Pattern Recognition Letters 34 (2013) 439–451

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

An algorithm for training a large scale support vector machine for regression based on linear programming and decomposition methods

Pablo Rivas-Perea [a,*], Juan Cota-Ruiz [b]

[a] Department of Computer Science, Baylor University, One Bear Place 97356. Waco, TX 76798-7356, United States
[b] Department of Electrical and Computer Engineering, Autonomous University of Ciudad Juarez (UACJ), Ave. del Charro #450 Nte. C.P. 32310. Ciudad Juarez, Chihuahua, Mexico

ARTICLE INFO

Article history:
Available online 16 November 2012

Keywords:
Support vector machines
Support Vector Regression
Linear programming
Interior point methods

ABSTRACT

This paper presents a method to train a Support Vector Regression (SVR) model for the large-scale case where the number of training samples supersedes the computational resources. The proposed scheme consists of posing the SVR problem entirely as a Linear Programming (LP) problem and on the development of a sequential optimization method based on variables decomposition, constraints decomposition, and the use of primal–dual interior point methods. Experimental results demonstrate that the proposed approach has comparable performance with other SV-based classifiers. Particularly, experiments demonstrate that as the problem size increases, the sparser the solution becomes, and more computational efficiency can be gained in comparison with other methods. This demonstrates that the proposed learning scheme and the LP-SVR model are robust and efficient when compared with other methodologies for large-scale problems.

1. Introduction

The Support Vector-based learning field is very active (see Hu et al., 2004; Bo et al., 2007; Shen et al., 2010; Mangasarian et al., 1999; Collobert and Bengio, 2004; Papageorgiou et al., 1999). As a consequence, the treatment of large-scale SVM is of interest to researchers in the field (see Drucker et al., 1996; Osuna et al., 1997; Scholkopf and Smola, 2002; Hazan et al., 2008; Du et al., 2009; Zhu and Sung, 2005; Debnath and Takahashi, 2006; Yan-zi and Hua, 2009; Huang and LeCun, 2006; Yongping and Dongtao, 2002; Nishida and Kurita, 2008; Papadonikolakis and Bouganis, 2008; Chen et al., 2008; Zhang et al., 2005). However, a number of open issues are yet to be addressed. Certainly, SVR algorithmic development seems to be at a more stable stage recently; but in spite of this, one open issue is to find whether linear programming (LP) SVR approaches will lead to more satisfactory results or other useful training strategies (Smola and Scholkopf, 2004; Zhang and Zhou, 2010). These strategies demand the usage of optimization techniques developed under the context of SVRs, as they could facilitate the treatment of large data sets (Smola and Scholkopf, 2004). This may be done in combination with reduced set methods for speeding up the training phase for large data sets. This topic is of huge importance as machine learning applications demand algorithms that are capable of dealing with data sets that are at least larger than 1 million samples (Smola and Scholkopf, 2004).

In this research the authors introduce Algorithm 1 as an alternative to address the particular issue of large-scale training of an LP-SVR. In the remaining sections of this paper, Algorithm 1 will be explained step by step, which involves three major parts that increases computational tractability of large-scale problems. First, the Large-Scale Linear Programming Support Vector Regression (LP-SVR) problem is reduced by variable decomposition, which exploits LP-SVR structure to produce a lower-dimensional representation of the decomposed LP sub-problem. The finite termination of the decomposition strategy is guaranteed by an adaptation of Torii and Abe (2009) infinite-loop prevention algorithm. Second, the authors will design a constraint decomposition strategy that generates a number of smaller sub-problems by again exploiting LP-SVR structure. The resulting LP problems are solved sequentially in a monotonically non-increasing objective function fashion. Convergence is guaranteed by adapting Bradley's et al. (Bradley and Mangasarian, 2000; Bradley et al., 2002) theorems. Bradley's et al. algorithm is known as Linear Programming Chunking (LPC), and it is depicted in Fig. 1. As shown in the figure, the algorithm consist on dividing an LP into blocks of smaller LPs without taking any advantage of the SVR structure.

Third, the two-way-reduced LP-SVR sub-problems (*i.e.*, variables and constraint reduction) are solved using interior point methods, which have a very fast rate of convergence. This is the key that balances the total computational time of performing two decompositions and solving several LPs. Using any other

* Corresponding author.
E-mail addresses: Pablo_Rivas_Perea@Baylor.edu (P. Rivas-Perea), jcota@uajc.mx (J. Cota-Ruiz).

0167-8655/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2012.10.026

approach (*i.e.*, simplex methods) would dramatically increase computational efforts. The method by Argáez and Velázquez (2003) is used.

Algorithm 1: Variables and Constraints Decomposition Strategy for a Large Scale Training Set

Require: B_0, initial working set size.
Require: $\mathcal{T} = \{\boldsymbol{x}_i, d_i\}_{i=1}^{N}$, a training set with N samples.
Require: τ, number of blocks for block decomposition.
$\mathcal{B} \leftarrow$randomly selected B_0 indices as the initial wk. set.
$\mathcal{M} \leftarrow$indices not in $\mathcal{B}$, that denotes the initial fixed set.
Begin ▷**Variables Decomposition**.
  $\mathcal{W} \leftarrow \{\boldsymbol{x}_i, d_i\}_{i\in\mathcal{B}}$
  Fix $\alpha_j = 0$ for all $j \in \mathcal{M}$. ▷Vars in prob. (3) ignored.
  Begin ▷**Constraints Decomposition**.
    Define $\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}$ with (4a)–(4d).
    $\boldsymbol{A}_{\mathcal{B}}, \boldsymbol{b}_{\mathcal{B}}, \boldsymbol{c}_{\mathcal{B}} \leftarrow \boldsymbol{A}, \boldsymbol{b}, \boldsymbol{c}$
    $\begin{pmatrix} \boldsymbol{A}_{\mathcal{R}}^1 & \boldsymbol{b}_{\mathcal{R}}^1 \\ \boldsymbol{A}_{\mathcal{R}}^2 & \boldsymbol{b}_{\mathcal{R}}^2 \\ \vdots & \vdots \\ \boldsymbol{A}_{\mathcal{R}}^\tau & \boldsymbol{b}_{\mathcal{R}}^\tau \end{pmatrix} \leftarrow$BlockPartition$\tau, \boldsymbol{A}_{\mathcal{B}}, \boldsymbol{b}_{\mathcal{B}}$
    $t = 0$ ▷Iterations counter.
    Repeat
      $t = t + 1$
      $\boldsymbol{z}_{\mathcal{R}}^{(t)} \leftarrow$ IPMSolveLP$\boldsymbol{c}_{\mathcal{R}}, \boldsymbol{A}_{\mathcal{R}}, \boldsymbol{b}_{\mathcal{R}}$ ▷Solves (20).
      $z_{i,\mathcal{B}}^{(t)} = \begin{cases} z_{j,\mathcal{R}}^{(t)} & \text{if } j = i, \text{ for all} j \in \mathcal{R} \\ 0 & \text{otherwise,} \end{cases}$
    until $\left(\boldsymbol{c}_{\mathcal{B}}^T \boldsymbol{z}_{\mathcal{B}}^{(t)} = \boldsymbol{c}_{\mathcal{B}}^T \boldsymbol{z}_{\mathcal{B}}^{(t+4)}\right)$ ▷Stops if no change in 4 it.
    $\boldsymbol{z}_{\mathcal{B}} \leftarrow \boldsymbol{z}_{\mathcal{B}}^{(t)}$ ▷Problem is solved for $\mathcal{W}$.
  End ▷End **Constraint Decomposition**.
  for all $j \in \mathcal{M}$ **do** ▷Verify if problem is solved for $\mathcal{M}$.
    Reconstruct u_j, ξ_j w/(13)-(14) verify primal LP.
    Fix $\lambda_j = 0$, reconstruct s_j w/ (15), verify dual LP.
    $\widetilde{\mathcal{B}} \leftarrow$ VerifyComplementarityz_j, s_j, B_0
  end for
  if $z_j s_j \neq 0$ ▷**If problem (3) is not solved then**
    $\mathcal{B} \leftarrow$ CreateNewWorkingSet$\mathcal{B}, B_0, \widetilde{\mathcal{B}}$
  else
    Stop Training
  end if
End ▷End **Variable Decomposition**.
$\begin{pmatrix} \boldsymbol{\alpha}^+ & \boldsymbol{\alpha}^- & b^+ & b^- & \boldsymbol{\xi} & \boldsymbol{u} \end{pmatrix}^T \leftarrow \boldsymbol{z}$
Ensure $(\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^-, b^+, b^-)$, the SVR solution.

In this work, an LP-SVR is trained over large-scale data sets and at the same time sparser solutions will be produced in terms of the number of Support Vectors. This results in a more efficient model in terms of future kernel evaluations which can be made faster. It will be shown that the proposed model is sparser than the regular SVR and other SV-based classifiers. Although this was studied by Zhang and Zhou (2010), it will be demonstrated that as the problem size increases, the more sparser the solution becomes. The proposed approach is comparable with other formulations in terms of performance, which is desirable.

This paper is organized as follows: Section 2 introduces the proposed LP-SVR formulation for large-scale problems, describing its optimality conditions and its boundaries which are an extension of Zhang, et al.'s work (Zhang and Zhou, 2010); Section 3 presents a variables decomposition strategy that is an extension of Torii, et al. research (Torii and Abe, 2009), that exploits the structure of the proposed LP-SVR problem increasing its efficiency; similarly, Section 4 explains a constraints decomposition strategy that is extended from the work of Bradley and Mangasarian (2000); Section 5 analyzes the results of the experiments performed using the proposed model and decomposition strategies for large-scale training; and finally, conclusions are drawn in Section 6.

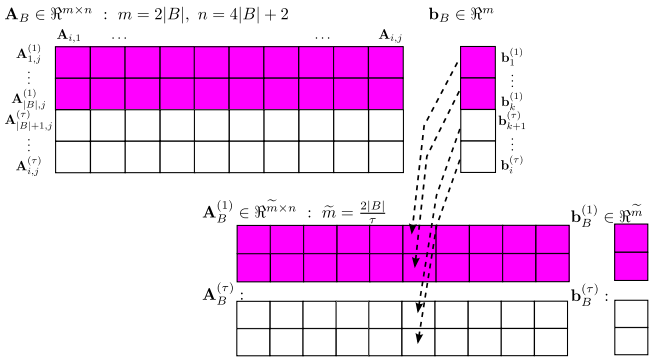

Fig. 1. LP-SVR constraints decomposition strategy. This figure shows an example of the decomposition of the LP constraints. Here the coefficients matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ are divided in τ blocks. This figure shows the linear programming chunking (LPC) approach by Bradley and Mangasarian (2000) and Bradley et al. (2002).

2. Large-Scale LP-SVR formulation

Let us assume that we have training samples $\{(\boldsymbol{x}_i, d_i)\}_{i=1}^N$, where $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^M$ is a regressor and d is the desired output. Then, one can define a non-linear SVR prediction function:

$$d_j \equiv f(\boldsymbol{x}_j) = \sum_{i=1}^{N} (\alpha_i^+ - \alpha_i^-) k(\boldsymbol{x}_i, \boldsymbol{x}_j) + (b^+ - b^-), \tag{1}$$

where $\boldsymbol{\alpha}^+, \boldsymbol{\alpha}^- \in \mathbb{R}_+^N; b^+, b^- \in \mathbb{R}_+; k(\cdot,\cdot)$ is a valid kernel function (Mercer et al., 1909; Courant and Hilbert, 1966; Lanckriet et al., 2004); $\boldsymbol{\alpha}^+ = \max\{\boldsymbol{\alpha}, 0\}$, $\boldsymbol{\alpha}^- = \max\{-\boldsymbol{\alpha}, 0\}$; $b^+ = \max\{b, 0\}$, $b^- = \max\{-b, 0\}$; $\boldsymbol{\alpha} \in \mathbb{R}^N$; and $b \in \mathbb{R}$; the motivation for splitting $\boldsymbol{\alpha} = \boldsymbol{\alpha}^+ - \boldsymbol{\alpha}^-$, is that we would need these variables to be strictly positive, *i.e.*, in the set $\mathbb{R}_+^N$, in order to pose the SVR problem as an LP problem, and the same applies for $b = b^+ - b^-$. Kernel functions (Scholkopf, 2001) map the input feature vectors to the kernel-induced feature space denoted $\mathcal{H}$ since these kernel functions follow the properties of Hilbert spaces (Mercer et al., 1909; Cristianini et al., 2006). The kernel-induced feature space for non-linear SVR can be defined as $\mathcal{H} = \left\{f(\boldsymbol{x}_j) : f(\boldsymbol{x}_j) = \sum_{i=1}^N (\alpha_i^+ - \alpha_i^-) k(\boldsymbol{x}_i, \boldsymbol{x}_j) + (b^+ - b^-)\right\}$, for all $\boldsymbol{x}_j \in \mathbb{R}^M$ and $j = \{1, 2, \ldots, N\}$. The objective is to find the set of parameters $\boldsymbol{\alpha}$ and b. One can find these parameters via constrained optimization.

