EFDA–JET–CP(09)04/01

J. Ramírez, S. Dormido-Canto, J. Vega and JET EFDA contributors

Automatic Parallelization of Classification Systems based on Support Vector Machines: Comparison and Application to JET Database

Automatic Parallelization of Classification Systems based on Support Vector Machines: Comparison and Application to JET Database

J. Ramírez[1], S. Dormido-Canto[1], J. Vega[2] and JET EFDA contributors*

JET-EFDA, Culham Science Centre, OX14 3DB, Abingdon, UK

[1]Dpto. Informática y Automática, UNED, Madrid, Spain
[2]Asociación EURATOM/CIEMAT para Fusión, Madrid, Spain
** See annex of F. Romanelli et al, "Overview of JET Results",*
(Proc. 22 nd IAEA Fusion Energy Conference, Geneva, Switzerland (2008)).

Preprint of Paper to be submitted for publication in Proceedings of the
7th Technical Meeting on Control, Data Acquisition and Remote Participation for Fusion Research,
Aix-en-Provence, France.
(15th June 2009 - 19th June 2009)

ABSTRACT.
In learning machines, the larger is the training dataset the better model can be obtained. Therefore, the training phase can be very demanding in terms of computational time in mono-processor computers. To overcome this difficulty, codes should be parallelized. This article describes two general purpose parallelization techniques of a classification system based on Support Vector Machines (SVM). Both of them have been applied to the recognition of the L-H confinement regime in JET. This has allowed reducing the training computation time from 70h to 3m.

1. INTRODUCTION
Due to the large amount of experimental data at JET, data mining tools are being considered for massive analysis. Currently, classification techniques based on support vector machines are providing remarkable results in several problems such as data-driven theory [1] and automatic determination of transition points [2]. Typically, the greater the training dataset the better generalization capability can be achieved. However, by increasing the number of training samples, the computation time to train the system rises in a non-linear way. From the computational point of view, this fact can render that the problem is intractable in single processor computers. The motivation of the present work has been the implementation of parallel algorithms for classification techniques based on SVM.

This paper contains 5 sections. Section 2 gives a brief introduction to SVM. Section 3 describes the two parallel algorithms that have been used in this article: Cascade-svm and Spread-svm. Also, the changes done in the LIBSVM [3] library code to implement these algorithms are outlined. Section 4 presents results regarding the L-H confinement regime in JET. Finally, in section 5, some conclusions are drawn.

2. SUPPORT VECTOR MACHINES
Support vector machines [4] are powerful classification and regression tools, but the computational requirements grow very rapidly with the increase in the number of input data.

In this work, binary classifiers based on SVM have been developed, where the two classes are separated by a hyperplane. The decision function that defines this hyperplane is the following:

$$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \tag{1}$$

The parameters α_i, $i = 1,..., n$ are the solution of the following quadratic optimization problem (QP-problem). Maximize the function:

$$L(\alpha) = -\frac{1}{2} \sum_{i=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j) + \sum_{i=1}^{n} \alpha_i \tag{2}$$

subject to these constraints

$$\sum_{i=1}^{n} \alpha_i \, y_i = 0, \quad 0 \quad \alpha_i \quad C/n \quad i = 1,....,n$$

where (x_i, y_i), i=1,...,n are training data, $x_i \in R^n$, $y_i \in \{-1,1\}$, K is a kernel function and C is a regularization parameter. The kernel function transforms the input data to a higher dimensional space to find the hyperplane that separates the two classes with the maximum margin. There are different kernel types: linear, polynomial or Radial Basis Function (RBF). In this article, only the RBF kernel has been used (Equation 3)

$$K(x, y) = e^{-\gamma \|x-y\|^2} \tag{3}$$

Only a reduced number of the α_i parameters is non-zero. These are called support vectors and determine the equation of the hyperplane. Obtaining a solution for Equation 1 requires solving the QP-problem of Equation 2. To do this, heavy computational resources can be needed, depending on the number of input data, n.

