Efficient Approximate Regularized Least Squares by Toeplitz Matrix

Sergio Decherchi, Paolo Gastaldo, and Rodolfo Zunino.

DIBE - Dept. Biophysical and Electronic Eng. - University of Genova, V. Opera Pia 11a – 16145, Genova – Italy
Email: { sergio.decherchi, paolo.gastaldo, rodolfo.zunino}@unige.it

Abstract

Machine Learning based on the Regularized Least Square (RLS) model requires one to solve a system of linear equations. Direct-solution methods exhibit predictable complexity and storage, but often prove impractical for large-scale problems; Iterative methods attain approximate solutions at lower complexities, but heavily depend on learning parameters. The paper shows that applying the properties of Toeplitz matrixes to RLS yields two benefits: first, both the computational cost and the memory space required to train an RLS-based machine reduce dramatically; secondly, timing and storage requirements are defined analytically. The paper proves this result formally for the one-dimensional case, and gives an analytical criterion for an effective approximation in multidimensional domains. The approach validity is demonstrated in several real-world problems involving huge data sets with highly dimensional data.

Keywords: Regularized Least Squares, Toeplitz matrix, Levinson-Trench-Zohar algorithm, Digital Signal Processor, Large Scale Learning, Resources Limited Device

1. Introduction

Large-scale learning represents a crucial topic in the research area of Machine Learning, and kernel methods are of particular interest for non-linear learning. Different approaches have been proposed to address such issue. Sequential Minimization Optimization (SMO) for Support Vector Machines (SVMs) (Platt, 1999), (Keerthi and Shevade, 2003) and the conjugate gradient method (Hestenes et Stiefel, 1952) represent well-known techniques that can prove useful for large-scale problems when high dimensional spaces are involved. An online recursive algorithm for training Support Vector Machines (SVMs) has been presented in (Cauwenberghs, and Poggio, 2001), while Nishida et al. (Nishida and Kurita, 2008) have introduced an heuristic training strategy based on the random selection of subset of patterns from a large dataset.

This work introduces a non-iterative method for the approximate solution of large-scale learning. The Regularized Least Squares (RLS) (Evgeniou et al., 2000) framework supports the learning principle. The rationale behind this choice is that RLS is a well-known and successful Machine-Learning algorithm, whose training procedure consists in solving a system of linear equations. Efficient solvers exist for the RLS paradigm; however, they cannot address effectively large-scale

problems. Solvers based on decomposition methods (Keerthi and Shevade, 2003) do not allow one to predict execution time or complexity easily. Direct-solution methods, which combine Gaussian elimination with a matrix factorization technique (Cholesky decomposition) (Press et al.,1992), exhibit predictable time and complexity, but suffer from two major drawbacks: first, the whole system matrix must be kept, hence storage requirement scales as $O(n^2)$, where n is the number of rows/columns of the linear system; secondly, the solver complexity scales with $O(n^3)$. Iterative methods (Hestenes et Stiefel, 1952) can outperform in speed the latter approaches in the presence of sparse matrixes, but still require the computation of the whole matrix; moreover, the speed-up is not predictable and heavily depends on the specific problem settings.

When tackling large-scale problems or when using devices with limited resources, one requires approximation schemes that can grant a trade-off between accuracy and computational complexity. Toeplitz matrix (Musicus, 1988) are particularly interesting, as the solution of a Toeplitz system with n variables reduces computational complexity to $O(n^2)$ and storage requirements to $O(n)$. Toeplitz solvers have been successfully used in Linear Predictive Coding, and in general where a Toeplitz system emerges, as in autocorrelation-based methods (Bäckström, 2004). This work exploits the properties of Toeplitz matrixes to significantly reduce both the computational cost and the memory space required to train an RLS-based machine.

The research presented in this paper first derives a sufficient condition that remaps the RLS learning problem into a Toeplitz system for one-dimensional problems; then, an approximation scheme for multivariate data is proposed. This general scheme balances efficiency versus accuracy and can be used to address large-scale classification problems effectively; indeed, it is showed that approximation accuracy is high as long as the kernel function leads to a kernel matrix that is very close to having a Toeplitz-based structure. The remarkable reduction of storage requirements and the exact predictability of memory usage represents in particular the crucial advantage provided by the present framework when compared with other approaches proposed in the literature (Hestenes et Stiefel, 1952); one should also consider that the method scales in memory as $O(n)$ using at the same time all the available patterns, differently from (Nishida and Kurita, 2008) or (Cauwenberghs, and Poggio, 2001). That feature makes actually the framework also amenable for limited-resource implementations involving embedded systems (Decherchi et al., 2006).

The paper is organized as follows: Section 2 introduces Regularized Least Squares learning, Section 3 analyzes the connections between RLS, Kernel Methods, and Toeplitz matrixes; this section also gives an operational approach to using Toeplitz matrixes for RLS-based Machine Learning. Section 4 presents the experimental verification of the approach, and Section 5 makes some concluding remarks.

2. Regularized Least Squares Learning

The learning problem underlying Regularized Least Squares (RLS) training can be formalized as follows: given a set, $\mathbf{X} = \{\mathbf{x}_i, i = 1,..,np\}$, of *np* samples and a vector, **y**, holding their associate labels ($y_i \in \boldsymbol{R}$), one wants to infer the function, $f(\mathbf{x})$, drawn from a family of admissible hypothesis, that associates the correct label with an unseen sample, **x**. The RLS method provides a powerful tool to solve this problem; its effectiveness and generalization ability rely on two main concepts:

- The function $f(\mathbf{x})$ belongs to a Reproducing Kernel Hilbert Space (RKHS) (Evgeniou et al., 2000).
- Regularization Theory is used as the conceptual basis (Evgeniou et al., 2000).

