Parallelization of filter BSS/WSS on GPGPU for classifying cancer subtypes with SVM

Iliana Castro-Liera, J. Enrique Luna-Taylor, Jacob E. Merecías-Pérez, Germán Meléndrez-Carballo
Departamento de Sistemas y Computación, Instituto Tecnológico de La Paz,
Boulevard Forjadores No. 4720, La Paz, B.C.S., México;
icastro@itlp.edu.mx, eluna@itlp.edu.mx

Abstract

A parallel version of BSS/WSS (Between Sum Square-BSS, Within Sum Square-WSS) filter using GPGPU (General Purpose Graphics Processing Units) is proposed. The application processes genetic expression data to select statistically relevant genes. A SVM (Support Vector Machine) is applied on the expression data of genes selected by the filter, to diagnose cancer. With the use of this combination of algorithms, a success rate of 92% in the diagnosis was achieved. With the parallel implementation on GPGPU with CUDA technology, a dramatic reduction of execution time of approximately 18 times compared to the sequential implementation on CPU was achieved.

Keywords: genetic expression, Support Vector Machine, parallel programming, GPGPU, CUDA.

I. INTRODUCTION

Cancer is a serious and complex problem that has caught the attention of the scientific community and is among the 10 main causes of death in the world [1]. Among the many fronts and efforts that are being made to address this problem, exists the development of algorithms that aim predict and classify different types of cancers using gene expression data. One of the main technologies that have provided large volumes of biological data are the DNA microarrays [2]. The datasets, on which the classification algorithms work, contain the expression levels of thousands of genes from a group of patients. For instance, the dataset for the analysis of Leukemia cancer [3], consists of expression values of 7129 genes for 72 patients. This dataset is divided in two groups according to the specific subtype of Leukemia cancer: AML (Acute Myeloid Leukemia) and ALL (Acute Lymphoblastic Leukemia). Within the group of patients with AML subtype, there is another division corresponding to subgroups of patients that were successful in the treatment and those that were not successful in the treatment. With these data, a classification algorithm has as goals either diagnoses the specific subtype of Leukemia, or predict the likelihood of success of treatment.

Previous works for the classification and prognosis of cancer have been done. Golub et al. [3], use the *"neighborhood analysis"* method for the distinction of classes, grouping the genes with similar expression patterns. Next, using the levels of expression, and the degree of correlation of genes, a method is applied for each gene to assign a score associated with each of the classes. With the sum of the scores, the winning class is determined. The results of this method are validated through a "cross-fold." For the classification experiments they used 50 *"informative genes"* from Leukemia cancer dataset, obtained these previously, achieving a total of 36 correct predictions from 38 patients. Nasimeh and Russel [4], use an algorithm to identify biclusters, based on the mathematical method called *"rank-1 matrix approximation"*, which is applied to reduce the size of the dataset. Each bicluster is a subset of genes and a subset of patients. The genes in a bicluster have expression values which are correlated. Based on the discovered biclusters, a Support Vector Machine is used to perform the classification. The Leukemia, Colon [5], Prostate [6], and Lung cancer datasets [7] were used to the diagnosis experiments, achieving a success rate of 84.72% for the Leukemia cancer dataset. The runtime of the method was approximately of 1 minute for obtaining each bicluster, from a data matrix of around 20,000 genes of 100 patients. Hernández et al. [8], use three filters to reduce the number of genes used in the classification. The filters are: BSS/WSS, T-Statistic and Wilcoxon. After applying the filters, the subset of genes obtained is passed to a genetic algorithm (GA), which with the help of an SVM, performs the final selection of genes to be used in the classification. The Leukemia and Colon cancer datasets were used for the experiments. With the Leukemia dataset, a success rate of 98.61% was obtained. It is unclear if in the final experiment performed, were used patients' data that were also used in the filtering phase. None of the methods mentioned above, have a parallel computing technique.

Our proposed method uses two algorithms to deal with the problem of classification and prognosis of cancer. The first is the BSS/WSS filter [9], which was used to select the most relevant genes for classification. The second algorithm is a Support Vector Machine [10], to perform the classification based on selected genes by the filter. Due SVM is unable to perform a correct classification when the number of genes is greater than the number of patients [11], it is necessary to reduce the number of genes before applying the SVM for classification or prognosis of cancer.

