encoding as an alternative to our approach, (b) The optimization problem CENG solves, (c) The basic tenets of multi-layer perceptron (MLP) networks and the EGA. In Sect. 3 we present the global methodology resulting in the parallel version of CENG, for which see [5]. In Sect. 4 we present some experimental results and, finally, in Sect. 5 we present our conclusions.

2 Encoding Mixed Databases

As stated in the introduction, the basic idea is to apply clustering algorithms designed for strictly numerical databases (ND) to MDs by encoding the instances of categorical variables with a number. This is by no means a new concept. MDs, however, offer a particular challenge when clustering is attempted because it is, in principle, impossible to impose a metric on CAs. There is no way in which numerical codes may be assigned to the CAs in general.

2.1 Pseudo-Binary Encoding

A common alternative is to replace every CA variable by a set of binary variables, each corresponding to the instances of the category. The CAs in the MD are replaced by numerical ones where every categorical variable is replaced by a set of t binary numerical

codes. An MD (with c categories and n numerical variables) will be replaced by an ND with $n - c + ct$ variables. This approach suffers from the following limitations:

- (a) The number of attributes of ND will be larger than that of MD. In many cases this leads to unwieldy databases which are more difficult to store and handle.
- (b) The type of coding system selected implies a subjective choice since all pseudo-binary variables may be assigned any two values (typically “0” denotes “absence”; “1” denotes “presence”). This choice is subjective. Any two different values are possible. Nevertheless, the mathematical properties of ND will vary with the different choices, thus leading to clusters which depend on the way in which “presence” or “absence” is encoded.
- (c) Finally, with this sort of scheme the pseudo-binary variables do no longer reflect the essence of the idea conveyed by category c . A variable corresponding to the t -th instance of the category reflects the way a tuple is “affected” by belonging to the t -th categorical value, which is correct. But now the original issue “How does the behavior of the individuals change according to the category?” is replaced by “How does the behavior of the individuals change when the category’s value is t ?” The two questions are not interchangeable.

2.2 Pattern Preserving Codes

An alternative goal is to assign codes (which we call Pattern Preserving Codes or PPCs) to each and all the instances of every class (category) which will preserve the patterns present for a given MD.

Consider a set of n -dimensional tuples (say U) whose cardinality is m . Assume there are n unknown functions of $n-1$ variables each, which we denote with

$$f_k(v_1, \dots, v_{k-1}, v_{k+1}, \dots, v_n); k = 1, \dots, n$$

Let us also assume that there is a method which allows us to approximate f_k (from the tuples) with F_k . Denote the resulting n functions of $n-1$ independent variables with F_i , thus

$$F_k = f(v_1, \dots, v_{k-1}, v_{k+1}, \dots, v_n); k = 1, \dots, n \quad (1)$$

The difference between f_k and F_k will be denoted with ε_k such that, for attribute k and the m tuples in the database

$$\varepsilon_k = \max[abs(f_{ki} - F_{ki})]; i = 1, \dots, m \quad (2)$$

Our contention is that the PPCs are the ones which minimize ε_k for all k . This is so because only those codes which retain the relationships between variable k and the remaining $n-1$ variables AND do this for ALL variables in the ensemble will preserve the whole set of relations (i.e. patterns) present in the data base, as in (3).

$$\Xi = \min[\max(\varepsilon_k; k = 1, \dots, n)] \quad (3)$$

Notice that this is a multi-objective optimization problem because complying with condition k in (2) for any given value of k may induce the non-compliance for a different possible k . Using the min-max expression of (3) equates to selecting one point in the Pareto's front [10].

To achieve the purported goal we must have a tool which is capable of identifying the F_k 's in (1) and the codes which attain the minimization of (3). This is possible using NNs and GAs.

Cybenko [6] proved the Universal Approximation Theorem which states that MLP networks with a single hidden layer are sufficient to compute a uniform ε approximation to a given training set represented by the set of inputs x_1, \dots, x_{m_O} and a desired (target) output $f(x_1, \dots, x_{m_O})$ where m_O denotes the number of independent variables and $F(x_1, \dots, x_{m_O})$ is the output of the NN, thus

$$\left| F(x_1, \dots, x_{m_O}) - f(x_1, \dots, x_{m_O}) \right| < \varepsilon \quad (4)$$

Rudolph [7] proved that the canonical GA (CGA) maintaining the best solution found over time after selection converges to the global optimum. This result ensures that CGA is an optimization tool which will always reach the best global value.

However, its time of convergence is not bounded. In [8] a statistical survey was conducted wherein 5 evolutionary algorithms were tested vs. a very large ($>10^6$) number of problems. The relative performance of these algorithms is shown in Fig. 1. The five algorithms selected for this study were: CHC (Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation GA), RMH (Random Mutation Hill Climber), SGA (Statistical GA), TGA (eliTist CGA) and EGA (Eclectic GA).

