Support Vector Regression for the simultaneous learning of a multivariate function and its derivatives

Marcelino Lázaro[†], Ignacio Santamaría[‡], Fernando Pérez-Cruz[†,*]

Antonio Artés-Rodríguez[†]

[†]*Departamento de Teoría de la Señal y Comunicaciones*

Universidad Carlos III, Leganés, 28911, Madrid, Spain

e-mail: {marce,fernandop,antonio}@ieee.org

[‡]*Departamento de Ingeniería de Comunicaciones*

Universidad de Cantabria, 39005 Santander, Spain

e-mail: nacho@gtas.dicom.unican.es

[*]*Gatsby Computational Neuroscience Unit, UCL.*

Alexandra House, 17 Queen Square, London WC1N 3AR, UK.

Abstract

In this paper, the problem of simultaneously approximating a function and its derivatives is formulated within the Support Vector Machine (SVM) framework. First, the problem is solved for a one-dimensional input space by using the ε-insensitive loss function and introducing additional constraints in the approximation of the derivative. Then, we extend the method to multi-dimensional input spaces by a multidimensional regression algorithm. In both cases, to optimize the regression estimation problem, we have derived an Iterative Re-Weighted Least Square (IR-WLS) procedure that works fast for moderate size problems. The proposed method shows that using the information about derivatives significantly improves the reconstruction of the function.

Key words:
SVM, IRWLS

1 Introduction

Regression approximation of a given data set is a very common problem in a number of applications. In some of these applications, like economy, device modeling, telemetry, etc, it is necessary to fit not only the underlying characteristic function but also its derivatives, which are often available. The problem of learning a function and its derivatives have been addressed, for instance, in the neural networks literature, to analyze the capability of several kind of networks [1], [2], or in some applications [3],[4]. Some other methods have been employed to simultaneously approximate a set of samples of a function and its derivative: splines, or filter bank-based methods are some examples (see [5] and references therein).

On the other hand, Support Vector Machines are state-of-the-art tools for linear and nonlinear input-output knowledge discovery [6,7]. The Support Vector Machines, given a labeled dataset $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \in \mathbb{R}^d$ for $i = 1, \ldots, N$, and a function $\boldsymbol{\phi}(\cdot)$ that nonlinearly transforms the input vector $\mathbf{x}_i$ to a higher dimensional space, solve either classification ($y_i \in \{\pm 1\}$) or regression ($y_i \in \mathbb{R}$)

[1] This work has been partially supported by grants CAM 07T/0016/2003, CYCIT TIC2003-2602, and TIC2001-0751-C04-03. Fernando Pérez Cruz is also supported by Spanish Ministry of Education Postdoctoral fellowship EX2003-0437

problems.

In this paper, we will deal with the regression approximation problem and we will extend the SVM framework when prior knowledge about the derivatives of the functional relationship between $\mathbf{x}$ and y is known.

First, we will solve the problem in a one-dimensional problem ($d = 1$) by using the ε-insensitive loss function and introducing linear constraint for the derivatives. Then, we will extend the method to multidimensional input spaces. In both cases, the corresponding method will lead to a solution similar to the SVM in which we have Support Vectors related to the function value and Support Vectors related to the derivatives values. Together, both kind of support vectors form the complete SVM expansion for regression approximation with information about the derivatives of the function. The solution to the proposed algorithms is obtained using an Iterative Re-Weighted Least Square (IRWLS) procedure, which has been successfully applied to the regular SVM for classification [8] and for regression [9]. This algorithm has been recently proven to converge to the SVM solution [10].

2 Proposed one-dimensional SVM-based approach

The one-dimensional problem can be stated as follows: to find the functional relation between x and y giving a labeled data set, $(x_i, y_i, y_i^{'})$, where $y_i \in \mathbb{R}$ and $y_i^{'} \in \mathbb{R}$ is the derivative of the function to be approximated at x_i. The proposed method is an extension of the Support Vector Machine for Regression (SVR) employing the Vapnik's ε-insensitive loss function [6]. The SVR obtains a linear regressor in the transformed space (feature space)

$$f(x) = \mathbf{w}^T \boldsymbol{\phi}(x) + b, \tag{1}$$

where $\mathbf{w}$ and b define the linear regression [2], which is nonlinear in the input space (unless $\boldsymbol{\phi}(x) = x$). Roughly speaking, the SVR minimizes the squared norm of the weight vector $\mathbf{w}$, while linearly penalizes deviations greater than ε.

With respect to the conventional SVR cost function, the proposed method adds a new penalty term: the errors in the derivative that are out of its associated insensitive region. In the general case, a different parameter is employed to define the insensitive region size for the function (ε) and for the derivative (ε'). Taking this extension into account, the proposed approach minimizes

$$L_P(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\tau}, \boldsymbol{\tau}^*) = \frac{1}{2}||\mathbf{w}||^2 + C_1 \sum_{i=1}^{N} (\xi_i + \xi_i^*) + C_2 \sum_{i=1}^{N} (\tau_i + \tau_i^*), \tag{2}$$

subject to

$$\mathbf{w}^T \boldsymbol{\phi}(x_i) + b - y_i \leq \varepsilon + \xi_i, \tag{3}$$

$$y_i - \mathbf{w}^T \boldsymbol{\phi}(x_i) - b \leq \varepsilon + \xi_i^*, \tag{4}$$

$$\mathbf{w}^T \boldsymbol{\phi}'(x_i) - y_i' \leq \varepsilon' + \tau_i, \tag{5}$$

$$y_i' - \mathbf{w}^T \boldsymbol{\phi}'(x_i) \leq \varepsilon' + \tau_i^*, \tag{6}$$

$$\xi_i, \xi_i^*, \tau_i, \tau_i^* \geq 0, \tag{7}$$

for $i = 1, 2, \cdots, N$. The positive slack variables ξ_i, ξ_i^*, τ_i and τ_i^* are responsible for penalizing errors greater than ε and ε', respectively, in the function and derivative, and $\boldsymbol{\phi}'(x)$ denotes the derivative of $\boldsymbol{\phi}(x)$. To solve this problem, a

[2] All vectors will be column-vectors. We will denote the scalar product as a matrix multiplication of a row-vector by a column-vector, and ^{T} denotes transpose.

Lagrangian functional is used to introduce the previous linear constraints, as usual in the classical SVM framework [7].

The Lagrangian has to be minimized with respect to $\mathbf{w}$, b, ξ, ξ^*, τ and τ^*, and maximized with respect to the Lagrange multipliers. The solution to this problem can be obtained considering the Karush-Kuhn-Tucker (KKT) complementary conditions, which lead to a weight vector $\mathbf{w}$ taking the form (see [11] for details)

$$\mathbf{w} = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)\phi(x_i) + \sum_{i=1}^{N}(\lambda_i^* - \lambda_i)\phi'(x_i). \tag{8}$$

where α_i, α_i^*, λ_i and λ_i^* are, respectively, the lagrange multipliers associated with constraints (3)-(6). Therefore, the regression estimation for a new sample x can be computed as follows:

$$f(x) = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)\boldsymbol{\phi}^T(x_i)\boldsymbol{\phi}(x) + \sum_{i=1}^{N}(\lambda_i^* - \lambda_i)\boldsymbol{\phi}'^T(x_i)\boldsymbol{\phi}(x) + b. \tag{9}$$

In the SVM framework, the nonlinear transformation $\boldsymbol{\phi}(x)$ is not needed to be explicitly known and it can be replaced by the kernel of the nonlinear transformation. In this case, $\boldsymbol{\phi}^T(x_i)\boldsymbol{\phi}(x_j)$ is substituted by $K(x_i, x_j)$, a kernel satisfying Mercer Theorem [7]. From this definition for the kernel, it is easy to demonstrate that,

$$\boldsymbol{\phi}'^T(x_i)\boldsymbol{\phi}(x_j) = \frac{\partial K(x_i, x_j)}{\partial x_i} \triangleq K'(x_i, x_j), \tag{10}$$

$$\boldsymbol{\phi}^T(x_i)\boldsymbol{\phi}'(x_j) = \frac{\partial K(x_i, x_j)}{\partial x_j} \triangleq G(x_i, x_j), \tag{11}$$

and

$$\boldsymbol{\phi}'^T(x_i)\boldsymbol{\phi}'(x_j) = \frac{\partial^2 K(x_i, x_j)}{\partial x_i \partial x_j} \triangleq J(x_i, x_j). \tag{12}$$

Though $K(\cdot,\cdot)$ must be a Mercer Kernel, its derivatives do not necessarily have to. Therefore, using a valid kernel $K(\cdot,\cdot)$, once the Lagrange multipliers have been obtained, the regression estimate takes the form

$$f(x) = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)K(x_i, x) + \sum_{i=1}^{N}(\lambda_i^* - \lambda_i)K'(x_i, x) + b. \tag{13}$$

where we have only used the kernel of the transformation without explicitly computing the nonlinear transformation. We will show, in the following subsection, that the resolution of the minimization problem can be done as well using kernels, so one does not need to know the nonlinear transformation, as in the regular SVM framework.