In the experiments performed, a success rate of 92% on the Leukemia cancer dataset was obtained. Due to high cost of time that this kind of analysis involves on large volumes of data, a parallel BSS/WSS filter implementation was done. Using CUDA technology on a GNU/Linux Fedora environment with CUDA C, a reduction of the execution time of approximately 18 times was achieved, with respect to a sequential implementation on a CPU with the C programming language. Both implementations of the filter were performed 1000 times using different sets of patients in each iteration.

II. DEVELOPMENT

A. BSS/WSS Filter

It has been shown that the use of BSS/WSS filter, is a useful method in the classification and prediction of cancer [9]. The function of this filter is to identify, statistically, those genes that behave differently between groups of patients of different classes.

In the filter, the selection of genes is based on the ratio of the sums of squares differences between groups (BSS), and within groups (WSS), calculated for each gene j. The filter's formula is shown in (1).

$$\frac{BSS(j)}{WSS(j)} = \frac{\Sigma_i \Sigma_k I(y_i=k)(\bar{x}_{kj}-\bar{x}_j)^2}{\Sigma_i \Sigma_k I(y_i=k)(x_{ij}-\bar{x}_{kj})^2} \quad (1)$$

Where y_i represents the class of the subtype of cancer for the ith patient. $\bar{x}_j$ represents the mean expression level of gene j for all patients. $\bar{x}_{kj}$ represents the mean of expression level of gen j for all patients that belong to the class k. x_{ij} represents the expression level of gene j for the patient i. The function $I(y_i = k)$ return 1 if the class of patient i is equal to the class k, and 0 otherwise.

Since our problem is to identify only two classes, and we know in advance the range of indexes of patients of both classes, we can omit the function $I(y_i = k)$ from (1), as shown on (2).

$$\frac{BSS(j)}{WSS(j)} = \frac{\Sigma_i \Sigma_k (\bar{x}_{kj}-\bar{x}_j)^2}{\Sigma_i \Sigma_k (x_{ij}-\bar{x}_{kj})^2} \quad (2)$$

B. Filter implementation in CUDA

Since the formula of BSS/WSS filter is applied independently for each gene, the parallel implementation was designed so that each gene is processed by a thread. Due that the CUDA device must process 7129 genes, the algorithm was designed to run on 14 thread´s blocks of 512 threads each one, giving a total of 7168 threads, where the last 30 threads are not used.

Each thread has its own ID, whose value is calculated as shown in (3).

$$tid = (BLK\_DIM * BLK\_ID) + THR\_ID \quad (3)$$

Where BLK_DIM represents the block´s dimension, BLK_ID is the ID of the thread´s block and THR_ID is the ID of the set of threads.

The Leukemia cancer dataset was loaded into memory in a one-dimensional array, including the class labels for each patient. Figure 1 shows the representation of the data in a one-dimensional array.

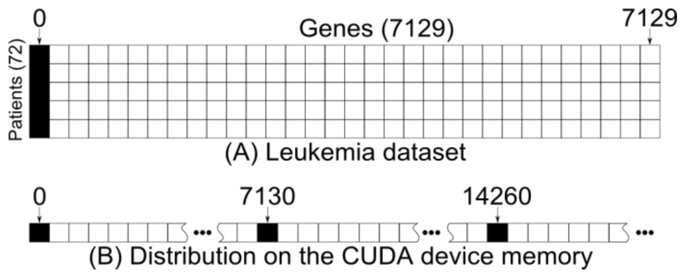

Figure 1. Representation of the Leukemia dataset. The black cells contain the labels of the patients, and white cells contain the expression levels of the genes. A) The two-dimensional Leukemia array representation. B) The distribution of the Leukemia dataset on the CUDA device memory, where each segment of 7130 cells correspond to the label and genes of a patient.

In the same way, the indexes of the training patients, used in each iteration, were stored in a one-dimensional array. The size of this array is equal to the number of patients multiplied by the number of iterations to be executed. The formula used to get the index of a patient, for a given iteration, is shown in (4).

$$rowlabel = (k * patients\_length) + i \tag{4}$$

Where k is the number of iteration, i is the number of patient, and $patients\_length$ is the number of patients used for each iteration of the filter. The formula for get the index of the expression level of a patient is shown in (5).

$$rowdata = ((rowlabel - 1) * COLS) + tid \tag{5}$$

Where $COLS$ is the total of columns that has the dataset, that is, the number of genes, plus one, for the label. A one-dimensional array to store the results was created. The size of the array is equal to the number of genes plus one, multiplied by the number of iterations to be executed. The formula for get the index to store a result is given in (6).

$$(num\_iter * COLS) + tid \tag{6}$$

Where num_iter is the number of the iteration. The results are real numbers that vary in the range from 0 to 3 for Leukemia cancer dataset. A result of 0, or close to 0, is considered a poor value, and is associated to a gene irrelevant for classification. A value close to 3 is considered as good, and its gene associated is relevant for the classification.