Given a Hilbert space, **H**, of functions *f:* $\mathbf{X} \rightarrow \boldsymbol{R}$, whose inner product is denoted as $\langle \cdot,\cdot \rangle_{\mathbf{H}}$, then **H** is a Reproducing Kernel Hilbert Space (RKHS) if, for every $\mathbf{x} \in \mathbf{X}$, **H** admits a function $K_{\mathbf{x}} \in \mathbf{H}$ that satisfies the following reproducing property:

$$f(\mathbf{x}) = \langle f, K_{\mathbf{x}} \rangle_{\mathbf{H}} \qquad f \in \mathbf{H} \tag{1}$$

The RKHS is uniquely defined by its kernel $K : \mathbf{X} \times \mathbf{X} \rightarrow \boldsymbol{R}$, $\forall$s,x $K(\mathbf{s},\mathbf{x}) = K_{\mathbf{x}}(\mathbf{s})$. The matrix, **K**, of elements $K(\mathbf{s},\mathbf{x})$ is symmetric and positive definite; moreover, each element $K(\mathbf{s},\mathbf{x})$ can be seen as a inner product $< \varphi(\mathbf{s}), \varphi(\mathbf{x}) >$ where $\varphi(\cdot)$ is a implicit non linear mapping function uniquely defined by **H**.

The Regularized Least Squares method minimizes a cost function; the cost sums a quadratic-loss term, which penalizes misclassified patterns, and a regularizing term, which favors 'smooth' solutions. In terms of the Structural Risk Minimization principle (Vapnik, 1998), this favors less complex functions. If one denotes with $\lambda > 0$ the regularization parameter, the minimization problem can be written as

$$\min_{f \in \mathbf{H}} \left\{ \sum_{i=1}^{np} (y_i - f(\mathbf{x}_i))^2 + \lambda \| f \|_{\mathbf{H}}^2 \right\} \tag{2}$$

The Representer Theorem (Evgeniou et al., 2000) shows that, in an infinite-dimensional RKHS space, **H**, the problem (2) has a solution, which can be expressed as an expansion series ruled by a vector of scalar coefficients, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_{np})$; the function $f(\mathbf{x})$ consequently can be written as

$$f(\mathbf{x}) = \sum_{i=1}^{np} \alpha_i K(\mathbf{x}, \mathbf{x_i}) \tag{3}$$

The coefficients α_i are obtained by substituting (3) in (2) and by minimizing the cost function. It can be shown (Evgeniou et al., 2000) that the desired vector of coefficients $\boldsymbol{\alpha}$ is obtained by solving the following system of linear equations:

$$(\mathbf{K} + \lambda \mathbf{I})\boldsymbol{\alpha} = \mathbf{y} \tag{4}$$

A first important aspect of the RLS method is that the prediction phase (3) involves a sum over all the *np* training patterns. That computation can prove impractical in large-scale problems. The literature offers sparsifying techniques such as the Reduced Set method (Burges, 1996) to shrink the number of training patterns actually used in (3), but these methods do not affect the complexity associated with the optimization process. The drawback represented by the non-sparse nature of the solution is counterbalanced by the simplicity of the learning process; this makes RLS appealing for the implementation of the learning procedure on low power, inexpensive embedded devices.

A second, crucial issue is represented by the structure of the matrix $\mathbf{K}$ in (4). If $\mathbf{K}$ is sparse, the system of equations (4) can be efficiently solved by an iterative method such as the Conjugate Gradient algorithm; however, $\mathbf{K}$ is usually dense, hence Conjugate Gradient loses much of its effectiveness. On the other hand, when $\mathbf{K}$ implies a linear kernel (or the kernel map $\varphi(\cdot)$ is explicitly known) and input data lie in a low-dimensional space, the Sherman-Morrison-Woodbury formula (Chua, 2003) can solve the system (4) efficiently.

In general, to solve (4) one typically adopts a direct method that combines Gaussian elimination with Cholesky decomposition (Press et al., 1992). This procedure involves a computational timing cost that scales as $O(np^3)$; moreover, keeping the entire matrix $\mathbf{K}$ in memory brings about a cost in storage that scales as $O(np^2)$. As a result, such an approach is not suitable when one has to tackle large-scale problems ($np \geq 10^4$). The following section shows the advantages of using Toeplitz systems in these cases for RLS learning, for both univariate and multivariate problems.

3. Toeplitz Matrixes for Regularized Least Squares

3.1. Toeplitz Linear Systems

A Toeplitz matrix, **T**, of size $n \times n$ is a diagonal-constant matrix. If the matrix is symmetric, n values completely specify **T**; thus, if one denotes with $\mathbf{k} \in \boldsymbol{R^n}$ the vector that contains the elements duplicated in each diagonal, one verifies that: $T_{i,j} = \mathbf{k}_{|i-j|}$. With these specifications, **k** spans the first row of the Toeplitz matrix, and the matrix takes the form:

$$\begin{pmatrix} k_0 & k_1 & k_2 & \dots & k_{n-2} & k_{n-1} \\ k_1 & k_0 & k_1 & \dots & \dots & k_{n-2} \\ \vdots & k_1 & k_0 & \dots & \dots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ k_{n-2} & \dots & \dots & \dots & \dots & k_1 \\ k_{n-1} & k_{n-2} & \dots & \dots & k_1 & k_0 \end{pmatrix} \tag{5}$$

Theory shows that any system of equations supported by a Toeplitz matrix, **T**, expressed as $\mathbf{T}\boldsymbol{\alpha} = \mathbf{y}$ can be solved efficiently by the Levinson-Trench-Zohar (LTZ) recursive algorithm (Musicus, 1988). The advantage of such an approach is that the complexity of the solution process scales as O(n^2) in time and as O(n) in memory, hence it outperforms any Gaussian elimination method. The LTZ algorithm is outlined in the pseudo-code given in the Appendix.

These features make the LTZ algorithm very appealing when one needs an efficient approach to solve (4). In the following, a theoretical analysis derives a sufficient condition to formulate (4) in terms of a Toeplitz matrix for one-dimensional problems, and an approximation scheme that, starting from a general kernel matrix **K**, yields a Toeplitz kernel **T** for multidimensional domains.

3.2 Toeplitz Kernels for Univariate RLS problems

Toeplitz matrixes naturally emerge when dealing with translation-invariant kernels and uniform (step-constant) sampling of univariate data. The following Lemma relates one-dimensional data distributions to Toeplitz matrixes.