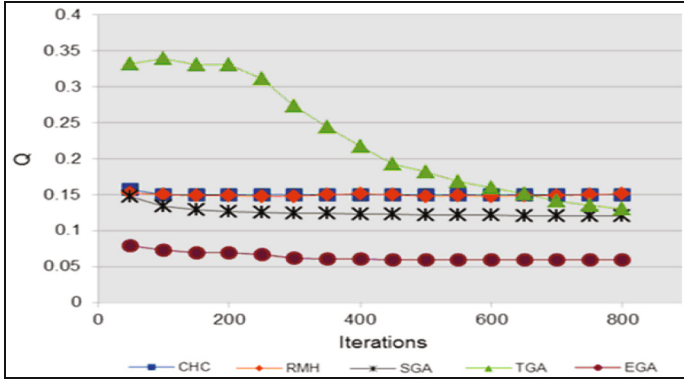


Fig. 1. Relative performance of different GAs.

Notice that EGA reached values very close (within 5%) to the global known optimum and did so in less than 400 generations. Therefore, from Rudolph's theorem and the previous results we have statistical proof that the best practical set of codes for the categorical instances will be found by EGA in a short number of generations.

3 General Methodology

The above mentioned tools are computationally intensive and when applied in CENG, yield an effective algorithm, albeit with relatively large execution times even on small databases.

3.1 The CENG Algorithm

The general algorithm for CENG is as follows:

- Specify the mixed database MD.
- MD is analyzed to determine c and t (see above). The size of the genome (the number of bits needed to encode the solution with a precision of 2^{-22}) is $L = 22 \text{ } ct$. EGA randomly generates a set of PS strings of size L . This is population P_0 . Every one of the PS strings is an individual of EGA.

For $i=1$ to G

For $j=1$ to PS

 From individual j , ct numerical codes are extracted and ND_{ij} is generated replacing the categorical instances by their numerical counterparts. Numerical variables are left undisturbed. MLP_{ij} 's architecture (corresponding to individual j) is determined as described above.

For $k=1$ to $n-1$

MLP_{ij} is fed with a data matrix in which the k -th attribute of ND is taken as a variable dependent on the remaining $n-1$. MLP_{ij} is trained and the maximum error e_{ijk} (i.e. the one resulting by feeding the already trained MLP_{ij} with all tuples vs. the dependent variable) is calculated.

endFor

 Fitness(j) = $[\max(e_{ijk})]$

endFor

 if Fitness(j) < 0.01 EGA ends. Otherwise the PS individuals of P_i are selected, crossed over and mutated.

This is the new P_i .

endFor

EGA yields the codes for every one of the ct instances of the categorical variables and ND: a version of MD in which all categorical instances are replaced by the proper numerical codes.

3.2 ParCENG

As opposed to other optimization tools, however, GAs are prone to easy parallelization. Every individual's evaluation is independent of every other one's, so that a multi-thread implementation of every individual's fitness evaluation is natural. So much so, that we may *a priori* determine that execution time will decrease basically as the inverse of the number of threads (cores).

Nevertheless, there is still the need to fine-tune the parameters of ParCENG. Two main issues affecting execution time were explored: (a) The parameters of the EGA and (b) Those of the NNs.

From the algorithm presented in Sect. 3.1, the process of training the $n - 1$ MLPs in the innermost loop of CENG is independent for every individual in the generation. Thus, it is natural to spawn PS parallel processes which will execute this portion of the algorithm.

3.3 ParCENG's Configuration Parameters

The maximum number of generations of the EGA has a direct and linear impact on the overall execution time of the algorithm. It becomes fundamental to let the algorithm run for enough generations to reach a global minimum but not for any more than that.

During the evolution of each generation in the EGA, every individual must train and evaluate $n - 1$ MLPs on the codes defined by the genome of the individual. The training process of an MLP using the backpropagation algorithm [9] depends on the number of epochs on the training data, which in turn also has a linear impact on the execution time required to effectively train it. Therefore, the MLPs should be trained using no more epochs than needed to correctly approximate the patterns present in ND. See Sect. 4.

4 Experiments

4.1 Parallel CENG

The training process of the $n - 1$ MLPs for each individual in a generation was programmed to be executed in parallel. This approach takes advantage of multi-core architectures by using the total number of cores available on the machine the code is executing on. In Fig. 2 we show the execution times for a database consisting of 3 numerical variables, 10 categorical variables with 4 instances each, and 500 tuples.

