Efficient Distributed Genetic Algorithm for Rule Extraction

Antonio Peregrin
Dept. of Information Technologies
University of Huelva
peregrin@dti.uhu.es

Miguel Angel Rodriguez
Dept. of Information Technologies
University of Huelva
miguel.rodriguez@dti.uhu.es

Abstract

This paper presents an efficient distributed genetic algorithm for classification rules extraction in data mining, which is based on a new method of dynamic data distribution applied to parallelism using networks of computers in order to mine large datasets. The presented algorithm shows many advantages when compared with other distributed algorithms proposed in the specific literature. In this way, some results are presented showing significant learning rate speed-up without compromising other features.

1. Introduction

Mining huge datasets for learning classification models with high prediction accuracy can be a very difficult task. The evaluation of data over large size datasets makes data mining algorithms inefficacy and inefficient. The effect produced by the size of the data in the algorithms is called scaling problem.

There are three main approaches for this problem:

- Use as much as possible apriori knowledge to search in subspaces small enough to be explored.
- Perform data reduction.
- Algorithm scalability.

In this way, there have been several efforts to make use of models based on distributed evolutionary algorithms in data mining for classification in large size datasets in order to emphasize on aspects of scalability and efficiency. REGAL [1] and NOW G-net [2] increase the computational resources via the use of data distribution. Nowadays, the use of collection of computers to achieve greater amount of computational resources become more popular because they are much more cost-effective than single computers of comparable speed.

In this paper we present an Efficient Distributed Genetic Algorithm for classification Rules extraction (EDGAR) with dynamic data partitioning that shows advantages in scalability for exploring high complexity search spaces with comparable classification quality.

The outline of the contribution is as follows: In Section 2 we review the distributed evolutionary models in rule induction. Section 3 is devoted to analyse the proposed algorithm. The experimental study developed is shown on Section 4 and finally, we reach conclusions in Section 5.

2. Preliminaries: genetic learning

Genetic Algorithms (GA) are search algorithms based on natural genetics that provide robust search capabilities in complex spaces, and thereby offer a valid approach to problems requiring efficient and effective search processes [8].

They have achieved reputation of robustness in rule induction, in commons problems associated to real world mining (noise, outliers, incomplete data, etc…). Initially GA were not designed as machine learning algorithms but can be easily dedicated to this task [10]. Typically the search space is seen as the entire possible hypothesis rule base that covers the data. The goodness is usually related to a coverage function over a number of learning examples.

Regarding the representation of the solutions, the proposals in the specialized literature follow two approaches in order to encode rules within a population of individuals:

- The ‘‘Chromosome = Set of rules’’, also called the Pittsburgh approach, in which each individual represents a rule set [18]. In this case, a chromosome evolves a complete RB and they compete among them along the evolutionary process. GABIL [9] and GA-MINER [7] are proposals that follow this approach.
- The ‘‘Chromosome = Rule’’ approach, in which each individual codifies a single rule, and the whole rule set is provided by combining several individuals in a population (rule cooperation) or via different evolutionary runs (rule competition).

In turn, within the ‘‘Chromosome = Rule’’ approach, there are three generic proposals:

978-0-7695-3326-1/08 $25.00 © 2008 IEEE
DOI 10.1109/HIS.2008.128

- The Michigan approach, in which each individual encodes a single rule. These kinds of systems are usually called learning classifier systems [17]. They are rule-based, message-passing systems that employ reinforcement learning and a GA to learn rules that guide their performance in a given environment. The GA is used for detecting new rules that replace the bad ones via the competition between the chromosomes in the evolutionary process.
- The IRL (Iterative Rule Learning) approach, in which each chromosome represents a rule. Chromosomes compete in every GA run, choosing the best rule per run. The global solution is formed by the best rules obtained when the algorithm is run multiple times. SIA [4] is a proposal that follows this approach.
- The GCCL (genetic cooperative-competitive learning) approach, in which the complete population or a subset of it encodes the RB. In this model the chromosomes compete and cooperate simultaneously. COGIN [19], REGAL [1] and NOW G-Net [2] are examples with this kind of representation.

