Incorporating Prior Knowledge in Support Vector Regression

Fabien Lauer, Gérard Bloch

Machine Learning manuscript No.
(will be inserted by the editor)

Incorporating Prior Knowledge in Support Vector Regression

Fabien Lauer and Gérard Bloch

Centre de Recherche en Automatique de Nancy (CRAN), Nancy-University, CNRS, CRAN-ESSTIN, rue Jean Lamour, 54519 Vandœuvre Cedex, France
e-mail: {fabien.lauer,gerard.bloch}@esstin.uhp-nancy.fr

Received: 12 June 2006 / Revised version: 9 May 2007 / Accepted: 9 October 2007

Abstract This paper explores the addition of constraints to the linear programming formulation of the support vector regression problem for the incorporation of prior knowledge. Equality and inequality constraints are studied with the corresponding types of prior knowledge that can be considered for the method. These include particular points with known values, prior knowledge on any derivative of the function either provided by a prior model or available only at some specific points and bounds on the function or any derivative in a given domain. Moreover, a new method for the simultaneous approximation of multiple outputs linked by some prior knowledge is proposed. This method also allows consideration of different types of prior knowledge on single outputs while training on multiple outputs. Synthetic examples show that incorporating a wide variety of prior knowledge becomes easy, as it leads to linear programs, and helps to improve the approximation in difficult cases. The benefits of the method are finally shown on a real-life application, the estimation of in-cylinder residual gas fraction in spark ignition engines, which is representative of numerous situations met in engineering.

Key words Support Vector Regression – kernel approximation – prior knowledge – multi-outputs

1 Introduction

In non-linear function approximation, Support Vector Regression (SVR) has proved to be able to give excellent performances in various applications [Müller et al., 1997, Stitson et al., 1999, Mattera and Haykin, 1999]. Based

Correspondence to: fabien.lauer@esstin.uhp-nancy.fr

on the statistical learning theory [Vapnik, 1995], SVR originally leads to a quadratic programming (QP) problem [Cristianini and Shawe-Taylor, 2000, Smola and Schölkopf, 2004]. Other formulations of the SVR problem minimizing the ℓ_1-norm of the parameters can be derived to yield linear programs (LP) [Weston et al., 1999, Bennett, 1999, Smola et al., 1999b, Mangasarian and Musicant, 2002]. Some advantages of this latter approach can be noticed compared to the QP formulation such as the sparsity of support vectors [Weston et al., 1999, Bennett, 1999, Smola et al., 1999b] or the ability to use more general kernels [Mangasarian, 2000].

Support Vector Regression aims at learning an unknown function based only on a training set of N input-output pairs $(\boldsymbol{x}_i, y_i)$ in a black box modelling approach. Nonetheless, in real world applications, such as system identification, some information is usually known beforehand. This prior knowledge can take many forms from the positiveness of a physical variable to the shape of the function on a particular region. Incorporating this knowledge into the learning scheme can improve the quality of the model in different ways. The function can be approximated in regions of the input space where the data are sparse, specific properties of the function such as a maximum or a curvature at some point can be introduced in the model, and so on. However, incorporating prior knowledge into support vector learning is not trivial and is still a partially open issue. The methods developed for neural networks and that use prior knowledge for the network initialization [Andrews and Geva, 1999] or structure selection (KBANN) [Towell and Shavlik, 1994] cannot be directly transferred to SVMs since these tasks are typically handled by support vector learning. The smoothness assumption is also another simple form of prior knowledge implicitly included in support vector machines (SVM) [Smola et al., 1998, Evgeniou et al., 2000]. On the other hand, a certain amount of work has been done in the past decade to build support vector machine classifiers or kernels, whose outputs are invariant under a known transformation of the input (see [Lauer and Bloch, 2007] for a recent overview). However, such type of prior knowledge is rarely available in the regression framework.

The present paper explores the addition of equality or inequality constraints of a general form, linear in the parameters, to the LP-SVR problem and exposes the different types of prior knowledge that can be included in the learning with this technique. Interesting issues are considered such as: knowledge in a region of the input space without data, knowledge on any derivative provided either by a prior model or only at particular points and prior knowledge between multiple outputs.

The first part of the paper presents the related work and discusses the differences and advantages of the proposed method. The linear programming formulation of the SVR problem is then recalled in section 3 before exposing the proposed method. In particular, section 4 is dedicated to the introduction of equality constraints to the optimization problem, while section 5 considers prior knowledge in the form of inequalities. Practical examples on synthetic data are given throughout the paper to show the interest and effi-

ciency of the methods, while section 6 presents their application to real-life problems.