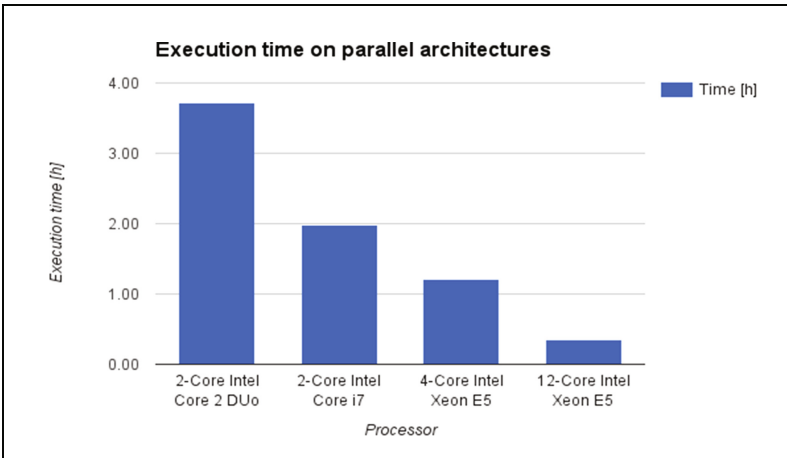


Fig. 2. ParCENG's execution times

Parallel CENG was run for the same database on 4 different architectures, as shown in Table 1.

Table 1. Execution times for ParCENG.

Processor	Clock Rate	Physical cores	Virtual cores
Intel Core 2 Duo	2.66 GHz	2	NA
Intel Core i3	1.7 GHz	2	4
Intel Xeon E5	2.6 GHz	4	8
Intel Xeon E5	2.7 GHz	12	24

For this experiment, the maximum number of generations for the EGA was set to 100 and the number of epochs the MLPs were trained for was set to 1000.

4.2 Determining the Number of Generations for the EGA

We ran ParCENG for 1,000 generations and recorded the fitness obtained after each generation for 3 databases of different sizes and with a different number of variables. Each of the databases included 3 numerical variables. However, one also contained 3 categorical variables with 4 instances each and consisted of 700 tuples, another contained 5 categorical variables with 4 instances each and 500 tuples. A last one contained 9 categorical variables with 4 instances each and 300 tuples. These databases were chosen since they reflected similar execution times in the sequential case.

In Fig. 3 we can see that database size in terms of variables and tuples directly affects the number of generations in which the EGA converges. We may see that $G = 800$ generations are enough. G is sensitive to the size of the input and complexity of the patterns present in it. It should be statistically determined depending on the data CENG works on.

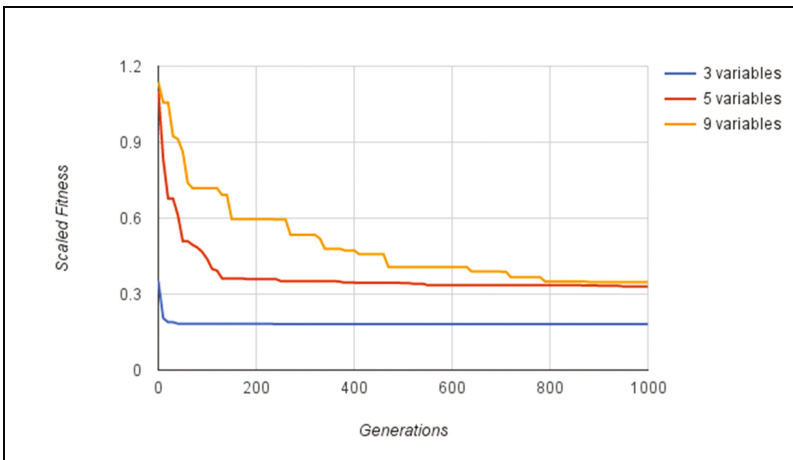


Fig. 3. Fitness varying maximum number of generations for the EGA

4.3 Determining the Number of Epochs to Train the MLPs

To determine the minimum number of epochs needed to train the MLPs in CENG, we ran the algorithm varying only the number of epochs. CENG was set to run for 100 generations and was fed a database consisting of 3 numerical variables, and 10 categorical variables with 4 instances each. The algorithm was then configured to run varying the number of epochs from 20 to 500 in steps of 20. Each configuration was run on 10 databases with tuples varying from 100 to 1,000 in steps of 100. The results are shown in Fig. 4.

