Efficient implementation of SVM training on Embedded Electronic Systems

Paolo Gastaldo, Giovanni Parodi, Sergio Decherchi and Rodolfo Zunino

Dept. of Biophysical and Electronic Engineering (DIBE), Genoa University
Via Opera Pia 11a, 16145 Genoa, Italy
{paolo.gastaldo, giovanni.parodi, rodolfo.zunino}@unige.it

Abstract. The implementation of training algorithms for SVMs on embedded architectures differs significantly from the electronic support of trained SVM systems. This mostly depends on the complexity and the computational intricacies brought about by the optimization process, which implies a Quadratic-Programming problem and usually involves large data sets. This work presents a general approach to the efficient implementation of SVM training on Digital Signal Processor (DSP) devices. The methodology optimizes efficiency by suitably adjusting the established, effective Keerthi's optimization algorithm for large data sets. Besides, the algorithm is reformulated to best exploit the computational features of DSP devices and boost efficiency accordingly. Experimental results tackle the training problem of SVMs by involving real-world benchmarks, and confirm both the computational efficiency of the approach.
Keywords: Support Vector Machine, SMO, embedded systems, DSP

1 Introduction

Support Vector Machines (SVMs) [1] are one of the most effective tools for tackling classification and regression problems in complex, nonlinear data distributions [1, 2]. Indeed, SVM has been recently successfully applied to the fuzzy methodology [3, 4]. The training of SVMs is characterized by convex optimization problem; thus local minima can be avoided by using polynomial-complexity Quadratic Programming (QP) methods. Besides, kernel-based representations allow SVMs to handle arbitrary distributions that may not be linearly separable in the data space. The ability to handle huge masses of data is indeed an important feature; in such significant cases, SVM training algorithms [5-7] typically adopt an iterative selection strategy: first, limited subsets of (supposedly) critical patterns are identified and undergo partial optimization; then the local solution thus obtained is projected onto the whole data set to verify consistency and global optimality. Such a process iterates until convergence.

The success of SVMs in real-world domains motivates continuing research toward embedded implementations on low-cost machinery. Programmable digital devices often represent the basic choice for the run-time support of trained SVMs [8], but FPGA technology may not prove efficient when dealing with systems that

require training capabilities. In these cases, the target hardware platforms should be endowed with: 1) agile memory handling for easy support of the pattern-selection strategies described in the previous section; 2) buffering and caching capabilities, for managing large sets of high-dimensional data effectively; 3) specialized arithmetic features to speed up computations and maximize efficiency; 4) limited cost, to ensure maximal market impact of the embedded technologies. Hence, the basic features of the SVM training model make the family of Digital Signal Processors (DSPs) possibly more appropriate. Therefore, this paper describes a methodology for the embedded support of SVM training by means of DSP-based architectures. The research reconsiders the overall training problem and privileges the viewpoint of embedded implementations. This resulted in a hardware-oriented version of Keerthi's well-known optimization algorithm [6]. A reformulation of the basic local-optimization steps allows the algorithm to exploit the architectural features of the target processors at best, thus attaining highest efficiency. The method has been tested on a set of standard benchmarks, to verify the algorithm's effectiveness and the computational efficiency of the embedded system.

2 Support Vector Machines for Classification

A binary classification problem involves a set of patterns Z ={$(\mathbf{x}_l, y_l)$; l=1,..,n_p} where $y_l \in$ {-1,+1}. To perform such task, SVM [1] requires the solution of the QP problem:

$$\min_{\alpha} \left\{ \frac{1}{2} \sum_{l,m=1}^{np} \alpha_l \alpha_m y_l y_m K(\mathbf{x}_l, \mathbf{x}_m) - \sum_{l=1}^{np} \alpha_l \right\} \textbf{ subject to: } \begin{cases} 0 \leq \alpha_l \leq C, \forall l \\ \sum_{l=1}^{np} y_l \alpha_l = 0 \end{cases} \tag{1}$$

