LEARNING SUPPORT VECTOR MACHINES USING PARALLELIZED EVOLUTIONARY ALGORITHMS

Gheorghe Radu, Ionut Balan and Stefan Gheorghe Pentiuc

Abstract. Support vector machines (SVMs) have proven to be a powerful tool for classification and regression (Vapnik, 1998; Mierswa, 2006). Despite the originality and performance of the learning vision, the inner training engine appears as intricate, constrained, rarely transparent and able to converge only for certain particular decision functions. Evolutionary approach to support vector machines (EASVMs) is a novel technique constructed as an alternative of the standard SVMs architecture (R. Stoean et al, 2009); this approach adopts the learning strategy of the latter but aims to simplify and generalize its training, by offering a transparent substitute to the initial black-box. EASVMs focuses on the evolution of the coefficients of the decision function within the geometrical learning concept of SVMs.

In this paper, we present a new hybridization between EASVMs and parallel algorithms. The experimentation were deployed on a high performance processor cluster with 96 nodes with IBM PowerXCell 8i processors and an hybrid architecture IBM Roadrunner that will permit to get better results in much smaller intervals of time. The computational results show the validity of new approach in terms of transparency of the training, runtime, accuracy and flexibility.

2010 *Mathematics Subject Classification*: 68M20, 68T05, 68W10, 91E40.

1.Introduction

Support vector machines, SVMs, (see [1-2]) have proven to be a powerful tool for classification and regression in the supervised learning context of pattern recognition. They turned on original instruments in the field of machine

learning. However, the training engine has a high computational complexity and its convergence is ensured only for some particular kernel functions. This has motivated researchers to investigate many alternatives to training approach, based on evolutionary algorithms, which are known to be flexible and robust.

There are other reported attempts to hybridize SVMs and evolutionary algorithms (EAs) in order to achieve better performances. For example, in the former paper [3], it is used a Penalty Function's Method to deal with constraints. In another approach, the evolution of kernel function to model training data is performed by means of genetic programming. Another example, the evolution strategies and particle swarm optimization (PSO) is used for computing the Lagrange multipliers involved in the expression of the dual problem (see [2]). The evolution approach to support vector machines (EASVMs), presented in [4], focuses on the evolution of the coefficients of the decision function within the geometrical learning concept of SVMs, with respect to the optimization objectives regarding accuracy and generalization.

The aim of this paper is to parallelize the EASVM algorithms to obtain better run times, especially for large training data sets.

The paper is structured as follows. The next section presents the approach of support vector machine learned with an evolutionary engine, following the paper [4]. This is followed by an outline of our approach including the parallelization of evolutionary algorithms. The last chapter presents the results obtained from the experiments.

2.The EASVM Algorithm

SVMs are well suited for classification and regression. Given $\{(x_i, y_i)\}$, $i = 1, 2, ..., m$ a training set where every represents a data sample and each $x_i \in \mathbf{R}^n$ corresponds to a target, a learning task is concerned with the discovery of the optimal function that minimizes the discrepancy between the given targets of data samples and the predicted ones, the outcome of previously unknown samples is the tested (see [4]).

The task for classification is to achieve an optimal separation of given data into classes. SVMs regard learning in this situation from a geometrical point of view: they assume the existence of a separating surface between every two classes labeled as -1 and 1 (see [4]). If training data were linearly separable, then there would exist a linear hyper-plane, $\langle w, x\rangle - b = 0$, which partitions the samples according to classes. Separation is achieved if each positive/negative

sample lies on the corresponding side of a matching supporting hyper-plane of respective class [5].

$$y_i(\langle w, x_i\rangle - b) > 1, i = 1, 2, ..., m \tag{1}$$

The general primal problem of finding the decision hyper-plane is consequently solved using an EA.

The evolutionary elements (see [6]) are set out below:

Representation: The coefficients of the hyper-plane are encoded in the structure of an individual: $c = (w_1, ., w_n, b)$. Individuals are initially randomly generated such that $w_i \in [-1, 1]$, $i = 1, 2, ..., n$, $b \in [-1, 1]$.

Fitness assignment: The fitness assignment derives from the objective function and is subject to the constraints of the optimization problem. By departing from the standard SVMs, a different nonlinear formulation is derived (see [4]). The parameter w is mapped through Φ into H. As a result, the squared norm that is involved in the generalization condition becomes $\|\Phi(w)\|^2$ and the equation of the hyper-plane is $\langle \Phi(w), \Phi(x_i)\rangle - b = 0$.