The Algorithm 1, shows the pseudocode of the CUDA kernel.

Each thread j starts getting the mean of the expression levels of its associated gene for all patients of each class. The mean of expression levels for all patients is obtained too. After, the square of the mean of each class minus the mean of gene j is obtained, and is multiplied by the number of patients in the corresponding class. These squares are summed and divided by the following summation. A summation of the squared, of the expression level of the gene j for the patient i minus the mean of the class of such patient, is performed.

Algorithm 1: BSS/WSS CUDA implementation

```
Input: one-dimensional array of gene expression, AML patients indexes, ALL patients indexes,  # of iterations
Output: one-dimensional array with the scores obtained by the filter for each gene
1.  allocate memory on GPU and CPU for the one-dimensional array of gene expression, AML/ALL patients indexes and one-dimensional array for the scores
2.  fill the one-dimensional array of gene expression with the data of the Leukemia cancer dataset
3.  fill the AML/ALL arrays with the round-cross method
4.  copy the one-dimensional array of gene expression and AML/ALL patients indexes from CPU to GPU
5.  invoke the CUDA kernel for perform the filter (Algorithm 2).
6.  copy back the one-dimensional array with the scores from GPU to CPU
7.  free both memories, GPU and CPU
```

C. Support Vector Machine

The Support Vector Machine (SVM) was introduced in 1992 [10], and is widely used in classification problems. The SVM works through of the construction of an N-dimensional hyperplane that is used to separate the data of different classes of a dataset.

Through an initial set of data used as *"training"*, where the classes are known in advance, the SVM creates a function, or a mathematical model, able to separate elements of different classes. Is recognized as *"support* *vectors"* to the subset of data which are taken as the basis for create the model.

Once generated the mathematical model, the test data are entered for the classification. This process is shown in Figure 2.

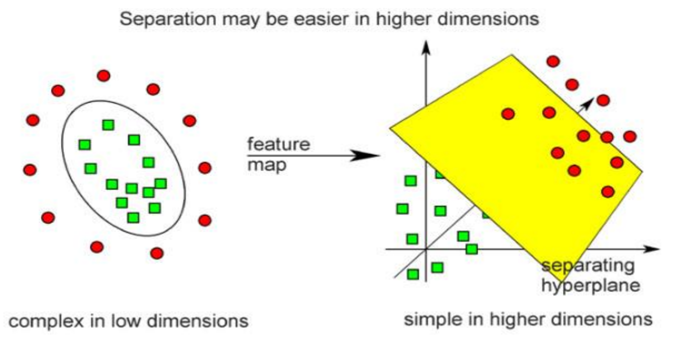
Separation may be easier in higher dimensions
feature map
separating hyperplane
complex in low dimensions
simple in higher dimensions

Figure 2. Process of a SVM. Gets training data, generates the mathematical model and applies it on testing data.

In our implementation, the set of patients used as training for the SVM, corresponds to the same patients used in the filter. The values entered to the SVM for training and classification, are the expression values of the 10 "best" genes obtained by the filter. With these data the training of the SVM was performed, and a function or mathematical model able to separate patients from different classes was obtained. After building the model, the final classification tests were performed. For these tests was selected a set of patients whose expression data, and class, were not used in the execution of the filter, nor in some stage prior to these final tests. On these test patients, and based on the generated model, the SVM estimates the subtype of cancer of each patient. Finally, with the information of the classes estimated by the SVM, and with the information of the actual classes of the patients, we calculate the success rate obtained by the method.

The SVM functions used in this project were taken from [13]. These functions are implemented in R programming language, and work on GPGPU using CUDA technology. To apply these functions, the dataset was scaled to a range of -1 to 1. This range is the recommended in [14]. The kernel used in the SVM was of linear type, and its *"C"* parameter was set to 2.0. This value of *"C"* was obtained running the script *"grid.py"* that belongs to the LIBSVM library [12].

III. Experiments and Results

A. Design of the Experiment

For the tests of the BSS/WSS filter, we structured the Leukemia cancer dataset according to the LIBSVM format [12]. In this format the columns correspond to the genes, and the rows to the patients. The first 25 patients have Leukemia type AML, and the last 47 patients have type ALL. The first column is reserved to identify the class of each patient. Figure 3 shows a representation of the structure of the Leukemia cancer dataset.