where α_l are the SVM parameters setting the class-separating surface and the scalar quantity C upper bounds the SVM parameters; $K()$ is a kernel function such that $K(\mathbf{x}_1, \mathbf{x}_2) = \langle \Phi(\mathbf{x}_1), \Phi(\mathbf{x}_2) \rangle$, where $\Phi(\mathbf{x}_1)$ and $\Phi(\mathbf{x}_2)$ are the points in the "feature" space that are associated with $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. An SVM supports a linear class separation in that feature space; the classification rule for a trained SVM is:

$$f(\mathbf{x}) = \sum_{l=1}^{np} \alpha_l y_l K(\mathbf{x}, \mathbf{x}_l) + b \tag{2}$$

where b is a bias. The n_{SV} patterns for which non-null parameters α_l are found by solving (1) are called support vectors. For a thorough presentation of SVM, see [1, 2].

The problem setting (1) involves a QP problem with linear constraints; thus, the solution is unique and it can be found in polynomial time. Since the problem formulation depends on the sample cardinality, n_p, an effective strategy for selecting the eventual support vectors is critical in the presence of large training

sets. In such cases, the widely accepted approach consists in focusing the optimization algorithm on subsets of training patterns in turns. Sequential Minimal Optimization (SMO) [6, 7] proved to be one of the most effective approaches for that purpose. It belongs to the class of "Decomposition" methods [5]; hence, the overall QP problem is decomposed into fixed size QP sub-problems. In particular, SMO involves the smallest possible subset of vectors, and considers pairs of patterns sequentially. The crucial advantage is that the solution of (1) when n_p=2 can be computed analytically and explicitly.

Lin's LibSVM [7] represents a highly efficient implementation of the SMO approach in terms of convergence speed. LibSVM applies Keerthi's SMO [6] on "working sets" of patterns that can be selected iteratively by a shrinking process. This approach tackles the convergence problem by measuring the cost gradient, ∇f_l, at the l-th pattern $\mathbf{x}_l$. The selection strategy identifies the pair of patterns whose Lagrange coefficients $\{\alpha_i, \alpha_j\}$ most violate the Karush-Kuhn-Tucker conditions (KKTs) [6]. Two quantities are associated with these patterns, and rule the stopping criterion:

$$g_i \equiv \begin{cases} -\nabla f(\alpha)_i \text{ if } y_i = 1, \alpha_i < C \\ \nabla f(\alpha)_i \quad \text{if } y_i = -1, \alpha_i > 0 \end{cases}, \quad g_j \equiv \begin{cases} -\nabla f(\alpha)_j \text{ if } y_j = -1, \alpha_j < C \\ \nabla f(\alpha)_j \quad \text{if } y_j = 1, \alpha_j > 0 \end{cases}; \tag{3}$$

the stopping condition that ensures convergence is written as $g_i \leq g_j$. Empirical practice shows that it is one of the most successful strategies to minimize the number of iterations in QP optimization.

3 SVM Training on DSP-based architectures

The basic algorithm for SVM training described in LibSVM involves three main procedures (Fig.1) : 1) the selection of the working set, 2) the verification of stopping criteria, and 3) the update of the crucial quantities, namely, α_i, α_j, and $\nabla f(\alpha)$. The first two steps clearly play a crucial role to the ultimate effectiveness of the training algorithm.

Nevertheless, one should note that the optimization efficiency of decompositions algorithms is conventionally measured by the involved reduction in the number of iterations until QP convergence. When implemented on specific embedded devices, however, selection-based algorithms might exhibit peculiar features that ultimately tend to limit the computational efficiency of the overall system.

