Paper 224-2012

Parallel Computing in SAS®: Genetic Algorithms Application

Alejandro Correa Bahnsen
Darwin Amezquita
Andres Gonzalez
Banco Colpatria, Bogotá, Colombia

ABSTRACT

Genetic Algorithms is a very powerful optimization technique that can be used in a wide variety of problems. But unfortunately the performance of this methodology relies heavily on computer power. We used Genetic Algorithms to select the architecture of a Multi-Layer Perceptron Neural Network and even though results indicated that it improves the predictive power of credit risk models, it is also very time consuming and computationally expensive. Because of this, we implemented a version of Parallel Genetic Algorithms in SAS® using PROC CONNECT procedure. Results show that parallel computing can drastically reduce the total execution time of the genetic algorithm.

INTRODUCTION

In previous works Correa, Gonzalez and Ladino (2011) and Correa and Gonzalez (2011) proposed the use of Genetic Algorithms (GA) in order to select the architecture of a Multi-Layer Perceptron Neural Network (MLP) in a credit scoring context. The idea behind those works is to codify the architecture of an MLP in a set of bits, called a chromosome, and then apply a GA and other algorithms to optimize that chromosome in order to maximize a defined cost function, in this case the predictive power of the scorecard is measured using the ROC statistic.

The resulting GA-MLP framework has shown to have a better predictive power than a MLP using the SAS default parameters and the traditional Logistic Regression. Unfortunately it takes too much time for the GA to converge. This is because the algorithm must calculate at each iteration, a large number of different MLP Neural Networks. In this paper, we propose to decrease the time of convergence of the GA using the PROC CONNECT features, which allows to split one process into different sub-processes that can be calculated in parallel and therefore make a better usage of the computational resources.

The GA can be parallelized by splitting in different sub-processes the cost calculation step for each chromosome. That is because the calculation of each MLP Neural Network can be done separately. The Parallel Genetic Algorithm (PGA) was run using a 16 processor Unix Server, which means that the GA can be split up to 16 sub processes. In order to make a comparison between the times of convergence of the GA versus the number of processors used, the algorithm was run using a different amount of processors.

This paper is divided into four sections. First, an introduction to general concepts of the GA, the theory behind the PGA, and SAS PROC CONNECT is done; then, second section explain the methodology using to parallelize GA; subsequently, we illustrate the experimental results of the algorithm and finally the conclusions are presented.

GENERAL CONCEPTS

- ### SAS PROC CONNECT

  The CONNECT procedure is one of the ways that a local computer can connect to a server when both have SAS installed. There are several advantages of having this type of structure, being the most important the scalability of the system, the ability to allow many users to connect to it, and the no need of any of the users to have a high performance computer, because all the processes are computed on the server (Buchecker, 2010).

  The standard way that the PROC CONNECT works is by sending instructions from the local computers to the server using the rsubmit code.

```
rsubmit server1;
        /* Instructions…. */
endrsubmit;
```

One issue with this code is that the local computer will stand by until the server ends its processes. The way to avoid this waiting time is to modify the code by setting the option wait to no. Using this code we are able to submit different codes that will be processed at the server simultaneously if there are available resources to do so.

```
rsubmit server1 wait=no;
                /* Instructions…. */
endrsubmit;

rsubmit server2 wait=no;
                /* Instructions…. */
endrsubmit;

rsubmit server3 wait=no;
                /* Instructions…. */
endrsubmit;
```

Lastly, one finally code line should be added, that will tell the local computer to wait until all the processes are finish.

```
waitfor server1 server2 server3;
```

- **GENETIC ALGORITHM**

As said in Haupt and Haupt (2004), a GA is an optimization technique that attempts to replicate natural evolution processes in which the individuals with the considered best characteristics to adapt to the environment are more likely to reproduce and survive. These advantageous individuals mate between them, producing descendants similarly characterized, therefore favorable characteristics are preserved and unfavorable ones destroyed, leading to the progressive evolution of species. GA is an optimization algorithm that can be applied to a wide range of problems. The flow diagram presented in Fig 1. describes the process. For further information please refer to Correa, Gonzalez and Ladino (2011).