Lemma 1: *Given a set, **X**, of mono-dimensional patterns drawn from uniform sampling, and a translation-invariant kernel such that: $K(u,v) = K(\|u - v\|)$, then the associate kernel matrix is a Toeplitz (symmetric) matrix.*

Proof: The assertion is proved by construction. Because of the uniform sampling, one can write the j-th sample as $x_j = x_0 + \Delta \cdot j$, where x_0 is the first sample of the set and Δ is the sampling step. To compute the kernel function one works out the element:

$$K_{i,j} = K(x_i, x_j) = K(|x_0 + i\Delta - (x_0 + j\Delta)|) = K(|i - j|\Delta) \tag{6}$$

In matrix form, this becomes:

$$\begin{pmatrix} K(0) & K(\Delta) & \dots & \dots & K((n-2)\Delta) & K((n-1)\Delta) \\ K(\Delta) & K(0) & K(\Delta) & \dots & \dots & K((n-2)\Delta) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ K((n-1)\Delta) & \dots & \dots & \dots & \dots & K(0) \end{pmatrix} \tag{7}$$

which is a Toepliz matrix like (5); this proves the assertion. **Q.E.D**.

It is easy to verify that, for any kernel matrix of Toeplitz type, $\mathbf{T}$, the system $(\mathbf{T}+\lambda\mathbf{I})\boldsymbol{\alpha}=\mathbf{y}$ is of Toeplitz type as well, hence a fast solution with the LTZ algorithm is attainable. This proves that regression problems can be solved exactly and rapidly by the RLS approach when dealing with univariate, uniformly sampled large training sets. The work by Steinke et al. (Steinke and Schölkopf, 2008) provides an example of application of Lemma 1. In that method, one builds a kernel matrix starting from a differential operator; if the differential operator is a full derivative one, then a Toeplitz kernel is obtained and Lemma 1 applies.

3.3. Approximated Toeplitz Systems for Multivariate RLS Problems

Tackling multivariate data makes it difficult, or even impossible, to set a sufficient condition ensuring that the kernel matrix always is in Toeplitz form. To benefit from the LTZ algorithm, one might yet approximate the original *np* x *np* kernel matrix, $\mathbf{K}$, by its nearest Toeplitz approximation, $\mathbf{T_K}$; the similarity between matrixes is ruled by a specified metric measure, *M*. This approach requires one to solve the following problem:

$$\min_{\mathbf{T_K}} \left\| \mathbf{T_K} - \mathbf{K} \right\|_M \tag{10}$$

with the constraint that $\mathbf{T_K}$ is positive semi-definite. To solve (10) one usually applies iterative algorithms that prove computationally expensive (Al-Homidan, 2002).

The research presented in the following yields an efficient approach that also gives an effective approximation scheme. The problem (10) has an analytical solution if one relaxes the constraint on the positive semi-definite property of $\mathbf{T_K}$, and the proximity is measured by the Frobenius norm, $\|\cdot\|_F$, of the difference matrix:

$$\left\| \mathbf{T_K} - \mathbf{K} \right\|_F = \left\| \mathbf{D} \right\|_F = \sqrt{\sum_i \sum_j | d_{ij} |^2} \tag{11}$$

Lemma 2 *Given a set, **X**, of multivariate samples, the solution of* $\min_{\mathbf{T_K}} \left\| \mathbf{T_K} - \mathbf{K} \right\|_F$ *is attained by the matrix* $\mathbf{T_K}$ *that is built by setting all elements of each constant-value diagonal to the mean value of the corresponding diagonal in* $\mathbf{K}$.

Proof: First, one unrolls in a diagonal-wise fashion the matrix $\mathbf{K}$, such that each diagonal is concatenated to each other. Considering the symmetry of $\mathbf{K}$ one only concatenates the upper diagonals; indicating by d_i each diagonal the resulting vector $\mathbf{v_K}$ is of length $(np(np+1))/2$.

$$\mathbf{v_K} = \left\{ \underbrace{K_{0,0}, K_{1,1}, ..., K_{np-1,np-1}}_{d0}, ..., \underbrace{K_{0,j}, K_{1,j+1}, ..., K_{np-1-j,np-1}}_{dj}, ..., \underbrace{K_{0,np-1}}_{d(np-1)} \right\}$$

Then one performs the same unrolling procedure for the Toeplitz matrix, $\mathbf{T_K}$, and builds $\mathbf{v_T}$

$$\mathbf{v_T} = \left\{ \underbrace{T_{0.0}, T_{0,0}, ..., T_{0,0}}_{d0}, ..., \underbrace{T_{0,j}, T_{0,j}, ..., T_{0,j}}_{dj}, ..., \underbrace{T_{0,np-1}}_{d(np-1)} \right\}$$

the difference vector $\boldsymbol{\delta} = \mathbf{v_K} - \mathbf{v_T}$ is:

$$\boldsymbol{\delta} = \left\{ \begin{array}{l} \underbrace{K_{0,0} - T_{0.0}, K_{1,1} - T_{0.0}, ..., K_{np-1,np-1} - T_{0.0}}_{d0}, ..., \\ \underbrace{K_{0,j} - T_{0,j}, K_{1,j+1} - T_{0,j}, ..., K_{np-1-j,np-1} - T_{0,j}}_{dj}, ..., \\ \underbrace{K_{0,np-1} - T_{0,np-1}}_{d(np-1)} \end{array} \right\}$$

Finally the problem $\min_{\mathbf{T_K}} \left\| \mathbf{T_K} - \mathbf{K} \right\|_M$ reduces to find $\min_{\mathbf{T_K}} \| \boldsymbol{\delta} \|_2^2$ where $\| \boldsymbol{\delta} \|_2^2 = \left\{ \sum_{j=0}^{np-1} \underbrace{\sum_{i=0}^{np-1-j} (K_{i,j+i} - T_{0.j})^2}_{dj} \right\}$. All terms in the summation are positive or zero, therefore the minimum is attained when each term in round brackets is minimum: however due to the Toeplitz diagonal-constant structure one has to consider together all the terms within each diagonal. Given a diagonal *j* then the minimum of $\underbrace{\sum_{i=0}^{np-1-j} (K_{i,j+i} - T_{0.j})^2}_{dj}$, is attained when $T_{0,j}$ is the sample mean of the diagonal *j* of the original matrix $\mathbf{K}$. This observation holds for each diagonal. **Q.E.D.**

Lemma 2 offers two principal advantages: first, the solution of problem (10) is expressed analytically; secondly, to work out $\mathbf{T_K}$ one should only compute the mean values of each diagonal of the matrix $\mathbf{K}$. Once the mean value of a diagonal is computed, the associate memory can be de-allocated and re-used; this leads to a memory occupation that scales as O(*n*).