2.1. Primal, dual, and KKT conditions

First, let us assume that the mapping $k(\boldsymbol{x}_i, \boldsymbol{x}_j) : \mathcal{X}^{(N\times M)\times(M\times N)} \mapsto \mathcal{H}^{N\times N}$ exists; *i.e.*, a kernel operation of order $(N \times M) \times (M \times N)$ in the input space $\mathcal{X}$ exists and maps to a Hilbert space $\mathcal{H}$ of order $N \times N$. Then, assume that the slack variables ξ_i, ξ_i^*, that correspond to the amount of positive and negative deviation between the approximated model and the desired output, can be expressed as simply $2\xi_i$ (since $\xi_i \xi_i^* = 0$, see (Lu et al., 2009)); in fact, $\xi_i = d_i - f(\boldsymbol{x}_i) - \epsilon$ and $\xi_i^* = f(\boldsymbol{x}_i) - d_i - \epsilon$, where ϵ is a prescribed parameter coming required by the ϵ-insensitive loss function defined as follows:

$$L_\epsilon(d, f(\boldsymbol{x})) \equiv |d - f(\boldsymbol{x})|_\epsilon = \begin{cases} |d - f(\boldsymbol{x})| - \epsilon & \text{for } |d - f(\boldsymbol{x})| \geqslant \epsilon \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Then, let us introduce a slack variable $\boldsymbol{u}$ to avoid inequalities in the SVR formulation. As a consequence of these assumptions, the following optimization problem is proposed:

$$\min_{\boldsymbol{\alpha}^+,\boldsymbol{\alpha}^-,b^+,b^-,\boldsymbol{\xi},\boldsymbol{u}} \quad \sum_{i=1}^{N}(\alpha_i^+ + \alpha_i^- + 2C\xi_i)$$
$$\text{s.t} \quad \begin{cases} -\sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-)k(\boldsymbol{x}_j,\boldsymbol{x}_i)\ldots \\ -b^+ + b^- - \xi_j + u_j = \epsilon - d_j \\ \sum_{i=1}^{N}(\alpha_i^+ - \alpha_i^-)k(\boldsymbol{x}_j,\boldsymbol{x}_i)\ldots \\ +b^+ - b^- - \xi_j + u_j = \epsilon + d_j \\ \alpha_j^+,\alpha_j^-,b^+,b^-,\xi_j,u_j \geqslant 0 \end{cases} \tag{3}$$
$$\text{for} \quad j = 1,2,\ldots,N.$$

Problem (3) has two groups of constraints because it has to account for errors in either side of the model, which is typical of any SV regression model. We claim that problem (3) can be posed as a linear programming problem. To do so, we may define the following equalities:

$$\boldsymbol{A} = \begin{pmatrix} -\boldsymbol{K} & \boldsymbol{K} & -1 & 1 & -\boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{K} & -\boldsymbol{K} & 1 & -1 & -\boldsymbol{I} & \boldsymbol{I} \end{pmatrix}, \tag{4a}$$

$$\boldsymbol{b} = \begin{pmatrix} \mathbf{1}\epsilon - \boldsymbol{d} \\ \mathbf{1}\epsilon + \boldsymbol{d} \end{pmatrix}, \tag{4b}$$

$$\boldsymbol{z} = \begin{pmatrix} \boldsymbol{\alpha}^+ & \boldsymbol{\alpha}^- & b^+ & b^- & \boldsymbol{\xi} & \boldsymbol{u} \end{pmatrix}^T, \tag{4c}$$

$$\boldsymbol{c} = \begin{pmatrix} \mathbf{1} & \mathbf{1} & 0 & 0 & \boldsymbol{2C} & \mathbf{0} \end{pmatrix}^T, \tag{4d}$$

where $\boldsymbol{A} \in \mathbb{R}^{(2N)\times(4N+2)}$, $\mathbf{b} \in \mathbb{R}^{2N}$, $\boldsymbol{z},\boldsymbol{c} \in \mathbb{R}^{4N+2}$. If we use the above equalities, then one can claim that the problem has been posed as an LP problem.

It is claimed that problem (3) is an original formulation for LP-SVR. In comparison with the v–LPR formulation by Smola et al. (1999) problem (3) (i) uses the canonical formulation, (ii) computes b, and $\boldsymbol{u}$ implicitly, (iii) does not compute ϵ implicitly, (iv) does not require the parameter v, (v) promotes efficiency in the sense of using only one ξ, and (vi) is a lower dimensional problem.

In comparison with Mangasarian and Musicant (2002), problem (3) (i) uses the canonical formulation, (ii) computes b implicitly, (iii) does not compute ϵ implicitly, and (iv) does not require the parameter μ. By (iii) and (iv) one provides the experimenter with more control of the sparseness of the solution (Zhang and Zhou, 2010). In this case sparseness means fewer number of Support Vectors.

Similarly, Problem (3) in comparison to Lu et al. (2009) our LP-SVR formulation (3) (i) uses the canonical formulation and (ii) computes b implicitly. By (ii) the linear program (LP) size is reduced by a factor of $N^2 + N$.

In comparison with the ℓ_1-norm LP-SVR formulation by Zhang and Zhou (2010) problem (3) does not require parameter δ and is more efficient in several ways: (i) uses only one ξ, (ii) avoids penalization of b, (iii) reduces computational efforts by forcing positivity in $\boldsymbol{u}$ which reduces the LP problem size by $2N^2 + 2N$, and (iv) is a smaller problem.

Using equalities (4a)–(4d), we can obtain the dual problem of (3) as follows:

$$\max_{\boldsymbol{\lambda}} \quad \boldsymbol{b}^T\boldsymbol{\lambda}$$
$$\text{s.t.} \quad \begin{cases} \boldsymbol{A}^T\boldsymbol{\lambda} + \boldsymbol{s} = \boldsymbol{c} \\ \boldsymbol{s} \geqslant \mathbf{0}, \end{cases} \tag{5}$$

where $\boldsymbol{\lambda}$ is a vector of dual variables defined over $\mathbb{R}^{2N}$, and $\mathbf{s}$ is a slack vector variable in $\mathbb{R}^{4N+2}$.

Similarly, for the primal (3) and dual (5), the KKT conditions are defined as follows:

$$\boldsymbol{A}^T\boldsymbol{\lambda} + \boldsymbol{s} = \boldsymbol{c}, \tag{6a}$$

$$\boldsymbol{A}\boldsymbol{z} = \boldsymbol{b}, \tag{6b}$$

$$z_i s_i = 0, \tag{6c}$$

$$(\boldsymbol{z},\boldsymbol{s}) \geqslant \mathbf{0}, \tag{6d}$$

for $i = 1,2,\ldots,n$,

where the equality $z_i s_i$ implies that one of both variables must be zero. This equality will be referred to as the *complementarity condition*. Note that the KKT conditions depend on the variables $(\boldsymbol{z},\boldsymbol{\lambda},\boldsymbol{s})$, and if the set of solutions $(\boldsymbol{z}^*,\boldsymbol{\lambda}^*,\boldsymbol{s}^*)$ satisfy all the conditions, the problem is said to be solved. The set $(\boldsymbol{z}^*,\boldsymbol{\lambda}^*,\boldsymbol{s}^*)$ is known as a *primal–dual solution*.

2.2. Optimality and sparseness

Let $\boldsymbol{z}^*$ be the solution to the primal problem (3), and let $(\boldsymbol{\lambda}^*,\boldsymbol{s}^*)$ be the solution to the dual problem (5). The proposed LP-SVR exhibits two important properties. First, that it has a *global solution*. That is, if $\boldsymbol{z}^*$ is a minimum for problem (3), then $\boldsymbol{z}^*$ is a global minimum since problem (3) is a convex problem (*i.e.*, a linear programming problem) (Dantzig and Thapa, 1997; Dantzig, 1998; Dennis and Schnabel, 1996; Ferris et al., 2007).

Second, its *optimality conditions* are well defined. That is, for problem (3), the KKT conditions (6a)–(6d) are necessary and sufficient for optimality since $(\boldsymbol{z}^*,\boldsymbol{\lambda}^*,\boldsymbol{s}^*)$ is a solution to the primal (3) and dual (5), then it follows that the KKT conditions (6a)–(6d) are necessary and sufficient for optimality (Neumann, 1945; Dantzig and Thapa, 1997; Dantzig, 1998; Ferris et al., 2007).

One concern of the work presented here is to demonstrate that the solution of the proposed LP-SVR is better than that of SVRs in the sense of solution sparseness. Sparseness in a solution is desired because any SV-based model relies on actual feature vectors $\boldsymbol{x}_i$ to define the optimal set of model parameters (α_i, b) for all $i : \alpha_i \neq 0$. Especially since the feature vectors $\boldsymbol{x}_i$ are required for kernel distances as shown in (1).

Zhang and Zhou (2010) performed a comprehensive study in regard to SVM sparseness. Zhang and Zhou (2010) explains that sparseness of a learning machine depends on the problem and the precision of the solution. Then, the authors prove (see Zhang and Zhou (2010) Theorems 1 and 2) that, for their proposed LP-SVR, the solution is always sparser than regular SVRs. In fact, Zhang's theorems also hold for our formulation. To demonstrate this, the following definitions are given.

Definition 1 (*Support Vectors*). Let $\mathcal{T} = \{\boldsymbol{x}_i, d_i\}_{i=1}^N$ be a training set; let $\boldsymbol{z}$ be a solution to problem (3); and let (1) be the regression function for problem (3). Then,

1. $\mathcal{V}_S = \{\boldsymbol{x}_i : d_i - \epsilon < f(\boldsymbol{x}_i) < d_i + \epsilon\}$ defines the set of Saturated Support Vectors (SSVs).
2. $\mathcal{V}_E = \{\boldsymbol{x}_i : f(\boldsymbol{x}_i) = d_i + \epsilon, \text{ or } f(\boldsymbol{x}_i) = d_i - \epsilon\}$ defines the set of Exact Support Vectors (ESVs).
3. $\mathcal{V}_N = \{\boldsymbol{x}_i : f(\boldsymbol{x}_i) < d_i + \epsilon, \text{ or } f(\boldsymbol{x}_i) > d_i - \epsilon\}$ defines the set of Non-Support Vectors (NSVs).
4. $\mathcal{V}_\alpha = \{\boldsymbol{x}_i : \alpha_i \neq 0\}$ defines the set of Sparse Vectors (SPVs).
5. $N = |\mathcal{V}_S| + |\mathcal{V}_E| + |\mathcal{V}_N|$.
6. $\mathcal{S} = \mathcal{V}_S \cup \mathcal{V}_E$, means that the union of the SSVs and the ESVs is the set of Support Vectors (SVs).
7. $\mathcal{A} = \{\alpha_i : \alpha_i \neq 0\}$ denotes the set of Non-zero Coefficients of the decision function (1) and of problem (3).

Then, since Definition 1 is equivalent to Definition 2 in (Zhang and Zhou, 2010) by Zhang, et al., then we can say that given an optimal solution $\boldsymbol{z}^*$ to (3), the number of nonzero α_i coefficients of (3) has the following upper bound:

$$|\mathcal{A}| \leqslant |\mathcal{V}_E| \tag{7}$$

for all $i : \alpha_i \neq 0$. This property states that ESVs characterize the sparseness of problem (3) just as SVs characterize the sparseness of any SVR formulation. The above property points out that the proposed LP-SVR problem (3) possess better sparseness than that of standard SVRs, since there are always several SSVs in standard SVRs, especially for practical noisy data sets used in recognition or regression problems (Zhang and Zhou, 2010).

Similarly, from Definition 2 in (Zhang and Zhou, 2010) by Zhang, et al., we have that the number of nonzero coefficients of (3) has the following upper bound:

$$|\mathcal{A}| \leqslant \operatorname{rank}(\boldsymbol{K}) \tag{8}$$

and the column vectors $k(\boldsymbol{x}_j, \boldsymbol{x}_1), k(\boldsymbol{x}_j, \boldsymbol{x}_2), \ldots, k(\boldsymbol{x}_j, \boldsymbol{x}_i)$, are linearly independent for all $j \in \mathcal{A}$. This means that the LP-SVR regression function (1) can be exactly reproduced using only those samples that are SVs, without affecting performance. This property also indicates that vectors $k(\boldsymbol{x}_j, \boldsymbol{x}_1), k(\boldsymbol{x}_j, \boldsymbol{x}_2), \ldots, k(\boldsymbol{x}_j, \boldsymbol{x}_i)$, for all $j \in \mathcal{A}$, in the decision function $f(\boldsymbol{x}) = \sum_{i\in\mathcal{A}}(\alpha_i^+ - \alpha_i^-)k(\boldsymbol{x}_i, \boldsymbol{x}) + (b^+ - b^-)$, are linearly independent. This means one cannot further reduce the number of basis functions in the regression function, and also indicates that the proposed LP-SVR (3) may lead to a sparser model representation.