2.1 IRWLS algorithm

The problem can be solved following the classical SVM method [7]: to arrive to the Wolfe's dual problem, which gives a quadratic functional depending only on the Lagrange multipliers that can be solved by Quadratic Programming (QP) techniques. However, the QP solution of the system can be computationally expensive, especially when a large number of samples is employed, which can make the problem unaffordable. In order to reduce the computational burden, an Iterative Re-Weighted Least Square (IRWLS) procedure has been developed. This IRWLS algorithm follows the same basic idea proposed in [9], but we are going to develop it following [10], which is much more comprehensible and from which the convergence naturally follows. We will first state it as an unconstrained optimization problem

$$L_P(\mathbf{w}, b) = \frac{1}{2}||\mathbf{w}||^2 + C_1 \sum_{i=1}^{N} \left(L(e_i) + L(e_i^*)\right) + C_2 \sum_{i=1}^{N} \left(L(d_i) + L(d_i^*)\right), \tag{14}$$

where

$$e_i = \mathbf{w}^T \boldsymbol{\phi}(x_i) + b - y_i - \varepsilon \tag{15}$$

$$e_i^* = y_i - \mathbf{w}^T \boldsymbol{\phi}(x_i) - b - \varepsilon \tag{16}$$

$$d_i = \mathbf{w}^T \boldsymbol{\phi}'(x_i) - y_i' - \varepsilon' \tag{17}$$

$$d_i^* = y_i' - \mathbf{w}^T \boldsymbol{\phi}'(x_i) - \varepsilon' \tag{18}$$

and $L(u) = \max(u, 0)$. The proof of convergence in [10] uses a differentiable approximation to this non-differentiable function:

$$L(u) = \begin{cases} 0, & u < 0 \\ Ku^2/2, & 0 \leq u < 1/K \\ u - 1/(2K), & u \geq 1/K \end{cases}$$

to ensure the converge of the algorithm, which tends to $\max(u, 0)$ as K tends to infinity. But it also shows that K can be made arbitrary large.

Optimization problems are solved using iterative procedures that relies in each iteration in the previous solution ($\mathbf{w}^k$ and b^k, in our case) to obtain the following one, until the optimal solution has been reached. To construct the IRWLS procedure, we modify (14) using a first order Taylor expansion of $L(u)$ over the previous solution, leading to:

$$\begin{aligned} L_P^{'}(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 &+ C_1 \left(\sum_{i=1}^{N} L(e_i^k) + \left.\frac{dL(u)}{du}\right|_{e_i^k} [e_i - e_i^k] \right) \\ &+ C_1 \left(\sum_{i=1}^{N} L({e_i}^{*k}) + \left.\frac{dL(u)}{du}\right|_{{e_i}^{*k}} [e_i^* - {e_i}^{*k}] \right) \\ &+ C_2 \left(\sum_{i=1}^{N} L(d_i^k) + \left.\frac{dL(u)}{du}\right|_{d_i^k} [d_i - d_i^k] \right) \\ &+ C_2 \left(\sum_{i=1}^{N} L({d_i}^{*k}) + \left.\frac{dL(u)}{du}\right|_{{d_i}^{*k}} [d_i^* - {d_i}^{*k}] \right), \end{aligned} \tag{19}$$

where $e_i^k = \mathbf{w}^{k^T}\boldsymbol{\phi}(x_i) + b^k - y_i - \varepsilon$ (the others follow the same definition), $L_P^{'}(\mathbf{w}^k, b^k) = L_P(\mathbf{w}^k, b^k)$ and $\nabla L_P^{'}(\mathbf{w}^k, b^k) = \nabla L_P(\mathbf{w}^k, b^k)$. Now, we construct a quadratic approximation imposing that $L_P^{''}(\mathbf{w}^k, b^k) = L_P(\mathbf{w}^k, b^k)$ and $\nabla L_P^{''}(\mathbf{w}^k, b^k) = \nabla L_P(\mathbf{w}^k, b^k)$, leading to:

$$\begin{aligned}
L_P^{''}(\mathbf{w}, b) = \frac{1}{2}\|\mathbf{w}\|^2 &+ C_1\left(\sum_{i=1}^{N} L(e_i^k) + \left.\frac{dL(u)}{du}\right|_{e_i^k}\frac{(e_i)^2 - (e_i^k)^2}{2e_i^k}\right) = \\
&+ C_1\left(\sum_{i=1}^{N} L({e_i}^{*k}) + \left.\frac{dL(u)}{du}\right|_{{e_i}^{*k}}\frac{({e_i}^*)^2 - ({e_i}^{*k})^2}{2{e_i}^{*k}}\right) = \\
&+ C_2\left(\sum_{i=1}^{N} L(d_i^k) + \left.\frac{dL(u)}{du}\right|_{d_i^k}\frac{(d_i)^2 - (d_i^k)^2}{2d_i^k}\right) = \\
&+ C_2\left(\sum_{i=1}^{N} L({d_i}^{*k}) + \left.\frac{dL(u)}{du}\right|_{{d_i}^{*k}}\frac{({d_i}^*)^2 - ({d_i}^{*k})^2}{2{d_i}^{*k}}\right) = \\
= \frac{1}{2}\|\mathbf{w}\|^2 &+ \frac{1}{2}\sum_{i=1}^{N} a_i e_i^2 + a_i^* {e_i}^{*2} + s_i d_i^2 + s_i^* {d_i}^{*2} + \text{CT}, \qquad (20)
\end{aligned}$$

where

$$a_i = \left.\frac{C_1}{e_i^k}\frac{dL(u)}{du}\right|_{e_i^k} \quad a_i^* = \left.\frac{C_1}{{e_i}^{*k}}\frac{dL(u)}{du}\right|_{{e_i}^{*k}} \quad s_i = \left.\frac{C_2}{d_i^k}\frac{dL(u)}{du}\right|_{d_i^k} \quad s_i^* = \left.\frac{C_2}{{d_i}^{*k}}\frac{dL(u)}{du}\right|_{{d_i}^{*k}}$$

and CT are constant terms that do not depend on neither $\mathbf{w}$ nor b.

The value of a_i can be computed as follows:

$$a_i = \left.\frac{C_1}{e_i^k}\frac{dL(u)}{du}\right|_{e_i^k} = \begin{cases} 0, & e_i^k < 0 \\ KC_1, & 0 \le e_i^k < 1/K \\ C_1/e_i^k, & e_i^k \ge 1/K \end{cases}$$

This definition can be readily extended to a_i^*, s_i and s_i^*.