3. PARALLEL ALGORITHMS

Increasing execution speed through parallelization is difficult due to dependencies between the steps which solve the optimization of the above equation. We have used two algorithms to increase the speed of the SVM algorithms.

3.1 CASCADE-SVM

In Cascade-SVM [5], we split data into smaller subsets and assign each of these subsets to different processors to obtain the support vectors. In this way, we eliminate the points that are not support vectors in a first phase, saving a large amount of memory and processing time.

As shown in Figure 1, the results of each machine are combined two by two and moved into a new process to obtain the new support vectors. The process continues until only a subset of support vectors is obtained. This process is repeated until a solution is reached; this occurs when the support vectors that have entered into first level are the same as those emerging from the last stage. In [5] it is proved that this process converges in a finite number of steps, although a single pass also produces good results, so we have chosen this alternative.

3.2 SPREAD-SVM

Typically in a SVM problem, the optimization process spends about 95% of the computational time to process the kernel matrix to map data into a higher dimensional space. Spread-SVM [6] is focused on parallelizing the kernel matrix, of dimension *nxn*, where n is the number of input data. The parallelization process distributes the computations among the number of available processors.

According to the Spread-svm method, the QP-problem is tackled in an iterative way. In each iteration, the individual processors solve the optimization problem with different elements of the kernel matrix (Figure 2).

Spread-svm is based on the Sequential Minimal Optimization (SMO) algorithm [6]. SMO breaks a large QP-problem into a series of smallest possible QP-problems. These small QP-problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop. The individual results are combined with a two by two process communication up to reach a global solution of the iteration. This global solution is sent to all processors to start a new iteration as shown in figure 3. The algorithm continues until a stop condition is met [6].

3.3 CHANGES IN LIBSVM

LIBSVM is a library that allows to solve problems with SVM and it has been the tool used to implement the above two techniques.

In the case of Cascade-SVM, we have split the dataset into smaller disjoint subsets without making any modifications to LIBSVM code. Results are integrated according to figure 1 and the process is finished at the bottom level. The resulting set of support vectors defines the separating hyper-plane.

In Spread-SVM, the optimization routine code provided with LIBSVM has been modified. The loop that computes the kernel matrix is distributed among several processors by means of MPI [7] communication calls. In this way, local optimizations are obtained with the aim of achieving a global solution in each iteration. This procedure greatly reduces the computational times and offers a high scalability with the number of processors.

4. EXPERIMENTAL RESULTS

To test the performance of both techniques, a test file of the United States Department of Forest has been used. It contains 580000 input vectors with 64 features for each input. Executions of these tests have been performed on the Lince cluster at CIEMAT. It is made up of nodes Xeon 3.2GHz with 2GB of RAM and a low-latency Infiniband network.

As shown in Figure 4, the scalability is very poor for the Cascade-svm method, as the evolution with the number of processors is far from the linear speed-up. However, Spread-svm has a very good scalability and, therefore, Spread-svm has been chosen for the example about the L-H confinement regime in JET.

Determining the confinement regime in fusion devices (L-mode or H-mode) can be important not only for the physics but also for control purposes. The specific application of this article is focused on the transition from L-mode to H-mode. The feature vectors are made up of temporal evolution samples of 5 signals: the coordinates of the X-point, the beta normalized with respect to the diamagnetic energy, the total heating power and the magneto-hydrodynamic energy. Therefore, each feature vector has 5 components of synchronous samples. The sampling period is 2ms.

Fifty six JET discharges (in the range 70028-72456) have been used. All of these discharges show a very clear transition time. Fifty of them (randomly selected) were chosen to train the SVM classifier and the remaining ones have been used for test purposes. The training was carried out in a symmetric interval of radius 2.7s around the transition, which means a total of 135000 input data.

Execution results of Spread-svm on the Lince cluster are shown in Table 1 and are plotted in Figure 5. It should be noted that the algorithm performance declines over four processors because there are relatively few input data (135000 with only five features).

Therefore, the time needed for calculating the local maximum in each step is lesser than the time spent in inter-processor communications. The Spread-svm algorithm requires at least 500K of data to maintain a good level of scalability as prove in [6].