It is important to remark that problem (3) was designed to maximize computational efficiency without sacrificing accuracy. This has been achieved by not introducing unnecessary parameters into the problem and by posing a problem that minimizes the number of SVs without affecting performance. In spite of this, the training phase still may be computationally expensive for applications with N larger than a few thousands. Therefore, the following section introduces a learning process for the case when N is very large.

3. LP-SVR variable decomposition

Let us consider the proposed Linear Program (3). In order to deal with large N problems, an idea is to divide the training set $\mathcal{T} = \{\boldsymbol{x}_i, d_i\}_{i=1}^N$ in two subsets: a *working set* $\mathcal{W} = \{\boldsymbol{x}_i, d_i\}_{i\in\mathcal{B}}$ and a *fixed set* $\mathcal{F} = \{\boldsymbol{x}_i, d_i\}_{i\in\mathcal{M}}$; where we identify the members of each set with two index sets $\mathcal{B}$ and $\mathcal{M}$, respectively. These two sets are disjoint: $\mathcal{B} \cap \mathcal{M} = \emptyset$; and their union contains all the training set indices: $\mathcal{B} \cup \mathcal{M} = \{1, 2, \ldots, N\}$. The initial size of $\mathcal{B}$ is given by a parameter B_0, $(2 \leqslant B_0 \leqslant N)$, chosen heuristically. Note that for classification problems with q number of classes, B_0 is bounded as $q \leqslant B_0 \leqslant N$.

Under these definitions, a method is proposed, in which problem (3) is solved using only a subset of the variables $\boldsymbol{z} \geqslant 0$ from (4c), and a subset of the constraints $\boldsymbol{Az} = \boldsymbol{b}$ from (4a and 4b). This approach is known as variables decomposition and constraints decomposition (Torii and Abe, 2009). The variable decomposition strategy presented in this research is an adaptation of the work by Torii and Abe (2009), in which the author decomposes the variables of a linear program (LP). However, instead of simply decomposing variables of the LP, one can exploit the properties of the LP-SVR to make the LP smaller. The constraints decomposition strategy is an adaptation of the linear programming chunking (LPC) algorithm introduced by Bradley and Mangasarian (2000) and Bradley et al. (2002) but modified for LP-SVR efficiency.

Section 4 presents the strategy for solving an LP-SVR using a subset of the constraints $\boldsymbol{Az} = \boldsymbol{b}$ from (4a and 4b). But first, let us address the problem of solving an LP-SVR using only a subset of the variables $\boldsymbol{z} \geqslant 0$ from (4c), which is summarized in Algorithm 2.

Algorithm 2: Variable Decomposition Strategy for Large-Scale LP-SVR Training. Modification of Torii and Abe (2009) Algorithm

1: Set $\mathcal{B}$ with the first B_0 indices from $\mathcal{T}$.
2: For all indices in $\mathcal{B}$ **Solve LP** sub-problem.
3: Verify if the current solution satisfies the KKT conditions for the indices in $\mathcal{M}$. If so, then **Stop**.
4: Move inactive constraints indices from $\mathcal{B}$ to $\mathcal{M}$. Then, if $|\mathcal{B}| + 1 + \lceil \log|\mathcal{B}| \rceil \leqslant B_{\max}$, move the **worst** $1 + \lceil \log|\mathcal{B}| \rceil$ violating indices from $\mathcal{M}$ into $\mathcal{B}$. Go to Step 2. Else, **Stop**.
5: Those indices that have been at least l times in and out of the working set $\mathcal{B}$ are moved permanently into $\mathcal{B}$.

Each step is explained in the following paragraphs.

In **Step 1**, a subset of the variables is chosen according to the indices in $\mathcal{B}$. Then one proceeds to fix $\boldsymbol{\alpha}_\mathcal{M} = \alpha_i^+ - \alpha_i^- = 0$, for all $i \in \mathcal{M}$. The problem is to find the variables $\boldsymbol{\alpha}_\mathcal{B} = \alpha_i^+ - \alpha_i^-$, for all $i \in \mathcal{B}$; and then, since every fixed variable is associated with two constraints, these can be ignored under the assumption they are not Support Vectors, which leads to the following sub-problem:

$$\min_{\boldsymbol{\alpha}_\mathcal{B}^+, \boldsymbol{\alpha}_\mathcal{B}^-, b^+, b^-, \xi_\mathcal{B}, \boldsymbol{u}_\mathcal{B}} \quad \sum_{i\in\mathcal{B}}(\alpha_i^+ - \alpha_i^- + 2C\xi_i)$$
$$\text{s.t.} \quad \begin{cases} -\sum_{i\in\mathcal{B}}(\alpha_i^+ - \alpha_i^-)k(\boldsymbol{x}_j, \boldsymbol{x}_i) \\ -b^+ + b^- - \xi_j + u_j = \epsilon - d_j \\ \sum_{i\in\mathcal{B}}(\alpha_i^+ - \alpha_i^-)k(\boldsymbol{x}_j, \boldsymbol{x}_i) \\ +b^+ - b^- - \xi_j + u_j = \epsilon + d_j \\ \alpha_j^+, \alpha_j^-, b^+, b^-, \xi_j, u_j \geqslant 0 \end{cases} \tag{9}$$
$$\text{for all } j \in \mathcal{B}.$$

Let $\boldsymbol{z}_\mathcal{B}^* = (\boldsymbol{\alpha}_\mathcal{B}^{+*}, \boldsymbol{\alpha}_\mathcal{B}^{-*}, b^{+*}, b^{-*}, \xi_\mathcal{B}^*, \boldsymbol{u}_\mathcal{B}^*)$, denote the solution to linear programming problem (9). Problem (9) is a size-reduced version of problem (3), for which a comparison between LP problems is illustrated in Fig. 2. The figure shows problem (3) in the left, and the reduced problem (9) is shown on the right. Fig. 2 shows the LP-SVR structure being exploited to reduce problem (3).

To derive the optimality conditions at a later stage, we obtain the following dual:

$$\min_{\boldsymbol{\lambda}, \boldsymbol{s}} \quad \sum_{i\in\mathcal{B}}\lambda_i(\epsilon - d_i) + \sum_{i\in\mathcal{B}}\lambda_{i+|\mathcal{B}|}(\epsilon + d_i)$$
$$\text{s.t.} \begin{cases} \sum_{i\in\mathcal{B}}\lambda_{i+|\mathcal{B}|}k(\boldsymbol{x}_j, \boldsymbol{x}_i) - \sum_{i\in\mathcal{B}}\lambda_i k(\boldsymbol{x}_j, \boldsymbol{x}_i) + s_j = 1_j \\ \sum_{i\in\mathcal{B}}\lambda_i k(\boldsymbol{x}_j, \boldsymbol{x}_i) - \sum_{i\in\mathcal{B}}\lambda_{i+|\mathcal{B}|}k(\boldsymbol{x}_j, \boldsymbol{x}_i) + s_{j+|\mathcal{B}|} = -1_j \\ \sum_{i\in\mathcal{B}}\lambda_{i+|\mathcal{B}|} - \lambda_i + s_{2|\mathcal{B}|+1} = 1 \\ \sum_{i\in\mathcal{B}}\lambda_i - \lambda_{i+|\mathcal{B}|} + s_{2|\mathcal{B}|+2} = -1 \\ -\lambda_i - \lambda_{i+|\mathcal{B}|} + s_{j+2|\mathcal{B}|+2} = 2C_i \\ \lambda_i + \lambda_{i+|\mathcal{B}|} + s_{j+3|\mathcal{B}|+2} = 1_i \\ \boldsymbol{s} \geqslant \boldsymbol{0} \end{cases} \tag{10}$$
$$\text{for all } j \in \mathcal{B}$$

and the KKT conditions (6a)–(6d) are rewritten as follows:

$$\sum_{i\in\mathcal{B}}\lambda_{i+|\mathcal{B}|}k(\boldsymbol{x}_j, \boldsymbol{x}_i) - \sum_{i\in\mathcal{B}}\lambda_i k(\boldsymbol{x}_j, \boldsymbol{x}_i) + s_j = 1_j, \tag{11a}$$

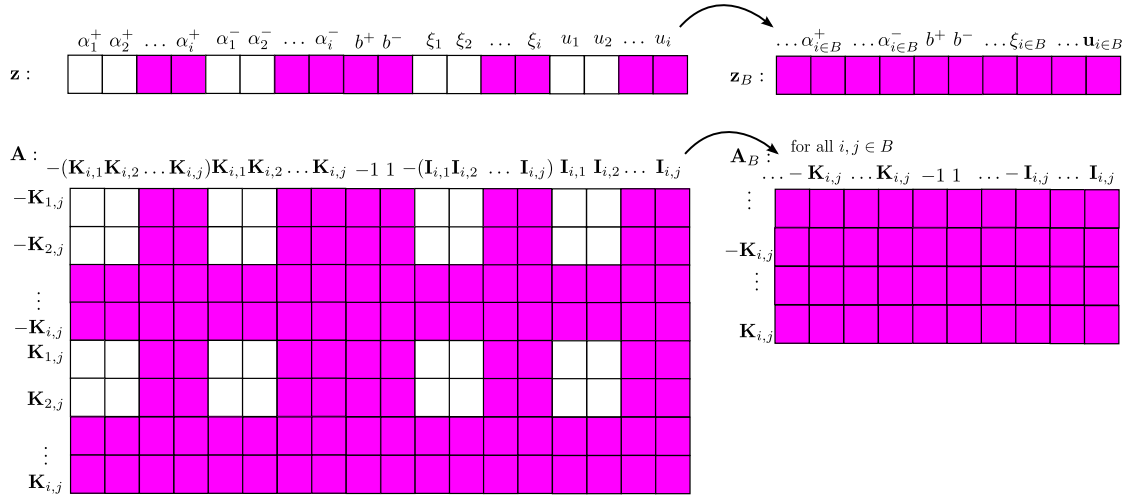

Fig. 2. LP-SVR variable decomposition strategy. This illustrates a decomposition of the LP variables vector $\boldsymbol{z}$ and coefficients matrix $\boldsymbol{A}$ in which a subset of the variables α_i is considered for the solution of the LP-SVR problem, for all $i \in \mathcal{B}$. The shaded area corresponds to an arbitrary selection of $|\mathcal{B}|$ indices that produce a reduced LP: $\boldsymbol{z}_B$ and $\boldsymbol{A}_B$.

$$\sum_{i\in\mathcal{B}}\lambda_i k(\boldsymbol{x}_j,\boldsymbol{x}_i) - \sum_{i\in\mathcal{B}}\lambda_{i+|\mathcal{B}|}k(\boldsymbol{x}_j,\boldsymbol{x}_i) + s_{j+|\mathcal{B}|} = -1_j, \tag{11b}$$

$$\sum_{i\in\mathcal{B}}\lambda_{i+|\mathcal{B}|} - \lambda_i + s_{2|\mathcal{B}|+1} = 1, \tag{11c}$$

$$\sum_{i\in\mathcal{B}}\lambda_i - \lambda_{i+|\mathcal{B}|} + s_{2|\mathcal{B}|+2} = -1, \tag{11d}$$

$$-\lambda_j - \lambda_{j+|\mathcal{B}|} + s_{j+2|\mathcal{B}|+2} = 2C_j, \tag{11e}$$

$$\lambda_j + \lambda_{j+|\mathcal{B}|} + s_{j+3|\mathcal{B}|+2} = 1_j, \tag{11f}$$

$$-\sum_{i\in\mathcal{B}}(\alpha_i^+ - \alpha_i^-)k(\boldsymbol{x}_j,\boldsymbol{x}_i) - b^+ + b^- - \xi_j + u_j = \epsilon - d_j, \tag{11g}$$

$$\sum_{i\in\mathcal{B}}(\alpha_i^+ - \alpha_i^-)k(\boldsymbol{x}_j,\boldsymbol{x}_i) + b^+ - b^- - \xi_j + u_j = \epsilon + d_j, \tag{11h}$$

$$s_j\alpha_j^+ + s_{j+|\mathcal{B}|}\alpha_j^- + s_{2|\mathcal{B}|+1}b^+ + s_{2|\mathcal{B}|+2}b^- + s_{j+2|\mathcal{B}|+2}\xi_i + s_{j+3|\mathcal{B}|+2}u_j = 0, \tag{11i}$$

$$s_i, \alpha_j^+, \alpha_j^-, b^+, b^-, \xi_j, u_j \geqslant 0 \quad \text{for all } i,j \in \mathcal{B}. \tag{11j}$$

The primal sub-problem (9) has $4|\mathcal{B}|+2$ variables and $2|\mathcal{B}|$ constraints; the dual sub-problem (10) has $2|\mathcal{B}|$ variables and $4|\mathcal{B}|+2$ constraints; and the sub-problem KKT conditions (11a)–(11j) are necessary and sufficient for optimality as explained before.