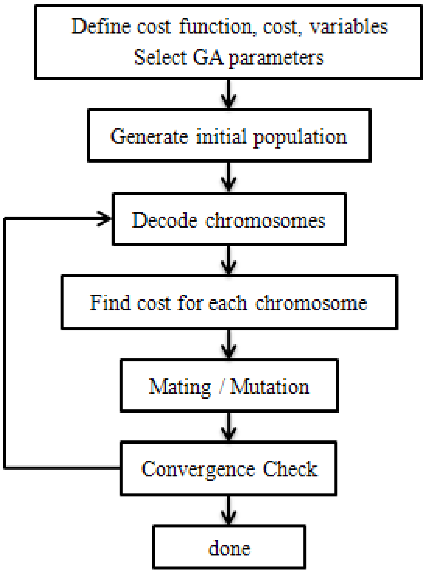
Define cost function, cost, variables
Select GA parameters
Generate initial population
Decode chromosomes
Find cost for each chromosome
Mating / Mutation
Convergence Check
done

Figure1. GA Flow Diagram

- **PARALLEL GENETIC ALGORITHM**

Parallel genetic algorithms are modifications made to the genetic algorithms in order to reduce the time consumption, making them more efficient (Nowostawski and Poli, 1999). Because GA is a serial algorithm it doesn't used the full computational resources available in a multi core computer, the PGA attempts to improve this weakness by paralyzing the GA process.

To parallelize a GA the process is sectioned, attempting to identify the steps where parallelization can be inserted, and split a long serial step into shorter sub-process. The applicability and the profit of make this kind of modifications to original genetic algorithms will depend of GA structure, limitations come from the need of results of one steps in the next one.

METHODOLOGY

As shown by Nowostawski and Poli (1999), there are several ways of parallelize a GA, for this specific case a Master-Slave parallelization with synchronous cost evaluation is used. This translates in creating t sub-groups at the moment of the neural network estimation and cost calculation, and calculates all sub-groups at the same time. This can be done because when calculating the cost function the only information the algorithm needs is the chromosome, so the cost function of different chromosomes can be calculated at the same time. Fig 3. Shows the PGA process.

This algorithm present a limitation in the parallelization, it must wait until all the subgroups cost functions are calculated in order to move to the next step. As a consequence, the time of the algorithm is going to be the time of the slowest sub-group cost function.

As shown in Fig 2. and Fig 3., the modification made to GA Flow diagram doesn't affect the results of the process; the profit must be quantified in time saving, and to be more accurate, that time saving must be relativized with initial time spent by the GA.

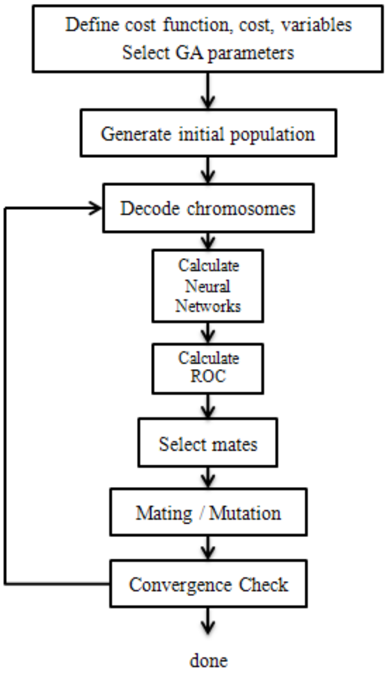

Figure 2. GA Flow Diagram

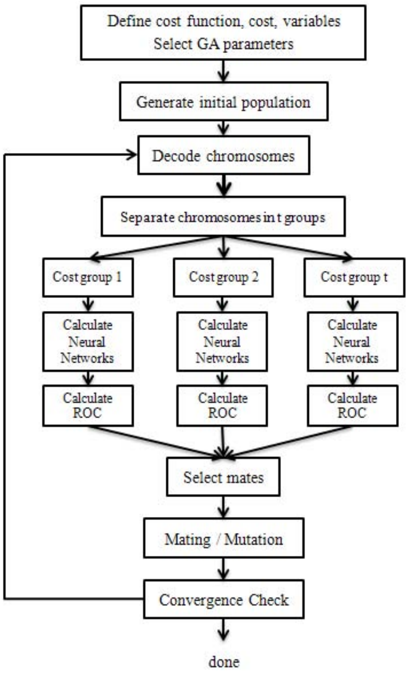
Define cost function, cost, variables
Select GA parameters
Generate initial population
Decode chromosomes
Separate chromosomes in t groups
Cost group 1
Cost group 2
Cost group t
Calculate Neural Networks
Calculate Neural Networks
Calculate Neural Networks
Calculate ROC
Calculate ROC
Calculate ROC
Select mates
Mating / Mutation
Convergence Check
done

Figure 3. PGA Flor Diagram

RESULTS

In this section we present the experimental results. For the purpose of testing the parallel algorithm we use the same data set used by Correa and Gonzalez (2011). This particular data set contains information of 125.557 clients of a financial institution and it was used to develop a credit risk scorecard using a MLP. Correa and Gonzalez demostrated that using a GA to select the architecture of a MLP improves the predictive power of the scorecard but calculating the GA takes significantly more time than using the MLP with the default parameters of SAS® Enterprise Miner.