The parameters used with a RBF kernel (Equation 3) appear in Table 2, where the success rate with the test dataset can reach 99.79%.

CONCLUSIONS

The Spread-svm algorithm provides a large scalability and has been applied to train classification systems of L-H transitions. Typically, the bottle neck in relation to computational times appears in the training process. Each individual case that appears in table 2 takes about 70h in a mono-processor personal computer with only 27000 training samples. With Spread-svm, times are reduced to 3m in the Lince cluster and 135000 training samples. Therefore, more discharges at higher sampling rates can be used to achieve high generalization capabilities in the training process.

ACKNOWLEDGEMENTS

This work was partially funded by the Spanish Ministry of Science and Innovation under the Project No. ENE2008-02894/FTN. This work, supported by the European Communities under the contract of Association between EURATOM/CIEMAT, was carried out within the framework of the European Fusion Development Agreement. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

REFERENCES

[1]. A. Murari, J. Vega, D. Mazon, N. Martin, G. Ratt·, G. Vagliasindi et al. “Data-driven determination of scaling laws in Nuclear Fusion: the threshold for accesing the High Confinement regime. Submitted to Nuclear Fusion.

[2]. J. Vega, A. Murari, S. Gonz·lez. “Universal method for automatic event location in waveforms and video-movies. Application to massive nuclear fusion databases”. To be submitted to Review of Scientific Instruments.

[3]. Chih-Chung Chang and Chih-Jen Lin, LIBSVM: a library for support vector machines, 2001. http://www.csie.ntu.edu.tw/~cjlin/libsvm

[4]. Vladimir Cherkassky, Filip Mulier. Learning from Data (2nd Ed). Wiley-IEEE Press, 2007.

[5]. H. Peter Graf, E. Cosatto, et al. Parallel Support Vector Machines: The cascade SVM. Advances in Neural Information Processing Systems 2004.

[6]. I. Durdanovic, E. Cosatto and H. Graf. Large-scale kernel machines pp. 107-138. MIT Press 2007.
[7]. William Gropp, Ewing Lusk and Anthony Skjellum. Using MPI (2^{nd} Ed). MIT Press 1999.

Processors	**Time (seconds)**
1	494
2	368
4	204
8	187
16	219
32	277

Table 1: Spread times of SVM for L-H transitions.

C	**Gamma (γ)**	**Success (%)**
30000	0.5	89.46
60000	0.5	98.19
90000	1	99.79
90000	0.5	98.09
120000	1	98.05

Table 2: Success rates in classification for different combinations of the C and gamma parameters.