The crucial issue is that the implemented strategy should also take into account the internal architecture of the target electronic device. Toward that end, the research presented in this paper proposes a hardware-oriented reformulation of the optimization algorithm [6] for DSP-based embedded electronics, which provide a suitable tradeoff between computational power and cost efficiency. Besides, in the specific context of SVMs, hardware-looping capabilities and SIMD architectures well fit the features of training algorithms, which are typically characterized by deeply nested loop cycles.

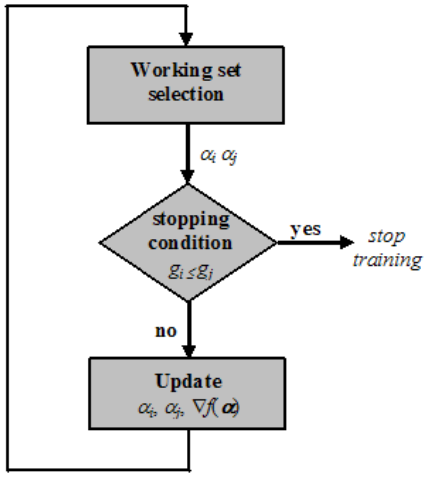

Fig. 1. The basic steps of the SVM training algorithm.

3.1 Training algorithm: reformulation and optimization

Keerthi's algorithm can be adapted by means of specific reformulations in the basic local-optimization steps, and the eventual optimization algorithm exploits a slightly modified heuristic. With respect to the original criterion formulated in the LibSVM library [7], the new heuristic proposed in this work replaces (3) with:

$$g_i \equiv \begin{cases} -\nabla f(\alpha)_i \text{ if } y_i = 1, \alpha_i < C \\ \nabla f(\alpha)_i \quad \text{if } y_i = -1, \alpha_i = C \end{cases}, \tag{4}$$

Thus, the new pattern-selection heuristic implied by (4) improves on computational efficiency by accepting the risk that a few patterns might violate one KKT condition. From a hardware-oriented viewpoint, the rationale behind this approach is that fulfilling entirely the KKTs results in a computationally demanding task. On the other hand, one runs the risk that the method might reach a sub-optimal solution. The reformulation (4) specifically aims to balance solution accuracy and computational efficiency.

From a cognitive viewpoint, one might justify the above modified heuristic in terms of region representation. The heuristic implied by (4) replaces the condition $\alpha_i > 0$ with $\alpha_i = C$; hence, only multipliers that are bounded support vectors are considered violators. In principle, one doesn't know how bounded support vectors and true support vectors are placed; however, in practice, a considerable percentage of the bounded support vectors that are violators with the original heuristic (3) will be subject to optimization process implied by (4). Indeed, true support vectors are not used in the optimization process; as a result, a loss of generalization ability is expected. Experimental evidence actually showed that the new heuristic embeds a sort of early stopping criterion.

The graphs in Fig.2 plot the dual cost (1) on the y-axis versus the number of iterations for two training sessions. The graphs compare the results obtained by training an SVM with the original criterion (gray line) with those resulting from the DSP-enhanced heuristic (4) (black line). Fig. 2a presents the costs for the "Pima-Indians Diabetes" testbed [9], whereas Fig. 2b is associated with tests on the "Sonar" testbed [10]. In both cases, the graphs show that the algorithm based on the DSP-enhanced selection strategy supports an early stopping criterion, as it terminates in a number of iterations that is significantly smaller than that required by the original criterion. Indeed, one notes that the training cost attained by the proposed algorithm is quite close to the optimal solution reached by the original algorithm. Further experiments on other testbeds confirmed these conclusions.

