Variable Selection in a GPU Cluster Using Delta Test

A. Guillén[1], M. van Heeswijk[2], D. Sovilj[2], M.G. Arenas[1], L.J. Herrera, H. Pomares[1], and I. Rojas[1]

[1] Department of Computer Architecture and Computer Technology Universidad de Granada, Spain
[2] Department of Information and Computer Science, Aalto University School of Science, Finland

Abstract. The work presented in this paper consists in an adaptation of a Genetic Algorithm (GA) to perform variable selection in an heterogeneous cluster where the nodes are themselves clusters of GPUs. Due to this heterogeneity, several mechanisms to perform a load balance will be discussed as well as the optimization of the fitness function to take advantage of the GPUs available. The algorithm will be compared with previous parallel implementations analysing the advantages and disadvantages of the approach, showing that for large data sets, the proposed approach is the only one that can provide a solution.

1 Introduction

The problem of variable selection is crucial by the time of design models that classify or perform regression so, the better the selection is, the more accurate models can be designed [3,13]. The approach to be taken in classification problems differs from the one taken in regression due to their differences in the output type (discrete and continuous respectively) so, this paper will consider the regression problem also known as function approximation. Formally, the function approximation problem is to determine, given a set of input/output pairs $(\mathbf{x}_i, y_i) \in R^d \times R \;\; i = 1...N$, design an unknown function F such as $F(\boldsymbol{x}_i) \approx y_i$. From this, the problem of variable selection can be defined as the search for the subset of variables that make possible to build a model that approximates the data as accurately as possible. Genetic Algorithms have been applied to many problems and variable selection is not an exception, however, the use of non-parametric noise estimators (independent from the models designed afterwards) has not been widely treated [11]. This paper presents a GA which has been implemented on a novel High Performance Computing (HPC) architecture using clusters of computers and Graphical Processing Units (GPUs) in order to compute the fitness of the individuals.

The rest of the paper is organised as follows: Section 2 introduces the Delta Test. Afterwards, Section 3 describes the design of the Parallel Genetic Algorithm that will perform the optimizations presented in the experiments included in Section 4. Finally, conclusions are discussed.

2 Delta Test in Variable Selection

In order to evaluate the goodness of an individual, the Delta Test (DT) [15] value obtained using the combination of variables will be used as it has been shown to be an adequate criterium [5]. The DT is a method to estimate the variance of the noise r_i, or the Mean Squared Error (MSE), that can be obtained without overfitting, this is:

$$y_i = F(\mathbf{x}_i) + r_i, \qquad i = 1, ..., N$$

where F is the unknown function. The DT can be formulated using the nearest neighbour formulation as

$$\mathrm{Var}[r] \approx \delta = \frac{1}{2N} \sum_{i=1}^{N} (y_i - y_{NN(i)})^2,$$
$$\text{with } \mathrm{Var}[\delta] \to 0 \text{ for } N \to \infty$$

where the first nearest neighbour of a point $\mathbf{x}_i$ in the R^d space is $\mathbf{x}_{NN(i)}$ and $y_{NN(i)}$ is the output of $\mathbf{x}_{NN(i)}$.

2.1 Computation of Delta Test Using Pre-calculated Distances

Computation of the nearest neighbour in the naive way involves calculating the distances between each pair of samples $\mathbf{d}_{i,j}2 = \sum_{m=1}^{d} (x_i^{(m)} - x_j^{(m)})2$ and returning the smallest $\mathbf{d}_{i,j}$ and the corresponding index $NN(i)$ for each sample. Since the focus is on examining non-empty subsets of variables which can share individual elements, a lot of time is wasted recomputing the squared differences to obtain $\mathbf{d}_{i,j}$. A simple solution to decrease running time is to store that information into a $N(N-1)/2 \times d$ matrix, where each row contains precomputed squared differences for a pair of samples (x_i, x_j). Given this matrix, computing all pairwise distances for a given variable subset $I \subseteq \{1, 2, \ldots, d\}$ involves summing precomputed values for those I variables (i.e. the I-th columns of the matrix).