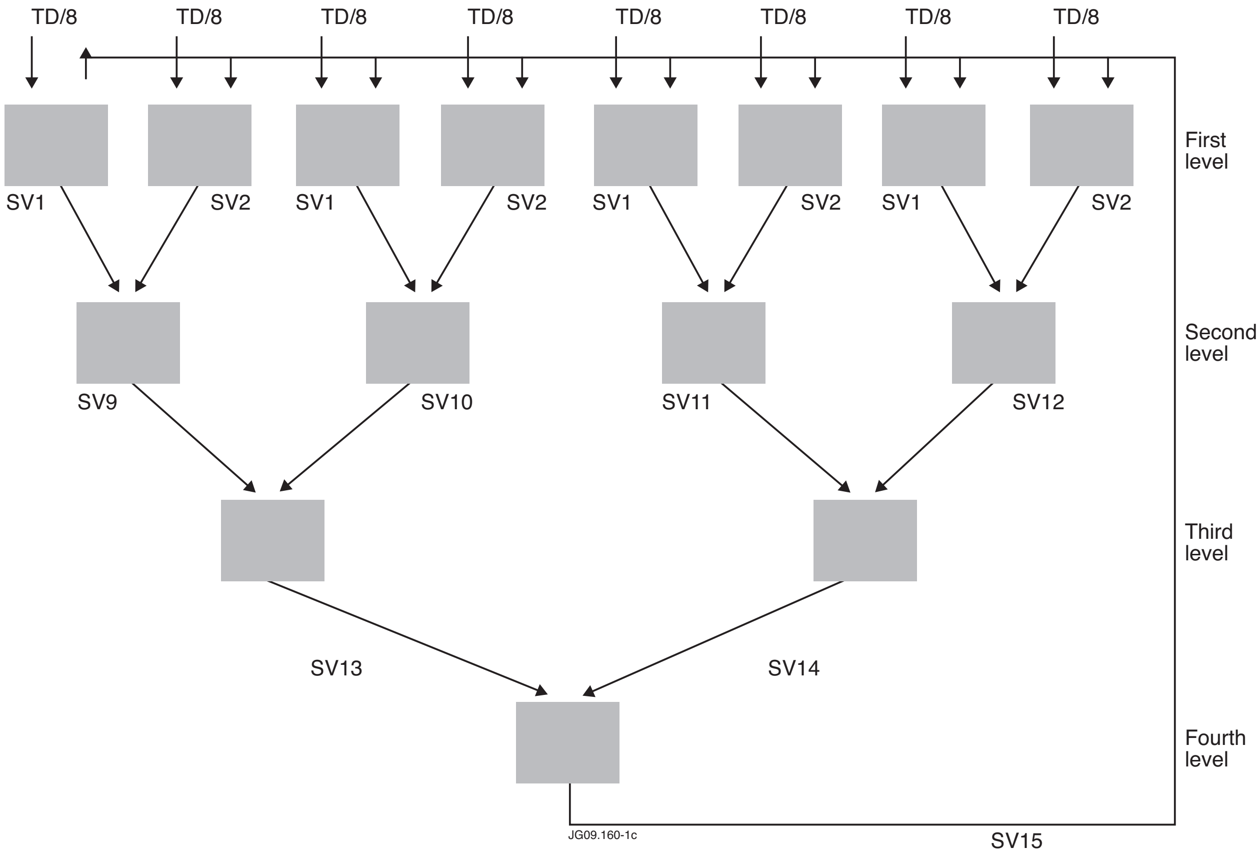

Figure 1: Schematic of a binary Cascade-svm architecture.

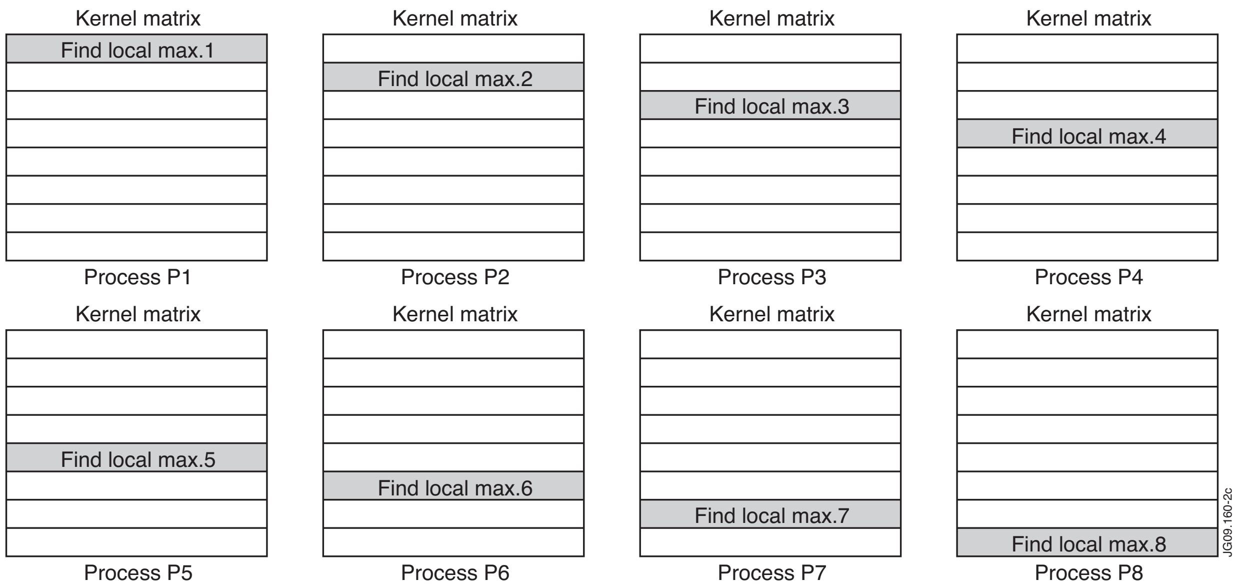

Figure 2: Spreads-svm data processing distribution with 8 processors.