From the definition of e_i and e_i^*, we can infer that either one of them is positive or both are negative, but they cannot be both positive at the same time. This property means that either a_i or a_i^* are nonzero or both are zero, but they cannot be both nonzero at the same time. The samples that present

$a_i = a_i^* = 0$ do not need to be considered in the resolution of the functional in (20), as they will not be support vectors and will not contribute to the value of $\mathbf{w}$. Therefore, the sum in (20) should only run for those samples that either a_i or a_i^* are nonzero. In the derivation of the algorithm, we will suppose that all of the samples present either a nonzero a_i or a_i^*. So we can work with a simple notation and we do not need to introduce a new index indicating which samples present either a nonzero a_i or a_i^*. But when implementing the procedure, we will only consider those samples that can be support vectors, basically in each iteration we can assume that the training samples are limited to those samples with a nonzero a_i or a_i^*, letting aside the rest of the samples. After, each iteration we will compute e_i or e_i^* for all the samples, so if any of the previous samples with $a_i = a_i^* = 0$ now presents a positive e_i or e_i^*, it can be recovered for the following iteration. This discussion about the values of a_i or a_i^* is very relevant as it will allow us to solve the problem with a reduced set of samples in each iteration, but we will obtain the SVM solution, when the algorithm stops, as we can recover samples that at some stage presented $a_i = a_i^* = 0$. Finally, this argument can be carried out for d_i and d_i^* and s_i and s_i^*, as well.

The IRWLS procedure consists in minimizing (20), then recomputing a_i, a_i^*, s_i and s_i^* with the obtained solution, and continue until the solution has been reached. Each iteration can be seen similar to Least Squares SVM [12], but the IRWLS preserves the sparseness property of the SVM because of the insensitivity region. To solve (20), we take derivative with respect to $\mathbf{w}$ and b, and equate to zero, thus giving as result

$$\begin{bmatrix} \mathbf{\Phi}^T\mathbf{D}_{\mathbf{a}+\mathbf{a}^*}\mathbf{\Phi} + \mathbf{\Phi}'^T\mathbf{D}_{\mathbf{s}+\mathbf{s}^*}\mathbf{\Phi}' + \mathbf{I} & \mathbf{\Phi}^T(\mathbf{a}+\mathbf{a}^*) \\ (\mathbf{a}+\mathbf{a}^*)^T\mathbf{\Phi} & (\mathbf{a}+\mathbf{a}^*)^T\mathbf{1} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{\Phi}^T[\mathbf{D}_{\mathbf{a}+\mathbf{a}^*}\mathbf{y} + (\mathbf{a}-\mathbf{a}^*)\varepsilon] + \mathbf{\Phi}'^T[\mathbf{D}_{\mathbf{s}+\mathbf{s}^*}\mathbf{y}' + (\mathbf{s}-\mathbf{s}^*)\varepsilon'] \\ (\mathbf{a}+\mathbf{a}^*)^T\mathbf{y} + (\mathbf{a}-\mathbf{a}^*)^T\mathbf{1}\varepsilon \end{bmatrix}, \quad (21)$$

where $\mathbf{a}$, $\mathbf{a}^*$, $\mathbf{s}$ and $\mathbf{s}^*$ are the column vectors containing the N corresponding weights in (20), $\mathbf{D_a}$ denotes a diagonal matrix ($\mathbf{D}_{\mathbf{a}_{ij}} = a_i\delta(i-j)$), and

$$\begin{array}{c} \mathbf{\Phi} = [\boldsymbol{\phi}(x_1), \boldsymbol{\phi}(x_2), \ldots, \boldsymbol{\phi}(x_N)]^T \\ \mathbf{\Phi}' = [\boldsymbol{\phi}'(x_1), \boldsymbol{\phi}'(x_2), \ldots, \boldsymbol{\phi}'(x_N)]^T \end{array}. \quad (22)$$

2.2 IRWLS with Kernels

The system (21) can be solved, as well, using kernels, when $\boldsymbol{\phi}(\cdot)$ is unknown or infinite dimensional. We can make use of the Representer theorem [7] that, under fairly general conditions, states that the best solution can be expressed as a linear combination of the training samples in the feature space:

$$\mathbf{w} = [\mathbf{\Phi}^T \quad \mathbf{\Phi}'^T] \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{\gamma} \end{bmatrix}. \quad (23)$$

From (8), we can notice that once the solution has been reached $\boldsymbol{\beta} = \boldsymbol{\alpha}^* - \boldsymbol{\alpha}$ and $\boldsymbol{\gamma} = \boldsymbol{\lambda}^* - \boldsymbol{\lambda}$. We will use (23) to replace $\mathbf{w}$ in (21). But, first, we have rewritten the first set of equations, moving b to the second term, so it will be simpler to obtain the kernel representation

$$\left([\mathbf{\Phi}^T \quad \mathbf{\Phi}'^T] \begin{bmatrix} \mathbf{D}_{\mathbf{a}+\mathbf{a}^*} & 0 \\ 0 & \mathbf{D}_{\mathbf{s}+\mathbf{s}^*} \end{bmatrix} \begin{bmatrix} \mathbf{\Phi} \\ \mathbf{\Phi}' \end{bmatrix} + \mathbf{I} \right) \mathbf{w} = \\ = [\mathbf{\Phi}^T \quad \mathbf{\Phi}'^T] \begin{bmatrix} \mathbf{D}_{\mathbf{a}+\mathbf{a}^*}(\mathbf{y} - \mathbf{1}b) + (\mathbf{a} - \mathbf{a}^*)\varepsilon \\ \mathbf{D}_{\mathbf{s}+\mathbf{s}^*}\mathbf{y}' + (\mathbf{s} - \mathbf{s}^*)\varepsilon' \end{bmatrix}. \tag{24}$$

Now we pre-multiply both sides by the pseudo-inverse $[\mathbf{\Phi}^T \quad \mathbf{\Phi}'^T]^T$ and using the definition of $\mathbf{w}$ in (23), we obtain:

$$\left[\begin{bmatrix} \mathbf{D}_{\mathbf{a}+\mathbf{a}^*} & 0 \\ 0 & \mathbf{D}_{\mathbf{s}+\mathbf{s}^*} \end{bmatrix} \mathbf{H} + \mathbf{I} \right] \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{\gamma} \end{bmatrix} = \begin{bmatrix} \mathbf{D}_{\mathbf{a}+\mathbf{a}^*} & 0 \\ 0 & \mathbf{D}_{\mathbf{s}+\mathbf{s}^*} \end{bmatrix} \begin{bmatrix} \mathbf{y} - \mathbf{1}b + \frac{\mathbf{a}-\mathbf{a}^*}{\mathbf{a}+\mathbf{a}^*}\varepsilon \\ \mathbf{y}' + \frac{\mathbf{s}-\mathbf{s}^*}{\mathbf{s}+\mathbf{s}^*}\varepsilon' \end{bmatrix}, \tag{25}$$

where $\frac{\mathbf{a}-\mathbf{a}^*}{\mathbf{a}+\mathbf{a}^*}$ and $\frac{\mathbf{s}-\mathbf{s}^*}{\mathbf{s}+\mathbf{s}^*}$ denote, respectively, a column-vector containing $(a_i - a_i^*)/(a_i + a_i^*)$ and $(s_i - s_i^*)/(s_i + s_i^*)$ in the i^{th} row, and

$$\mathbf{H} = \begin{bmatrix} \mathbf{\Phi}\mathbf{\Phi}^T & \mathbf{\Phi}\mathbf{\Phi}'^T \\ \mathbf{\Phi}'\mathbf{\Phi}^T & \mathbf{\Phi}'\mathbf{\Phi}'^T \end{bmatrix} = \begin{bmatrix} \mathbf{K} & \mathbf{G} \\ \mathbf{K}' & \mathbf{J} \end{bmatrix}, \tag{26}$$

where $(\mathbf{K})_{ij} = K(x_i, x_j)$, $(\mathbf{K}')_{ij} = K'(x_i, x_j)$, $(\mathbf{G})_{ij} = G(x_i, x_j)$ and $(\mathbf{J})_{ij} = J(x_i, x_j)$. Multiplying by the inverse of the diagonal matrix and moving b back to the first term, the equation can be simplified to:

$$\begin{bmatrix} \mathbf{K}+\mathbf{D}_{\mathbf{a}+\mathbf{a}^*}^{-1} & \mathbf{G} & \mathbf{1} \\ \mathbf{K}' & \mathbf{J}+\mathbf{D}_{\mathbf{s}+\mathbf{s}^*}^{-1} & \mathbf{0} \\ \mathbf{1}^T & \mathbf{0}^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{\gamma} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{y}+\frac{\mathbf{a}-\mathbf{a}^*}{\mathbf{a}+\mathbf{a}^*}\varepsilon \\ \mathbf{y}'+\frac{\mathbf{s}-\mathbf{s}^*}{\mathbf{s}+\mathbf{s}^*}\varepsilon' \\ 0 \end{bmatrix}. \tag{27}$$

Finally, instead of using the last equation in (21), we have made use of a simpler constraint: $\sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0$, which is obtained equating $\frac{\partial L_P}{\partial b}$ to zero, and using the relationship between $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}^*$ introduced at the beginning of this subsection. We show an algorithmic implementation of the IRWLS procedure in Table 1.