In relation with scalability, one of the approaches to scale supervised data mining algorithms is to use data distribution in a number of processors [3]. Then each processor performs genetic learning using a single training data subset and a single subpopulation. It seems that each sub-population is likely to over fit the corresponding training data subset. Some approaches have been applied to avoid this problem:

- Dynamically redistribute training data and rules. A master process controls the data assigned trough the learning process [1][2] to each client processor.
- The data subsets assigned to each client processors changes after a pre-specified number of generations without removing the existing population. The individuals will try to cover then the newly arrived training data [6].

The proposal presented in this work follows GCCL approach with a new dynamic data distribution technique in order to get scalability without the previously mentioned over fitting, and a central elite pool to integrate the partial solutions.

3. EDGAR

This section describes the characteristics of EDGAR. This algorithm uses the inherent parallelism of GA for distributing the population and the training data in order to improve the scalability on large datasets.

This section details the distributed model and the components of the genetic algorithm: representation, genetic operators, genetic search and data reduction. Last part of this section is devoted to the greedy algorithm used for determine the best set of rules that will make up the classifier from the redundant population of rules generated by a GCCL algorithm.

3.1. Model

Using the intrinsic parallelism of GA, various distributed GA have been proposed to reduce the computational effort needed in solving complex optimization problems. Instead of evolving the entire population in a single processor, parallel GA applies the concept of multiple intercommunicating subpopulations [10] in analogy with the natural evolution of spatially distributed populations namely island model.

In order to improve scalability, we have assigned different partitions of the learning data to each node. Each node will tend to cover the local data proposing a concept description. The communication of the best individuals between subpopulations will enforce those individuals that perform properly in more than one node.

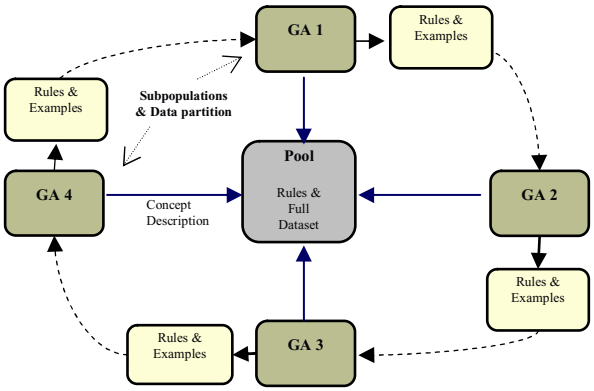

Figure 1. Distributed model

To avoid the disruption due to data partitioning we propose a novel technique called data learning flow (DLF). DLF it is based in the idea that a training example not properly covered may be better covered in another data partition. DLF copies the training examples covered by low fitness rules to the neighborhood. DLF allows recreating the original concept represented by those learning examples in one of the existing data partitions.

When the data does not have small disjunts [15] and the classes are nearly balanced each node will produce

a valid set of rules. But datasets with small dijunts will be even more difficult to learn after data partitioning.

Another difference with the island model is the extraction of the rule set that will form the final solution. In an island model all the nodes will converge to a similar population, but with data partitioning each node will generate a different set of rules reflecting the assigned data and may not perform properly against the entire dataset.

This proposal uses an elite pool in a central node with the full dataset for this purpose (see figure 1). The nodes send their best rules to the elite pool. When the classification ratio does not improve in the elite pool for some generations the current set of rules is extracted from this pool as the proposed classifier.

As a summary, the distributed model proposed is based on:

- Multiple intercommunicating subpopulations.
- Distributed data and DLF.
- Central elite pool.

3.2. Local genetic algorithm

EDGAR uses a local GA in each node with some communications with the neighborhood for individuals and poorly covered examples.

```
Generate initial population using seeding
While (Stop Criteria)
   For a number of generations
      Select g individuals by US
         For each individual
            If % Perform recombination
            If % Perform mutation
          end
         replace g individual from population
         Exchange individuals
         Exchange training examples
      end
    end
    Extract set of rules by greedy algorithm
    Send set of rules to Central Pool
    If (not improving) reduce training data
  end
```

Figure 2.Local Genetic Algorithm

Each node receives a subset of the training examples randomly selected from the initial dataset. Each training example only will exist initially in one partition. The initial population is created with rules covering some of the examples of the initial population (seeding). In each cycle some individual are selected using the Universal Suffrage (US) operator for recombination and mutation. Each offspring will replace a randomly selected individual in the current population.