In **Step 2**, one solves the sub-problem (9), (10), (11a)–(11j), *e.g.*, using LP interior point methods (IPM). In **Step 3**, one determines if problem (3), (5),(6a)–(6d) has been solved successfully. To do this, one needs to check the KKT conditions of the original problem. Since variables $\alpha_i^+, \alpha_i^- = 0$, for all $i \in \mathcal{M}$, and since we have a sub-problem solution, then, the values for the primal variables ξ_i and u_i for $i \in \mathcal{M}$ can be estimated according to the following cases:

Case 1: When the following inequalities holds true for $j \in \mathcal{M}$:

$$-\sum_{i\in\mathcal{B}}(\alpha_i^+ - \alpha_i^-)k(\boldsymbol{x}_j,\boldsymbol{x}_i) - b^+ + b^- - \epsilon + d_j \geqslant 0, \tag{12a}$$

$$\sum_{i\in\mathcal{B}}(\alpha_i^+ - \alpha_i^-)k(\boldsymbol{x}_j,\boldsymbol{x}_i) + b^+ - b^- - \epsilon - d_j \geqslant 0, \tag{12b}$$

then, the values for the jth index can be computed from (6b) as follows:

$$u_j = 2\left(\sum_{i\in\mathcal{B}}(\alpha_i^+ - \alpha_i^-)k(\boldsymbol{x}_j,\boldsymbol{x}_i) + b^+ - b^- - d_j\right), \tag{13a}$$

$$\xi_j = 0, \tag{13b}$$

where $\boldsymbol{u}$ is the vector of slacks that preserves the equality in the constraints.

Case 2: When the inequalities (12) hold false, then the values for the j-th index are computed as follows:

$$u_j = 0, \tag{14a}$$

$$\xi_j = -2\left(\sum_{i\in\mathcal{B}}(\alpha_i^+ - \alpha_i^-)k(\mathbf{x}_j,\boldsymbol{x}_i) + b^+ - b^- - d_j\right). \tag{14b}$$

Next, the values for the dual variable are fixed $\lambda_i = 0$ for all $i \in \mathcal{M}$. Then, the values for the dual slack s_i for $i = 1,2,\ldots,4|\mathcal{M}|+2$ are estimated from (6a) as follows:

$$s_j = 1_j - \sum_{i\in\mathcal{B}}\lambda_{i+|\mathcal{B}|}k(\boldsymbol{x}_j,\boldsymbol{x}_i) + \sum_{i\in\mathcal{B}}\lambda_i k(\boldsymbol{x}_j,\boldsymbol{x}_i), \tag{15a}$$

$$s_{j+|\mathcal{B}|} = -1_j - \sum_{i\in\mathcal{B}}\lambda_i k(\boldsymbol{x}_j,\boldsymbol{x}_i) + \sum_{i\in\mathcal{B}}\lambda_{i+|\mathcal{B}|}k(\boldsymbol{x}_j,\boldsymbol{x}_i), \tag{15b}$$

$$s_{2|\mathcal{B}|+1} = 1 - \sum_{i\in\mathcal{B}}\lambda_{i+|\mathcal{B}|} + \lambda_i, \tag{15c}$$

$$s_{2|\mathcal{B}|+2} = -1 - \sum_{i\in\mathcal{B}}\lambda_i + \lambda_{i+|\mathcal{B}|}, \tag{15d}$$

$$s_{j+2|\mathcal{B}|+2} = 2C_j + \lambda_j + \lambda_{j+|\mathcal{B}|}, \tag{15e}$$

$$s_{j+3|\mathcal{B}|+2} = 1_j - \lambda_j - \lambda_{j+|\mathcal{B}|} \quad \text{for all } j \in \mathcal{M}. \tag{15f}$$

Note that KKT primal and dual conditions (6a)–(6d) have been already satisfied by (13a), (14a), and (15a); however, the complementarity conditions have not. Then, we verify if the following conditions hold:

$$z_i s_i = 0, \tag{16a}$$

$$\boldsymbol{z}, \boldsymbol{s} \geqslant \mathbf{0} \tag{16b}$$

for all $i = 1, 2, \ldots, 4|\mathcal{M}| + 2$.

If there were no violations to (16), the global problem is said to be solved, and the method has converged for the set of parameters given. The global LP-SVR Support Vectors $\boldsymbol{x}_i$ are those whose $(\alpha_i^+ - \alpha_i^-) \neq 0$ for all $i \in \mathcal{B}$.

In **Step 4**, we verify if there were any violations to (16a); in such case, a new working set is created. To do this, we look for inactive constraint indices (*i.e.*, those $(\alpha_i^+ - \alpha_i^-) = 0$ for all $i \in \mathcal{B}$) and move them into $\mathcal{M}$ and then, we replace those indices with the indices that *most* violate the complementarity conditions (16a) from $\mathcal{M}$ into $\mathcal{B}$. By "*most* violation" we mean that the the complementarity condition (16a) has been sorted and we chose the largest violations first.

For practical purposes, a record is kept indicating which indices have been moved from $\mathcal{M}$ into $\mathcal{B}$. In the case that all the constraints in $\mathcal{B}$ are active (*i.e.*, all are Support Vectors), then, the size of $\mathcal{B}$ is incremented by a scaling exponent as follows:

$$|\mathcal{B}|^{(t+1)} = \begin{cases} |\mathcal{B}|^{(t)} + 1 + \lceil \log |\mathcal{B}|^{(t)} \rceil, & \text{if } |\mathcal{B}|^{(t)} + 1 + \lceil \log |\mathcal{B}|^{(t)} \rceil \leqslant B_{\max} \\ B_{\max}, & \text{otherwise} \end{cases} \tag{17}$$

where $B_{\max}$ is the maximum working set size allowed by the researcher, which is bounded to $B_0 < B_{\max} \leqslant |\mathcal{M}|$, and t is the current iteration at the variable decomposition strategy. Eq. (17) is proposed here to smooth the increments in the working set size.

Steps 1–4 complete one iteration. These steps should be repeated until convergence. However, if after l iterations, the method has not converged, a check is performed to see if there are any indices that have been moving from $\mathcal{M}$ into $\mathcal{B}$ for at least l times. The constant l is an arbitrary parameter given by the researcher: typically $l = 10$.

Finally, in **Step 5**, we check if the condition $t > l$ holds true, if it does, the indices will be added to $\mathcal{B}$ permanently, thus, preventing infinite loops.

The fact that our algorithm stops when the KKT conditions are satisfied guarantees the convergence to an optimal solution. Furthermore, our algorithm avoids a possible infinite loop by limiting indices from going in and out of the set $\mathcal{B}$ for an undefined number of iterations. This guarantees that the algorithm will converge in a finite number of iterations. Of course, the solution will be suboptimal if the algorithm stops when the maximum number of iterations $t^{\max}$ is reached, or if the maximum working set size $B_{\max}$ has been reached without a solution.

The next section explains the method for chunking the constraints.

4. LP-SVR constraints decomposition

Although the variable decomposition approach reduces the complexity of the linear program (LP) solution, the problem is likely to be increasing in size until it reaches the maximum working set size $B_{\max}$. As the LP size increases at each iterate the problem will become much slower proportionally to the current working set size $|\mathcal{B}|$. To overcome this difficulty, we present a modification of the constraint decomposition algorithm originally introduced by Bradley and Mangasarian (2000). In this algorithm which an LP-SVR subproblem is solved using a subset of the constraints. This can be achieved by dividing the LP-sub-problem into blocks of smaller size and then we modify it such that the LP-SVR properties can be exploited one more time to further reduce the sub-problem.

The process of decomposing the constraints is illustrated in Fig. 3 and summarized in Algorithm 3. The next paragraphs explain the processes involved in the completion of Algorithm 3, and propose some definitions and equivalences necessary to the success of the algorithm.

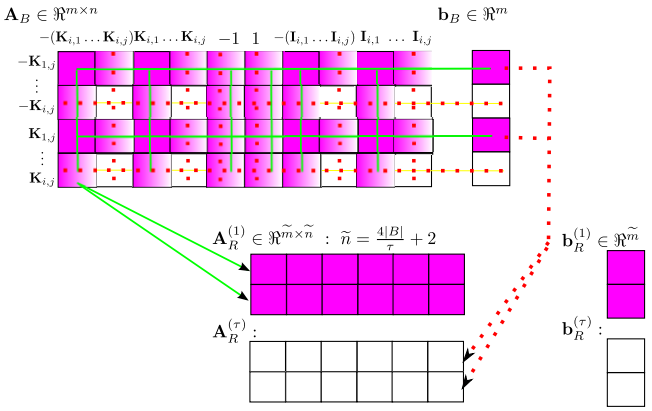

Fig. 3. Proposed LP-SVR constraints decomposition strategy. The proposed decomposition strategy exploits the LP-SVR structure to further reduce the problem size. Note that the reduced problem size is inversely proportional to τ. The notation $\boldsymbol{A}_{ij}$ refers to an element of the matrix $\boldsymbol{A}$ at the i-row and jth column.

Algorithm 3: Constraints Decomposition Strategy for Large-Scale LP-SVR Training: a modification of the algorithm introduced by Bradley and Mangasarian (2000)

Require: LP-SVR with subset of variables: $\boldsymbol{c}_{\mathcal{B}}, \boldsymbol{b}_{\mathcal{B}}, \boldsymbol{A}_{\mathcal{B}}$.
Require: Parameters: $t_{\max}$, and τ.
1: Partition $(\boldsymbol{A}_{\mathcal{B}} \mid \boldsymbol{b}_{\mathcal{B}})$ into τ blocks with (22), where $1 < \tau < |\mathcal{B}|$. Then obtain constraint-sub-blocks $(\boldsymbol{A}_{\mathcal{R}} \mid \boldsymbol{b}_{\mathcal{R}})$.
2: For all indices in $\mathcal{R}$, **Solve LP** sub-sub-problem (23).
3: If $t \leqslant t_{\max}$, then go to Step 2 with a new block.
4: With (24) obtain $\boldsymbol{c}^T\boldsymbol{z}^{(t)}$ for $\mathcal{B}$.
5: If $\boldsymbol{c}^T\boldsymbol{z}^{(t)} = \boldsymbol{c}^T\boldsymbol{z}^{(t+t_{\max})}$, then **Stop**. Else, go to Step 2 with a new block.
Ensure: $\boldsymbol{z}^{(t)}$.

One can pose problem (9) as an LP problem by defining the following equalities:

$$\boldsymbol{A}_{\mathcal{B}} = \begin{pmatrix} -\boldsymbol{K}_{\mathcal{B}} & \boldsymbol{K}_{\mathcal{B}} & -1 & 1 - \boldsymbol{I}_{\mathcal{B}} & \boldsymbol{I}_{\mathcal{B}} \\ \boldsymbol{K}_{\mathcal{B}} & -\boldsymbol{K}_{\mathcal{B}}1 & -1 & -\boldsymbol{I}_{\mathcal{B}} & \boldsymbol{I}_{\mathcal{B}} \end{pmatrix}, \tag{18a}$$

$$\boldsymbol{b}_{\mathcal{B}} = \begin{pmatrix} \mathbf{1}_{\mathcal{B}}\epsilon - \boldsymbol{d}_{\mathcal{B}} \\ \mathbf{1}_{\mathcal{B}}\epsilon + \boldsymbol{d}_{\mathcal{B}} \end{pmatrix}, \tag{18b}$$

$$\boldsymbol{z}_{\mathcal{B}} = \begin{pmatrix} \boldsymbol{\alpha}_{\mathcal{B}}^+ & \boldsymbol{\alpha}_{\mathcal{B}}^- & b^+ & b^- & \boldsymbol{\xi}_{\mathcal{B}} & \boldsymbol{u}_{\mathcal{B}} \end{pmatrix}^T, \tag{18c}$$

$$\boldsymbol{c}_{\mathcal{B}} = \begin{pmatrix} \mathbf{1}_{\mathcal{B}} & \mathbf{1}_{\mathcal{B}} & 0 & 0 & \mathbf{2C}_{\mathcal{B}} & \mathbf{0}_{\mathcal{B}} \end{pmatrix}^T, \tag{18d}$$

where $\mathbf{A}_{\mathcal{B}} \in \mathbb{R}^{(2|\mathcal{B}|)\times(4|\mathcal{B}|+2)}$, $\boldsymbol{b}_{\mathcal{B}} \in \mathbb{R}^{2|\mathcal{B}|}$, $\boldsymbol{z}_{\mathcal{B}}$, $\boldsymbol{c}_{\mathcal{B}} \in \mathbb{R}^{4|\mathcal{B}|+2}$. In this manner, problem (9) is posed as an LP problem.