We have completed the definition of the algorithm and we can now recover our argument that the procedure only needs to work with the samples that present a nonzero a_i or a_i^*. In this case the matrix $\mathbf{D}_{\mathbf{a}+\mathbf{a}^*}$ will be full-rank and very easily inverted as it is a diagonal matrix. The vector $\frac{\mathbf{a}-\mathbf{a}^*}{\mathbf{a}+\mathbf{a}^*}$ can be easily computed as it will contain either a 1 in the ith position, if a_i is nonzero, or a -1 in the ith position, if a_i^* is nonzero. Finally the matrix $\mathbf{D}_{\mathbf{a}+\mathbf{a}^*}^{-1}$ can be defined as:

$$\left(\mathbf{D}_{\mathbf{a}+\mathbf{a}^*}^{-1}\right)_{ii} = \begin{cases} 1/KC_1, & 0 \le e_i^k < 1/K \\ e_i^k/C_1, & e_i^k \ge 1/K \end{cases}$$

if a_i is nonzero (identical definition with e_i^*, if a_i^* is nonzero). We can see that as K tends to infinity $\left(\mathbf{D}_{\mathbf{a}+\mathbf{a}^*}^{-1}\right)_{ii} = e_i^k/C_1$ and we will not have any problems with the nonlinearity in the definition of $L(u)$ when solving the procedure with kernels. Finally, we can repeat this argument for s_i and s_i^* with identical results.

```
(1) Initialization:
    • Compute $\mathbf{H}$ (from $\mathbf{K}$, $\mathbf{K}'$, $\mathbf{G}$ and $\mathbf{J}$)
    • $a_i = C_1$, $s_i = C_2$ for odd $i$; $a_i^* = C_1$, $s_i^* = C_2$ for even $i$.
(2) Solve (27)
(3) Evaluate
    $\mathbf{e} = \mathbf{K}^T\boldsymbol{\beta} + \mathbf{K}'^T\boldsymbol{\gamma} + \mathbf{1}b - \mathbf{y} - \mathbf{1}\varepsilon$,    $\mathbf{e}^* = \mathbf{y} - \mathbf{K}^T\boldsymbol{\beta} - \mathbf{K}'^T\boldsymbol{\gamma} - \mathbf{1}b - \mathbf{1}\varepsilon$
    $\mathbf{d} = \mathbf{G}^T\boldsymbol{\beta} + \mathbf{J}^T\boldsymbol{\gamma} - \mathbf{y}' - \mathbf{1}\varepsilon'$,    $\mathbf{d}^* = \mathbf{y}' - \mathbf{G}^T\boldsymbol{\beta} - \mathbf{J}^T\boldsymbol{\gamma} - \mathbf{1}\varepsilon'$
(4) Recalculate $a_i$, $a_i^*$, $s_i^*$ and $s_i^*$.
(5) Go to step 2 until convergence is achieved.
```

Table 1

IRWLS algorithm pseudocode for a one-dimensional input space.

3 Extension to d-dimensional input spaces

The proposed method can be easily extended to d-dimensional input spaces and to consider up to k^{th} order derivatives following the simple idea of the proposed one-dimensional method. It is only necessary to incorporate the corresponding constraints. Because of the space limitation we have omitted the development, but in this case the solution takes the form

$$\mathbf{w} = \sum_{i=1}^{N}\sum_{j_1=0}^{k}\cdots\sum_{j_d=0}^{k}(\lambda^*_{ij_1\cdots j_d} - \lambda_{ij_1\cdots j_d})\frac{\partial^{(j_1+\cdots+j_d)}\phi(\mathbf{x}_i)}{\partial x_1^{j_1},\cdots,x_d^{j_d}}, \tag{28}$$

where $\mathbf{x}_i = [x_{i1}, x_{i2}, \cdots, x_{id}]^T$ and $\lambda^*_{ij_1\cdots j_d}$ and $\lambda_{ij_1\cdots j_d}$ are the Lagrange multipliers associated to the constraint in the i^{th} sample of

$$\frac{\partial^{(j_1+\cdots+j_d)} f(\mathbf{x})}{\partial x_1^{j_1}, \cdots, x_d^{j_d}}.$$

However, the number of constraints grows linearly with the input dimension and exponentially with the number of derivatives considered, which will make this formulation very difficult to solve for standard problems. Anyhow, this problem can be cast as a multidimensional regression estimation in which we will have a single constraint per sample, as in the regular SVM.

3.1 Multidimensional regression formulation

Without lack of generality, in the following we will formulate the problem of estimating $f(\mathbf{x})$ from its first order derivatives, given N data points $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ in a d dimensional space $\mathbf{x}_i \in \mathbb{R}^d$. The extension to include higher order derivatives or the function itself is straightforward, as we will show in Section 3.3. In this case, for each input vector a d-dimensional label vector $\mathbf{y}_i \in \mathbb{R}^d$ will be available, where

$$\mathbf{y}_i = \left[\left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_{i1}}\right|_{\mathbf{x}_i}, \left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_{i2}}\right|_{\mathbf{x}_i}, \ldots, \left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_{id}}\right|_{\mathbf{x}_i} \right] = \nabla_{\mathbf{x}} f(\mathbf{x}_i).$$

We define the estimated function $\hat{f}(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$, where $\mathbf{w}$ is a weight vector and $\boldsymbol{\phi}(\cdot)$ is a nonlinear transformation of the input vector $\mathbf{x}$ to a higher dimensional space (the feature space, $\boldsymbol{\phi}(\mathbf{x}) \in \mathcal{H}$). We need to solve a multidimensional regression problem for finding $\mathbf{w}$, in which we need to reduce the error between the derivatives of the estimated function and the $\mathbf{y}_i$ vector:

$$\mathbf{e}_i = \mathbf{y}_i - \nabla_{\mathbf{x}} \hat{f}(\mathbf{x}_i) = \left[y_{i1} - \mathbf{w}^T \boldsymbol{\phi}_1^{'}(\mathbf{x}_i), y_{i2} - \mathbf{w}^T \boldsymbol{\phi}_2^{'}(\mathbf{x}_i), \ldots, y_{id} - \mathbf{w}^T \boldsymbol{\phi}_d^{'}(\mathbf{x}_i) \right],$$

where we have defined $\boldsymbol{\phi}_j^{'}(\mathbf{x}_i) = \left. \frac{\partial \boldsymbol{\phi}(\mathbf{x})}{\partial \mathbf{x}_{ij}} \right|_{\mathbf{x}_i}$.