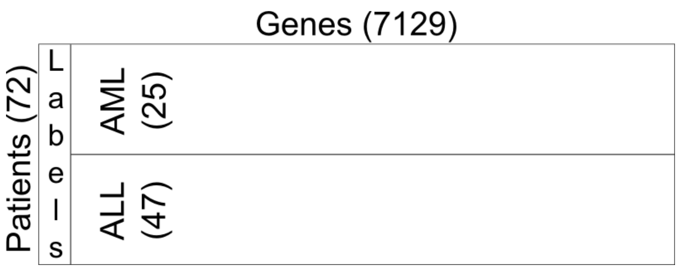

Figure 3. Structure of the Leukemia cancer dataset.

The evaluation of the CUDA implementation consisted in tests of 1000 iterations, performed on a set of training data, which were selected by a method which we called *"round-cross"*. In each iteration were selected a total of 15 patients of class AML, and 27 patients of ALL class. The first 15 patients selected of the AML class correspond to the indexes 1 to 15 of the dataset, and for the ALL class the indexes 26 to 52. In each iteration was incremented, by one, the start of the indexes of the subsets, so that, for the second iteration, the indexes of the subset of AML patients are from 2 to 16 and from 27 to 53 for the subset of ALL patients. In Table 1, are shown the selected sets by the *"round-cross"* method for the first 5 iterations.

Table 1. The first 5 subsets of AML/ALL patients obtained from round-cross for the BSS/WSS filter.

AML	ALL
1 – 15	26 – 52
2 – 16	27 – 53
3 – 17	28 – 54
4 – 18	29 – 55
5 – 19	30 – 56

The experiments were performed with the following equipment:

- Graphic card: Nvidia GeForce GTX 670 with 2GB RAM memory, 1344 CUDA cores, compute capability of 3.0.
- Driver version: 319.49.
- CUDA Toolkit version: 5.5.
- GCC version: 4.7.0 20120507 (Red Hat 4.7.0-5).
- Kernel version: 3.3.4-5.fc17.x86_64.
- GNU/Linux distribution: Fedora release 17 (Beefy Miracle) 64 bits.
- Processor: Intel(R) Core(TM) i5-2500 CPU @ 3.30GHz.
- System ram memory: 4GB.

B. Results

The genes considered as "best" for classification, and for which the filter assigns a higher score, have a similar behavior for patients of the same class, and dissimilar behavior for patients of different classes. Figure 4 shows a graph with the expression levels of the 10 best qualified genes by the filter. It is clear that the behavior of these 10 genes is very different for the first 25 patients, that are of type AML, with respect to the remaining 47 patients that are of type ALL. Therefore these 10 genes are most suitable for the classification of subtypes of Leukemia.

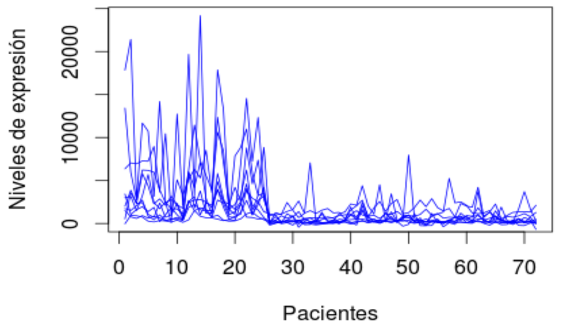

Figure 4. Graph of the expression levels of the best 10 genes qualified by BSS/WSS filter.

Figure 5 shows a graph of the expression levels of the 10 worse qualified genes by the filter. It is appreciable that the behavior of the genes cannot distinguish between the groups of patients with type AML from those of type ALL. Therefore these 10 genes are not suitable to perform the classification of subtypes of Leukemia.

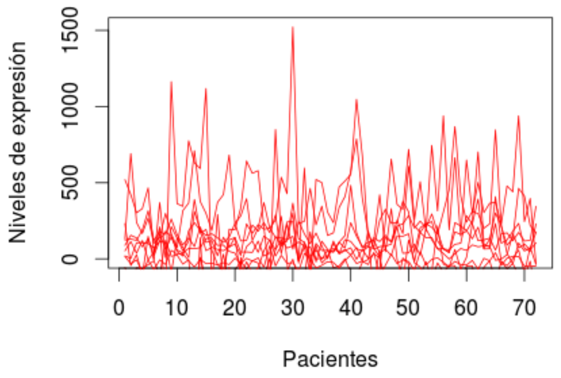

Figure 5. Graph of the expression levels of the worse 10 genes qualified by BSS/WSS filter.