2.2 Computation of Delta Test on GPU

The computation of the k Nearest Neighbours (KNN) requires a big computational effort since it has to compute the pairwise distances between all the points. In [6] an implementation of the k-NN algorithm on a GPU[1] was presented, showing very large speed-ups compared to CPU times. We use this algorithm to determine the nearest neighbours ($\mathbf{x}_{NN(i)}$) to all input points ($\mathbf{x}_i$).

Differently from the approach in the previous subsection, the pairwise squared differences between all points are not pre-calculated, since it would not be feasible to keep this entire matrix in memory. However, even though we do not make

[1] The code is available at: http://www.i3s.unice.fr/∼creative/KNN/

this optimization, computing the pairwise distances between the points can still be many times faster when using a fast GPU instead of the CPU.

Once all pairwise distances have been computed, a partial sort is performed in order to determine the nearest neighbour $\mathbf{x}_{NN(i)}$ (and their index $NN(i)$) to each of points $\mathbf{x}_i$. Finally, given these indices of the nearest neighbours, we can compute the Delta Test as explained in the beginning of this section.

3 Design of the Genetic Algorithm

The Genetic Algorithm (GA) is a well known optimization tool that has been applied to many problems.

3.1 GA Operators Description

When a GA is designed, there is a set of elements/operators that has to be defined depending on the problem to be solved. The first design decision is how an individual will represent a solution, this is, the solution encoding. The variable selection problem has a straight forward encoding using a binary chromosome whose length is equal to the number of variables. If a gene within the chromosome equals 1, the variable is selected, if it is 0, then the variable is discarded. This binary encoding has been widely treated in the literature so the classical, but effective, operators have been chosen:

1. Crossover: two-points binary
2. Mutation: Gene level, this is, if the individual is mutating select-unselect a random variable.
3. Selection: binary tournament selection
4. Elitism: Keep the best individuals from the previous generation.

3.2 Parallelizing the GA

Apart from the optimization of the computation of the fitness function using GPUs, the architecture available allows the algorithm to be distributed through several machines in a classical cluster manner. GAs are intrinsically parallel although modifications in the flowchart, distributing the populations, might improve results [2,9,10,8].

Island Model. Among the several approaches proposed in the literature, the multi-deme distributed GA is one of the most popular due to its good behaviour [4,8,12]. This implementation consists of evolving isolated populations on each island which is usually mapped into one processor. In our concrete case, since a processor might have several cores and several GPUs it might allocate several populations, one per each CPU/GPU pair.

In this type of algorithm, it is quite common to determine a migration operator or mechanism so the isolated islands share some individuals in order to collaborate and take advantage of the progress made by the others.

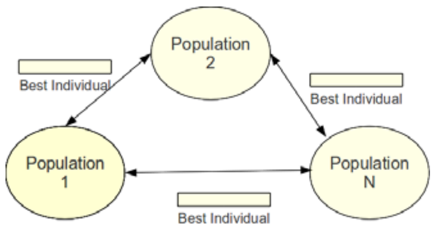

Fig. 1. Island migration scheme

The migration rate is a parameter that can be random [14], fixed [8], or autoregulated depending on the diversity of the population although this last approach can be looked from the perspective of the replacement policy, this is, migrate after a certain number of generations and, depending on the diversity of the population, accept the new individual.

The implementation selected in this algorithm was to migrate after 5 generations and always replaces the worst individual in the population by the incoming ones (in the same order they arrive form the other islands) as in [9]. The migration scheme is depicted in 1.