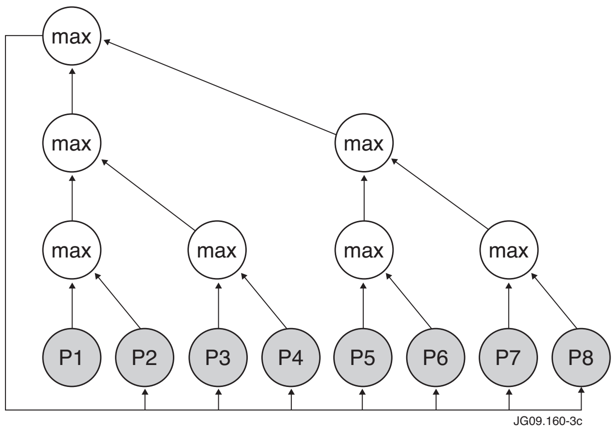

Figure 3: Process communication with 8 processors.

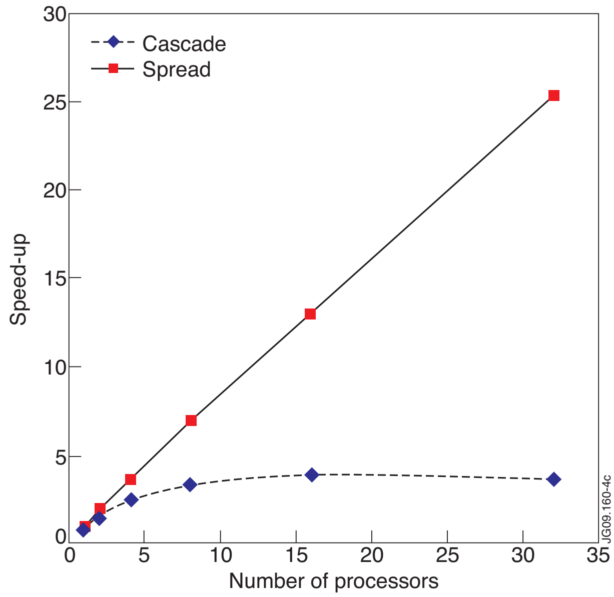

Figure 4: Speed-up comparative.

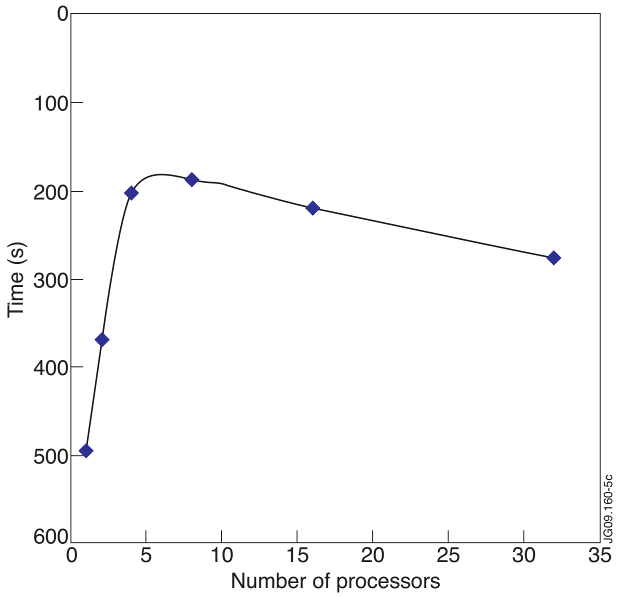

Figure 5: Spread-svm time evolution for L-H transitions.