A Multidimensional Support Vector Regressor (M-SVR) has been recently proposed in [13]. This multidimensional problem needs to be modified to solve our particular function approximation problem. First, instead of having a vector $\boldsymbol{\phi}(\mathbf{x})$ and a matrix $\mathbf{W}$ to construct the error vector, we have a unique weight vector $\mathbf{w}$ and a matrix that contains the derivatives of $\boldsymbol{\phi}(\mathbf{x})$. The second modification slightly changes the quadratic cost function to make its derivative continuous to avoid numerical instabilities:

$$L(u) = \begin{cases} 0, & u < \varepsilon \\ u^2 - 2u\varepsilon + \varepsilon^2, & u \geq \varepsilon \end{cases}, \tag{29}$$

To summarize, the problem at hand is reduced to find the vector $\mathbf{w}$ that minimizes the following unconstrained functional

$$L_P(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} L(u_i), \tag{30}$$

where

$$u_i = \|\mathbf{e}_i\| = \sqrt{\mathbf{e}_i^T \mathbf{e}_i}, \quad \mathbf{e}_i = \mathbf{y}_i - \boldsymbol{\Phi}_i \mathbf{w}, \quad \boldsymbol{\Phi}_i = [\boldsymbol{\phi}_1^{'}(\mathbf{x}_i), \ldots, \boldsymbol{\phi}_d^{'}(\mathbf{x}_i)]^T.$$

3.2 Resolution of the Multidimensional Support Vector Regressor

To optimize the proposed multidimensional regression estimation problem, we are going to follow again an Iterative Re-Weighted Least Square (IRWLS) procedure.

To construct this procedure, we first obtain a first order Taylor expansion of $L(u)$ over the previous step solution u_i^k, leading to the minimization of

$$L_P^{'}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{N} L(u_i^k) + \left.\frac{dL(u)}{du}\right|_{u_i^k}[u_i - u_i^k]\right) \tag{31}$$

where $u_i^k = \|\mathbf{e}_i^k\|$ and $\mathbf{e}_i^k = \mathbf{y}_i - \mathbf{\Phi}_i\mathbf{w}^k$. Then, the following quadratic approximation is constructed:

$$\begin{aligned} L_P^{''}(\mathbf{w}) &= \frac{1}{2}\|\mathbf{w}\|^2 + C\left(\sum_{i=1}^{N} L(u_i^k) + \left.\frac{dL(u)}{du}\right|_{u_i^k}\frac{(u_i)^2-(u_i^k)^2}{2u_i^k}\right) \\ &= \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}\sum_{i=1}^{N} a_i(\mathbf{e}_i^T\mathbf{e}_i) + CT, \end{aligned} \tag{32}$$

where

$$a_i = \frac{C}{u_i^k}\left.\frac{dL(u)}{du}\right|_{u_i^k} = \begin{cases} 0, & u_i^k < \varepsilon \\ \frac{2C(u_i^k-\varepsilon)}{u_i^k}, & u_i^k \geq \varepsilon \end{cases}, \tag{33}$$

and CT comprises constant terms that do not depend on $\mathbf{w}$. This is a regularized weighted least square problem in which the weight a_i depends on the previous solution. This dependence implies to iterate the process until the fixed point solution is reached.

The functional $L_P^{''}(\mathbf{w})$ is a quadratic approximation to $L_P(\mathbf{w})$ in (30) that presents the same value $L_P^{''}(\mathbf{w}^k) = L_P(\mathbf{w}^k)$ and gradient $\nabla_{\mathbf{w}}L_P^{''}(\mathbf{w}^k) = \nabla_{\mathbf{w}}L_P(\mathbf{w}^k)$ for $\mathbf{w} = \mathbf{w}^k$. Therefore, we can define $\mathbf{p}^k = \mathbf{w}^s - \mathbf{w}^k$ as a descending direction for $L_P(\mathbf{w})$, where $\mathbf{w}^s$ is the least square solution to (32), and we can use it to construct a line search method [14], i.e. $\mathbf{w}^{k+1} = \mathbf{w}^k + \eta^k\mathbf{p}^k$. The value of η^k can be computed using a backtracking line search [14], in which η^k is initially set to 1 and if $L_P(\mathbf{w}^{k+1}) \geq L_p(\mathbf{w}^k)$, it is iteratively reduced until a strict decrease in the functional in (30) is observed.

To obtain $\mathbf{w}^s$, the solution to $L''_P(\mathbf{w})$ in (32), its gradient is equated to zero

$$\nabla_{\mathbf{w}} L''_P(\mathbf{w}) = \mathbf{w} - \sum_{i=1}^{N} \mathbf{\Phi}_i^T \mathbf{e}_i a_i = \mathbf{0},$$

equation that can be written down as:

$$\mathbf{w} + \sum_{i=1}^{N} \mathbf{\Phi}_i^T \mathbf{D}_{\mathbf{a}_i} \mathbf{\Phi}_i \mathbf{w} = \sum_{i=1}^{N} \mathbf{\Phi}_i^T \mathbf{D}_{\mathbf{a}_i} \mathbf{y}_i. \tag{34}$$

Here, $\mathbf{D}_{\mathbf{a}_i}$ denotes the $d \times d$ diagonal matrix with a_i as its diagonal elements, $(\mathbf{D}_{\mathbf{a}_i})_{lk} = a_i \delta(l-k)$ $(\mathbf{D}_{\mathbf{a}_i} = a_i \mathbf{I})$. In matrix notation (34) becomes:

$$[\mathbf{\Phi}^T \mathbf{D}_{\mathbf{a}} \mathbf{\Phi} + \mathbf{I}]\mathbf{w} = \mathbf{\Phi}^T \mathbf{D}_{\mathbf{a}} \mathbf{Y}. \tag{35}$$

where $\mathbf{\Phi} = [\mathbf{\Phi}_1^T, \ldots, \mathbf{\Phi}_N^T]^T$, $\mathbf{D}_{\mathbf{a}}$ is a $Nd \times Nd$ diagonal matrix where each $d \times d$ submatrix is defined as $(\mathbf{D}_{\mathbf{a}})_{ij} = \mathbf{D}_{\mathbf{a}_i} \delta(i-j)$ and $\mathbf{Y} = [\mathbf{y}_1^T, \ldots, \mathbf{y}_N^T]^T$ is a Nd-dimensional column vector.

The system in (35) can be solved using kernels. The Representer theorem [7] states that the optimal solution can be constructed as a linear combination of the training samples in the feature space, i.e. $\mathbf{w} = \mathbf{\Phi}^T \boldsymbol{\beta}$. By replacing this expression into (35) we obtain:

$$[\mathbf{\Phi}^T \mathbf{D}_{\mathbf{a}} \mathbf{\Phi} + \mathbf{I}]\mathbf{\Phi}^T \boldsymbol{\beta} = \mathbf{\Phi}^T \mathbf{D}_{\mathbf{a}} \mathbf{Y}. \tag{36}$$

Now, pre-multiplying (36) by $\mathbf{\Phi}$, we get to

$$[\mathbf{H} \mathbf{D}_{\mathbf{a}} \mathbf{H} + \mathbf{H}]\boldsymbol{\beta} = \mathbf{H} \mathbf{D}_{\mathbf{a}} \mathbf{Y}.$$

where $\mathbf{H} = \mathbf{\Phi}\mathbf{\Phi}^T$. Cancelling out $\mathbf{H}$ and pre-multiplying by the inverse of $\mathbf{D}_{\mathbf{a}}$, we arrive at:

$$[\mathbf{H} + \mathbf{D}_{\mathbf{a}}{}^{-1}]\boldsymbol{\beta} = \mathbf{Y}, \tag{37}$$

The IRWLS procedure for solving the multidimensional regression problem to find a function from its derivatives is summarized in Table 2.

```
(1) Initialization: set $\boldsymbol{\beta}^0 = \mathbf{0}$, $u_i = \|\mathbf{y}_i\|$ and compute $a_i$ from (33).
(2) Compute $\boldsymbol{\beta}^s = [\mathbf{H} + \mathbf{D_a}^{-1}]^{-1}\mathbf{Y}$ and set $\eta^k = 1$.
(3) Set $\boldsymbol{\beta}^{k+1} = \boldsymbol{\beta}^k + \eta^k[\boldsymbol{\beta}^s - \boldsymbol{\beta}^k]$ if $L(\boldsymbol{\beta}^{k+1}) < L(\boldsymbol{\beta}^k)$ go to Step 5.
(4) Set $\eta^k = \rho\eta^k$ with $0 < \rho < 1$ and go to Step 3.
(5) Recompute $u_i$ and $a_i$, set $k = k + 1$ and go to Step 2 until convergence.
```