The form $\langle u, w\rangle = u^T w$ is used and the kernel is employed to transform the norm in its simplistic equivalence to a scalar product. The fitness formulation (to be minimized) embodies the objective function and the constraints are handled by penalizing the worst chromosomes through a function t that returns the value of the argument, if negative, and 0 otherwise. Its expression (2) for classification is the follow (see [4]):

$$f(w, b) = K(w, w) + C\sum_{i=1}^{m}\xi_i + \sum_{i=1}^{m}[t(y_i(K(w, x_i) - b) - 1 + \xi_i)]^2 \tag{2}$$

Selection and variation operators: Widely used schemes for real encoding were applied. These are: tournament selection, intermediate crossover and mutation with normal perturbation.

Stop condition: The algorithm stops after a predefined number of generations. Once the near optimal values for the coefficients of the decision hyper-plane are found, the target for a new, unseen test data sample can be determined by the following equation (3):

$$class(x_i) = sgn(K(w, x_i - b) \tag{3}$$

The classification accuracy is defined as the number of correctly labeled cases over the total number of test samples.

3.Parallelizing evolutionary algorithms

By using the evolutionary algorithms for solving the proposed problem, the results are obtained in a pretty large amount of time with an important consumption of computing resources. Because these kinds of algorithms can be easily parallelized, we have chosen this option firstly for obtaining better and better results, in a quite short time frame, following the same mode used in our work [7].

The tests have been made on a cluster having the hybrid architecture of supercomputer IBM Roadrunner [8], the first places in the Top 500 of the fastest supercomputers in the world, since 2009. The system has 48 nodes and each node has 2 processors, in total 768 computing cores, disposed in IBM Blade Center LS22 and IBM Blade Center QS22.

The supercomputer, having 6.53 TFlops in double precision proved by Linpack, is installed in the High Performance Computing Laboratory of the University "Stefan cel Mare" of Suceava [9].

The communication links among the processing nodes used in the tests we have conducted are made using Open MPI library [10]. The parallelization of this algorithm was accomplished on a data level, meaning that the population is divided equally among the MPI processes that run on each node of the cluster, choosing an insular model that allows this.

The initial population is generated in the main node, after that, the population is evolving separately in each node. In this condition, the evolutionary operators are applied to each population and after a prefixed number of generations they send one to each other the best chromosomes. Communications between the processes are shown in Figure 1.

4.The experimental results

For testing the proposed algorithm, four sets of data have been used taken from the University of California Machine Learning Repository [11]:

- Pima Indians Diabetes Data Set;
- Iris Data Set;
- Spambase Data Set;
- Soybean (Small) Data Set.

In each considered case, 5 runs have been made and considering the average values of the obtained results.

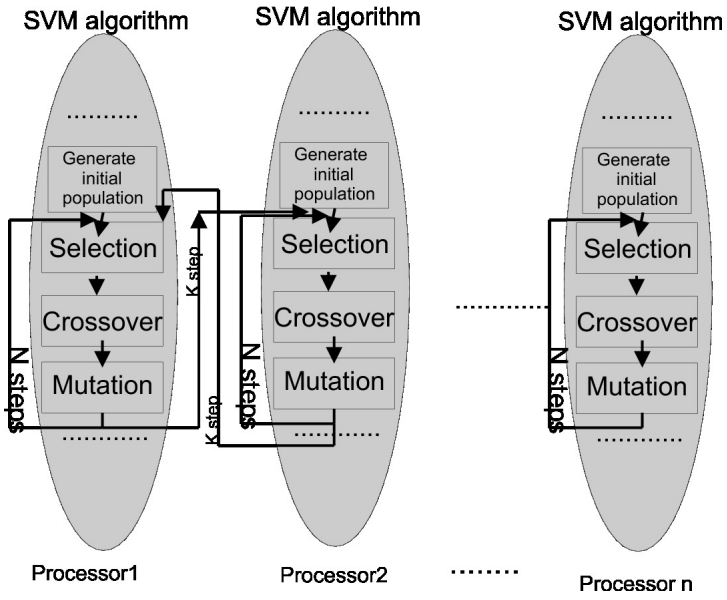

Figure 1: Communications between the populations.

Data set	**Kernel**	**Number of training patterns**	**Number of testing patterns**
Pima	polynomial	576	192
Iris	Radial	105	45
Spambase	polynomial	3451	1150
Soybean	polynomial	30	17

Table 1: Data sets characteristics.

In Table 1 (see also [4]) there we are presented the main characteristics that appeared in the evolutionary algorithm for each separate case.

We have to note that the dimension of the population in each separate case is 200 individuals, the number of generations in which this populations will evolve is 250, while both the recombination probability and the mutation probability is 0.4.