The approximation scheme based on Lemma 2 is most effective when the original kernel matrix $\mathbf{K}$, has an "almost-Toeplitz" structure, so that alterations in the diagonal elements marginally distort the overall information carried by $\mathbf{K}$. To evaluate the accuracy attained by $\mathbf{T_K}$ and to measure the distortion brought about by the approximation, one should ultimately estimate the error performed on a test set.

The relaxation of the constraint on the positive semi-definiteness of $\mathbf{T_K}$ is of minor importance in practice, for several reasons. First, even if $\mathbf{T_K}$ is indefinite (that is a matrix neither positive- nor negative semi-definite), the regularization constant, $\lambda > 0$, that is added to each term in the main diagonal contributes to make the matrix positive semi-definite (singularities are in fact very unlikely). Secondly, if the matrix $[\mathbf{T_K} + \lambda\mathbf{I}]$ still results indefinite, it can be shown that the associate RLS training problem is equivalent to learning in a Reproducing Kernel Krein Space (Ong et al.,2004). Theory shows (Ong et al.,2004) that learning is possible in such a space, and Rademacher bounds to the generalization error can be estimated accordingly (Ong et al.,2004). Finally, the Look-Ahead-Levinson algorithm (Chan and Hansen, 1992), which is a version of the LTZ method, can effectively deal with indefinite Toeplitz matrixes almost without extra costs.

The advantage of the Toeplitz-based approach becomes apparent when one considers the analysis summarized in Table 1, where the features of the conventional and of the Toeplitz-based solutions of the RLS learning problem are compared (for the conjugate gradient a underestimation of the computational cost is given). The crucial aspect is that the sharp reduction in all requirements makes the direct-solution approach viable for large data sets, which would otherwise prove inaccessible.

Table 1

Comparison among Gaussian Elimination (GE), Conjugate Gradient (CG) and the Toeplitz-based (TB) solution approaches to RLS learning: *n* is the number of patterns and *k* is the number of CG iterations.

	GE	**CG**	**TB**
Computational complexity	$O(n^3)$	$O(kn^2)$	$O(n^2)$
Storage requirement	$O(n^2)$	$O(n^2)$	$O(n)$

3.4. Effects of RBF Kernels on Generalization

The distortion introduced by the Toeplitz kernel, $\mathbf{T_K}$, in approximating **K** affects the accuracy of the classifier machine. Therefore, the kernel function should ensure a satisfactory trade-off between accuracy and computational complexity. This section shows that the Radial-Basis-Function (RBF) kernel can accomplish this requirement and is suitable for Toeplitz approximations.

The kernel formulation, $K(\mathbf{u},\mathbf{v}) = \exp\left(-\|\mathbf{u}-\mathbf{v}\|^2/(2\sigma^2)\right)$, implies that inner products vary in the range (0,1]. By varying the kernel parameter, σ, one can drive the numerical distribution of similarity results towards either extremum of the range. Let $\varphi(\mathbf{u})$, $\varphi(\mathbf{v})$ the non-linear mappings induced by the Gaussian kernel on pattern vectors **u**, **v**, respectively; then, the two extreme situations can be represented as follows (Fig. 1):

- $\sigma \to 0$: then $K(\mathbf{u},\mathbf{v}) \to 0 \ \ \forall \mathbf{u},\mathbf{v}$; thus all distances in the Hilbert space collapse to a constant $\| \varphi(\mathbf{u}) - \varphi(\mathbf{v}) \|^2 = 2 \cdot [1 - K(\mathbf{u},\mathbf{v})] \cong 2$. The images of all patterns in the infinite- dimensional space lay on a hyper-sphere, and the kernel matrix tends to the identity matrix.
- $\sigma \to \infty$, then $K(\mathbf{u},\mathbf{v}) \to 1 \ \ \forall \mathbf{u},\mathbf{v}$; all distances in the feature space collapse to $\| \varphi(\mathbf{u}) - \varphi(\mathbf{v}) \|^2 = 2 \cdot [1 - K(\mathbf{u},\mathbf{v})] \cong 0$. All images collapse onto one point and the entries in the kernel matrix are all set to 1.

In both those situations, one expects the resulting generalization performance to be far from optimal, due to over-fitting in the former case, and over-smoothing in the latter. It is however intriguing that, in both cases, the kernel matrix yet tends to a Toeplitz matrix. The case $\sigma \to 0$ may be especially interesting because images do not concentrate in one point of the kernel space, and the key issue is to set σ to a value that is small enough to drive $\mathbf{K}$ toward a Toeplitz matrix and, at the time, to ensure that over-fitting is not severe. Thus the Gaussian kernel provides a suitable kernel function which gives an effective generalization ability and also benefits from a Toeplitz-like structure matrix.

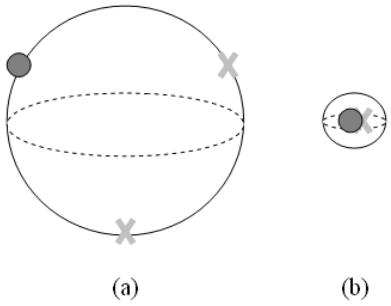


Figure 1. Kernel space data representation (crosses are +1 data and balls are -1 data): (a) is the case $\sigma \to 0$, (b) is the case $\sigma \to \infty$

4. Experimental Results

4.1 Experimental Setup

The experimental session of the presented research aimed at: 1) checking the consistency of the approximated solution, 2) assessing the gap in accuracy between classical solvers and the approximated one; 3) evaluating the average speed-up attained by the Toeplitz-based approach. To achieve those goals, the proposed framework was tested on several classification problems, and compared with both a conventional solver (Press et al., 1992) and a solver based on the conjugate-gradient method (Hestenes et Stiefel, 1952). As the approximation scheme may coarsely alter some elements of the original kernel matrix, classification problems represent a suitable applicative

domain for a learning strategy that addresses a trade-off between accuracy and complexity.

Experimental verifications involved real-world, binary datasets, namely: Covertype, Ijcnn, w8a (Chang and Lin, 2001), Daimler (Munder and Gavrila, 2006), Manuscript NIST (3 vs 8) (Hettich and Bay, 1999), and MIT face database (CBCL Face Database). In all cases, the patterns were shuffled, each coordinate was normalized into the range [-1,+1], and each data set was randomly split into a training set and a test set. Table 2 gives the partitioning criteria and the number of variables for each dataset in the experiments. For each dataset, a grid-based model selection tuned kernel parameters; in the following, σ^C and σ^T will denote the best kernel parameter for the classical and the Toeplitz-based approach, respectively. Optimal settings for parameters λ and σ have been set by adopting a conventional cross-validation strategy based on a test set.