After a number of iterations, some operations are performed (see figure 2):

- The rules that better cover the local data are copied to other nodes and the elite pool.
- Some individuals from other nodes replace randomly selected individuals in the current population.
- DLF: The learning examples not covered or covered by low fitness rules are copied to other nodes.
- If the best rules are the same for a number of iterations, the best individual and their covered cases are removed.

The process finalizes whether all the training data is removed or the classification ratio in the central pool does not improve after a number of generations.

3.3. Representation

EDGAR uses a fixed length bit string representation to code a disjunctive rule. The use of fix length chromosomes allows having simpler genetic operators. In order to keep the maxima that a local change on the genotype implies a proportionate change on the phenotype, each different possible value of an attribute in a rule is represented as a single bit. Meaning a bit set to 1 the presence of the value in the rule and a bit set to 0 the absence of this value in the rule.

A rule is composed by characteristics $C = c_1, c_2, \ldots, c_j$, each one can take just one value in each instance of the data mined but the rule may have more than one value for this characteristic.

For example, a rule which covers a dataset with three antecedents $c_1\ (v_1,v_2,v_3)$, $c_2(v_4,v_5)$, $c_3\ (v_6,v_7,v_8)$ and the consequent $class(v_9,v_{10})$ will be represented as seen in figure 3.

if c_1 in (v_1,v_3) and c_3 in (v_1) then class is v_2

c_1			c_2		c_3			Class	
V_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
1	0	1	0	0	1	0	0	0	1

Figure 3. Example of rule representation

3.4. Fitness

The fitness function is based on the following measurements:

- Simplicity: considered as the number of conditions in a rule. In a bit string representation, as more zeros are presents in the formula, as fewer conditions in the rule.
- Quality: is inversely the number of misclassifications. This means examples covered with a different assigned class.

The fitness function is

$$f(r) = \left(1 + \frac{zeros(r)}{lenght(r)}\right)^{-1*Cases-}$$

Where $Cases^-$ means the number of covered examples predicted has false positives (different consequent as the rule). *Zeros(r)* is the number of zeroes in the bit string representation of the rule *r* and *length* is the chromosome length in bits.

3.5. Genetic operators

The species formation is of great importance in a GCCL algorithm. For this purpose EDGAR uses the selection operator universal suffrage (US) first used in [1]. This mechanism creates niches of coverage that do not compete between them (co-evolution).

The recombination and mutation operators are based on standard bit string representation and are applied on the selected parents based on a probability. The recombination operator used is the two-point crossover. The mutation operator changes one random bit in the chromosome.

3.6. Data training set reduction under evolution and covering

US is a powerful and flexible operator for most situations, but if training examples representing a concept are a in a smaller amount than other concepts, US could makes the rule disappear under the attraction of the rules with more voters.

In order to dedicate more computational effort to the more difficult to learn examples the algorithm deletes the examples already learned. The process is the following: when the algorithm detects that the rule set has the same rules in a number of generation, the best rule is removed from the population. The criteria of selection for the best rule is based on number of positive cases and the fitness. The worst rules covering fewer examples will receive more computational efforts allowing a better coverage.

This strategy also makes the algorithm less dependent on the example-population ratio because it guarantees that all the examples will be selected either in the standard phase with the initial dataset, or in the final phase, with the reduced dataset.

3.7. Greedy algorithm for rule set extraction

This algorithm is used by the nodes to send their partial model to the pool and also to extract the final classifier from the central pool.