Notations: all vectors are column vectors written in boldface and lowercase letters whereas matrices are boldface and uppercase, except for the ith column of a matrix $\boldsymbol{A}$ that is denoted $\boldsymbol{A}_i$. The vectors $\mathbf{0}$ and $\mathbf{1}$ are vectors of appropriate dimensions with all their components respectively equal to 0 and 1. For $\boldsymbol{A} \in \mathbb{R}^{d\times m}$ and $\boldsymbol{B} \in \mathbb{R}^{d\times n}$ containing d-dimensional sample vectors, the "kernel" $\boldsymbol{K}(\boldsymbol{A}, \boldsymbol{B})$ maps $\mathbb{R}^{d\times m} \times \mathbb{R}^{d\times n}$ in $\mathbb{R}^{m\times n}$ with $\boldsymbol{K}(\boldsymbol{A}, \boldsymbol{B})_{i,j} = k(\boldsymbol{A}_i, \boldsymbol{B}_j)$, where $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is the kernel function. In particular, if $\boldsymbol{x} \in \mathbb{R}^d$ is a column vector then $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{B})$ is a row vector in $\mathbb{R}^{1\times n}$. The matrix $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ contains all the training samples $\boldsymbol{x}_i$, $i = 1, \ldots, N$, as rows. The vector $\boldsymbol{y} \in \mathbb{R}^N$ gathers all the target values y_i for these samples. The kernel matrix $\boldsymbol{K}(\boldsymbol{X}^T, \boldsymbol{X}^T)$ will be written $\boldsymbol{K}$ for short. Uppercase Z is a set containing $|Z|$ vectors that constitute the rows of the matrix $\boldsymbol{Z}$. The function sinc is defined as $\mathrm{sinc}(x) = \sin(\pi x)/(\pi x)$, for all $x \neq 0$ and $\mathrm{sinc}(0) = 1$.

2 Related work

Classification. A certain amount of work has been done in the past decade to build SVM classifiers or kernels, whose outputs are invariant under a known transformation of the input. Such invariances can be, for instance, incorporated by creating new samples to enlarge the training set. This idea was first introduced in [Poggio and Vetter, 1992] as virtual samples. It is based on a simple observation that the generalization ability of the obtained model depends on the number of data at hand. The more relevant samples we have, the better we learn. This method can be easily implemented in the context of pattern recognition. For instance, in image recognition, invariance of the output to a translation or rotation of the input image is often considered. In [Schölkopf et al., 1996], the virtual sample approach was combined with SVMs. The idea was to generate the virtual samples from support vectors only since they contain all the information about the problem. The virtual sample method can also directly be applied to regression, but, in this case, prior knowledge considering invariance of the output is not easily available (except for a few special cases such as symmetric or periodic functions). Nonetheless the idea of virtual samples can be modified to be used for regression when prior knowledge on the function is given in terms of output values or derivatives instead of invariance as in the proposed method of section 4.1.

Another simple and straightforward way to incorporate prior knowledge is to weight the samples. In practice, this amounts to weight the errors or to choose a different trade-off parameter C for different samples in the criterion to minimize. Originally, for pattern recognition, different misclassification costs C_i were introduced in [Joachims, 2002] to deal with unbalanced data, i.e. providing much more samples of a class compared to the other. Another

approach for the weighting of SVM can be found in [Wu and Srihari, 2004] where a confidence value is assigned to every sample based on some knowledge of the data acquisition or labeling procedure. The application to regression is straightforward. In [Tay and Cao, 2002], the weighting of samples is applied to non-stationary time-series forecasting where it was noticed that the distant past data were less significant than the recent past data. To include this forgetting effect, the weights C_i applied on the errors are given by an increasing function of time.

Constrained regression. The topic of linearly constrained least squares regression has a long history and is involved in a variety of applications, both with equality constraints or inequality constraints. Incorporating linear equality constraints is useful whenever one or several coefficients must be expressed as a linear combination of the others. This is the case for instance when the intercept is known to be zero, when the sum of the coefficients must be equal to 1 or when the steady state gain is known in the dynamical system identification framework [Söderström and Stoica, 1988]. Inequality constraints may arise from requirements such as positivity, monotonicity, and convexity [Lawson and Hanson, 1995]. Many points presented here are close in spirit to these approaches, in the sense that prior knowledge is also incorporated by the addition of constraints. However, even if the least squares formulation can be extended to be applied to non-linear models by considering expansions of basis functions [Johansen, 1996], it does not handle non-quadratic error criteria nor select the regressors or basis functions as done in the SVM framework.