Number of nodes	**Iris (%)**	**Spam (%)**	**Pima (%)**	**Soybean (%)**
1	95.11	76.08	71.09	91.76
2	96.88	78.23	76.82	94.11
4	98.66	78.32	79.43	98.82
8	98.22	79.00	79.68	100
20	100	80.48	80.21	100

Table 2: The average accuracy obtained.

The experimental results obtained after running on different architectures are presented in Table 2.

The data from Table 2 can be graphically represented like in Figure 2, for pointing out the gains obtained by parallelizing the evolutionary algorithm from the solved problem.

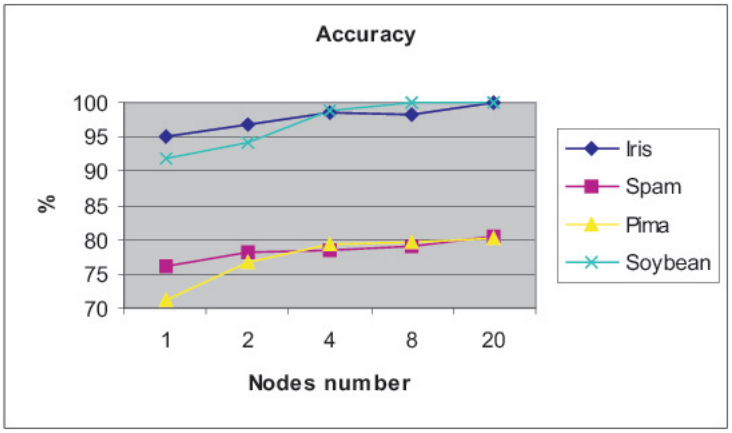

Figure 2: The obtained accuracy in each of the 4 cases.

Analyzing the studied data sets we can notice that the introduction of the parallelization of evolutionary algorithms lead to improved results in the great

majority of the cases observed. Particularly for Iris and Soybean data sets we can notice that on 20, respectively 8 nodes used for running we obtain a 100% accuracy.

This result is obtained due to the small number of patterns that need to be included in a certain category, and so this is happening with a much greater probability than when using much larger data sets.

A certain cluster has enough resources for solving certain problems. Therefore, there is no point in overloading with tasks certain processors just for obtaining better results and in exchange enlarging the running time frame. For confirming this thing we have used the same number of individuals in a generation of evolution but on different configurations. The obtained results by running the algorithm on Iris data set, meeting the conditions described above are further presented in Table 3.

Population's dimension	4000	2000	800	400	200
No. of processors	1	2	5	10	20
Time(s)	202.19	110.47	45.23	22.87	12.18
Accuracy (%)	95.55	97.03	97.77	97.77	100

Table 3: The results obtained by running under different configurations of Iris data set.

From Table 3 we can draw the conclusion that it is not necessary to use huge populations that can lead to choking of a certain calculus unit in the moment that we dispose more processing units that are parallel connected. By appealing to these methods, using a same number of individuals/generation we obtain much better results in shorter running time frames.

Another experiment was achieved (only) on Pima Indians Diabetes Data Set, composed by 768 patterns, each pattern having 8 features represented by 8 attributes. Also we have to mention that the patterns in this data set belong to two classes (binary classification). We have proceeded to the execution of the algorithm seven times (with the possibility that to obtain different results at each execution).

In the sequential case we have used a population made by 200 chromosomes, which is evolving during 250 generations.

The results of the parallelized EASVM algorithm were obtained on a parallel architecture composed by 5 machines with characteristics resembling the ones of the machine the sequential version runed on. On each of the 5 ma-

chines a number of 200 chromosomes were generated, that evolved during 250 generations. Crossover probability, in this case also will have the value 0.4, mutation probability for w and b will be 0.4, while mutation step for w and b will be equal to 0.1. We have used a 200 chromosomes population on each of the parallel architecture's nodes, for proving the fact that one run on a n processor architecture will provide much better results than n runs in the sequential version. Similar problems are described for the multi-grid cellular genetic algorithms as in [12].

Used architecture	Average	Worst (%)	Best (%)
Sequential	76.97	70.31	**82.81**
Parallel with 5 nodes	**79.89**	**77.60**	**84.34**

Table 4: The accuracy obtained in the sequential and parallel version with 5 nodes (Pima Set).

The best results, the worse and the average, from both versions analyzed here are presented in Table 4.

For answering the question: is the parallelization of evolutionary algorithms beneficial for solving the suggested problems, we will compute the speedup obtained based on Iris data set.