Table 2

IRWLS algorithm pseudocode for multidimensional input spaces

3.3 Extensions

The extension of the proposed method to include samples from the function and from higher order derivatives is straightforward. In this case, the vectors $\mathbf{y}_i$ and $\mathbf{e}_i$ will be constructed with all the available information and the procedure in Sections 3.1 and 3.2 can be easily replicated. To illustrate this point, we propose the following example, where $\mathbf{y}_i$ and $\mathbf{e}_i$ are, respectively

$$\mathbf{y}_i = \left[f(\mathbf{x}_i), \left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_{i1}}\right|_{\mathbf{x}_i}, \left.\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}_{i2}}\right|_{\mathbf{x}_i}, \left.\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}_{i1}\partial \mathbf{x}_{i2}}\right|_{\mathbf{x}_i} \right],$$

$$\mathbf{e}_i \quad = \quad \left[y_{i1} - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i), y_{i1} - \mathbf{w}^T\boldsymbol{\phi}_1^{'}(\mathbf{x}_i), y_{i2} - \mathbf{w}^T\boldsymbol{\phi}_2^{'}(\mathbf{x}_i), y_{i3} - \mathbf{w}^T\boldsymbol{\phi}_{1,2}^{''}(\mathbf{x}_i) \right].$$

It must be noted that the one-dimensional resulting method is different from the one presented in Section 2.

Finally, when some data is more reliable or less noisy, or the range of the derivatives is clearly different, a weighted norm is more convenient for u_i, i.e. $u_i = \sqrt{\sum_{j=1}^{Q} c_j e_{ij}^2}$, where Q is the dimension of $\mathbf{y}_i$ and c_j are the corresponding weights with each dimension of $\mathbf{y}_i$. It is straightforward to find out that, in the algorithm, this just means to include the weights in the diagonal matrix $\mathbf{D_a}$ as $(\mathbf{D}_{\mathbf{a}_i})_{lk} = a_i \delta[l-k] c_k$.

4 Results

In this section, some experimental results show the advantages of this method in the reconstruction of the derivative with respect to the conventional SVR approach.

4.1 One-dimensional input space

As test functions, we have selected a set of band-limited functions: specifically, in each experiment a linear combination of 100 sinusoids with random amplitudes, frequencies (between 0 and 1 Hz) and phases has been generated. In the first example, 100 equally spaced sampling points in the range 0-5 have been employed by the SVR (100 samples of the function) and by the proposed method, labeled SVM-D (in this case 200 total samples: 100 samples of the function + 100 samples of the derivative). Moreover, we have tested the proposed method using the same number of total samples, which means to subsample (we will label this option by SVM-D^s). This method uses 50 sampling points (50 samples of the function + 50 samples of the derivative). In this way, the number of total available data is the same. 1000 independent experiments have been considered, with a signal to noise ratio (SNR) of 20 dB in the samples of both the function and the derivative. Parameters C and σ have been selected by cross-validation. Figure 1 plots the mean values of signal to error ratio (SER) in the reconstruction of the function (a) and of the derivative (b) as a function of the insensitivity parameter ε. In this case, $\varepsilon' = \pi\varepsilon$ has been considered to take into account the different amplitude range of function and derivative (the mean amplitude of the derivative is π times higher than the mean amplitude of the function).

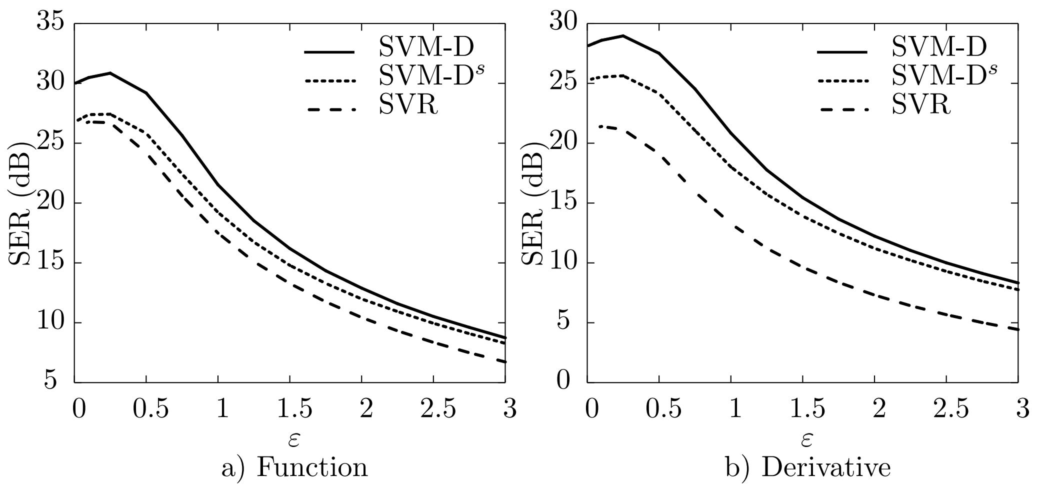

Fig. 1. SER in the reconstruction of function and derivative.

It can be seen that the proposed method, using twice data (SVM-D), but more interestingly, even using the same amount of data (SVM-D^s), provides better results than the SVR, specially in the reconstruction of the derivative. Obviously, using twice data, the improvement is bigger.

Moreover, when the same amount of data is used (SVM-D^s), a similar number of support vectors has been observed for both methods in all simulations (in the proposed method: support vectors related to the function + support vectors related to the derivative). Therefore, the proposed method, when using the same amount of data, does not increment the storage requirements of the model. The number of support vectors, as a function of ε, is plotted in Figure 2.

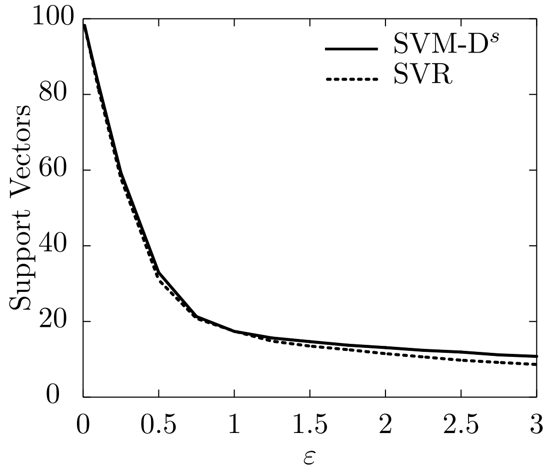

Fig. 2. Number ot total support vectors as a function of ε.

This method also reduces the sensitivity to the selection of σ for the Gaussian kernels. Figure 3 shows the SER in the reconstruction of the function as a function of σ for $\varepsilon = 0.5$. It can be seen that the σ range to obtain high SER values is increased by using the proposed method.

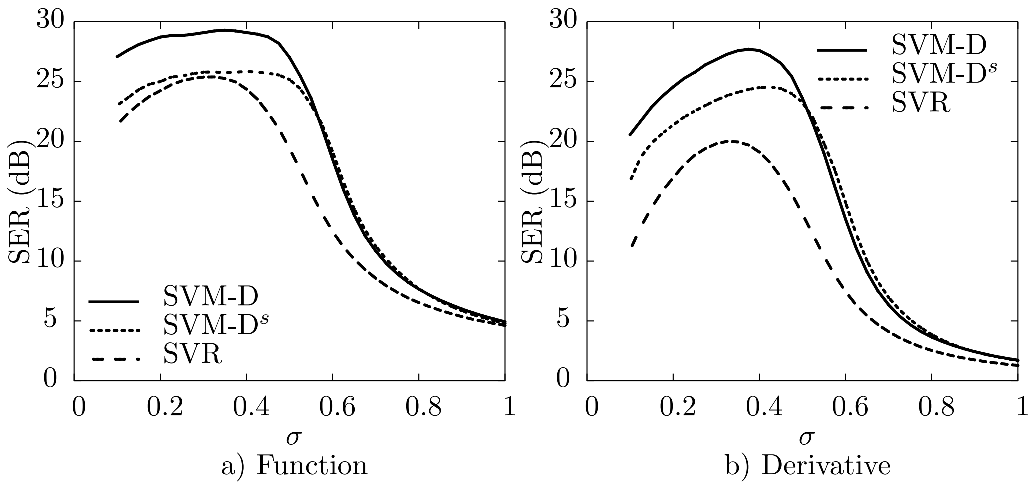

Fig. 3. SER as a function of the Kernel size.