Table 2 - Data splitting criteria for the data sets used in the experiments and number of variables. The numbers of patterns in the table are intended multiplied by 10^3

Dataset	#Training set	#Test set	#Variables
MNist3vs8	1,5,10,20	4	80
Ijcnn	1,5,10,20,50,100	5	22
Daimler	1,2,5,7	1	648
w8a	1,5,10,20,40	5	300
Covertype	1,5,10,20,50,100	5	54
Faces	1,5,10,20	1	361

The tests have been implemented in a Matlab 2009a environment: source files were C-coded and compiled as 'mex' files. The standard Gaussian Elimination solver and the standard conjugate gradient solver of the Matlab environment have been used; the latter was implemented by exploiting the default stopping criterion and by setting 100 as the maximum number iterations. The LTZ algorithm for the approximated solution was coded with no parallelism.

4.2 Result Analysis

The graphs in Figure 2 give, for each testbed, the classification error scored by the 'best' model on the test set, for a varying number of training patterns. In each graph, the solid black line gives the best test error attained by the approximation method, the dashed black line marks the best test error obtained by the classical linear solver, and the grey line is the best test error obtained by the conjugate-gradient (CG) method. Figure 3 gives the speed-up factors obtained by the Toeplitz-based RLS method with respect to the comparison strategies.

As predicted by theory, the approximation method proved very effective whenever the optimal parameter setting, σ^C, lead to a matrix that was close to a Toeplitz matrix; this situation is

exemplified by the Covertype and Daimler datasets; in both cases, the approximated model almost matched the classical method.

The conventional linear-system solver and conjugate gradient required to allocate the entire kernel matrix, whose storage cost scaled as $O(n^2)$. This set severe limitations for those problems involving more than $2 \cdot 10^4$ patterns, as memory storage for the kernel matrix exceeded 3GB; memory occupation further increased when taking into account the overhead brought about by the linear system solver (e.g. Cholesky decomposition) and the dataset matrix. This made memory storage quite a demanding constraint even on powerful machinery. Conversely, the experiments showed that the approximated method required less than 300 MB of RAM, even for datasets including more than 10^5 patterns.

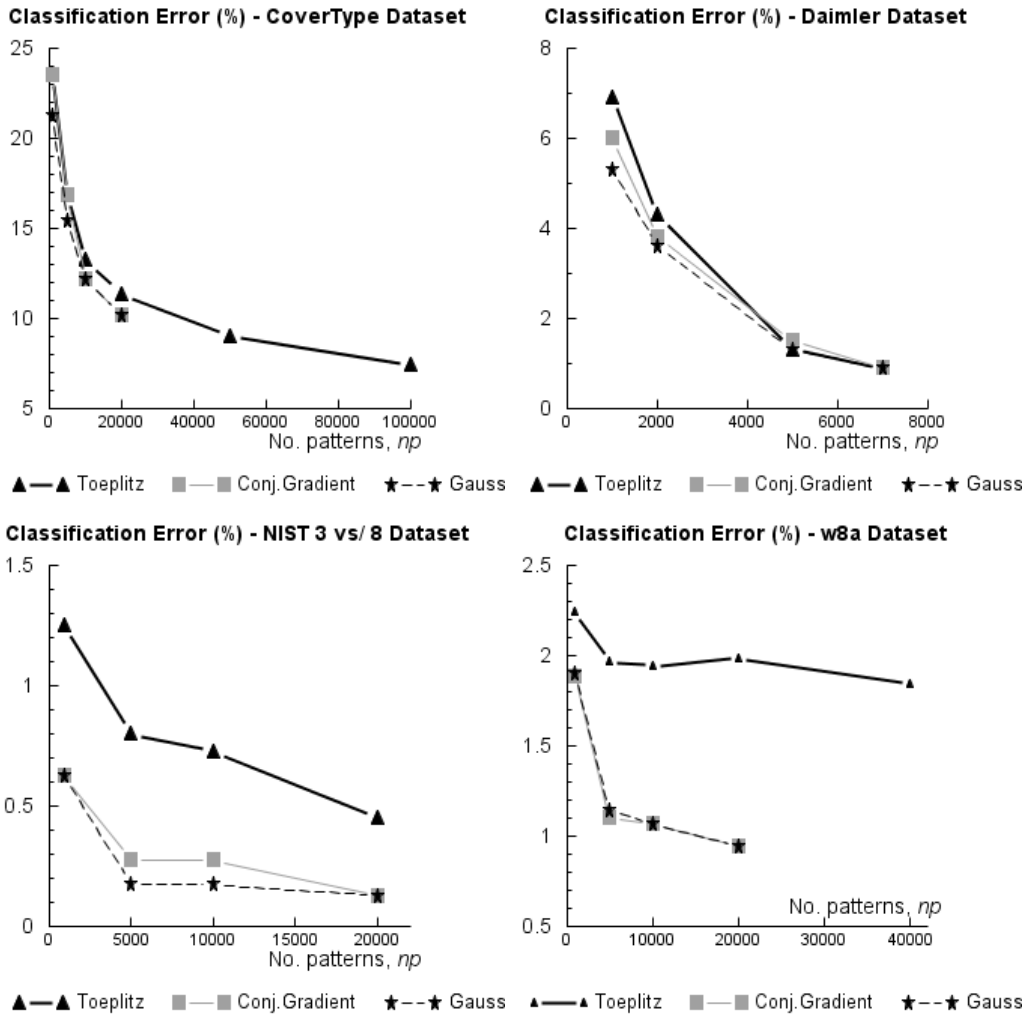
The Toeplitz-based approach made it possible to train classifiers also with large data sets including 50,000 and 100,000 patterns, as was the case for Covertype; in those situations, empirical evidence showed that the best model obtained by the approximated method outperformed the "exact" solution obtained by classical methods (which had been trained on fewer patterns). In other cases, the approximated approach yielded a sub-optimal solution, although classification errors always kept within a reasonable range.

The (successful) event $\sigma^C \cong \sigma^T$ could be observed when the size of the training sets reached 10^3 patterns or more. Whenever σ^C differed significantly from σ^T, the results on large-scale data sets did not match those attained by classical solutions (even with fewer patterns); conversely, whenever $\sigma^C \cong \sigma^T$, the approximation for large-scale learning proved effective. These considerations provide a designer with an operative criterion to verify the advantage of the approximation scheme in tackling large-scale problems or in supporting limited-resource implementations.