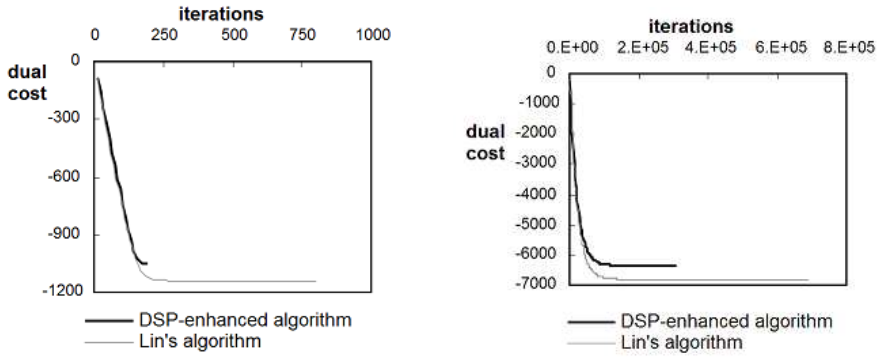

Fig. 2. Results on the "Diabetes" and "Sonar" datasets using the LibSVM criterion and the DSP-enhanced heuristic.

3.2 Training algorithm: basic porting

Step 3) indeed involves some critical aspects for the DSP implementation, as updating $\nabla f(\alpha)$ requires the access to the Hessian matrix Q. In general, the Hessian matrix needs an amount of memory that is available only in the memory external to the DSP. Thus, to implement the updating task efficiently, bandwidth becomes a critical aspect in transferring data from external RAM to the on-DSP memory.

The specific internal architectures of DSP devices allow one to attain optimal performance, as the Harvard schema provides separate, independent memory sections for program and data buses. Moreover, DMA channels allow independent memory loading from external to internal memory. Hence, in the proposed implementation, the DSP-internal RAM operates as a fast caching device, where matrix subsections are copied in turns. The eventual design strategy follows the architectural approach sketched in Fig. 2. The working set selection defines the

columns of the Hessian matrix involved in the update procedure; then, DMA-channels support data loading from external to internal memory. As a result, data transfer from the cache proceeds in parallel with ongoing computations. DMA channels are then used for data transfer from the external to the on-chip memory, so that the core can operate with limited latency. These features allow embedded implementations to exploit properly the limited amount of on-chip memory even when large data sets are involved. Incidentally, one notes that such a characteristic is hardly found on conventional microprocessors, which do not support directly addressable on-chip memory.

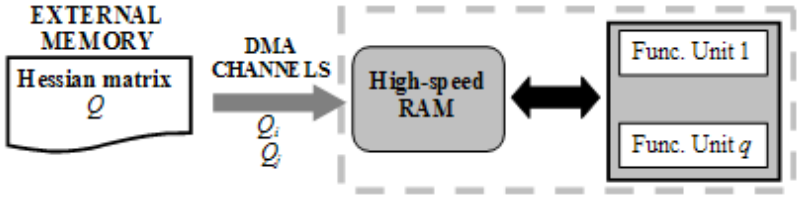

Fig. 3. The design strategy supporting data loading from the Hessian matrix.

4 Experimental Results

The proposed training algorithm has been implemented on an "ADSP-BF533 Blackfin" Analog Devices® DSP (denoted, in the following, as 'Blackfin' for brevity), running at a clock speed of 270 MHz. This device combine a 32-bit RISC instruction set with a dual 16-bit multiply accumulate (MAC) digital signal processing functionality. The memory architecture is hierarchical and provides a fast on chip memory (L1) and a slower off-chip memory. This platform supports DMA data transfer between internal and external data memory, thereby fulfilling the bandwidth condition for optimal data-transfer rates. Fixed-point representation brings about some loss in precision, as compared with the performance that could be attained by the higher resolutions provided by floating-point representations. Preliminary analysis pointed out that a 16-bit quantization level conveyed an acceptable degradation for the problem at hand. At the same time, the Q15 format representation allowed one to exploit the available 16-bit multipliers in parallel within the Blackfin DSP core.