Using the capabilities of SAS® PROC CONNECT we calculate the PGA using 2, 4, 8 and 16 CPU's. Results are shown in Fig 4. Also, Table 1. show the predictive power of the MLP that each PGA selects. Is important to note that all the PGA's reach the same MLP architecture.

Using a simple regression the time saving when using n CPU's can be express as: $Time(n) \approx \frac{Time(1)}{n}$.

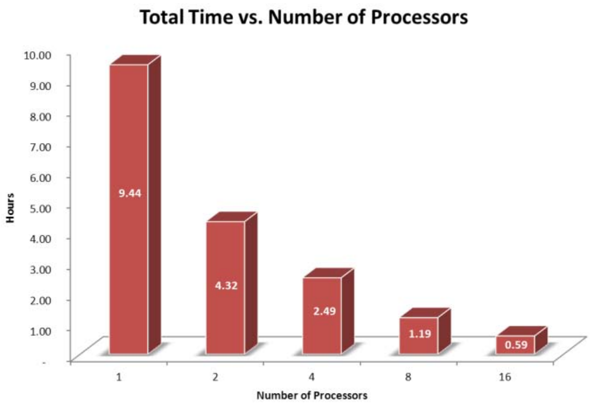

Figure 4. Total Time vs. Number of CPU's.

Number of CPU's	Time	Predictive Power
1	9:26:11	71.25%
2	4:19:17	71.25%
4	2:29:32	71.25%
8	1:11:35	71.25%
16	0:35:24	71.25%

Table 1. Total Time and Predictive Power

CONCLUSION

The experimental results have shown that using PGA to optimize the architecture of a MLP neural network, reach the same results like the serial GA, but the time spend is reduced approximately by a factor of n, where n is the number of CPU's used. This is particularly helpful because the spent time is reduced from 9.4 to 0.6 hours, which represents a reduction of around 94%.

The results of GA will depend of the parameters selection such as the mutation rate, the elite number, or the type of mating; time saving due to parallelizing allows testing a broader number of parameters maximizing the probability of arriving to the global optimum.

It is also important to note that there's still room for testing different parallelized versions of the GA. In fact, the one used in this paper, the Master-Slave with synchronous cost function evaluation, have the limitation that the total time of one iteration is the time of the slowest sub-group, and that lead to an inefficiency.

Finally, although the PGA outperforms the GA, this could be improved by also parallelizing the MLP estimation.

REFERENCES

- Correa, A. Gonzalez, C. Ladino. Genetic Algorithm Optimization for Selecting the Best Architecture of a Multi-Layer Perceptron Neural Network: A Credit Scoring Case. SAS Global Forum, 2011.
- Correa, A. Gonzalez. Evolutionary algorithms for selecting the architecture of a MLP Neural Network: A Credit Scoring Case. Fourth Workshop on Data Mining Case Studies and Practice Prize, IEEE International Conference in Data Mining, 2011.
- Buchecker, M. Michelle. Parallel Processing Hands-On Workshop. SUGI29. 2010.
- Mariusz, Nowostawski. Ricardo, Poli. Parallel Genetic Algorithm Taxonomy, KES'99, May 1999.
- R. Haupt, S. Haupt. Practical Genetic Algorithms, second edition. John Wiley & Sons. New York. 2004.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Name: Alejandro Correa Bahnsen
Enterprise: Banco Colpatria
City: Bogotá, Colombia
Phone: (+57) 3208306606
E-mail: al.bahnsen@gmail.com

Name: Andres Gonzalez
Enterprise: Banco Colpatria
City: Bogotá, Colombia
Phone: (+57) 3103595239
E-mail: correaal@colpatria.com

Name: Darwin Amezquita
Enterprise: Banco Colpatria
City: Bogotá, Colombia
Phone: (+57) 3013372763
E-mail: amezqud@colpatria.com