Let $\boldsymbol{z}_{\mathcal{B}}$ be the unknown in the linear program given by $\min_{\boldsymbol{z}_{\mathcal{B}}}\{\boldsymbol{c}_{\mathcal{B}}^T\boldsymbol{z}_{\mathcal{B}}\}$ subject to $\{\boldsymbol{A}_{\mathcal{B}}\boldsymbol{z}_{\mathcal{B}} = \boldsymbol{b}_{\mathcal{B}}, \boldsymbol{z}_{\mathcal{B}} \geqslant \mathbf{0}\}$; which is equivalent to (9). Then, allow the augmented matrix $(\boldsymbol{A}_{\mathcal{B}} \mid \boldsymbol{b}_{\mathcal{B}})$ to be decomposed into the following τ blocks:

$$(\boldsymbol{A}_{\mathcal{B}} \mid \boldsymbol{b}_{\mathcal{B}}) = \left(\begin{array}{c|c} \boldsymbol{A}_{\mathcal{B}}^{(1)} & \boldsymbol{b}_{\mathcal{B}}^{(1)} \\ \hline \boldsymbol{A}_{\mathcal{B}}^{(2)} & \boldsymbol{b}_{\mathcal{B}}^{(2)} \\ \hline \vdots & \vdots \\ \hline \boldsymbol{A}_{\mathcal{B}}^{(\tau)} & \boldsymbol{b}_{\boldsymbol{B}}^{(\tau)} \end{array}\right), \tag{19}$$

for all $1 < \tau < |\mathcal{B}|$.

Let $t = 1, 2, \ldots$, denote the current iteration and let $\boldsymbol{z}_{\mathcal{B}}^{(t)}$ be the solution to the following linear program:

$$\boldsymbol{z}_{\mathcal{B}}^{(t)} = \arg\min_{\boldsymbol{z}_{\mathcal{B}}} \left\{ \boldsymbol{c}_{\mathcal{B}}^T \boldsymbol{z}_{\mathcal{B}} \;\middle|\; \begin{array}{l} \boldsymbol{A}_{\mathcal{B}}^{(t \bmod \tau)} \boldsymbol{z}_{\mathcal{B}} - \boldsymbol{b}_{\mathcal{B}}^{(t \bmod \tau)} = e_p \\ \bar{\boldsymbol{A}}_{\mathcal{B}}^{(t \bmod \tau)-1} \boldsymbol{z}_{\mathcal{B}} - \bar{\boldsymbol{b}}_{\mathcal{B}}^{(t \bmod \tau)-1} = \bar{e}_p \end{array} \right\}, \tag{20}$$

where e_p, and $\bar{e}_p$ are errors for the LP primal (20), $\left(\bar{\boldsymbol{A}}_{\mathcal{B}}^{(0)} \mid \bar{\boldsymbol{b}}_{\mathcal{B}}^{(0)}\right) = \emptyset$, and $\left(\bar{\boldsymbol{A}}_{\mathcal{B}}^{(t)} \mid \bar{\boldsymbol{b}}_{\mathcal{B}}^{(t)}\right)$ is the set of active constraints (*i.e.*, those equalities in (20) with $e_p, \bar{e}_p = 0$ for $\boldsymbol{z}_{\mathcal{B}}^{(t)}$) with positive Lagrange multipliers. When $\boldsymbol{c}_{\mathcal{B}}^T \boldsymbol{z}_{\mathcal{B}}^{(t)}$ is equal to $\boldsymbol{c}_{\mathcal{B}}^T \boldsymbol{z}_{\mathcal{B}}^{(t+t_{\max})}$, then stop. Typically, the integer $t_{\max}$, controlling the stopping criteria, is set to $t_{\max} = 4$ or any small integer strictly positive (Bradley and Mangasarian, 2000). In (20), the bar notation, $(\bar{\cdot})$, is used to differentiate which blocks of the problem are being involved in a particular iteration.

As an example, let us consider $t = 1,\ \tau = 4$; at this point, $(1 \bmod 4) = 1$ and $(1 \bmod 4) - 1 = 0$, therefore, the LP in (20) only requires to satisfy the first part and not the second since it is empty by definition. In the next iteration $t = 2,\ \tau = 4, (2 \bmod 4) = 2$ and $(1 \bmod 4) - 1 = 1$, therefore, the LP in (20) requires both the first and the second part to be satisfied; however, recall that in the previous iterations only active constraints were preserved for that block. Bradley and Mangasarian (2000) demonstrated that this problem converges iteratively to the solution $\boldsymbol{z}_{\mathcal{B}}$ (See Theorem 3.2 in (Bradley and Mangasarian, 2000)).

However, for the proposed LP-SVR it is obvious that if the decomposition τ is carefully chosen, variables can be further reduced without affecting the solution at all. The key is for constraints associated with the same variable to be within the same block; when they are, all other variables can be omitted. Consider the constraints (18a) and consider the block-decomposition choosing the first $\lceil \frac{2N}{\tau} \rceil$ rows; then, the variables in the first $\lceil \frac{2N}{\tau} \rceil$ rows may have constraints related to the same variable in a different block. Therefore, it makes more sense (to our problem) to choose the blocks (19) in such a way that the properties of the proposed LP-SVR are exploited.

Clearly, if one uses (19) to select the elements of the τ-th block, the block itself can be further reduced by taking into account that variables associated with constraints that do no appear in the τ-th block do not play any role in the problem solution. To do so, define $\mathcal{R}$ as the set of indices of variables associated with the $\tau-$th block, where $\mathcal{R} \not\subseteq \mathcal{B}$ and $|\mathcal{R}| < |\mathcal{B}|$. Then, redefine (19) as follows:

$$\left(\boldsymbol{A}_{\mathcal{B}} \mid \boldsymbol{b}_{\mathcal{B}}\right) \equiv \left(\begin{array}{c|c} \boldsymbol{A}_{\mathcal{R}}^{(1)} & \boldsymbol{b}_{\mathcal{R}}^{(1)} \\ \hline \boldsymbol{A}_{\mathcal{R}}^{(2)} & \boldsymbol{b}_{\boldsymbol{R}}^{(2)} \\ \hline \vdots & \vdots \\ \hline \boldsymbol{A}_{\mathcal{R}}^{(\tau)} & \lfloor_{\mathcal{R}}^{(\tau)} \end{array}\right), \tag{21}$$

for all $1 < \tau < |\mathcal{B}|$, where

$$\left(\boldsymbol{A}_{\mathcal{R}}^{(\tau)} \mid \boldsymbol{b}_{\mathcal{R}}^{(\tau)}\right) = \left(\begin{array}{cccc|c} -\boldsymbol{K}_{i,j,\mathcal{B}} & \boldsymbol{K}_{i,j,\mathcal{B}} & -1 & 1 & -\boldsymbol{I}_{i,j,\mathcal{B}} \quad \boldsymbol{I}_{i,j,\mathcal{B}} & \boldsymbol{1}_{i,\mathcal{B}}\epsilon - \boldsymbol{d}_{i,\mathcal{B}} \\ \boldsymbol{K}_{i,j,\mathcal{B}} & -\boldsymbol{K}_{i,j,\mathcal{B}} & 1 & -1 & -\boldsymbol{I}_{i,j,\mathcal{B}} \quad \boldsymbol{I}_{i,j,\mathcal{B}} & \boldsymbol{1}_{i,\mathcal{B}}\epsilon + \boldsymbol{d}_{i,\mathcal{B}} \end{array}\right) \tag{22}$$

for all $i, j \in \mathcal{R}$, where $\boldsymbol{A}_{\mathcal{R}}^{(\tau)} \in \mathbb{R}^{(2|\mathcal{R}|)\times(4|\mathcal{R}|+2)}, \boldsymbol{b}_{\mathcal{R}}^{(\tau)} \in \mathbb{R}^{2|\mathcal{R}|}$, and consequently $\boldsymbol{z}_{\mathcal{R}}^{(\tau)}, \boldsymbol{c}_{\mathcal{R}}^{(\tau)} \in \mathbb{R}^{4|\mathcal{R}|+2}$. This can greatly reduce the size of the problem since $|\mathcal{R}| < |\mathcal{B}|$.

Next, the LP solution needs to be redefined. At iteration $t = 1, 2, \ldots$, let $\boldsymbol{z}_{\mathcal{R}}^{(t)}$ be the solution to the following linear program:

$$\boldsymbol{z}_{\mathcal{R}}^{(t)} = \arg\min_{\boldsymbol{z}_{\mathcal{R}}} \left\{ \boldsymbol{c}_{\mathcal{R}}^T \boldsymbol{z}_{\mathcal{R}} \;\middle|\; \begin{array}{l} \boldsymbol{A}_{\mathcal{R}}^{(t \bmod \tau)} \boldsymbol{z}_{\mathcal{R}} - \boldsymbol{b}_{\mathcal{R}}^{(t \bmod \tau)} = e_p \\ \bar{\boldsymbol{A}}_{\mathcal{R}}^{(t \bmod \tau)-1} \boldsymbol{z}_{\mathcal{R}} - \bar{\boldsymbol{b}}_{\mathcal{R}}^{(t \bmod \tau)-1} = \bar{e}_p \end{array} \right\}, \tag{23}$$

where $e_p, \bar{e}_p$ are errors for the LP primal (23), $\left(\bar{\boldsymbol{A}}_{\mathcal{R}}^{(0)} \mid \bar{\boldsymbol{b}}_{\mathcal{R}}^{(0)}\right) = \emptyset$, and $\left(\bar{\boldsymbol{A}}_{\mathcal{R}}^{(t)} \mid \bar{\boldsymbol{b}}_{\mathcal{R}}^{(t)}\right)$ is the set of active constraints (*i.e.*, those equalities in (23) with $e_p, \bar{e}_p = 0$ for $\boldsymbol{z}_{\mathcal{R}}^{(t)}$) with positive Lagrange multipliers. Then, for the linear program (20), the solution $\boldsymbol{z}_{\mathcal{B}}^{(t)}$ at iteration t is given by

$$z_{i,\mathcal{B}}^{(t)} = \begin{cases} z_{j,\mathcal{R}}^{(t)} & \text{if } j = i, \text{ for all } j \in \mathcal{R} \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

for all $i \in \mathcal{B}$.

One of the main advantages is that the problem size can be very small, especially if the number of blocks is large. However, the size may increase as iterations progress. In the worst case scenario the size will be the same as in the traditional linear programming chunking (LPC) algorithm in which no SVR properties are exploited.

Table 1
Summary of the dimensions and properties of the data sets.