The population in any node is a redundant set of rules that does not specify how they perform the classification. The classifier must be extracted from this pool taking into account coverage, classification and simplicity. As two of them are already managed by the fitness at rule level, a derived expression gives order criteria (Π) to extract the proposed rule set:

$$\Pi : \left(1 + \frac{zeros(r)}{length(r)}\right)^{-1*Cases-} * Cases^+$$

Once selected the rule, all the positive cases of this one are removed and the rest of candidate rules are newly reordered. When no examples are uncovered the process returns the rule set. This rule set represents the current concept description in an ordered set where the first applicable rule predicts the result. This set is calculated frequently to decide if the model is more accurate than the previous one. When the model does not longer improve, it returns the last concept description extracted.

4. Experimental study

In this section we describe the experimental study developed. Subsection 4.1 give details of dataset and algorithms used for comparison. Subsection 4.2 shows the methodology followed in the experiments, subsection 4.3 shows the results, and finally, in subsection 4.3 we analyze them.

4.1 Datasets and algorithms

For the experimental study, a well known problem has been chosen from UCI [16]: Nursery. This dataset has 12.960 instances, big enough to test data distribution. Nursery is a complex dataset with 6 characteristics and 5 not balanced classes, representing three of them more than 97% of the dataset.

The algorithm chosen for comparison is REGAL. As was commented before, this algorithm is a distributed evolutionary algorithm using US. However the parallelization strategies is different than the one proposed in this work: REGAL centralizes the search

trough the direct assignation of learning examples to nodes and EDGAR let the local heuristics and the enhanced island model to drive the search.

4.2 Comparison Methodology

We have evaluated both algorithms using 5 partitions with 50% of training and testing.

The comparison will measure the quality of the classifier and speed up achieved relative to the number of processors. The first one is given by the number of rules and classification ratio. The speed up is measured trough the total execution time having the same parameters in each execution.

The comparison has been carried out with 1600 individual as sum of all local node population. This population has been proved to be big enough for both algorithms to get the maximum accuracy and lower number of rules.

Table 1: Execution parameters

	REGAL	EDGAR
Stopping criteria	500 gen.	Not improve in 5 gen.
Mutation%	0,01%	1%
Recombination%	60%	90%
Communication ratio	10%	10% max
Generation Gap	10%	20%

The experiments have been realized having configurations from 4 to 64 nodes in order to study the impact of the distribution on the referenced variables. Table 2 shows value parameters used. The stopping criteria for REGAL has been calculated experimentally using the same criteria as with the population.

4.3 Results

Tables 2 and 3 represent average on 150 executions (5 for each partition, 6 different seeds and 5 node configurations). First column is the number of nodes. Second is the execution time in minutes. Finally, third and fourth columns show the classification results for test and training dataset.

Table 2: Results of REGAL

Nodes	Time	Rules	%Test	%Training
4	1,78	290	98,6	98,9
8	1,97	251	99,0	99,3
16	2,32	250	98,9	99,2
32	2,81	268	98,5	98,7
64	3,08	316	97,9	98,0

Table 3: Results of EDGAR

Nodes	Time	Rules	%Test	%Training
4	2,89	173	99,4	99,7
8	1,59	209	98,5	98,8
16	1,20	206	98,9	99,2
32	1,11	231	98,3	98,8
64	1,21	199	98,5	98,6

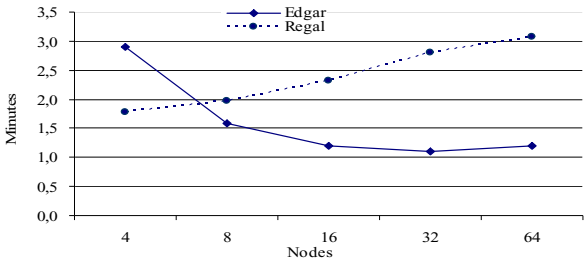

Figure 4. Compared time Nursery execution

4.4 Analysis

Analyzing tables 2 and 3, we can point out:

- The time of execution (see figure 4) of the proposed has a considerable speedup and a better behavior than the compared algorithm when the number of processors increases.
- Classification accuracy is similar in both algorithms and does not follow any tendency relative to the number of processors
- The number of rules generated is between 60% and 80% smaller in EDGAR.

