"HENRI COANDA"
AIR FORCE ACADEMY
ROMANIA

"GENERAL M.R. STEFANIK"
ARMED FORCES ACADEMY
SLOVAK REPUBLIC

INTERNATIONAL CONFERENCE of SCIENTIFIC PAPER
AFASES 2015
Brasov, 28-30 May 2015

PERFORMANCE ANALYSIS OF PARALLEL SVM TRAINING ON CLUSTER HAVING SIMILAR IBM ROADRUNNER ARCHITECTURE

Ştefania-Iuliana Şoiman*, Gheorghe Radu, Ştefan-Gheorghe Pentiuc*, Ionuț Balan*****

*Faculty of Electrical Engineering and Computer Science, "Stefan cel Mare" University, Suceava, Romania, ** Faculty of Aeronautical Management, "Henri Coandă" Air Force Academy, Brasov, Romania, ***Faculty of Economics and Public Administration, "Stefan cel Mare" University, Suceava, Romania

Abstract: *The paper presents an analysis of the efficiency of parallelization of the SVM training algorithms. The researches were focused on computing time reduction in the training stage, and on good accuracy of the resulted classifier. Firstly it was described the algorithm Parallel Cutting Plane Support Vector Machines (PCPSVM) that ensures the parallelism of both data and calculations. Another approach for training SVM was based on the evolutionary algorithms (EASVM). These algorithms are very suitable for parallelization at data level. The experiments were conducted on a supercomputer having a similar architecture to the IBM Roadrunner supercomputer from DOE of USA, the first supercomputer that surpassed 1 petaflops barrier. The experimental results validated the fact that the parallelization of the two different training algorithms of SVM led to very good time performances and a very good accuracy of the solutions.*

Keywords: *Parallel algorithms, Parallel computation, Classification and discrimination; cluster analysis, IBM Roadrunner cluster, Support Vector Machines training.*
***MSC2010*:** *68W10, 65Y05, 62H30, 68M99.*

1. INTRODUCTION

Support Vector Machine (SVM) is a successfully method applied in pattern recognition supervised learning on large data sets with a great number of patterns to be classified and/or having a large number of features thereof. SVM classification method was derived from statistical learning theory by Vapnik et al. in 1992 [1]. Initially used for handwriting character recognition, it is increasingly used so that in the last decade emerged many new and diverse applications using SVM.

The method involves a quadratic problem (QP) solving and generally produces a binary classifier usually for linear separable classes, but may be applicable for multi-class classification or for nonlinear separable classes. SVM is trained to find the optimal separation hyper-plane in the pattern space that ensures an area as large as possible for classes' separation. One of the disadvantages of SVM training algorithm is the large amount of memory required for classification of huge data sets.

Undertaken researches are focused on improving the classification accuracy and

reducing the computation time during the training stage.

An efficient algorithm for training the SVM is presented in the paper of Dai and Fletcher [2], about Quadratic Programs subject to Lower and Upper Bounds which demonstrated the usefulness on the medium and large training sets. Another interesting approach may be found in [3] that uses the solution of the problem division into several smaller quadratic sub problems in SVM training.

The Evolutionary Optimization technique enables the adaptation of the decision hyper-plane to the available training patterns, directly solving the optimization of the primary problem. It is considered the basic idea of SVM, geometric concept of training [4], but the proposed approach deviates from the standard mathematical treatment. In addition, this technique open the way of evolutionary generalizations involving non-standard, nonlinear decision surfaces. It proposes a new approach: the training follows the standard SVM, while the optimal hyper-plane coefficients (*w* and *b*) are determined directly by an evolutionary algorithm with a balance between classification accuracy and generalization ability [5, 6].

2. PARALLEL ALGORITHM FOR TRAINING SVM

2.1. The Parallel Cutting Plane Support Vector Machines algorithm. The PCPSVM algorithm was proposed in [7], and represents a parallel implementation of the training algorithm of SVM [8].