In Figure 6 is shown a comparative graph of the execution time for 1000 iterations, between the parallel implementation with CUDA technology and the sequential implementation with C. It is appreciable that CUDA C had a greater performance compared to C, resulting a difference of performance of about 18 times faster.

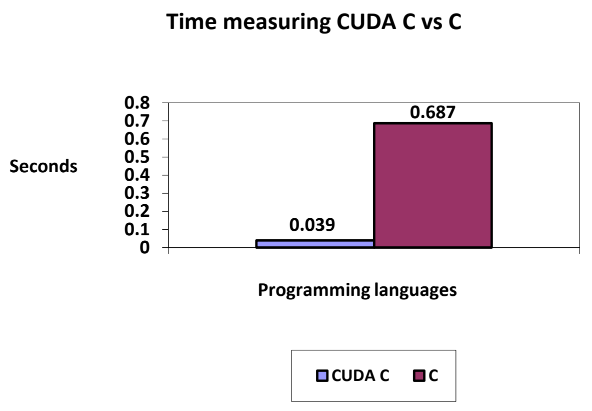

Figure 6. Time measuring of BSS/WSS filter execution. CUDA C vs C for 1000 iterations.

The final classification tests were performed with the SVM, based on the 10 best genes discovered by the filter. These tests were performed using information from 42 patients taken as training, and 15 patients used as test. The results show a success rate of classification of 92%, that is, in average in each iteration, 13.8 of 15 patients of test were correctly classified.

IV. CONCLUSIONS

The use of the BSS/WSS filter is very useful for the reduction of the size of the gene expression datasets used for the classification of subtypes of cancer. The significant reduction of the number of genes achieved for the filter makes possible the use of a Support Vector Machine, to perform the classification of cancer with an acceptable success rate.

With the parallel implementation of the BSS/WSS filter on GPGPU using CUDA technology, we have achieved a significant reduction of the execution time, compared to the sequential implementation.

As future work, we plan to implement the filters T-Statistic and Wilcoxon in parallel. It is expected that with the use of these two filters, together with the BSS/WSS filter, we could achieve best success rates in the classification. It is also proposed perform tests of classification, for both, diagnostic and prognostic of cancer, with others cancer datasets.

REFERENCES

[1] World Health Organization, Fact Sheet 310.

[2] Ying Lu, and Jiawei Han, "Cancer Classification Using Gene Expression Data," Journal Systems, vol. 28, no. 4, 2003, pp. 243–268.

[3] Golub et al., "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring," Science, vol. 286, 1999, pp. 531–537.

[4] Nasimeh Asgarian, and Russell Greiner. "Using Rank-1 Biclusters to Classify Microarray Data," Bioinformatics, vol. 00, 2007, pp. 1–10.

[5] U. Alon et al., "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," Proceedings of the National Academy of Sciences of USA, vol. 96, no. 12, 1999, pp. 6745–6750.

[6] Dinnesh Singh et al., "Gene expression correlates of clinical prostate cancer behavior," Cancer Cell, vol. 1, 2002, pp. 203–209.

[7] Gavin G. Gordon et al., "Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma," Cancer Research, vol. 62, 2002, pp. 4963–4967.

[8] Hernández Montiel L.A., Bonilla Huerta E., and Morales Caporal R., "A multiple-filter-GA-SVM method for dimension reduction and classification of DNA-microarray data," Revista Mexicana de Ingeniería Biomédica, vol. 32, no. 1, 2011, pp. 32–39.

[9] Sandrine Dudoit, Yee Hwa Yang, Matthew J. Callow, and Terence P. Speed, "Statistical Methods For Identifying Differentially Expressed Genes In Replicated cDNA Microarray Experiments," Statistica Sinica 12, 2002, pp. 111–139.

[10] B.E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5, 1991, pp. 144–152.

[11] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, "A Practical Guide to Support Vector Classication," 2010.

[12] Chih-Chung Chang, and Chih-Jen Lin, "LIBSVM: a library for support vector machines." ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011.

[13] (2009) The R's MVS website. [Online]. Available: http://www.r-tutor.com/

[14] Part 2 of Sarle's Neural Networks FAQ Sarle, 1997.