Speed-up values always proved very satisfactory, also considering that the classical Matlab Gaussian Elimination solver could benefit from a parallel implementation whereas the LTZ algorithm version was not parallelized. An important remark concerns the analysis of timing results, as the reported speed-up values only took into account the computational process involved by the solution of the linear system. Therefore, one might argue that, if one also considers the time required to work out the kernel matrix, speed-up values should be adjusted and would decrease accordingly. Actually, the total number of kernel evaluations is constant and scales exactly as $O(l \cdot n^2)$ (where l is the number of variables); therefore it does not compromise the advantage in complexity that is conveyed by the Toeplitz-based approximation. At the same time, modern technology approaches to kernel matrix computation make that process easily parallelizable (Catanzaro et al., 2008); on the contrary, parallelization of the optimization engine is a difficult task.

The experimental results presented in Fig. 3 confirmed the expected behaviors in terms of computational complexity as per Table 1. The speed-up provided by the Toeplitz solver over the conjugate-gradient method was roughly proportional to the number of iterations of the CG algorithm. Actually, in half of the experiments the CG didn't attain the required solution accuracy before reaching the maximum number of iterations. Thus, the average speed-up provided by the proposed strategy is somewhat underestimated.

A comparative analysis with iterative methods for SVM (Keerthi and Shevade, 2003) showed that the computational performances attained by the latter ones were heavily affected by the specific values of the regularizing parameter, λ (*C* SVM parameter in that case), whereas the performances of the Toeplitz-based method are guaranteed to keep constant and independent of that quantity. Likewise, the CG was heavily influenced by both parameters λ and σ: the computational complexity of CG increases when σ takes on high values and λ takes on low values. At the same time, empirical evidence pointed out that the accuracy provided by the approximated method matched satisfactorily the accuracy attained by iterative methods (CG), especially when large data set were involved.



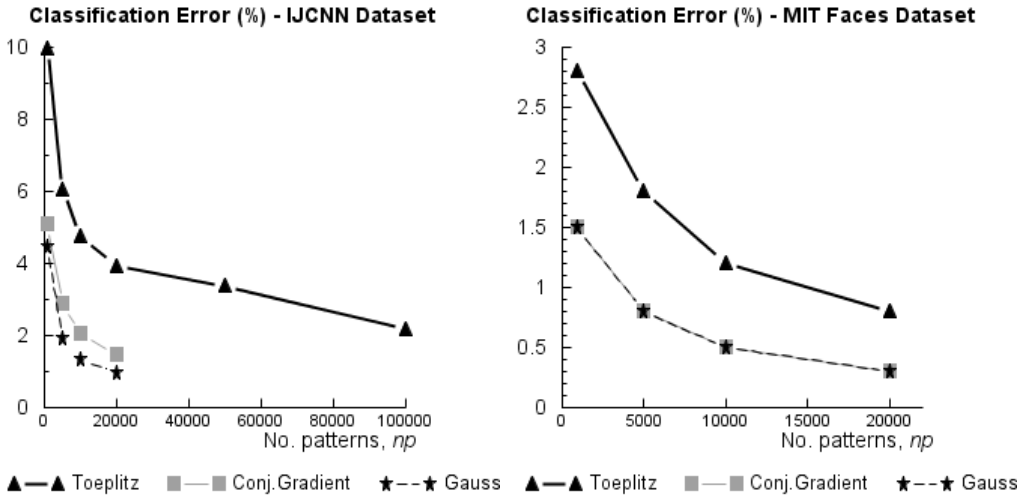
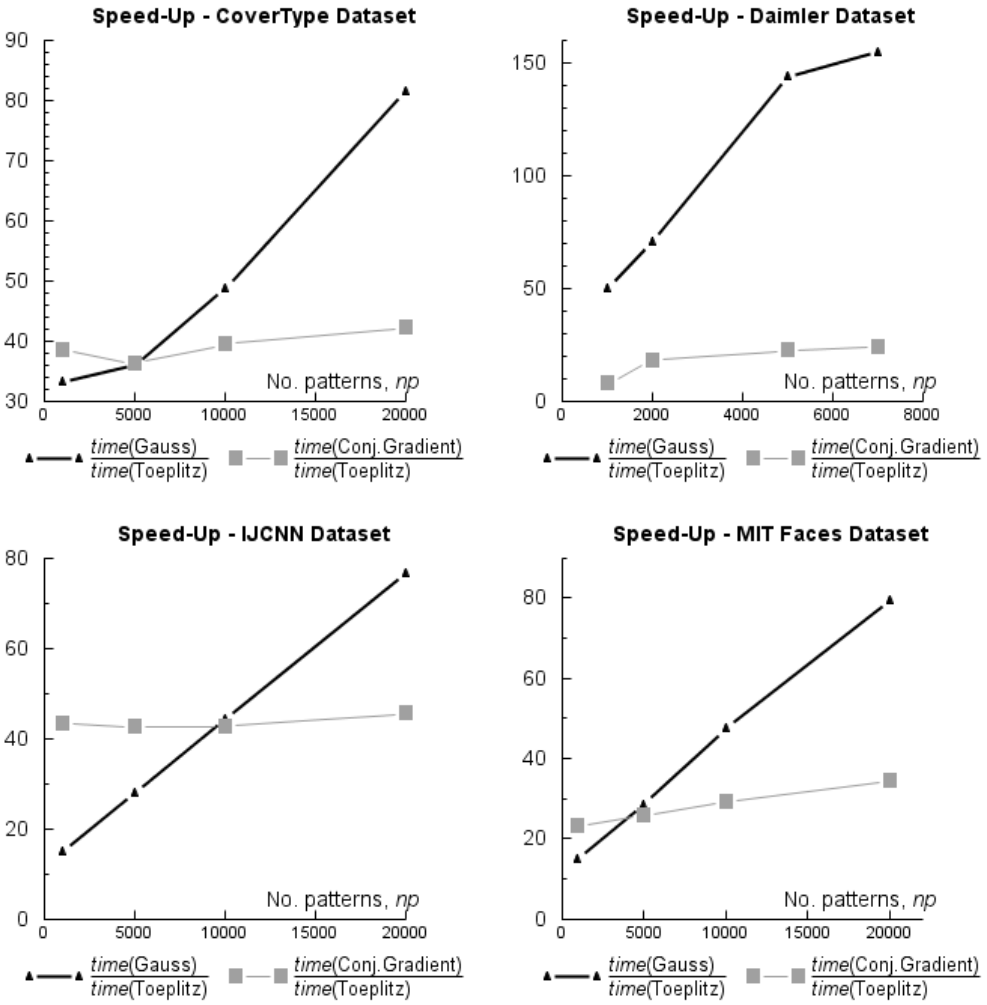