Problem Formulation

Given the training set (x_i,y_i) with $i=1,n$ where $x_i=(x_{i1},x_{i2},..,x_{ip})$ is an input pattern vector from a real space $X \subseteq \mathcal{R}^m$, and $y_i \in \{-1,1\}$ is the output, representing the class label, SVM will find the separation hyper-plane (between the two classes) $f_{(x)} = w^T\phi(x_i) + b$ by solving the optimization problem:

$$min_{w,b,\xi}\ \left(\frac{1}{2}w^Tw + C\sum_{i=1}^{n}\xi_i\right) \quad (1)$$

with the conditions:

$$y_i(w^T\phi(x_i)+b) \geq 1-\xi_i i = 1..n, \xi_i \geq 0 \quad (2)$$

The discriminant function may have the form of

$$f_{w(x,y)}=w^T\Psi(x,y) \quad (3)$$

where the parameter $w \in \mathcal{R}^m$ is the weight vector, and $\Psi(x,y)$ is the feature vector of the input pattern x and the output y as in [8]. The flexible definition of Ψ allows to the SVM to create models for various problems: natural language analysis, bioinformatics (Sequence alignment of proteins), feature selection, image segmentation.

The dual problem may be formulated as follows:

$$max_\alpha W_{(\alpha)} = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}y_iy_j\alpha_i\alpha_jk(x_i,x_j)$$

with the conditions $0\leq\alpha_i\leq C$, $i=1,n$

$\sum_{i=1}^{n}\alpha_i y_i = 0$.

The Cutting-Plane algorithms as presented by Joachims et al. in [7] starts from an empty set of constraints W=0. At the first iteration w and ξ are set to zero (5). The algorithm iteratively builds the working set $W = W_1 \cup ... \cup W_n$, within the constraints imposed by (2) and (4), reformulating the optimization problem as follows:

$$min_{w,\xi>0}\ \frac{1}{2}w^Tw + C\xi,\ \xi = \frac{1}{n}\sum_{i=1}^{n}\xi_i \quad (5)$$

with the conditions:

$$\frac{1}{n}w^T\sum_{i=1}^{n}[\Psi(x_i,y_i) - \Psi(x_i,\overline{y_i})] \geq \frac{1}{n}\sum_{i=1}^{n}\Delta(y_i,\overline{y_i}) - \xi, \forall(\overline{y_1}..\overline{y_n}) \in W \quad (6)$$

where $W = \{-1,1\}^n$.

The solution found by the algorithm is the couple (w, ξ) in the condition of a classification error ε.

The algorithm needs a number of iterations (C / ε), until reaching the optimum solution, which means that the working set W is independently of the size of the pattern training set.

Parallel Cutting Plane Support Vector Machines algorithm (PCPSVM) divides the input set of available processors ensuring so the parallelism of both data and calculations. PCPSVM algorithm was tested on the cluster RedPower from the Laboratory of High Performance Computing USV Suceava.

2.2. **Evolutive Algorithm for Training SVM**. The general problem of finding the separation hyper-plane between 2 classes is solved using systematically an evolutionary algorithm. The main elements of this evolutionary approach are as follows.

Representation: hyper-plane coefficients, w and b are coded in a chromosome structure: $c = (w_1 \ldots, w_n, b)$. Initially, the chromosomes are generated randomly, so $w_i \in [-1,1]$ for i=1,..,n and $b \in [-1,1]$.

Assignment of fitness function. The fitness function derives from the objective function and is subject to restrictions of the optimization problem. Departing, however, from standard SVM, for the fitness function is derived a different nonlinear formulation. The parameter is mapped from Φ in H. It follows, then, that the quadratic norm involved in the generalization condition becomes $\|\Phi(w)\|^2$, and the equation of the separation hyper-plane is

$$\langle \Phi(w), \Phi(x_i) \rangle - b = 0 \qquad (7)$$

It is used the notation $\langle u, w \rangle = u^T w$ and the nucleus is used to turn the rule into a scalar product. The formulation of the fitness condition (function) to be minimized incorporates the objective function. The restrictions are formulated by penalizing chromosomes that are incorrectly classified by a *t()*function that returns the value of the argument if it is negative, and 0 otherwise. For classification, the expression of the fitness function is

$$f(w,b) = \frac{1}{2} K(w,w) + C \sum_{i=1}^{m} \xi_i + \sum_{i=1}^{m} [t(y_i (K(w,x_i) - b) - 1 + \xi_i)]^2 \qquad (8)$$

The *selection* and *variation operators*: are applied the most commonly used coding scheme: the tournament selection, intermediated crossing and mutation by disruption of normal distribution.