Dataset	Classes	Features M	Training N	Testing $N^{\star}$	Reference
Ripley	2	2	250	1000	Osuna and De Castro (2002)
Wine	2	13	110	20	Kinzett et al. (2008)
ADA	2	48	4147	415	Platt (1999); Joachims, 1999
GINA	2	970	3153	315	Platt (1999); Collobert and Bengio, 2001
HIVA	2	1617	3845	384	Cawley (2006)
NOVA	2	16,969	1754	175	Cawley (2006)
SYLVA	2	216	13,086	1308	Collobert and Bengio (2001); Cawley, 2006
Iris	3	4	130	20	McGarry et al. (1999)
FERET	1199	393,216	3323	3816	Phillips et al. (1998) and Phillips et al. (2000)
Color FERET	1199	393,216	12,927	1199	Phillips et al. (1998) and Phillips et al. (2000)
Textures A	2	32,768	98,304	98,304	Rosiles (2004)
Textures B	5	65,536	327,680	327,680	Rosiles (2004)
Textures C	10	131,072	262,144	262,144	Rosiles (2004)
Textures D	16	262,144	524,288	524,288	Rosiles (2004)
NEPOOL	$\mathbb{R}$	24	1461	366	Cheung et al. (1999)
MODIS A	$\mathbb{R}$	3	40 million	80 million	Rivas-Perea and Rosiles (2010)
MODIS B	$\mathbb{R}$	4	2.7 million	40 million	Rivas-Perea and Rosiles (2010)
Spiral	2	2	200	101	Xu et al. (2009)
Synthetic S	3	2	3 million	3 million	–
Synthetic NS	3	2	3 million	3 million	–
$f(x) = \mathrm{sinc}(x)$	$\mathbb{R}$	1	200	200	Peng (2010)
$f(x) = \mathrm{sinc}(x) \times \pi$	$\mathbb{R}$	1	1 million	1 million	–

5. Experimental evaluation of LS LP-SVR

To show the effectiveness and efficiency of the proposed algorithms, simulations were performed over different data sets. The summary of the properties of these data sets are shown in Table 1. Note that the simulations include classification in two and multiple classes, as well as regression problems. The *Synthetic S* is a non-linearly separable three-class problem whose classes are normally distributed. The *Synthetic NS* is identical, however, the classes are non-separable. The two last examples are for regression purposes. The data sets objective is to fit the "sinc" function, which is a typical function to approximate (Peng, 2010). The $f(x) = \text{sinc}(x)$ consists of unevenly sampled points from the sinc function. Similarly, $f(x) = \text{sinc}(x) \times \pi$ consists of unevenly spaced points from the sinc function that are affected by multiplicative white Gaussian noise (WGN); this makes it a very difficult function to fit. For all other data sets we have provided a reference where the reader can find the details of each dataset. The kernel choice in this research is only the following radial basis function (RBF) kernel: $e^{-\frac{1}{2\sigma^2}\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}$. This choice is justified since RBF kernels are the most discriminant kernels (Bermani et al., 2005; Hadzic et al., 2000) and since RBF kernels produce very large data sets, fitting perfectly with what we intend to demonstrate in this research. These experiments show results using a testing set $\mathcal{D} = \{\boldsymbol{x}_i, d_i\}_{i=1}^{N^\star}$, where $N^\star$ is the number of samples available for testing. The testing set $\mathcal{D}$ has never been shown to the LP-SVR model before. In regard to the parameters required by the proposed model and kernel, *i.e.*, ϵ, σ, and C, those were determined using an inexact globalized quasi-Newton strategy that searches for the best set of parameters minimizing error metrics as an optimization problem (Rivas-Perea et al., 2013). The architecture of the Feed-Forward Neural Network (FFNN) was determined on a trial and error basis and in every single case the best architecture was a three-layered one: input → hidden → output, that is, one input layer, one hidden layer, and one output layer; however, the number of neurons in the input and output layer varied in each case. Note that in some cases researchers prefer to treat the input layer as part of the hidden layer, in which case, this research would have two hidden layers and an output layer. Specifically, in this research the input layer and the hidden layer have a variable number of neurons, while the output layer has always one neuron which brings fairness to further comparisons with other methods. The number of neurons was determined by exhaustively searching all possible combinations of neurons in the range of $[1, 100]$, repeating the experiments ten times for each possible combination, averaging the result, and observing the combination of neurons that produced the lowest average classification error. That combination of neurons, thus, was used for that particular problem. The ten times that each possible combination is tried is implemented using an "external" 10-fold cross-validation. That is, the training set is sliced into ten different parts of (ideally) equal size, then the network is trained with nine slices and tested with the remaining slice of data. At the end the results are averaged and presented as classification error. For consistency, in all cases we used the sigmoid activation function for the input and hidden layer; it is well known that although certain activation functions may perform well for particular problems, in the average case a sigmoidal kind of function can also perform well for most of the problems (Haykin, 2009). Nevertheless, the output layer's activation function varied according to the need of each classification or problem: linear for most classification problems and softmax for most regression problems (Hart et al., 2001). This research uses the "Levenberg–Marquardt" algorithm along with with a back-propagation strategy for all cases.

Fig. 4 shows how the total training time (in minutes) varies as the size of a problem increases. The figure summarizes the results of solving several different problems in which we vary the number of samples used for training, then we average the training time to produce the curves shown. Note that there are some approaches that did not continue further in the training; this is because they reached a state of computational intractability when dealing with the problem, and they had to stop. For this reason, there is no record of such methods going further in the training as the problem size increases. With respect to the problems considered when producing Fig. 4, we have used all the problems presented in this research and also we kept drawing samples from the synthetic data sets "Synthetic S" and "Synthetic NS" as well as from both the "$f(x) = \text{sinc}(x)$" and the "$f(x) = \text{sinc}(x) \times \pi$" data sets until all other methods would break due to computational intractability. We drew up to 130 million samples of these data sets. Fig. 4 suggests that the proposed approach is slower at small to medium size problems when compared with a FFNN and a LS SVM; however, it is clearly faster for large size problems.

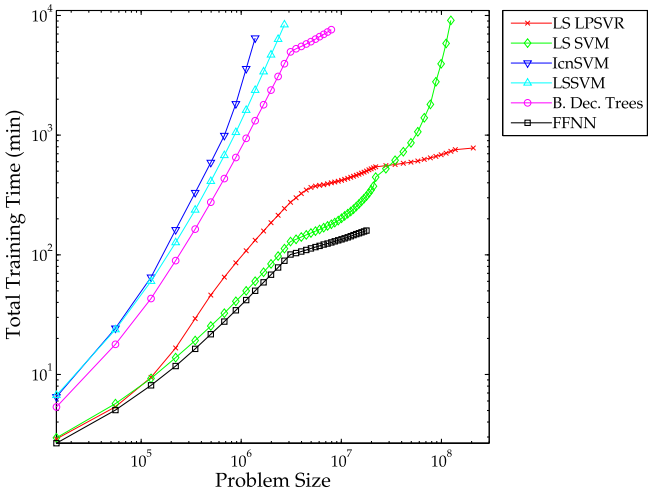

Fig. 4. Training time as a function of the problem size. LS LPSVR = Large-Scale LPSVR (proposed); LS SVM = Large-Scale SVM (Collobert and Bengio, 2001); IncSVM = Incremental SVM (Yang et al., 2005); LSSVM = Least-Squares SVM (Wang and Hu, 2005; Cawley, 2006); B.Dec.Trees = Bagged Decision (Regression) Trees (Cutler et al., 2009); FFNN = FeedForward Neural Network.

At this point it is important to remark that in every experiment, the linear program being solved exhibited a monotonic non-increasing behavior of the objective function as iterations progressed. As mentioned before in Section 1, the proposed sequential optimization has the property of a never increasing objective function, similar to Bradley's et al. work (Bradley and Mangasarian, 2000; Bradley et al., 2002). Also it was observed that most small size and small number of classes were solved in very few iterates; the converse is also true, as the problem size and number of classes increase, the model takes more iterations to solve the problem. Nevertheless, the objective function is always monotonically decreasing.

5.1. Classification

In order to measure the quality of the classifier one can also measure its generalization ability, compared with other classifiers, by observing a number of different criteria. The most used criteria is the accuracy of the classifier that is computed by counting the number of correct classifications and dividing it by the number of observations (Haykin, 2009). Table 2 shows an analysis of the accuracy of the proposed method by comparing it with other approaches. The table shows, in each row, the accuracy of the

Table 2
Comparison of the accuracy of the different approaches. The name of the approaches in each column is as follows: LS SVM = Large-Scale SVM (Collobert and Bengio, 2001); LS LPSVR = Large-Scale LPSVR (proposed); IncSVM = Incremental SVM (Yang et al., 2005); LSSVM = Least-Squares SVM (Wang and Hu, 2005; Cawley, 2006); FFNN = FeedForward Neural Network.

Dataset	Classifiers									
	LS SVM		LS LPSVR		IncSVM		LSSVM		FFNN	
Ripley	0.916	(1)	0.915	(2)	0.910	(3)	0.906	(5)	0.908	(4)
Wine	1.000	(2.5)	1.000	(2.5)	1.000	(2.5)	1.000	(2.5)	0.990	(5)
ADA	0.841	(3)	0.851	(1)	–	(5)	0.843	(2)	0.839	(4)
GINA	0.997	(2)	0.997	(2)	–	(5)	0.990	(4)	0.997	(2)
HIVA	0.870	(1.5)	0.870	(1.5)	–	(4)	–	(4)	–	(4)
NOVA	1.000	(2)	1.000	(2)	1.000	(2)	–	(4.5)	–	(4.5)
SYLVA	0.998	(2)	0.999	(1)	–	(4.5)	–	(4.5)	0.996	(3)
Iris	1.000	(1.5)	1.000	(1.5)	0.950	(4)	0.950	(4)	0.950	(4)
FERET	0.772	(1)	0.770	(2)	–	(4.5)	–	(4.5)	0.768	(3)
Color FERET	0.879	(1)	0.878	(2)	–	(4)	–	(4)	–	(4)
Textures A	0.966	(1.5)	0.966	(1.5)	–	(4)	–	(4)	–	(4)
Textures B	0.958	(1.5)	0.958	(1.5)	–	(4)	–	(4)	–	(4)
Textures C	0.927	(2)	0.928	(1)	–	(4)	–	(4)	–	(4)
Textures D	0.925	(1.5)	0.925	(1.5)	–	(4)	–	(4)	–	(4)
Spiral	1.000	(2.5)	1.000	(2.5)	1.000	(2.5)	1.000	(2.5)	0.990	(5)
Synthetic S	0.989	(2)	0.990	(1)	–	(4)	–	(4)	–	(4)
Synthetic NS	0.984	(2)	0.985	(1)	–	(4)	–	(4)	–	(4)
Avg.	0.942	(1.79)	0.943	(1.61)	–	(3.82)	–	(3.85)	–	(3.91)

Table 3
Balanced Error Rate (BER) for two-class classification problems. The rank of each classifier for each problem is shown in parenthesis. The average BER and rank is shown below; these suggest that the proposed approach is the second best ranked, yet, the difference between the best ranked method and the proposed is still less than the CD; that is, $2.05 - 1.61 = 0.44 < 0.65$, suggesting that they do not perform significantly different, which in this research is desired. In this case, the CD for only two classifiers for a level of significance of $\alpha = 0.05$ is the following: $CD = 1.960\sqrt{\frac{2\times 3}{6\times 9}} = 0.65$.

Dataset	Classifiers									
	LS SVM		LS LPSVR		IncSVM		LSSVM		FFNN	
Ripley	0.084	(1)	0.085	(2)	0.09	(3)	0.094	(5)	0.092	(4)
Wine	0.000	(2.5)	0.000	(2.5)	0.000	(2.5)	0.000	(2.5)	0.008	(5)
ADA	0.135	(1)	0.148	(2)	–	(5)	0.15	(3.5)	0.15	(3.5)
GINA	0.003	(2)	0.003	(2)	–	(5)	0.009	(4)	0.003	(2)
HIVA	0.171	(1.5)	0.171	(1.5)	–	(4)	–	(4)	–	(4)
NOVA	0.000	(2)	0.000	(2)	0.000	(2)	–	(4.5)	–	(4.5)
SYLVA	0.001	(1)	0.006	(2)	–	(4.5)	–	(4.5)	0.014	(3)
Textures A	0.034	(1)	0.035	(2)	–	(4)	–	(4)	–	(4)
Spiral	0.000	(2.5)	0.000	(2.5)	0.000	(2.5)	0.000	(2.5)	0.01	(5)
Avg.	0.047	(1.61)	0.049	(2.05)	–	(3.61)	–	(3.83)	–	(3.88)

classifier for the data set associated with that row accompanied by the rank of the classifier (shown in parenthesis) in comparison with the other classifiers. The ranking is carried following Friedman's test method (Demšar, 2006); in the case of a tie in the rank is equally divided between the tied elements (Zar, 1996). The last row includes an average of the accuracy (if it can be computed) and of the rank. It can be observed that the proposed approach has comparable results to other SVR/SVM-based and neural approaches. As problem size increases, a slight increase in accuracy can be observed. The proposed method is ranked above all the others in the average case. Also, in average, the proposed approach posses a slightly higher accuracy. However, the meaningfulness of the table can be compromised when classes are unbalanced, that is, when there are more samples of one class than of the others. With this in mind this research also considers the estimation and comparison of the Balanced Error Rate (BER) for the different methods and data sets; Table 3 shows such comparison. The BER is generally defined as follows:

$$\mathrm{BER} = \frac{1}{2}\left(\frac{FP}{TN+FP} + \frac{FN}{FN+TP}\right), \tag{25}$$

where FP, TN, FN, and TP denote the total count of "false positives", "true negatives", "false negatives", and "true positives" respectively. The advantage of using BER over the accuracy is that, the overall accuracy can be biased if classes are unbalanced, but BER offers a "fair" (*i.e.* non biased) performance error measure for two-class problems. It can be seen from Table 3 that the proposed LP-SVR learning model does not affect dramatically the performance of the classifier. On the contrary, we can observe that the error has negligible difference when compared to the others, that is, in the average case.

For the performance analysis of multi-class classification problems, the Cohen's kappa measure, κ, is better suited (Carletta, 1996). The κ measure scores the number of correct classifications independently for each class and aggregates them. This way of scoring is less sensitive to randomness caused by different number of examples in each class, therefore, it is less sensitive to become biased. Table 4 compares the different κ values for each problem and classifier. The table indicates that in the average case the performance of the proposed approach negligible compared with the other methods; however, note that for problems of a larger-scale the proposed approach performs better. Clearly, the average rank of the proposed approach is also the best.