Figure 2. Classification Error (%) comparison for Gaussian Elimination, Conjugate Gradient and Toeplitz Approximation



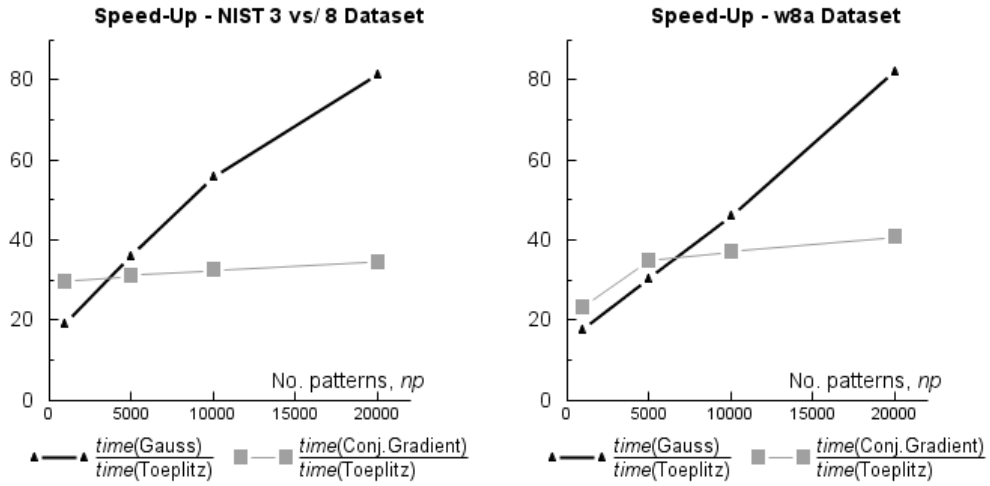


Figure 3. Experimental results for the RLS Toeplitz-based acceleration.

4.3 A Practical Procedure to Validate the Approximation for Large Datasets

The theoretical analysis and the experimental verifications point out that the use of an RBF kernel most likely enhances the Toeplitz-based scheme in RLS learning tasks. In practical applications, one is interested in the accuracy of the eventual system set-up, and the effectiveness of the Toeplitz-based approach ultimately depends on the consistency of the model-selection process. In other words, one would know in advance whether the approximated schema yields classification results that are close to the solution attained by direct methods.

The above discussion and the analysis of empirical evidence provides an operational, reliable procedure to assess the validity of the approximation in a practical fashion, and is especially useful in large-scale training problems.

The method baseline is the verification that, whenever the approximated model parameter proves close to the 'correct' value ($\sigma^T \cong \sigma^C$) for a relatively small training set, then this property also holds for the larger data set; thus one has $\mathbf{T_K} \cong \mathbf{K}$ and the Toeplitz-based approach dramatically simplifies the training task on the actual domain. The operational procedure is therefore the following.

1. Input: a large set, **X**, of training patterns;
2. Subsample **X** and assemble a reduced training set, **X'**, holding a number of patterns, *np*', that can be managed by direct-solution methods (e.g. $np' \propto 10^3$);
3. Perform a model selection process on **X'** by using both the direct-solution and the Toeplitz-based method, yielding model parameters $(\sigma^C)'$ and $(\sigma^T)'$, respectively;
4. if model selection is consistent, i.e., $(\sigma^C)' \cong (\sigma^T)'$,

   train an RLS machine on the entire set, **X**, by using the Toeplitz-based method;

otherwise the approximated method might not apply.

The advantage induced by the low computational complexity of the approximation scheme makes the above procedure also valid when dealing with limited-resource devices, since in those circumstances one has to use an algorithm that can support the learning process on the target device.

5. Concluding Remarks

The paper has analyzed the application of Toeplitz-related algorithms to the training problems involved by Regularized Least Squares. When considering the associate linear-system problem setting, the basic advantage of the proposed approach lies in the dramatic reduction in both computational complexity and storage requirements involved by a Toeplitz-based problem formulation.

The paper has proved a sufficient condition that yields a Toeplitz kernel matrix for mono-dimensional problems, with the result of a very efficient learning algorithm leading to the exact solution. An approximated problem setting has been described for the general case of multivariate domains, where the role of RBF kernels has been analyzed.

The theoretical analysis and experimental results have shown that, whenever the actual model implies a kernel matrix that is close to the approximating Toeplitz matrix, the proposed approach attains a marginal degradation in accuracy and, by contrast, allows one to tackle large-scale problems that would have been otherwise inaccessible. The approximation scheme is also appealing for embedded implementations, involving for instance low-cost DSPs, that are strongly constrained in resources and that can tolerate sub-optimal accuracy performances.

Future works will concern the development of other approximate light-weighted learning schemes for kernel machines and an optimized Digital Signal Processing based implementation of the proposed method.

Appendix

The following pseudo-code outlines the LTZ algorithm. The nesting level of the innermost loop is 2, thus confirming the overall timing complexity $\mathrm{O}(N^2)$. Only vectors of length at most N are stored, thus conveying a $\mathrm{O}(N)$ complexity in storage space.