The results presented in Table I compare the performances of the original training algorithm with the enhanced version implemented on the target DSP platform. To allow a fair comparison, training sessions were always performed by using the following parameter settings: C=1, $2\sigma^2$=100. Table 1 (a) and (b) compare the performances of the two algorithms on a random subset ($n_p = 100$) of four different testbeds: "Spam," "Ionosphere," "Banana," and "Pima-Indians

Diabetes" [9, 11]. The following quantities are reported: the number of iterations required by the training procedure, the dual cost (1) attained at convergence, the classification performance of error percentage (CE), the number of support vectors (SV), and the clock cycles required. The last column gives the results obtained by completing the SVM training on a high-end PC supporting the original heuristic included in LibSVM.

Empirical evidence confirms that the DSP-enhanced heuristic improves computational efficiency by attaining satisfactory performances in term of dual cost and digital cost. Experimental evidence shows that when implemented on a DSP LibSVM requires a computational effort that is five times larger than the effort required by the proposed heuristic. The proposed heuristic always outperforms the original heuristic in term of computational efficiency. Nonetheless, the dual cost and the classification error are very close to the reference values obtained by running LibSVM on a PC.

Table 1 Performance comparison between the enhanced algorithm and LibSVM.

	Spam Testbed			Ionosphere Testbed		
	Proposed heuristic (DSP)	LibSVM criterion (DSP)	LibSVM criterion (PC)	Proposed heuristic (DSP)	LibSVM criterion (DSP)	LibSVM criterion (PC)
Iterations	86	172	55	54	69	47
Dual cost	-31.894	-31.978	-31.916	-49.514	-49.707	-49.606
CE	1	1	1	12	12	12
SV	63	65	66	73	72	71
Clock cycles	598010	3194066	-	378150	1301331	-

	Banana Testbed			Diabetes Testbed		
	Proposed heuristic (DSP)	LibSVM criterion (DSP)	LibSVM criterion (PC)	Proposed heuristic (DSP)	LibSVM criterion (DSP)	LibSVM criterion (PC)
Iterations	52	97	51	49	49	49
Dual cost	-96.772	-96.649	-96.863	-92.043	-92.043	-91.828
CE	49	51	51	51	51	50
SV	99	98	98	98	98	98
Clock cycles	357755	1799341	-	334770	1254342	-

References

1. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
2. Cristianini, N. and Shawe-Taylor J.: An introduction to Support Vector Machines. Cambridge University Press, Cambridge, U.K. (2000).
3. Yixin Chen, and Wang, J.Z.: Support vector learning for fuzzy rule-based classification systems. IEEE Trans. Fuzzy System 11 (2003) 716-28
4. Chaves, Ad.C.F., Vellasco, M.M.B.R., and Tanscheit, R.: Fuzzy rule extraction from support vector machines. In: Proc. HIS05 (2005)
5. Joachims, T.: Making large-Scale SVM Learning Practical. In: Scholkopf, B., Burges, C.J.C. and Smola, A.J. (eds.): Advances in Kernel Methods - Support Vector Learning. MIT Press, Cambridge, MA (1999)
6. Keerthi, S., Shevade, S., Bhattacharyya, C., and Murthy, K.: Improvements to Platt's SMO algorithm for SVM classifier design. Neural Computation 13 (2001) 637-649
7. Chang, C.-C., and Lin, C.-J.: LIBSVM: a Library for Support Vector Machines. Available at: http://www.csie.ntu.edu.tw/ cjlin/libsvm.
8. Anguita, D., Boni, A., and Ridella, S.: A digital architecture for support vector machines: theory, algorithm and FPGA implementation. IEEE TNN 14 (2003) 993-1009
9. Rätsch, G., Onoda, T., and Müller, K.-R.: Soft margins for AdaBoost. Machine Learning 42 (2001) 287-320
10. Torres-Moreno, J.M., and Gordon, M.B.: Characterization of the sonar signals benchmark. Neural Processing Lett. 7 (1998) 1–4
11. Newman, D.J., Hettich, S., Blake, C.L., and Merz, C. J.: UCI repository of machine learning databases. Available at: http://www.ics.uci.edu/∼mlearn/MLRepository.html