Table 2–4, as a whole, suggest the following: (i) that the difference in performance between the best classifier and the proposed approach is very small, negligible in the average case; and (ii) that the rank of the classifiers, in the average case and for unbiased estimators, seems to favor the proposed approach for large-scale data sets in multi-class problems. Furthermore, this research

Table 4
Analysis of the Kappa coefficient (κ) for multi-class classification problems. The rank is also shown in parenthesis. The last row indicates the average κ and rank for each classifier. The last row suggests that the proposed approach performs better than the rest, in the average case. However, the difference between the two best methods is still under the CD, *i.e.*, $1.69 - 1.31 = 0.38 < 0.69$, this indicates that there is no significant difference between the two methods, which is desired. The value for the CD, in the case of comparing only two classifiers with a level of significance of $\alpha = 0.05$, is computed as follows: $CD = 1.960\sqrt{\frac{2\times3}{6\times8}} = 0.69$.

Dataset	Classifiers									
	LS SVM		LS LPSVR		IncSVM		LSSVM		FFNN	
Iris	1.000	(1.5)	1.000	(1.5)	0.950	(4)	0.950	(4)	0.950	(4)
FERET	0.772	(1)	0.770	(2)	–	(4.5)	–	(4.5)	0.768	(3)
Color FERET	0.879	(1)	0.878	(2)	–	(4)	–	(4)	–	(4)
Textures B	0.944	(2)	0.953	(1)	–	(4)	–	(4)	–	(4)
Textures C	0.919	(2)	0.923	(1)	–	(4)	–	(4)	–	(4)
Textures D	0.920	(2)	0.924	(1)	–	(4)	–	(4)	–	(4)
Synthetic S	0.979	(2)	0.984	(1)	–	(4)	–	(4)	–	(4)
Synthetic NS	0.973	(2)	0.989	(1)	–	(4)	–	(4)	–	(4)
Avg.	0.923	(1.69)	0.928	(1.31)	–	(4.06)	–	(4.06)	–	(3.88)

Table 5
Nemenyi's test, Holm's and Shaffer's procedures to analyze the Accuracy of the method. Here R_i denotes the rank of the ith approach, the level of significance is $\alpha = 0.05$, k is the number of approaches, and t_i is the maximum number of hypotheses that can be true given that $(i-1)$ hypotheses are false. The modified α value for the Nemenyi's test, Holm's and Shaffer's procedures is represented as α_{NM}, α_{HM}, and α_{SH} respectively.

i	Approaches compared	$z = \frac{R_i - R_j}{\sqrt{\frac{5\times6}{6\times17}}}$	p	α_{NM}	$\frac{\alpha_{HM}}{\frac{k(k-1)}{2}-i+1}$	$\frac{\alpha_{SH}}{t_i}$
1	LS LPSVR & FFNN	$\frac{1.61-3.91}{0.2941} = -7.8005$	6.9356×10^{-154}	0.0050	0.0050	0.0050
2	LS LPSVR & LSSVM	$\frac{1.61-3.85}{0.2941} = -7.6005$	3.7497×10^{-146}	0.0050	0.0056	0.0083
3	LS LPSVR & IncSVM	$\frac{1.61-3.82}{0.2941} = -7.5005$	2.3182×10^{-142}	0.0050	0.0063	0.0083
4	LS SVM & FFNN	$\frac{1.79-3.91}{0.2941} = -7.2084$	1.4175×10^{-131}	0.0050	0.0071	0.0083
5	LS SVM & LSSVM	$\frac{1.79-3.85}{0.2941} = -7.0044$	2.6951×10^{-124}	0.0050	0.0083	0.0083
6	LS SVM & IncSVM	$\frac{1.79-3.82}{0.2941} = -6.9024$	9.8114×10^{-121}	0.0050	0.0100	0.0083
7	LS SVM & LS LPSVR	$\frac{1.79-1.61}{0.2941} = 0.6120$	0.0458	0.0050	0.0125	0.0083
8	IncSCM & FFNN	$\frac{3.82-3.91}{0.2941} = -0.3060$	0.2322	0.0050	0.0167	0.0167
9	LSSVM & FFNN	$\frac{3.85-3.91}{0.2941} = -0.2040$	0.3136	0.0050	0.0250	0.0250
10	IncSVM & LSSVM	$\frac{3.82-3.85}{0.2941} = -0.1020$	0.3757	0.0050	0.0500	0.0500

estimated the statistical significance of the experiments; first, using the data in Table 2, Friedman's test determined that the results are statistically significant with $p = 8.0614 \times 10^{-10}$ rejecting the null-hypothesis; second, Friedman's test over the data shown in Table 3 and 4 combined also determined statistical significance with $p = 3.4845 \times 10^{-10}$ rejecting the null-hypothesis as well. The null-hypothesis being tested here is that the different approaches, presented in the comparison, perform the same and that their performance differences are random. Then, since the null hypothesis was rejected it follows to perform the post hoc Nemenyi's test (Nemenyi, 1963), the Holm's procedure, and finally the Shaffer's procedure (see (Demšar, 2006 and Garcıa and Herrera, 2008) for complete details of how to perform these tests). In the case of the Nemenyi test for the data in Table 2, the critical difference (CD), corresponds to $CD = 2.728\sqrt{\frac{5\times6}{6\times17}} = 1.48$ for a significance level of $\alpha = 0.05$; then, if the pairwise difference between classifiers rank is greater than the CD, the difference is considered significant. Clearly, there are significant differences between the large-scale-based classifiers (*i.e.*, LS SVM and LS LPSVR) and the rest; the same conclusion can be drawn from Table 3 and 4. This difference is evidently the result of the inability of the classifiers not being able to complete the training phase and produce a result; however, comparing only the first two classifiers, the CD would change to $CD = 1.960\sqrt{\frac{2\times3}{6\times17}} = 0.48$, and thus it can be concluded that there is no significant difference between the two classifiers since the difference between the two ranks is $1.79 - 1.61 = 0.18 < 0.48$. For the Nemenyi's test, the p value for each pairwise comparison is compared with an adjusted value of significance of 0.005 as shown in Table 5. If $p < 0.005$ for a particular comparison, then, the hypothesis is rejected; as can be seen from Table 5, the null hypothesis is accepted for the LS SVM and LS LPSVR methods. Thus, it is appropriate to proceed with the more refined Holm's and Shaffer's procedures to confirm this conclusion. As can be seen from Table 5, every null hypothesis is rejected until we compare the

Table 6
Mean Absolute Error. LS SVR = Large-Scale SVRCollobert and Bengio, 2001; LS LPSVR = Large-Scale LPSVR (proposed); B.Dec.Trees = Bagged Decision (Regression) TreesCutler et al., 2009; FFNN = FeedForward Neural Network.

Dataset	Classifiers							
	LS SVR		LS LPSVR		B. Dec. Trees		FFNN	
Mean absolute error								
NEPOOL (MWh)	491.38	(4)	238.69	(1)	330.08	(3)	243.18	(2)
MODIS A (Prob.)	0.2093	(2)	0.1705	(1)	0.4745	(4)	0.2336	(3)
MODIS B (Prob.)	0.1322	(2)	0.0896	(1)	0.3221	(4)	0.1588	(3)
$f(x) = \text{sinc}(x)$	0.000938	(2.5)	0.000938	(2.5)	0.000948	(4)	0.000808	(1)
$f(x) = \text{sinc}(x) \times \pi$	0.093108	(3)	0.091457	(1)	0.093721	(4)	0.092401	(2)
Avg.		(2.7)		(1.3)		(3.8)		(2.2)

Table 7
MAE analysis using Nemenyi's test, Holm's and Shaffer's procedures. Here R_i denotes the rank of the i-th approach, the level of significance is $\alpha = 0.05, k$ is the number of approaches, and t_i is the maximum number of hypotheses that can be true given that $(i-1)$ hypotheses are false. The modified α value for the Nemenyi's test, Holm's and Shaffer's procedures is represented as α_{NM}, α_{HM}, and α_{SH} respectively.

i	Approaches compared	$z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6N}}}$	p	α_{NM}	$\frac{\alpha_{HM}}{\frac{k(k-1)}{2}-i+1}$	$\frac{\alpha_{SH}}{t_i}$
1	LS LPSVR & B. Dec. Trees	$\frac{1.3-3.8}{0.8165} = -3.0618$	0.0037	0.0050	0.0083	0.0083
2	B. Dec. Trees & FFNN	$\frac{3.8-2.2}{0.8165} = 1.9596$	0.0585	0.0050	0.0100	0.0083
3	LS SVR & LS LPSVR	$\frac{2.7-1.3}{0.8165} = 1.7146$	0.0917	0.0050	0.0125	0.0125
4	LS SVR & B. Dec. Trees	$\frac{2.7-3.8}{0.8165} = -1.3472$	0.1610	0.0050	0.0167	0.0167
5	LS LPSVR & FFNN	$\frac{1.3-2.2}{0.8165} = -1.1023$	0.2173	0.0050	0.0250	0.0250
6	LS SVR & FFNN	$\frac{2.7-2.2}{0.8165} = 0.6124$	0.3307	0.0050	0.0500	0.0500

LS SVM and LS LPSVR methods; this suggests that the proposed approach performs significantly different (better in this case) than all the others, except for the LS SVM method. The same occur for the data in Table 3 and 4.

5.2. Regression

Note that Table 2–5 refer exclusively to classification problems. For regression cases, Table 6 is presented, which shows a comparison of the mean absolute testing error among neural, binary, and SVM-based learning methods. The table elucidates the idea that the proposed scheme produces results comparable to other SV-based approaches. The proposed model shows smaller errors even when compared with the neural network-based approach at a large-scale. The rank is also shown in Table 6; the ranking indicates that the proposed approach performs better in the average case, obtaining the highest ranking. The critical difference for this case corresponds to $CD = 2.569\sqrt{\frac{4\times5}{6\times5}} = 2.0976$. Therefore, we can conclude that there is no significant difference between all methods except between the proposed approach and the Binary Decision Trees for Regression (BDTR); in that case, the null hypothesis is rejected. However, considering only two methods, *i.e.*, the proposed LS LPSVR and the LS SVR, which would be direct competitors, the CD would be $CD = 1.960\sqrt{\frac{2\times3}{6\times5}} = 0.8765$; and using this CD, the null hypothesis can be rejected.

Proceeding with the Nemenyi's test, Holm's and Shaffer's procedures, it can be concluded that in general the null hypothesis is rejected for the proposed approach when compared with the BDTR, as shown in Table 7. In all the other cases, the null hypothesis is accepted and no significant difference can be appreciated in terms of these procedures. The reader should be reminded at this point, that the objective of this research is not to provide with a method that performs better than the others, instead, the objective here is to provide with an alternative method that makes large-scale training computationally tractable without loosing generalization capabilities, *e.g.*, sacrificing performance. Therefore, according to the results shown here, this research seems to accomplish the proposed objective. This can be confirmed using more and different error/performance metrics and datasets. In this regard, the author want to point out that the proposed LP-SVR model with the large-scale training strategy introduced here was also evaluated using other performance metrics, such as (i) *TP* rate, (ii) *FP* rate, (iii) accuracy, (iv) specificity, (v) positive predictive value, (vi) negative predictive value, (vii) false discovery rate, (viii) Matthews correlation coefficient, (ix) F_1-score, (x) estimate of scaled error rate, (xi) root mean squared error, (xii) normalized root mean squared error, and (xiii) area under the receiver operating characteristics curve. Nonetheless, the results lead to the same conclusions as those tables shown in this article; however, all this results will be included in a follow-up article that will include more experiments and other theoretical developments that space constraints preclude us to address in this paper.

6. Conclusion

6.1. Large-Scale learning

The proposed large-scale model involved three major parts that increases computational tractability of large-scale problems. First, the Large-Scale LP-SVR problem is reduced by variable decomposition a.k.a. column/variable chunking which exploits LP-SVR structure to produce a lower-dimensional representation of the decomposed LP sub-problem. The proof of finite termination of the decomposition strategy is guaranteed by an infinite-loop prevention algorithm and by previous work from Torii, et al. in (Torii and Abe, 2009).

Second, the authors take a constraint decomposition strategy a.k.a. row/constraint chunking that generates a number of smaller sub-problems by exploiting again LP-SVR structure. The resulting LP problems are solved sequentially in a monotonically non-increasing objective function fashion. Proof of convergence is given applying Bradley's (Bradley et al., 2002) theorems under the correct assumptions.