Population distribution in the cluster. Since we have a heterogeneous grid of computers, it is obvious that the time to complete a generation during the run will be different on each computer [1]. As the distributed populations perform a collective communication broadcasting individuals, the global performance of the algorithm would be injured if the fastest machines had to wait for the slower ones, wasting computing resources. In order to ameliorate this fact, the decision of setting different population sizes has been taken. Slower or overloaded machines will process less individuals, allowing these processes to require a smaller time to complete a generation. Therefore, during the collective communications, the waiting time for the synchronization will be reduced. Obviously, this approach is the same as increasing the predefined population size in the more powerful machines.

The question that arises from this policy is: how much the size of the population should be decreased/increased? The answer can be obtained empirically by measuring the time of one generation on each machine and obtaining the fraction between the fastest/slowest and the other time measurements. For example, if the time on $Machine_1$ is the double of $Machine_2$ the population size for $Machine_1$ should be the half (or the population size for $Machine_2$ should be doubled).

4 Experiments

4.1 Cluster Architecture

The cluster that was configured had the components described below that were interconnected as Figure 2 shows.

1. **1 Master node with 2 GPUs:**
   Processor:
   - *model name* : (26) Intel(R) Core(TM) i7 CPU 930 @ 2.80GHz — *cache size* : 8192 KB
   - *cpu cores* : 4 - *siblings* : 8

   2 GPUs:
   - *Graphics Processor*:GeForce GTS 450 — *CUDA Cores*: 192
   - *Memory*: 1024 MB - *Memory Interface*: 128-bit
   - *Bus Type*: PCIExpress x16 Gen1 - *PCI-E Max Link Speed*: 2500
2. **2 Local network node with 1 GPU:**
   Processor:
   - *model name* : (23) Intel(R) Core(TM)2 Quad CPU Q9550 @ 2.83GHz — *cache size* : 6144 KB
   - *cpu cores* : 4 - siblings : 4

   GPU:
   - *Graphics Processor*:GeForce 9800 GTX — *CUDA Cores*: 128
   - *Memory*: 512 MB - *Memory Interface*: 256-bit
   - *Bus Type*: PCIExpress x16 Gen2 - *PCI-E Max Link Speed*: 5000

   Processor:
   - *model name* : (15) Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz — *cache size* : 4096 KB
   - *cpu cores* : 4 - siblings : 4

   GPU:
   - *Graphics Processor*: GeForce 8400 GS — *CUDA Cores*: 16
   - *Memory*: 512 MB - *Memory Interface*: 64-bit
   - *Bus Type*: PCIExpress x16 - *PCI-E Max Link Speed*: not available

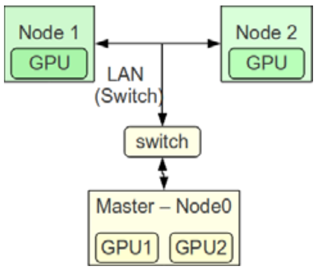

Fig. 2. Cluster of GPUs used in the experiments

4.2 Comparison with Previous Approaches

This section will analyse the performance and behaviour of the proposed algorithm. First, small data sets are compared with a previous work, afterwards, the algorithm is applied to a large real-world data set.

Small Datasets. The algorithm was compared with the approaches in [11] that was a previous parallel version of the algorithm implemented in a regular cluster. For the sake of a fair comparison, the stop criteria will be the same: execution time. In [11], the time limit is set to 600 seconds based on studies and recommendations from the industry arguing that is the maximum time that an operator is willing to wait to see a solution.

The same population sizes were used (50,100 and 150) and data sets processed were:1) The Tecator data set[2]: the Tecator data set aims at performing the task

[2] http://lib.stat.cmu.edu/data sets/tecator.

of predicting the fat content of a meat sample on the basis of its near infrared absorbance spectrum. The data set contains 215 useful instances for interpolation problems, with 100 input channels, and 3 outputs, although only one is going to be used (fat content). 2) The Anthrokids modified data set[3]: This data set consists on several measures to predict child's weight, it has 1019 instances and 53 variables.