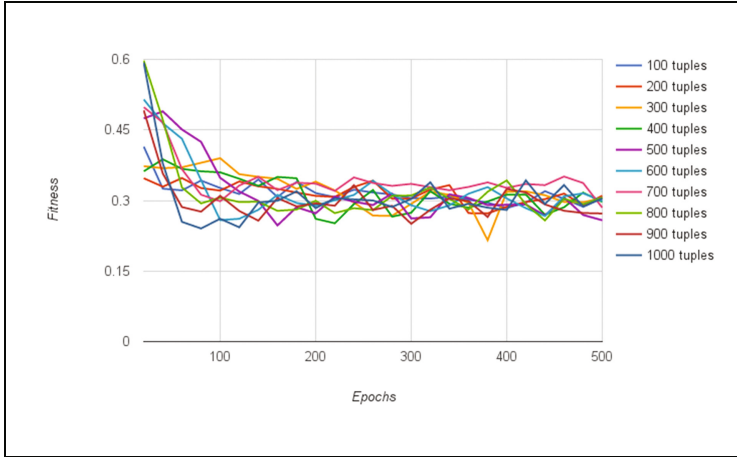


Fig. 4. Fitness varying number of epochs for MLP training

After 200 epochs, there appears to be no further improvement in terms of final fitness. We also explored the effect of running the algorithm for more than 500 epochs and concluded that no further improvement was possible. Thus, no more than 200 epochs are needed to train the MLPs in CENG to zero-in on the patterns present in the input data. Nonetheless, as is the case with the number of generations, we believe this configuration to be dependent on the complexity of the patterns present in the input database. Hence, case by case statistical determination of this parameter is necessary.

5 Conclusions

Preserving the information embedded in categorical attributes while replacing every instance of every class of a mixed database with a numerical value requires the identification of complex patterns tacit in the database. This has been achieved by using NNs and GAs in CENG, an algorithm essentially based on soft computing tools. An obvious disadvantage of the method is that the best codes change for every new database and, therefore, optimizing the conversion process from MDs to NDs becomes a priority. The algorithm is computationally intensive and we have appealed to the obvious possible parallel evaluation of the individuals in a version called ParCENG.

By using n physical cores execution time decreased almost linearly with n . However, to exploit the parallel alternative to its utmost we ought to fine-tune the parameters of, both, the EGA and the NNs. From the experiments conducted we were able to set upper bounds of the number of epochs to train the NNs as well as the number of generations of the EGA. These bounds are particular to ParCENG and, when selected as per our results, lead to execution times (for complex databases which originally necessitated of hours of CPU time) of a few minutes. This result is hardly surprising. However, the quantitative determination of the relation between the execution times and the parameters involved in ParCENG allows us to determine its behavior *a priori*. Thus, ParCENG specifications are freed from the purely heuristic estimation of its parameters. This will allow us to tackle much larger databases setting ParCENG's parameters accordingly.

One has to keep in mind that ParCENG offers a very interesting alternative to condition mixed databases into purely numerical ones. Thus, the resulting data, while retaining the patterns originally contained in the categories, are now susceptible to be applied to numerically cluster the data.

References

1. Norusis, M.: SPSS 16.0 statistical procedures companion. Prentice Hall Press, Upper Saddle River (2008)
2. Goebel, M., Gruenwald, L.: A survey of data mining and knowledge discovery software tools. *ACM SIGKDD Explor. Newslett.* **1**(1), 20–33 (1999)
3. Sokal, R.R.: The principles of numerical taxonomy: twenty-five years later. *Comput.-Assist. Bacterial Syst.* **15**, 1 (1985)
4. Barbará, D., Yi, L., Julia C.: COOLCAT: an entropy-based algorithm for categorical clustering. In: *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pp. 582–589. ACM (2002)
5. Kuri-Morales, A.F.: Categorical encoding with neural networks and genetic algorithms. In: Zhuang, X., Guarnaccia, C. (eds.) *WSEAS Proceedings of the 6th International Conference on Applied Informatics and Computing Theory*, pp. 167–175, 01 July 2015. ISBN 9781618043139, ISSN: 1790-5109
6. Cybenko, G.: Approximation by superpositions of a sigmoidal function. *Math. Control, Signals Syst.* **2**(4), 303–314 (1989)
7. Rudolph, G.: Convergence analysis of canonical genetic algorithms. *IEEE Trans. Neural Networks* **5**(1), 96–101 (1994)
8. Castro, F., Gelbukh, A., González, M. (eds.): MICAI 2013. LNCS (LNAI), vol. 8266. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-45111-9](https://doi.org/10.1007/978-3-642-45111-9)
9. Widrow, B., Lehr, M.A.: 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proc. IEEE* **78**(9), 1415–1442 (1990)
10. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J.J., Schwefel, H.-P. (eds.) *PPSN 2000*. LNCS, vol. 1917, pp. 849–858. Springer, Heidelberg (2000). doi:[10.1007/3-540-45356-3_83](https://doi.org/10.1007/3-540-45356-3_83)

Pattern Recognition

9th Mexican Conference, MCPR 2017, Huatulco, Mexico,

June 21-24, 2017, Proceedings

Carrasco-Ochoa, J.A.; Martínez-Trinidad, J.F.;

Olvera-López, J.A. (Eds.)

2017, XI, 310 p. 124 illus., Softcover

ISBN: 978-3-319-59225-1