Stop condition: the algorithm stops after a predefined number of generations.

Once the closest values to the optimal values of hyper-plane coefficients are found, a new pattern can be classified directly, using the equation:

$$class(x_i) = \text{sgn}(K(w, x_i) - b) \qquad (9)$$

Classification accuracy is defined as the ratio of the number of cases / patterns correctly labeled and total test examples.

The optimization problem solving requires the treatment of errors indicators that were included in the representation in the initial version of evolutionary algorithm. The proposed algorithm (EASVM) provides interactively the hyper-plane coefficients at any time, so it is possible to calculate the errors on each iteration. In this way the error indicators may be removed from the representation of the evolutionary algorithm.

As a consequence, the representation of an individual (hyper-plane) contains only w and b. In addition, all indicators will be calculated in reference to their fitness function. Regarding the classification, the current individual (separating hyper-plane) is taken and the support hyper-planes are determined by the proposed mechanism of Bosch and Smith, as in the paper [9].

By using evolutionary algorithms to solve such a problem, the results are obtained in a fairly large time, with a significant consumption of resources. Due to the fact that this type of algorithms can be easily parallelized, we resorted to this embodiment, firstly in order to achieve better results in a shorter training time.

The parallelization of the algorithm for SVM training in the evolutionary approach is presented in the paper [10]. The parallelization

was performed upon the general stages of an evolutionary algorithm: generation of the initial population, selection, crossover, mutation being executed in individual loops on each node. The parallelization is mainly achieved at data level: the population being equally divided between MPI processes running on each node of the cluster.

3. EXPERIMENTAL RESULTS

The experiments were conducted on a supercomputer from the High Performance Computing Laboratory in the „Stefan cel Mare" University of Suceava. This supercomputer has a very similar architecture as the supercomputer IBM Roadrunner installed at Los Alamos National Labs, New Mexico, USA, the first supercomputer in the world that surpassed 1 petaflops [11].

The endowment of the University "Stefan cel Mare" Suceava, is a computer cluster consisting of 48 blade servers QS22 (nodes) with 96 PowerXCell 8i processors at 3.2 GHz and 8 server blades (LS22) with AMD Opteron. The storage capacity is 10 TB cluster, and the theoretical processing power is 9.98 TFlops and was able to sustain a rate of 6.53 TFlops for LINPACK operations). Each PowerXCell 8i processor consists of 8 units of account SPE (Synergistic Processor Element) and a computing unit PPE (PowerPC Processor Element). The PPE processor contains a PowerPC core 64-bit. On this core can run operating systems and 32 and 64 bits applications? SPE processor is optimized for applications requiring intensive calculations, like SIMD (Single Instruction Multiple Data). These cores are not created for the implementation of operating systems. SPEs are independent processing elements, each running threads or individual applications. SPEs depend on the PPE sites in that the latter run the operating system and some applications are controlled, while the PPE utilize SPEs to achieve higher speeds in data processing.

Experiments with PCVSM

The results validate the fact that the parallelization of the algorithm reduces the complexity of $O(n)$ to $O(n/p)$, for p processors and n the size of the training set. The training of the SVM classifiers was implemented by using the MPICH platform. The PCPSVM algorithm was parallelized on one level (without using the SPE processors).

The speed-up factor was calculated by reference to the PCPSVM algorithm running time on a single PowerXCell 8i processor, using a single thread.

Table 1

Training set	Number of Patterns	Number of Features
covtype	581012	54
ijcnn1	49990	22
realsim	72309	20958
web-a	49479	300

It was selected 4 sets of binary data (Table 1) to evaluate the proposed parallel algorithm for training SVM: *covtype* (areas covered by forests), *real-sim* (articles discussion groups on real vs simulated) [12], *ijcnn1* - IJCNN 2001 (International Joint Conference on Neural Networks)] *web-a* (the web page classification) [13,14].