The results are shown in Table 1. As the table reflects, the performance of the proposed algorithm is not impressive in comparison with the previous algorithm. In fact, it does not outperform the previous approach. However, it is remarkable that the difference between the solutions is not too big and, in some cases, the new algorithm outperforms the previous one demonstrating that the proposed approach is a good algorithm. The reason because the previous algorithm obtains, in general, better solutions for these data sets is because it is able to perform more generations due to the pre-calculation of the distance matrix as it was described in Section 2.

Table 1. Performance of the proposed algorithm (pGPU) against sequential and parallel algorithms for different data sets. Values of the DT obtained.

Data set	Population	Measurement	seq.	parallel(np=2)	parallel(np=4)	pGPU(np=4,4GPUs)
Anthrokids	50	Mean (DT)	0.01278 (11.5e-4)	0.01269 (14.2e-4)	0.01204 (12.6e-4)	0.01587 (8.1e-3)
	100	Mean (DT)	0.01351 (11.6e-4)	0.01266 (86.4e-4)	0.01202 (17.4e-4)	0.014553 (5.6e-4)
	150	Mean (DT)	0.01475 (12.1e-4)	0.01318 (11.2e-4)	0.01148 (9.9e-4)	0.01556(12e-4)
Tecator	50	Mean (DT)	0.13158(7.9e-4)	0.14297 (7.7e-3)	0.13976 (7.8e-3)	0.123803 (3.7e-3)
	100	Mean (DT)	0.13321 (3.1e-3)	0.13587 (2.4e-3)	0.13914 (8.6e-3)	0.132501 (3e-4)
	150	Mean (DT)	0.13146 (8.5e-4)	0.1345 (2.4e-3)	0.13522 (6.9e-3)	0.13197 (9.9e-4)

Large Datasets. The analysis of the previous results might discourage the use of GPUs instead of using a pre-calculation of the distances, however, when the data set starts becoming a little bit bigger, this second approach is not possible any more. The reason is because the application runs out of memory so, to use a cluster of GPUs it is not a matter of performance in time or quality results, it is a matter of being able to provide a solution.

As an example, it will be used part of the data set provided by the Spanish Institute of Statistics (Instituto Nacional de Estadística, INE) that contains data about marital dissolutions in Spain. Several problems arise from this data, and one of them is predicting the dissolution process length which is translated into the regression problem.

The data to be used consists of 19967 input samples of 20 variables which was divided into training (of 15385) a test (4568) sets. The size of the training set is too big to pre-calculate the distance matrix used in previous approaches [11].

The proposed approach was executed during 600 seconds and was able to finish only one generation with 50 individuals providing a DT value of 0.001642 using 9 variables. Due to the high memory requirements, one of the computers with the

[3] http://research.ics.tkk.fi/eiml/datasets.shtml

oldest GPU model was delaying the other two because of the synchronization step in the migration. Therefore, the experiments were repeated only with two nodes instead of three. The performance was increased significantly, allowing the algorithm to evolve a mean of 7.6 generations with a DT value of 0.001589 using only 4 variables.

Table 2. A) Delta test values for the large size data set ; B) Approximation errors (NRMSE) of the large data set with and without variable selection using an RBFNN with 15 neurons

A)

Running Time	DT value (std)	# vars	# generations (std)
600 secs (3 nodes, 4 GPUs)	0.001629 (2e-4)	9	1 (0)
600 secs (2 nodes, 3 GPUs)	0.001592 (1e-4)	4	7.6 (0.5)

B)

	Train Error	Test Error
with var. selec.	0.4846	0.5017
without var. selec.	1.3086	1.3197

To test the validity of the selection provided, a model (Radial Basis Function Neural Network with 15 neurons) was designed using the methodology proposed in [7] without local search optimization. The experiments were done using all the variables and using the ones selected by the algorithm, obtaining the results shown in Table 2 B).