Input: **k, y**
Output: $\boldsymbol{\alpha}$
Vector Variables: $\boldsymbol{\gamma}$, $\bar{\boldsymbol{\gamma}}$, $\boldsymbol{\tau}_a$, $\boldsymbol{\tau}_b$,of size N.-1; $\mathbf{a}$, $\mathbf{b}$ of size N (all vectors entries initialized to 0)
$\mathbf{a}(0)=1$, $\mathbf{b}(0)=1$, $\varepsilon=\mathbf{k}(0)$, $\boldsymbol{\alpha}(0)=\mathbf{y}(0)/\varepsilon$

FOR *n=1* ***TO*** *n=N-1* ***STEP*** *n=n+1*

FOR *i=0* ***TO*** *i=n-1* ***STEP*** *i=i+1*

$\boldsymbol{\gamma}(i) = \mathbf{k}(n-i)\ ,\ \overline{\boldsymbol{\gamma}}(i) = \mathbf{k}(i+1)$

END FOR

$\delta = 0$

FOR *i=0* ***TO*** *i=n-1* ***STEP*** *i=i+1*

$\delta = \delta + \boldsymbol{\gamma}(i) * \mathbf{a}(i)$

END FOR

$\xi = -\delta / \varepsilon$

$\delta = 0$

FOR *i=0* ***TO*** *i=n-1* ***STEP*** *i=i+1*

$\delta = \delta + \overline{\boldsymbol{\gamma}}(i) * \mathbf{b}(i)$

END FOR

$\nu = -\delta / \varepsilon$

FOR *i=0* ***TO*** *i=n-1* ***STEP*** *i=i+1*

$\boldsymbol{\tau}_a(i) = \mathbf{a}(i)$

$\boldsymbol{\tau}_b(i) = \mathbf{b}(i)$

END FOR

$\mathbf{b}(0) = \nu\, \boldsymbol{\tau}_a(0)$

FOR *i=1* ***TO*** *i=n-1* ***STEP*** *i=i+1*

$\mathbf{a}(i) = \boldsymbol{\tau}_a(i) + \xi\, \boldsymbol{\tau}_b(i-1)$

$\mathbf{b}(i) = \boldsymbol{\tau}_b(i-1) + \nu\, \boldsymbol{\tau}_a(i)$

END FOR

$\mathbf{a}(n) = \xi\, \boldsymbol{\tau}_b(n-1)$

$\mathbf{b}(n) = \boldsymbol{\tau}_b(n-1)$

$\varepsilon = \varepsilon(1 - \xi\nu)$

$\delta = 0$

FOR *i=1* ***TO*** *i=n-1* ***STEP*** *i=i+1*

$\delta = \delta + \boldsymbol{\gamma}(i) * \mathbf{x}(i)$

END FOR

$\mu = \mathbf{y}(n) - \delta$

FOR *i=1* ***TO*** *i=n-1* ***STEP*** *i=i+1*

$\mathbf{x}(i) = \mathbf{x}(i) + (\mu / \varepsilon)\, \mathbf{b}(i)$

END FOR

$\mathbf{x}(n) = (\mu / \varepsilon)\, \mathbf{b}(n)$

END FOR

References

Evgeniou, T., Pontil, M., Poggio, T., 2000. Regularization Networks and Support Vector Machines, Journal Advances in Computational Mathematics, Springer, vol. 13(1), pp. 1-50.

Keerthi ,S. S., Shevade, S. K. 2003. SMO Algorithm for Least-Squares SVM Formulation. Neural Computation, MIT Press,vol. 15(2), pp. 487-507.

Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T. 1992. Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed. Cambridge, England: Cambridge University Press.

Musicus, B. R., 1988. Levinson and Fast Cholesky Algorithms for Toeplitz and Almost Toeplitz Matrices., RLE TR No. 538, MIT, 1988.

Vapnik, V. 1998. Statistical Learning Theory. Wiley-Interscience Pub., New York.

Burges, C. J. C. 1996. Simplified support vector decision rules. In: Proceedings 13th International Conference on Machine Learning, Bari, Italy , pp 71-77.

Chua K. S., 2003. Efficient computations for large least square support vector machine classifiers. Pattern Recognition Letters, vol. 24(1-3), pp. 75-80.

Steinke, F., Schölkopf , B. 2008. Kernels, regularization and differential equations. Pattern Recognition, Elsevier Science Inc. New York, NY, USA, vol. 41(11),, pp. 3271-3286.

Al-Homidan S.S 2002. SQP algorithms for solving Toeplitz matrix approximation problem. Numerical Linear Algebra with Applications, vol.9(8), pp. 619 – 627.

Ong, C. S. , Mary, X. , Canu, S., Smola, A. J. 2004. Learning with Non-Positive Kernels. In: International Conference of Machine Learning

Chan, T.F., Hansen, P.C. 1992. A look-ahead Levinson algorithm for general Toeplitz systems. IEEE Transactions on Signal Processing, vol.40(5), pp. 1079-1090.

Chang, C.C., Lin, C.J. 2001. LIBSVM : A Library for Support Vector Machines. Datasets available: http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/.

Munder, S., Gavrila, D.M. 2006. An Experimental Study on Pedestrian Classification. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, pp. 1863-1868.

Hettich, S. , Bay, S. D. 1999. The UCI KDD Archive, [http://kdd.ics.uci.edu]. University of California, Dept. Inf. and Comput. Sci., Irvine, CA.

CBCL Face Database #1, MIT Center For Biological and Computation Learning http://www.ai.mit.edu/projects/cbcl

Catanzaro, B. , Sundaram, N. , Keutzer, K. 2008. Fast support vector machine training and classication on graphics processors. In ICML '08: Proceedings of the 25th international conference on Machine learning, pp. 104-111. Software available at http://www.cs. berkeley.edu/~catanzar/GPUSVM/.

Bäckström, T.2004. Linear Predictive Modelling of Speech -- Constraints and Line Spectrum Pair Decomposition. Doctoral thesis. Report no. 71 / Helsinki University of Technology.

Hestenes, Magnus R., Stiefel, Eduard .1952. Methods of Conjugate Gradients for Solving Linear Systems. Journal of Research of the National Bureau of Standards 49 (6) http://nvl.nist.gov/pub/nistpubs/jres/049/6/V49.N06.A08.pdf.

Nishida, K., Kurita, T., RANSAC-SVM. 2008. for Large-Scale Datesets, Proc. of International Conference on Pattern Recognition, 2008, Tampa Convention Center, Tampa, FL, USA, 2008.12.

Cauwenberghs, G. and Poggio, T. 2001. Incremental and Decremental Support Vector Machine Learning. In: Leen, T.K., Dietterich, T.G., and Tresp, V. (eds) Advances in Neural Information Processing Systems, vol. 13, pp. 409-415. MIT Press

Platt, J.C. .1999. Fast training of support vector machines using sequential minimal optimization

Decherchi S., Parodi G., Gastaldo P. and Zunino R.. 2006. Embedded Electronics System for Training Support Vector Machine, IJCNN 2006, Vancouver.