5. Conclusions

This work presents an evolutionary distributed genetic algorithm for classification rules extraction based on the island model and enhanced for scalability with data training partitioning. To be able to generate an accurate classifier with data partition two techniques has been proposed: an elitist pool for rule selection and a novel technique of data distribution (DLF) that uses heuristics based on the local data to dynamically redistribute the training data in the node neighbourhood.

In this preliminary study EDGAR shows a considerable speed up and even more, this improvement does not compromised the accuracy and quality of the classifier.

Finally, we would like to remark the absence of a master process to guide the search. This architecture suggests a better scalability by avoiding idle time due

to synchronization issues or network bottleneck typically associated to master-slave relation.

6. Acknowledgements

This work has been supported by the Spanish Ministry of Education and Science under grant No. TIN2005-08386-C05-01, and the Andalusian government under grant No. P05-TIC-00531 and No. P07-TIC-03179.

7. References

[1] Giordana A, Neri F, "Search-intensive concept induction". *Evolutionary Computation*, 1995, pp. 375–416.

[2] C. Anglano, M. Botta, "NOW G-Net: Learning Classification Programs on Networks of Workstations". *IEEE Transactions on Evolutionary Computation*, October 2002, pp. 463-480.

[3] Freitas, A.A. and Lavington, S.H., *Mining Very Large Databases with Parallel Processing*, Kluwer Academic publishers, 1998.

[4] Venturini G "SIA: a supervised inductive algorithm with genetic search for learning attribute based concepts", *Proceedings of European conference on machine learning,* Vienna, 1993, pp. 280–296.

[5] Erick Cantu-Paz, *Efficient and accurate parallel genetic algorithms*, Kluwer Academic publishers, 2000.

[6] Nojima Y, Kuwajima I, Ishibuchi H, "Data set subdivision for parallel distribution implementation of genetic fuzzy rule selection", *IEEE international conference on fuzzy systems (FUZZ-IEEE'07)*, London, 2007, pp. 2006–2011.

[7] Flockhart, I.W. and Radcliffe, N.J., "GA-MINER: parallel data mining with hierarchical genetic algorithms – final report", *EPCC-AIKMS-GA-MINER Report1.0*, University of Edinburgh, UK, 1995.

[8] J.H. Holland, "Adaptation in Natural and Artificial Systems", *University of Michigan Press*, Ann Arbor, MI, 1975.

[9] De Jong KA, Spears WM, Gordon DF, "Using genetic algorithms for concept learning". *Machine Learning,* 1993 pp. 161–188.

[10] D.E. Goldberg, "Genetic algorithms in search, optimization and machine learning", *Addison-Wesley*, New York, 1989.

[13] M. J. Quinn, *Parallel Computing: theory and practice*, McGraw-Hill, 1994.

[14] C. Schaffer, "When does over fitting decrease prediction accuracy in induced decision trees and rule sets?", *Proceedings of the European Working Session on Learning (EWSL-91)*,1991, pp. 192–205.

[15] G. M. Weiss, "Learning with rare cases and small disjunts", *Proc. 12th Int. Conf. Machine Learning (ML-95)*, 1995, pp. 558-565.

[16] UCI C. J. Merz and P. M. Murphy, *UCI repository of machine learning databases*. University of California Irvine, Department of Information and Computer Science, http://kdd.ics.uci.edu, 1996.

[17] J.H. Holland and J.S. Reitman, *Cognition Systems Based on Adaptive Algorithms, in Pattern-Directed Inference Systems,* D.A. Waterman and F. Hayes-Roth, Eds., Academic Press, New York, 1978.

[18] Smith S, *A learning system based on genetic algorithms*. PhD Thesis, University of Pittsburgh, Pittsburgh, 1980.

[19] Greene DP, Smith SF, "Competition-based induction of decision models from examples", *Machine Learning, 1993,* pp. 229–257.

[20] Provost, F. and D. Hennessy, "Distributed Machine Learning: Scaling up with Coarse-grained Parallelism" *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology (ISMB-94)*, Stanford, 1994.