5 Conclusions

The problem of variable selection remains as an unsolved problem, being a crucial step before designing the models to classify or approximate. In this paper, a new algorithm that takes advantage of the new High Performance Computing technologies has been presented. Concretely, the main novelty is the use of a cluster where the nodes have graphical processing units to compute the k-Nearest Neighbours. The performance of the algorithm for small data sets is acceptable when compared with previous methods plus it is the only one of its kind that is able to provide good results in a reasonable time, when the size of the dataset becomes large.

Acknowledgments. This research has been supported by the projects by the Spanish CICYT Project TIN2007-60587 and TEC2008-04920 and Junta Andalucia Projects P08-TIC-03674 and P08-TIC03928 and PYR-2010-17 of CEI BioTIC GENIL (CEB09-0010) of the MICINN.

References

1. Alba, E., Nebro, A.J., Troya, J.M.: Heterogeneous computing and parallel genetic algorithms. Journal of Parallel and Distributed Computing 62, 1362–1385 (2002)
2. Alba, E., Tomassini, M.: Parallelism and evolutionary algorithms. IEEE Trans. on Evolutionary Computation 6(5), 443–462 (2002)

3. Bellman, R.E.: Adaptive control processes - A guided tour. Princeton University Press, Princeton (1961)
4. Cantú-Paz, E.: Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Massachusetts (2000)
5. Eirola, E., Liitiäinen, E., Lendasse, A., Corona, F., Verleysen, M.: Using the delta test for variable selection. In: ESANN 2008: European Symposium on Artificial Neural Networks, Bruges, Belgium, pp. 25–30 (April 2008)
6. Garcia, V., Debreuve, E., Barlaud, M.: Fast k nearest neighbor search using GPU. In: CVPR Workshop on Computer Vision on GPU (2008)
7. Guillén, A., González, J., Rojas, I., Pomares, H., Herrera, L.J., Valenzuela, O., Rojas, F.: Output Value-Based Initialization For Radial Basis Function Neural Networks. Neural Processing Letters (June 2007), doi:10.1007/s11063-007-9039-8
8. Guillén, A., Rojas, I., González, J., Pomares, H., Herrera, L.J., Paechter, B.: Improving the Performance of Multi-objective Genetic Algorithm for Function Approximation Through Parallel Islands Specialisation. In: Sattar, A., Kang, B.-h. (eds.) AI 2006. LNCS (LNAI), vol. 4304, pp. 1127–1132. Springer, Heidelberg (2006)
9. Guillén, A., Rojas, I., González, J., Pomares, H., Herrera, L.J., Paechter, B.: Boosting the performance of a multiobjective algorithm to design rBFNNs through parallelization. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds.) ICANNGA 2007. LNCS, vol. 4431, pp. 85–92. Springer, Heidelberg (2007)
10. Guillén, A., Pomares, H., González, J., Rojas, I., Valenzuela, O., Prieto, B.: Parallel multiobjective memetic rbfnns design and feature selection for function approximation problems. Neurocomputing 72(16-18), 3541–3555 (2009)
11. Guillen, A., Sovilj, D., Lendasse, A., Mateo, F., Rojas, I.: Minimising the delta test for variable selection in regression problems. Int. J. High Perform. Syst. Archit. 1, 269–281 (2008)
12. Herrera, F., Lozano, M.: Gradual distributed real-coded genetic algorithms. IEEE Transactions on Evolutionary Computation 4(1), 43 (2000)
13. Herrera, L.J., Pomares, H., Rojas, I., Verleysen, M., Guilén, A.: Effective input variable selection for function approximation. In: Kollias, S.D., Stafylopatis, A., Duch, W., Oja, E. (eds.) ICANN 2006. LNCS, vol. 4131, pp. 41–50. Springer, Heidelberg (2006)
14. Hiroyasu, T., Miki, M., Negami, M.: Distributed genetic algorithms with randomized migration rate. In: Proceedings of the IEEE Conf. Systems, Man and Cybernetics, pp. 689–694 (1999)
15. Pi, H., Peterson, C.: Finding the embedding dimension and variable dependencies in time series. Neural Computation 6(3), 509–520 (1994)