Spline models. In function modeling by fitting separate models in different regions of the input space, the introduction of constraints to take into account continuity knowledge at the boundary points, denoted knots, is the basic idea in splines models [Hastie et al., 2001]. Moreover, in smoothing splines, this idea is extended for avoiding to select the knots by using a maximal set of knots and controlling the resulting function smoothness by regularization. Thus some of the ideas presented here have strong links with splines models. In particular, section 4.4.2 focuses on building multi-models for continuous function approximation. However, it presents in a general way the nature of the submodels and extends the fitting criterion to the ℓ_1-norm. Beside this, a certain amount of work has been done to incorporate more prior knowledge in the form of linear inequalities into spline models, see for instance [Villalobos and Wahba, 1987] and [Micchelli and Utreras, 1988].

Semiparametric modeling. In [Smola et al., 1999a], prior knowledge is incorporated by moving from nonparametric to semiparametric modeling. In this scheme, the bias term of the the kernel expansion is replaced by a parametric model whose parameters are determined by the optimization procedure. Semiparametric modeling can be included either in the quadratic or linear programming form of SVR.

Knowledge in a region of input space. In [Fung et al., 2002], the authors use a different approach and introduce prior knowledge on polyhedral regions in the context of linear programming SVM for linear classification. It was then extended to the non-linear case in [Fung et al., 2003] via a reformulation of the kernel. In [Mangasarian et al., 2004], it was finally adapted for regression and later on developed to apply non-linear bounds on the model in any non-linear region [Mangasarian and Wild, 2007]. An advantage of this method is that the problem remains in a linear programming (LP) form. However, whenever the non-linear regions are not given as explicit sets of a finite number of points, these regions have to be discretized before including the prior knowledge in the learning as a finite set of inequalities. Though also requiring the knowledge to be in the form of a discrete set of constraints, the method proposed here is more simple and involves less variables in the optimization program. The increase of complexity for Mangasarian's method is justified in the case of polyhedral regions, for which a finite set of constraints ensures that the prior knowledge is satisfied for all points in these regions. But when discretization occurs, as often required for non-linear regions, this is no more true since adding constraints for the points of discretization only cannot guarantee that the non-linear bound on the function holds in the whole non-linear region. Thus, in this case, it is sufficient to consider simpler constraints as will be described in this paper. The method proposed here is also more general in the sense that it allows for the inclusion of knowledge on the derivatives of the model. Though Mangasarian's method could be extended to include prior knowledge on derivatives of the function, it has not been done so far.

Knowledge on the derivatives. In [Lázaro et al., 2005b], prior knowledge on the derivatives has been incorporated in the QP formulation of the SVR problem by considering a training set containing at every point the target values, not only for the function but also for the derivative. The optimization problem is extended to include the minimization of the error on the derivative and thus simultaneously approximate the function and its derivative. As a result, the derivative of the kernel function is involved in the approximation function, which may slow down the estimations in the test phase. However, another formulation [Lázaro et al., 2005a] has been proposed to circumvent this drawback. Whereas these methods consider knowledge on the derivatives at the training points only, the method proposed here allows to impose derivative values for any point. This is of particular interest when incorporating prior knowledge in regions not covered by the training data.

Support vector regression with multiple outputs. The issue of multiple outputs regression has been previously studied for SVR in [Sánchez-Fernández et al., 2004]. In this approach, the multiple ε-insensitive loss functions are replaced by a ℓ_2-norm based loss function, resulting in a single scalar error value taking into account the errors on all the outputs. Though providing a mean for the regression of multiple outputs with depen-

dencies, the method does not allow for the inclusion of prior knowledge on these dependencies. Another approach, proposed in [Weston et al., 2003], allows to incorporate prior knowledge between multiple outputs. In this approach, the loss function is considered as a distance in an output feature space and thus computed by a kernel function defined over the output space. Prior information on the outputs such as specific loss functions or invariances can be embedded in this output kernel. Being a very general framework, applying to regression as well as classification, this method also requires more complex steps. These include the decomposition of the outputs $\boldsymbol{\Phi}_{out}(\boldsymbol{y}_i)$ in the output feature space by Kernel Principal Component Analysis (KPCA), learning multiple mappings from the input to the resulting principal components and, finally, solving the pre-image problem for every new test sample, i.e. finding the pre-image $\boldsymbol{y}_t$ of the output $\boldsymbol{\Phi}_{out}(\boldsymbol{y}_t)$ estimated in feature space. In particular, the pre-image problem is still a partially open issue. The method proposed in section 4.4 allows for a simple formulation of the dependencies between multiple outputs and their inclusion as a finite set of constraints that leaves the nature of the learning problem unchanged. In [Maclin et al., 2005], the method of [Mangasarian et al., 2004] for knowledge-based kernel approximation is extended to provide advice to a reinforcement learner. In this setting, the value of each output determines if a specific action must be taken. Prior knowledge may indicate that under a set of conditions, one action is preferable to another. This amounts to consider the inequality: $f_1(\boldsymbol{x}) \geq f_2(\boldsymbol{x}) + \beta$, for $\boldsymbol{x}$ satisfying a set of conditions, where the output f_1 represents the preferred action to the one represented by f_2 and β accounts for how much preferable it is. The general framework proposed in section 5.2 includes this form of prior knowledge as a particular case.