Finally, in all the above experiments we have used $\varepsilon' = \pi\varepsilon$ to consider the amplitude of the function/amplitude of the derivative ratio (which can easily be estimated from data samples). Figure 4 plots the results in the reconstruction of function and derivative, using $\sigma = 0.4$ and $\varepsilon = 0.5$, as a function of ε', using 50 sampling points. It can be seen that the optimal value for parameter

ε' is around $\pi\varepsilon$ ($\varepsilon' \approx \pi/2$ in this case).

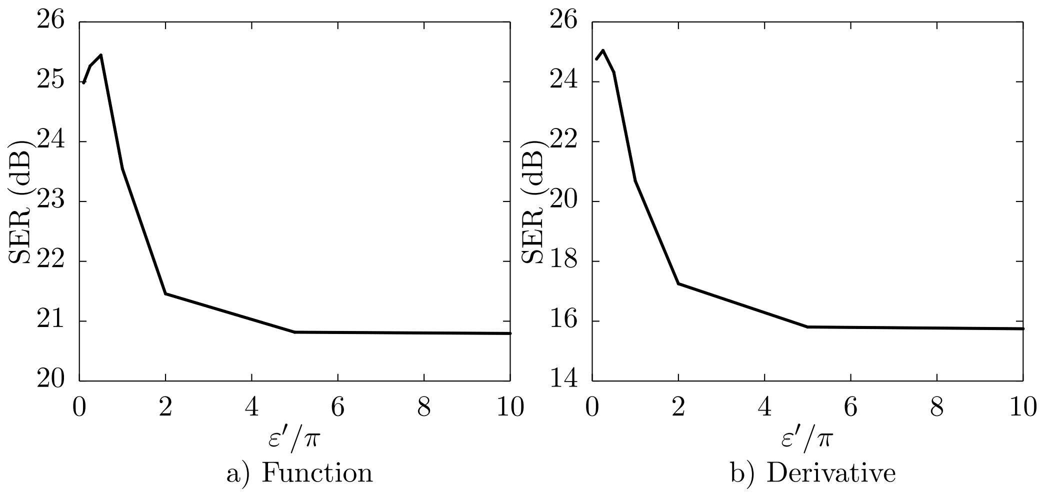

Fig. 4. SER as a function of ε' using 50 sampling points and $\varepsilon = 0.5$ and $\sigma = 0.4$.

This simple approach of selecting ε' to consider the relative function/derivative amplitude has shown good results when samples of function and derivative have a similar SNR. For noisier derivative samples (if this information is available or can be estimated), this parameter has to be increased proportionally to the ratio between SNR's.

4.2 Two-dimensional input space

The performance of the proposed multidimensional method has been tested using 8 functions, proposed in [15], sampled from a two-dimensional input space. The analytical expression of each function is provided in Table 3.

We want to show the benefits of using the samples of the derivatives in the reconstruction of the derivatives themselves as well as in the reconstruction of the function. On the one hand, we will show the advantage of adding samples of the derivatives to the set of samples of the function. On the other hand,

NAME	FUNCTION	DOMAIN
Fun 1	$y = \sin(x_1 x_2)$	[-2,2]
Fun 2	$y = \exp(x_1 \sin(\pi x_2))$	[-1,1]
Fun 3	$y = \frac{40{*}\exp(8((x_1-0.5)^2+(x_2-0.5)^2))}{\exp(8((x_1-0.2)^2+(x_2-0.7)^2))+\exp(8((x_1-0.7^2+(x_2-0.2)^2))}$	[0,1]
Fun 4	$y = (1+\sin(2x_1+3x_2))/(3.5+\sin(x_1-x_2))$	[-2,2]
Fun 5	$y = 42.659(0.1+x_1(0.05+x_1^4-10x_1^2x_2^2+5x_2^4))$	[-0.5,0.5]
Fun 6	$y = 1.3356\big[\exp(3(x_2-0.5))\sin(4\pi(x_2-0.9)^2)$ $+1.5(1-x_1)+\exp(2x_1-1)\sin(3\pi(x_1-0.6)^2)\big]$	[0,1]
Fun 7	$y = 1.9[1.35+\exp(x_1)\sin(13(x_1-0.6)^2)$ $+\exp(3(x_2-0.5))\sin(4\pi(x_2-0.9)^2)]$	[0,1]
Fun 8	$y = \sin(2\pi\sqrt{x_1^2+x_2^2})$	[-1,1]

Table 3
Two-dimensional Test Functions ($y = f(x_1, x_2)$).

the advantage of replacing some of the samples of the function by samples of the derivatives, which is more interesting, will also be showed.

The following methods will be compared. The conventional SVR will be used when only the samples of the function are used. When the samples of the function and the first order derivatives are jointly used, the proposed method (labeled “M-SVR” in the following) is employed. Finally, when only the samples of the two first order derivatives are used, again the proposed method (labeled “M-SVRd” in this case) is applied. In all cases, Gaussian kernels are employed. The Signal to Error Ratio (SER), expressed in dB, between the true function/derivatives and its corresponding reconstruction has been used as figure of merit.

It must be noted that using the same sampling points the M-SVRd is using twice as many samples as SVR and the M-SVR three times more samples than SVR, and this can be helpful to improve the results in a noisy environment. Therefore, we want also to compare results when the three methods use a similar amount of data, i.e, samples of the derivatives are replacing to (instead of being added to) samples of the function. Conventional SVR, as well as M-SVR and M-SVRd have been trained using a uniform grid of 19×19 sampling points (361 samples for SVR, 722 samples for M-SVRd and 1083 samples for M-SVR). Moreover, M-SVRd has been trained with 13×14 sampling points (this option, labeled M-SVRd^s, uses 363 samples), and M-SVR with 11×11 sampling points (this option, labeled M-SVR^s, uses 364 samples). Noisy samples with SNR=10 dB are considered. The parameters of the algorithms (C, σ and ε) have been selected by cross-validation, and a weighted norm has been used for the methods using the derivatives. The weights have been selected to compensate the different variances of function and derivatives (which are estimated from the samples). Table 4 compares the poerformance of all methods.

Approximation of the function

	Fun 1	Fun 2	Fun 3	Fun 4	Fun 5	Fun 6	Fun 7	Fun 8	**Mean**
SVR	19.1	27.0	24.0	21.9	29.3	25.6	26.8	16.3	**23.8**
M-SVRd	23.6	31.8	30.5	23.2	31.7	29.2	29.9	18.3	**27.3**
M-SVR	26.2	34.8	33.0	27.7	35.3	32.1	33.3	21.9	**30.5**
M-SVRd^s	21.0	29.0	27.1	20.1	27.7	26.2	27.2	16.4	**24.3**
M-SVR^s	21.5	29.9	27.7	23.1	29.6	27.3	28.5	16.7	**25.5**

Approximation of the first order derivatives (mean value)

	Fun 1	Fun 2	Fun 3	Fun 4	Fun 5	Fun 6	Fun 7	Fun 8	**Mean**
SVR	12.3	11.3	11.2	14.9	12.1	12.2	11.4	11.6	**12.1**
M-SVRd	20.5	20.6	21.3	20.6	19.2	19.9	18.8	16.4	**19.7**
M-SVR	21.1	21.0	21.7	22.1	20.3	20.6	19.5	17.1	**20.4**
M-SVRd^s	17.8	18.0	18.7	17.9	16.1	17.3	16.3	14.6	**17.1**
M-SVR^s	17.0	17.0	17.9	18.2	15.6	16.5	15.5	13.3	**16.4**

Table 4
SER (dB) for the reconstruction of the function and first order derivatives.