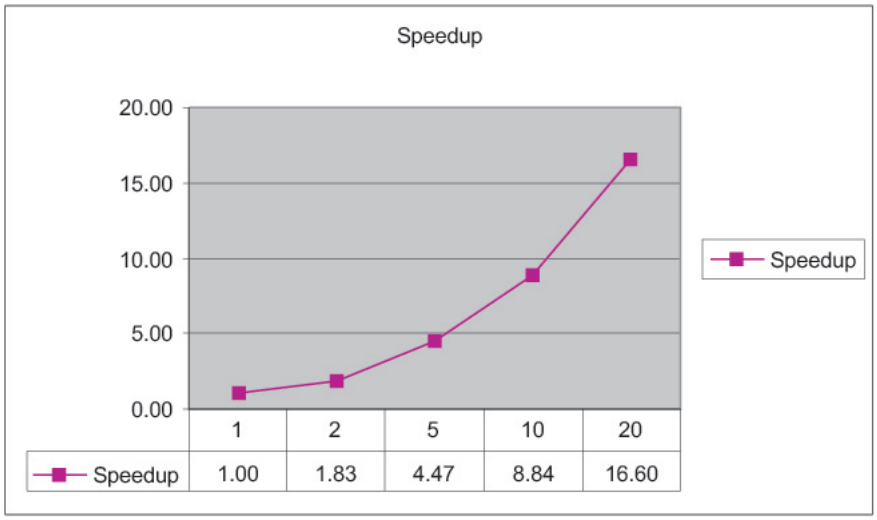

Figure 3: The acceleration obtained by parallelizing evolutionary algorithms.

The obtained values are presented in Figure 3.

Theoretically, the speedup cannot overcome the number of processors that the parallelization was made on, but it is preferred to get as close as possible to this. In our case we can notice certain differences between the speedup obtained and the number of processors that the run was made on. The main cause that leads to this thing is the existence of large program areas (different from the ones reserved for evolutionary algorithms) that weren't parallelized. Another major cause for this difference can also be the communication between processors.

5. Conclusions

The execution times, as well as the number of steps needed for obtaining the final result have different values, from one execution to the other, because when dealing with these types of algorithms with evolutionary techniques the random process has significant importance. That's why, significant for the present paper is the average value of the obtained results. By parallelizing the evolutionary algorithms, besides the gain in the execution time area, we can also register some gains in the number of individuals used for touching improved results.

6. Acknowledgements

We thank Ruxandra Stoean for some valuable suggestions and for her kind help in preparing this paper.

References

[1] V. Vapnik, Statistical Learning Theory, New York: Wiley, 1998.

[2] I. Mierswa, Making indefinite kernel learning practical, Technical Report, University of Dortmund, 2006.

[3] Gh. Radu, Evolutionary Techniques for Training of SVMs, Proceedings UNIVERSITARIA-ROPET-2003, Petrosani, ISBN 973-8260-37-X, October 16-18, (2003), 97-102.

[4] R. Stoean, M. Preuss, C. Stoean, E. El-Darzi and D. Dumitrescu, Support Vector Machine Learning with an Evolutionary Engine, Journal of the Operational Research Society, Vol.60, (2009), 1116-1122.

[5] R.A. Bosch and J.A. Smith, Separating hyper-planes and the authorship of the disputed federalist papers, Amer. Math. Month. Vol. 105(7), (1998), 601-608.

[6] D. Dumitrescu, B.Lazzerini, L.C.Jain and A.Dumitrescu, Evolutionary Computation, USA: CRC Press, 2000.

[7] Gh. Radu, I. Balan and I. Ungurean, Tuning Fuzzy Perceptron using Parallelized Evolutionary Algorithms, Electronics and Electrical Engineering, Kaunas: Technologija No. 1(107), (2011), 51-54.

[8] A. Komornicki, G. Mullen-Schultz and D. Landon, Roadrunner: Hardware and Software Overview, USA: IBM Redbooks, REDP-4477-00, 2009.

[9]*** Grid for developing pattern recognition and distributed artificial intelligence applications, [http://eed.usv.ro/gridnord/en/html].

[10] G. Burns, R. Daoud and J. Vaigl, LAM: An Open Cluster Environment for MPI, Proceedings of Supercomputing Symposium, [http://www.lam-mpi.org/ download/files/lam-papers.tar.gz], (1994), 379-386.

[11] A. Frank and A. Asuncion, UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Comp. Science,[http://archive.ics.uci.edu/ml], 2010.

[12] O. Brudaru, D. Popovici and C. Copaceanu, Cellular Genetic Algorithm with Communicating Grids for Assembly Line Balancing Problems, Advances in Electrical and Computer Engineering, vol. 10, no. 2, (2010), 87-93.

Gh. Radu, I. Balan and St. Gh. Pentiuc
Department of Electrical Engineering and Computer Science
"Stefan cel Mare" University of Suceava
University Street No. 13, 720229 Suceava, Romania
email: *gh.radu@gmail.com, iobalan@yahoo.com, pentiuc@eed.usv.ro*