Conclusion. All the related approaches, that allow the use of prior knowledge for SVR, focus on particular types of prior knowledge. The strength of the proposed method thus lies in its generality, but also in its simplicity as it amounts to the addition of linear constraints to the problem. Incorporating prior knowledge by the addition of constraints has been extensively studied in other areas such as smoothing splines or least squares regression. The present paper proposes to transfer these techniques to the LP-SVR framework and explores all the consequences of this transfer, from the inclusion of basic forms of prior knowledge to the handling of constrained multi-output regression.

3 Kernel approximation of functions

In kernel regression, the function of input $\boldsymbol{x} \in \mathbb{R}^d$ is approximated by a kernel expansion

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b = \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}^T)\boldsymbol{\alpha} + b \ , \tag{1}$$

where the α_i, contained in the vector $\boldsymbol{\alpha}$, and b are the parameters of the model and $k(.,.)$ is the kernel function. Typical kernel functions are the linear, Gaussian RBF, polynomial and sigmoidal kernels. In all the numerical examples of this paper, a Gaussian RBF kernel $k(\boldsymbol{x}, \boldsymbol{x}_i) = \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}_i\|^2/2\sigma^2\right)$ is used, except when mentioned otherwise.

3.1 Linear programming (LP)

In kernel regression via linear programming (LP), the ℓ_1-norm of the parameters $\boldsymbol{\alpha}$ of the kernel expansion is minimized together with the ℓ_1-norm of the errors by

$$\min_{(\boldsymbol{\alpha},b)} \|\boldsymbol{\alpha}\|_1 + C\sum_{i=1}^{N} |f(\boldsymbol{x}_i) - y_i| \ , \tag{2}$$

where a hyperparameter C is introduced to tune the trade-off between the error minimization and the flatness maximization. This problem can be implemented as the linear program

$$\begin{aligned} \min_{(\boldsymbol{\alpha},b,\boldsymbol{\xi},\boldsymbol{a})} \quad & \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} \\ \text{s.t.} \quad & -\boldsymbol{\xi} \leq \boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} \leq \boldsymbol{\xi} \\ & -\boldsymbol{a} \leq \quad \boldsymbol{\alpha} \quad \leq \boldsymbol{a} \ , \end{aligned} \tag{3}$$

where $\boldsymbol{\xi}$ and $\boldsymbol{a}$ are vectors of N positive variables. Instead of the ℓ_1-norm of the errors, the ε-insensitive loss function defined as

$$|\xi|_\varepsilon = \begin{cases} 0 & \text{if } |\xi| \leq \varepsilon \ , \\ |\xi| - \varepsilon & \text{otherwise,} \end{cases} \tag{4}$$

can also be used. A possible formulation of the corresponding problem involves $4N + 1$ variables [Smola et al., 1999b]. In this paper, we will follow the approach of [Mangasarian and Musicant, 2002] that involves only $3N+1$ variables. In this scheme, the optimization problem becomes in matrix form

$$\begin{aligned} \min_{(\boldsymbol{\alpha},b,\boldsymbol{\xi},\boldsymbol{a})} \quad & \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} \\ \text{s.t.} \quad & -\boldsymbol{\xi} \leq \boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} \leq \boldsymbol{\xi} \\ & 0 \leq \quad \mathbf{1}\varepsilon \quad \leq \boldsymbol{\xi} \\ & -\boldsymbol{a} \leq \quad \boldsymbol{\alpha} \quad \leq \boldsymbol{a} \ . \end{aligned} \tag{5}$$