The methods including the samples of the derivatives out-perform the SVR in the reconstruction of the derivatives and also in the reconstruction of the function. The best performance is obtained by M-SVR, which outperforms SVR in almost 7 dB in the reconstruction of the function and in more than 8 dB in the reconstruction of the derivatives. The advantage is reasonable if it is taken into account that M-SVR is the method using the biggest amount of data, and that also M-SVRd uses more data than SVR. However, the advantage holds when a similar number of total samples is used by all method. In this case, the M-SVRd^s provides better results in the reconstruction of the derivatives while the M-SVR^s is better for the reconstruction of the function.

In the previous experiment, the samples of function and derivatives have the same SNR. We also want to show that the benefit of including the samples of the derivatives holds when they are noisier than the samples of the function, which is the usual in a real application. Figure 5 compares the performance of SVR and M-SVR^s (in the same sampling conditions than the previous experiment) as a function of the increasing of SNR (dB) in the samples of the derivatives with respect to the SNR in the samples of the function. This example corresponds to the reconstruction of Function 1 with SNR = 10 dB.

The reconstruction of the function is improved even when the samples of the derivatives are around 4 dB noisier than the samples of the function. For the derivatives, even a higher margin (around 6 dB) is obtained. Moreover, if the samples of the derivatives are more accurate than the samples of the function (although this is not the usual in real applications) the reconstruction is clearly improved. Similar results have been obtained for all functions. Table 5 shows the margin of dB in the SNR in the samples of the derivative to improve in the reconstruction of function and derivatives.

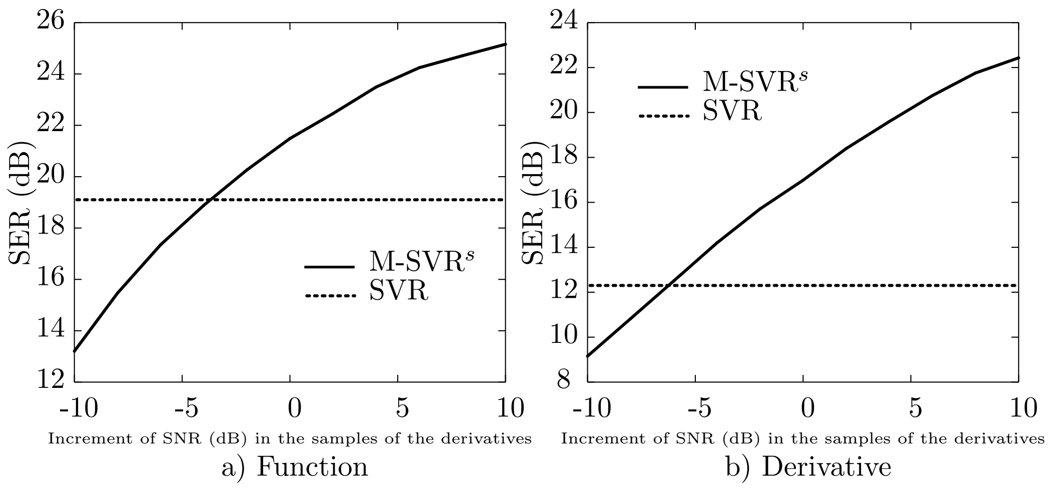

Fig. 5. SER (dB) in the reconstruction of Function 1 and its derivatives as a function of the SNR in the samples of the derivatives for a SNR=10 dB in the samples of the function.

	Fun 1	Fun 2	Fun 3	Fun 4	Fun 5	Fun 6	Fun 7	Fun 8	**Mean**
Function	3.9	4.3	3.4	2.3	0.7	2.9	3.3	1	**2.7**
Derivatives	6.4	7.4	7.7	4.4	6	6.5	7	4	**6.2**

Table 5
Margin (dB) in the SNR of the samples of the derivatives to improve the reconstruction of the function and of the derivatives.

5 Conclusions

Two variants of a new SVM-based method for the simultaneous reconstruction of a function and its derivatives have been presented: one for a one-dimensional input space and the other for a multidimensional input space. Computationally efficient IRWLS algorithms have been derived to allow the application of both variants to large data sets. This method provides better results than the conventional SVR approach in the reconstruction of function and derivatives even when the same number of labeled data is employed in both methods, or when the samples of the derivatives are noisier than the samples of the function. The proposed method needs a similar number of support vectors than conventional SVR. Moreover, the inclusion of the information of the derivatives reduces the dependence on the kernel size for Gaussian kernels.

It is necessary to mention that the multidimensional model is also valid for one-dimensional input spaces. However, it provides a slightly different solution that the one-dimensional proposed method since the loss function is different: linear versus quadratic. Anyway, the performance and accuracy is very similar for both methods in a one-dimensional input space.

Results obtained using this method show that the introduction of information about the derivatives is mandatory to obtain an accurate estimate of the derivatives of the function, which is necessary in a number of applications. Moreover, even when the approximation of the derivatives is not mandatory, this information can be useful in the reconstruction of the function without having to increase the total number of data and even when the samples of the derivatives are slightly noisier than the samples of the function. This can be clearly useful in applications where the available sampling rate is limited.

Finally, the method holds one of the limitations of traditional kernel method, since the computational burden allows its implementation only for moderate size problems.

References

[1] K. Hornik, M. Stinchcombe, H. White, Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks, Neural Networks 3 (1990) 551–560.

[2] A. R. Gallant, H. White, On learning the derivatives of an unknown mapping with multilayer feedforward networks, Neural Networks 5 (1992) 129–138.

[3] X. Li, Simultaneous approximations of multivariate functions and their derivatives by neural networks with one hidden layer, Neurocomputing 12 (1996) 327–343.

[4] T. Nguyen - Thien, T. Tran-Cong, Approximation of functions and their derivatives: A neural network implementation with applications, Neurocomputing 23 (1999) 687–704.

[5] M. Lázaro, I. Santamaría, C. Pantaleón, J. Ibáñez, L. Vielva, A regularized technique for the simultaneous reconstruction of a function and its derivatives with application to nonlinear transistor modeling, Signal Processing (83) (2003) 1859–1870.

[6] V. N. Vapnik, Statistical Learning Theory, John Wiley & Sons, New York, 1998.

[7] B. Schölkopf, A. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002.

[8] F. Pérez-Cruz, A. Navia-Vázquez, J. L. Rojo-Álvarez, A. Artés-Rodríguez, A new training algorithm for support vector machines, in: Proceedings of the Fifth Bayona Workshop on Emerging Technologies in Telecommunications, Baiona, Spain, 1999, pp. 116–120.

[9] F. Pérez-Cruz, A. Navia-Vázquez, P. Alarcón-Diana, A. Artés-Rodríguez, An IRWLS procedure for SVR, in: Proceedings of the EUSIPCO'00, Tampere, Finland, 2000.

[10] F. Pérez-Cruz, C. Bousoño-Calzón, A. Artés-Rodríguez, Convergence of the IRWLS procedure to the support vector machine solution, Neural Computation (January, 2005).

[11] M. Lázaro, I. Santamaría, F. Pérez-Cruz, A. Artés-Rodríguez, SVM for the simultaneous approximation of a function and its derivative, in: Proceedings of the 2003 IEEE International Workshop on Neural Networks for Signal Processing (NNSP), Toulouse, France, 2003.

[12] J. Suykens, J. De Brabanter, L. Lukas, J. Wandewalle, Weighted least squares support vector machines: robustness and sparse approximation, Neurocomputing - Special issue on fundamental and information processing aspects of neurocomputing 48 (2002) 85–105.

[13] F. Pérez-Cruz, G. Camps, E. Soria, J. Pérez, A. R. Figueiras-Vidal, A. Artés-Rodríguez, Multi-dimensional function approximation and regression estimation, in: ICANN02, Springer, Madrid, Spain, 2002.

[14] J. Nocedal, S. J. Wright, Numerical Optimization, Springer, 1999.

[15] V. Cherkassky, D. Gehring, F. Mulier, Comparison of adaptive methods for function estimation from samples, IEEE Transactions on Neural Networks 7 (4) (1996) 969–984.