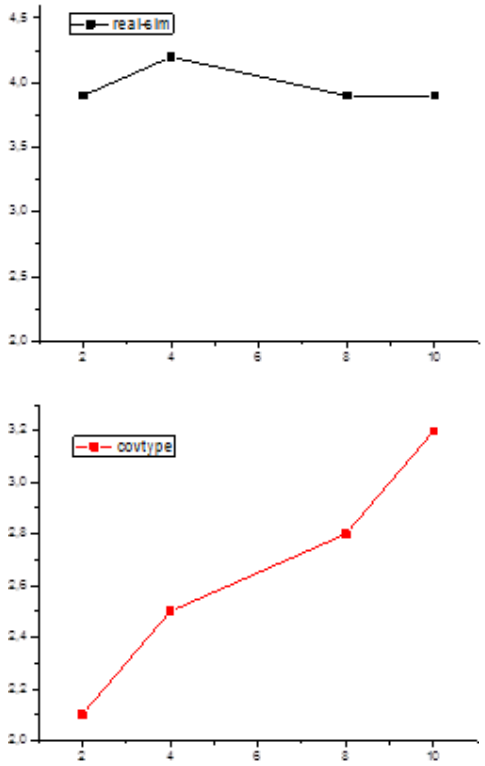

Figure 1. The speed-up obtained by PCPSVM algorithm for *covtype* and *real-sim* databases.

Tests show an increase execution speed of the parallel algorithm in relation to the number of processors involved in training the classifier

on the MPI platform. The results in Figure 1 show a linear growth of the acceleration factor obtained after the parallel execution of the program at the training with *covtype* database.

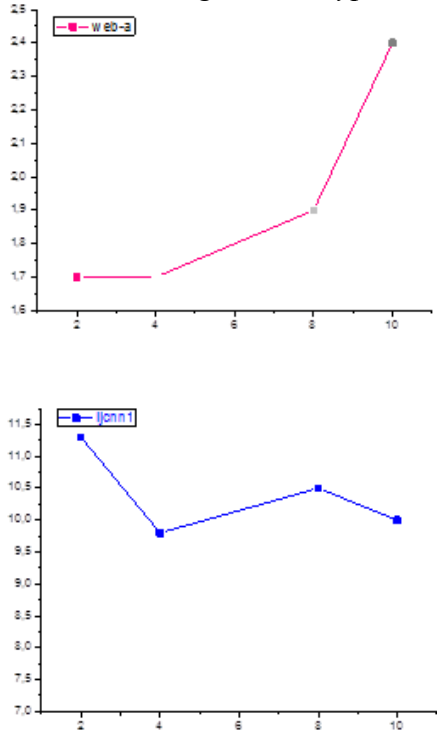

Figure 2. The speed-up for *web* and *ijcnn1* database.

The results in Figure 2 show an acceleration factor of 11 obtained by the program execution PCPSVM on 2 PPE processors using the training data set *ijcnn1* containing 50,000 patterns with 22 features. Lowest acceleration factor, 2.5, is obtained from the classification of web pages; an average acceleration factor of 4, is obtained by the PCPSVM algorithm during the tests with *real-sim* database containing 72,309 patterns.

Experiments with EAVSM

For the experiments were chosen four test problems from UCI Repository of Machine Learning Databases [15], their characteristics are shown in Table 2. In each case of these 4 databases many runs were made, taking into account the average values of the results.

Table 2.

Database	**Kernel**	**Size of training set**	**Size of test set**
Pima	polynomial	576	192
Iris	radial	105	45
Spambase	polinomyal	3451	1150
Soybean	polinomyal	30	17

The population size in each case was 200 individuals; the number of generations evolves these populations being 250, while the probability of recombination and the mutation is 0.4.

The average accuracy obtained in each case is shown in the following table.

Table 3.

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**

It may be observed that on the studied datasets, the parallelization of the evolutionary algorithm led to improved results in most considered cases.