Third, the two-way-reduced LP-SVR sub-problems are solved using interior point methods, which have a very high rate of convergence. This is the key that balances the total computational time of performing two decompositions and solving several LPs. Using other approaches (*e.g.* the simplex method) would dramatically increase computational efforts.

6.2. Model performance and sparseness

Finally, from the experimental results shown in this research we can conclude that the proposed approach is comparable with other formulations in terms of performance, which is desirable. However, in this work the authors are not looking for better performances, instead, it is desired to train large-scale data sets and at the same time produce sparser solutions in terms of the number of Support Vectors. As is known, if the solution is sparser, the model is more efficient in the testing phase. That is, a sparse solution implies fewer number of Support Vectors and hence, future kernel evaluations can be made faster. The point here is that it was demonstrated that the proposed model provides sparse solutions; this was achieved by extending the research by Zhang and Zhou (2010), to our LP-SVR problem formulation. Future work includes experimental demonstration that as the problem size increases, the sparser the solution becomes.

Acknowledgement

Portions of the research in this paper use the FERET database of facial images collected under the FERET program, sponsored by the DOD Counterdrug Technology Development Program Office.

The author P.R.P performed part of this work while at NASA Goddard Space Flight Center as part of the Graduate Student Summer Program (GSSP 2009) under the supervision of Dr. James C. Tilton. This work was supported in part by the National Council for Science and Technology (CONACyT), Mexico, under grant 193324/303732, and by the University of Texas El Paso Graduate School Cotton Memorial Funding. The partial support of the SEP DGRI complementary scholarship is also acknowledged. Finally, the authors acknowledge the support of the Large–Scale Multispectral Multidimensional Analysis (LSMMA) Laboratory (www.lsmma.com).

References

Argáez, M., Velázquez, L., 2003. A new infeasible interior-point algorithm for linear programming. In: Proc. 2003 Conf. on Diversity in Computing, TAPIA'03. ACM, New York, USA, pp. 12–14.

Bermani, E., Boni, A., Kerhet, A., Massa, A., 2005. Kernels evaluation of SVM based estimators for inverse scattering problems. Prog. Electromagnetics Res. 53, 167–188.

Bo, L., Jiao, L., Wang, L., 2007. Working set selection using functional gain for ls-svm. IEEE Trans. Neural Networks 18, 1541–1544.

Bradley, P., Mangasarian, O., 2000. Massive data discrimination via linear support vector machines. Optim. Methods Software 13, 1–10.

Bradley, P., Mangasarian, O., Musicant, D., 2002. Optimization methods in massive data sets. Handbook of Massive Data Sets. Kluwer Academic Publishers, pp. 439–471.

Carletta, J., 1996. Squibs and discussions assessing agreement on classification tasks: The kappa statistic. Comput. Linguistics 22, 249–254.

Cawley, G., 2006. Leave-one-out cross-validation based model selection criteria for weighted ls-svms. In: IEEE Internat. Conf. on Neural Networks (IJCNN'06).

Chen, G., Xu, J., Xiang, X., 2008. Neighborhood preprocessing svm for large-scale data sets classification. In: Fifth Internat. Conf. on Fuzzy Systems and Knowledge Discovery (FSKD'08), vol. 2, pp. 245–249.

Cheung, K., Shamsollahi, P., Sun, D., Milligan, J., Potishnak, M., 1999. Energy and ancillary service dispatch for the interim iso new england electricity market. In: Proc. 21st IEEE Internat. Conf. on Power Industry Computer Applications (PICA'99), pp. 47–53.

Collobert, R., Bengio, S., 2001. Svmtorch: Support vector machines for large-scale regression problems. J. Machine Learn. Res. 1, 143–160.

Collobert, R., Bengio, S., 2004. A gentle Hessian for efficient gradient descent. In: Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing (ICASSP'04), vol. 5, pp. V-517-20

Courant, R., Hilbert, D., 1966. Methods of Mathematical Physics. Interscience, New York.

Cristianini, N., Kandola, J., Elisseeff, A., Shawe-Taylor, J., 2006. On kernel target alignment. Innov. Machine Learn., 205–256.

Cutler, A., Cutler, D., Stevens, J., 2009. Tree-based methods. High-Dimensional Data Analysis in Cancer Research, 1–19.

Dantzig, G., 1998. Linear Programming and Extensions. Princeton Univ. Press.

Dantzig, G., Thapa, M., 1997. Linear Programming: Introduction. Springer Verlag.

Debnath, R., Takahashi, H., 2006. Svm training: Second-order cone programming versus quadratic programming. In: Internat. Joint Conf. on Neural Networks (IJCNN'06), pp. 1162–1168.

Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Machine Learn. Res. 7, 1–30.

Dennis, J., Schnabel, R., 1996. Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Society for Industrial Mathematics.

Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A.J., Vapnik, V., 1996. Support vector regression machines. In: Neural Information Processing Systems (NIPS), pp. 155–161.

Du, H., Teng, S., Fu, X., Zhang, W., Pu, Y., 2009. A cooperative intrusion detection system based on improved parallel svm. In: Joint Conf. on Pervasive Computing (JCPC), pp. 515–518.

Ferris, M.C., Mangasarian, O.L., Wright, S.J., 2007. Linear Programming with MATLAB. Society for Industrial and Applied Mathematics.

Garcıa, S., Herrera, F., 2008. An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. J. Machine Learn. Res. 9, 66.

Hadzic, I., Kecman, V., 2000. Support vector machines trained by linear programming: Theory and application in image compression and data classification. In: Proc. 5th Seminar on Neural Network Applications in Electrical Engineering (NEUREL 2000), pp. 18–23.

Hart, P., Duda, R., Stork, D., 2001. Pattern Classification. Wiley.

Haykin, S.S., 2009. Neural Networks and Learning Machines. Prentice Hall.

Hazan, T., Man, A., Shashua, A., 2008. A parallel decomposition solver for svm: Distributed dual ascend using fenchel duality. In: IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2008), pp. 1–8.

Hu, W., Song, Q., 2004. An accelerated decomposition algorithm for robust support vector machines. IEEE Trans. Circ. Systems II: Express Briefs 51, 234–240.

Huang, F.J., LeCun, Y., 2006. Large-scale learning with svm and convolutional for generic object categorization. In: IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, vol. 1, pp. 284–291.

Joachims, T., 1999. Making Large Scale Svm Learning Practical. Advances in Kernel Methods. MIT Press, pp. 169–184.

Kinzett, D., Zhang, M., Johnston, M., 2008. Using numerical simplification to control bloat in genetic programming. Simul. Evol. Learn., 493–502.

Lanckriet, G., Cristianini, N., Bartlett, P., Ghaoui, L., Jordan, M., 2004. Learning the kernel matrix with semidefinite programming. J. Machine Learn. Res. 5, 27–72.

Lu, Z., Sun, J., Butts, K.R., 2009. Linear programming support vector regression with wavelet kernel: A new approach to nonlinear dynamical systems identification. Math. Comput. Simul. 79, 2051–2063.

Mangasarian, O., Musicant, D., 1999. Successive overrelaxation for support vector machines. IEEE Trans. Neural Networks 10, 1032–1037.

Mangasarian, O., Musicant, D., 2002. Large scale kernel regression via linear programming. Machine Learn. 46, 255–269.

McGarry, K., Wermter, S., MacIntyre, J., 1999. Knowledge extraction from radial basis function networks and multilayer perceptrons. In: IEEE Internat. Joint Conf. on Neural Networks (IJCNN'99), vol. 4, pp. 2494–2497.

Mercer, J., 1909. Functions of positive and negative type, and their connection with the theory of integral equations. Philos. Trans. Roy. Soc. London. Ser. A (Containing Papers of a Mathematical or Physical Character) 209, 415–446.

Nemenyi, P.B., 1963. Distribution-free multiple comparisons, Ph.D. thesis, Princeton University.

Neumann, J., 1945. A model of general economic equilibrium. Rev. Econ. Stud. 13, 1–9.

Nishida, K., Kurita, T., 2008. Ransac-svm for large-scale datasets. In: 19th Internat. Conf. on Pattern Recognition (ICPR 2008), pp. 1–4.

Osuna, E., De Castro, O., 2002. Convex hull in feature space for support vector machines. Adv. Artif. Intell. – IBERAMIA 2002, 411–419.

Osuna, E., Freund, R., Girosi, F., 1997. An improved training algorithm for support vector machines. In: Proc. IEEE Workshop Neural Networks Signal Process VII, pp. 276–285.

Papadonikolakis, M., Bouganis, C.-S., 2008. Efficient fpga mapping of gilbert algorithm for svm training on large-scale classification problems. In: Internat. Conf. on Field Programmable Logic and Applications (FPL 2008), pp. 385 –390.

Papageorgiou, C., Girosi, F., Poggio, T., 1999. Sparse correlation kernel reconstruction, In: Proc. IEEE Internat. Conf. on Acoustics, Speech and Signal Processing (ICASSP'99), vol. 3, pp. 1633–1636

Peng, X., 2010. Tsvr: An efficient twin support vector machine for regression. Neural Networks 23, 365–372.

Phillips, P., Wechsler, H., Huang, J., Rauss, P., 1998. The feret database and evaluation procedure for face-recognition algorithms. Image Vision Comput. 16, 295–306.

Phillips, P., Moon, H., Rizvi, S., Rauss, P., 2000. The feret evaluation methodology for face-recognition algorithms. IEEE Trans. Pattern Anal. Machine Intell. 22, 1090–1104.

Platt, J., 1999. Using analytic qp and sparseness to speed training of support vector machines. Adv. Neural Inf. Process. Systems, 557–563.

Rivas-Perea, P., Cota-Ruiz, J., Perez Venzor, J.A., Garcia Chaparro, D., 2013. LP-SVR model selection using an nexact globalized quasi-Newton strategy. J. Intell. Learn. Syst. Appl. Vol.5, No.1.

Rivas-Perea, P., Rosiles, J., 2010. A probabilistic model for stratospheric soil-independent dust aerosol detection. Digital Image Processing and Analysis. Optical Society of America.

Rosiles, J., 2004. Image and texture analysis using biorthogonal angular filter banks, Ph.D. thesis, Georgia Institute of Technology.

Scholkopf, B., 2001. The kernel trick for distances. Adv. Neural Inf. Process. Systems, 301–307.

Scholkopf, B., Smola, A.J., 2002. Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond. The MIT Press.

Shen, M., Lin, L., Chen, J., Chang, C.Q., 2010. A prediction approach for multichannel eeg signals modeling using local wavelet svm. IEEE Trans. Instrum. Meas. 59, 1485–1492.

Smola, A.J., Scholkopf, B., 2004. A tutorial on support vector regression. Statist. Comput. 14, 199–222.

Smola, A., Scholkopf, B., Ratsch, G., 1999. Linear programs for automatic accuracy control in regression. In: 9th Internat. Conf. on Artificial Neural Networks (ICANN 99). (Conf. Publ. No. 470).

Torii, Y., Abe, S., 2009. Decomposition techniques for training linear programming support vector machines. Neurocomputing 72, 973–984.

Wang, H., Hu, D., 2005. Comparison of svm and ls-svm for regression. In: Internat. Conf. on Neural Networks and Brain (ICNN B'05), vol. 1, pp. 279–283.

Xu, Z., Huang, K., Zhu, J., King, I., Lyu, M.R., 2009. A novel kernel-based maximum a posteriori classification method. Neural Networks 22, 977–987.

Yang, J., Li, Z.-W., Zhang, J.-P., 2005. A training algorithm of incremental supprot vector machine with recombining method [supprot read support]. In: Proc. Internat. Conf. on Machine Learning and Cybernetics, vol. 7, pp. 4285–4288.

Yan-zi, X., Hua, Q., 2009. A new optimization method of large-scale svms based on kernel distance clustering. In: Internat. Conf. on Computational Intelligence and Software Engineering (CiSE 2009), pp. 1–4.

Yongping, L., Dongtao, Q., 2002. Accelerating svm on large scale regression problem. In: Proc. IEEE Region 10 Conf. on Computers, Communications, Control and Power Engineering, vol. 1, pp. 65–68.
Zar, J.H., 1996. Biostatistical Analysis. Prentice Hall.
Zhang, L., Zhou, W., 2010. On the sparseness of 1-norm support vector machines. Neural Networks 23, 373–385.
Zhang, J.-P., Li, Z.-W., Yang, J. 2005. A parallel svm training algorithm on large-scale classification problems. In: Proceedings of 2005 Internat. Conf. on Machine Learning and Cybernetics, vol. 3, pp. 1637–1641.
Li, X., Zhu, Y., Sung, E., 2005. Sequential bootstrapped support vector machines – a svm accelerator. In: Proc. IEEE Internat. Joint Conf. on Neural Networks (IJCNN'05), vol. 3, pp. 1437–1442.