3. CONCLUSIONS

The parallelization of the training algorithms of SVM proves to be an extremely promising approach, both in terms of computing time performance shown in solving the very large problems, and in terms of solution accuracy.

Tests aimed at training SVM classifier recorded the execution times when used only the PPE processors of PowerXCell 8i processors. Better results for the speed-up of

SVM algorithm may be obtained by activating accelerators SPE of the PowerXCell 8i processors.

The evolutionary solution of the optimization problem provides on every execution step the function coefficients. The EASVM algorithm performance is comparable to that of SVM canonical one, and in some cases offers better results, being widely applicable [~~14,~~15,16].The evolutionary approach to SVM training proves to be a promising method leading to good performance in solving large problems with very good solution accuracy.

REFERENCES

1. Boser, B.E., Guyon, I., Vapnik, V.N. (1992). A training algorithm for optimal margin classifiers. In Proceedings of the *Fifth Annual Workshop of Computational Learning Theory*, 5, 144-152, Pittsburgh, ACM.
2. ai, Y.-H. & Fletcher, R., New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds. *Math. Program.,* 106**,** 403-421 (2006).
3. Zanghirati, G. & Zanni, L. A parallel solver for large quadratic programs in training support vector machines. *Parallel Comput.*, 29, 535-551 (2003).
4. Gheorghe Radu, Ştefan-Gheorghe Pentiuc, Ionuţ Balan – Paralelizarea algoritmilor evolutivi pentru instruirea maşinilor cu vector suport, *Sisteme distribuite, Suceava, 2011*, ISSN 1842-6808, p. 24-26 (2011).
5. R. Stoean, M. Preuss, C. Stoean, D. Dumitrescu, Evolutionary Support Vector Machines and their Application for Classification, *Technical Report Nr. CI-212/06,* Collaborative Research *Center on Computational Intelligence, University of Dortmund* (June, 2006).
6. S.H. Jun, K.W. Oh, An evolutionary statistical learning theory, in *Comput. Intell*, 3(3): 249-256 (2006)
7. Yuan, G., PCPSVM: A parallel cutting plane algorithm for training SVMs. *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on (Volume:4)*, pp. V4-655 to V4-659 (2010).
8. oachims, T., Finley, T. & Yu, C.-N. J. 2009. Cutting-plane training of structural SVMs. *Mach. Learn.,* 77,27-59 (2009).
9. R.A. Bosch, J.A. Smith, Separating Hyper-planes and the Authorship of the Disputed Federalist Papers, *American Mathematical Monthly*, Volume 105, Number 7, pp. 601-608, (1998).
10. Gheorghe Radu, Ionuţ Balan, Ştefan-Gheorghe Pentiuc – Learning Support Vector Machine Using Parallelized Evolutionary Algorithms*, Acta Universitas Apulensis, Mathematics-Informatics*, Nr. Special, ISSN 1582-5329, p. 331-240 (2011).
11. arker, K. J., Davis, K., Hoisie, A., Kerbyson, D. K., Lang, M., Pakin, S. & Sancho, J. C. Year. Entering the petaflop era: The architecture and performance of Roadrunner. In: *High Performance Computing, Networking, Storage and Analysis*, 2008. SC 2008. International Conference for, 15-21 Nov. 2008 2008. 1-11 (2008).
12. McCallum, A. K., Data for binary classification separating real from simulated, and auto from aviation. *http://people.cs.umass.edu/~mccallum/data.html* .
13. Prokhorov D., IJCNN 2001 neural network competition. Ford Research Laboratory, *http://www.geocities.com/ijcnn/nnc_ijcnn01.pdf,* (2001).
14. Platt, J. C. (1998). Fast training of support vector machines using sequential minimal optimization. MIT Press**:** 185-208.
15. A. Frank, A. Asuncion, UCI Machine Learning Repository, University of California, Irvine, School of Inform. and Comp. Sc., *http://archive.ics.uci.edu/ml/* .
16. Schipor, O.A., Pentiuc, S.G., Schipor, M.D., Improving Computer Based Speech Therapy Using a Fuzzy Expert System, *Computing and Informatics* Volume: 29 Issue: 2 Pages: 303-318 (2010)