The parameter ε can be introduced as a variable in the cost function to be tuned automatically by the algorithm

$$\begin{aligned} \min_{(\boldsymbol{\alpha},b,\boldsymbol{\xi},\boldsymbol{a},\varepsilon)} \quad & \tfrac{1}{N}\mathbf{1}^T\boldsymbol{a} + \tfrac{C}{N}\mathbf{1}^T\boldsymbol{\xi} - C\mu\varepsilon \\ \text{s.t.} \quad & -\boldsymbol{\xi} \leq \quad \boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} \quad \leq \boldsymbol{\xi} \\ & 0 \leq \quad \mathbf{1}\varepsilon \quad \leq \boldsymbol{\xi} \\ & -\boldsymbol{a} \leq \quad \boldsymbol{\alpha} \quad \leq \boldsymbol{a} \ . \end{aligned} \tag{6}$$

It can be shown [Mangasarian and Musicant, 2002] that this formulation is equivalent to the ν-LPR given in [Smola et al., 1999b].

In the LP formulation, only symmetry of the kernel is required [Mangasarian and Musicant, 2002]. It is not necessary for the kernel to satisfy Mercer's conditions (or positive semidefiniteness) as in the original QP form of the SVR problem where a norm induced by the kernel in the feature space is used over the weights. Here the norm of the parameters to be minimized is simply the ℓ_1-norm in $\mathbb{R}^N$.

3.2 Linearity of the derivatives of a kernel expansion

Noticing that the kernel expansion (1) is linear in the parameters $\boldsymbol{\alpha}$ allows to write the derivative of the model output with respect to the jth component x^j of $\boldsymbol{x} \in \mathbb{R}^d$ as

$$\frac{\partial f(\boldsymbol{x})}{\partial x^j} = \sum_{i=1}^{N} \alpha_i \frac{\partial k(\boldsymbol{x}, \boldsymbol{x}_i)}{\partial x^j} = \boldsymbol{r}_1(\boldsymbol{x})^T \boldsymbol{\alpha} \ , \tag{7}$$

where $\boldsymbol{r}_1(\boldsymbol{x}) = [\partial k(\boldsymbol{x}, \boldsymbol{x}_1)/\partial x^j \ \dots \ \partial k(\boldsymbol{x}, \boldsymbol{x}_i)/\partial x^j \ \dots \ \partial k(\boldsymbol{x}, \boldsymbol{x}_N)/\partial x^j]^T$. The derivative (7) is also linear in $\boldsymbol{\alpha}$. In fact, the form of the kernel expansion implies that all the derivatives are linear in $\boldsymbol{\alpha}$ such as, for instance, the Laplacian

$$\nabla^2 f(\boldsymbol{x}) = \sum_{j=1}^{d} \frac{\partial^2 f(x)}{\partial {x^j}^2} \ . \tag{8}$$

This derivative of the second order is a measure of the roughness of the function on which prior knowledge might be considered. The linearity in $\boldsymbol{\alpha}$ allows to write any scalar derivative $f^{(k)}$ of order k as

$$f^{(k)}(\boldsymbol{x}) = \boldsymbol{r}_k(\boldsymbol{x})^T \boldsymbol{\alpha} \tag{9}$$

where $\boldsymbol{r}_k(\boldsymbol{x})$ contains the coefficients for the kth order derivative that only depend on the kernel and the training set. $\boldsymbol{r}_1(\boldsymbol{x})$ and $\boldsymbol{r}_2(\boldsymbol{x})$ are given for a RBF kernel in Appendix A.

Setting up $\boldsymbol{r}(\boldsymbol{x})$ accordingly, $\boldsymbol{r}(\boldsymbol{x})^T \boldsymbol{\alpha}$ can be any component or linear combination of components of any derivative of f. This general remark allows the incorporation of constraints on any derivative of a kernel expansion f into a linear program such as (5).

4 Adding equality constraints

In this section, prior knowledge on the function is considered via n_{SC} sets of equality constraints of the type

$$g_k(\boldsymbol{x}) = h_k(\boldsymbol{x}), \ \forall \boldsymbol{x} \in Z_k \ , \tag{10}$$

where the set $Z_k = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p, \ldots, \boldsymbol{x}_{|Z_k|}\}$ contains the points of interest for the kth set of constraints. The points of Z_k can be chosen in the training set but are not restricted to these data. As will be seen in the following, the equalities (10) can be, for many interesting prior knowledge forms, reformulated as equality constraints that are linear in the parameters $\boldsymbol{\theta} = [\boldsymbol{\alpha}^T \ b]^T$ and of the general form

$$\boldsymbol{\Gamma}_k(Z_k)\boldsymbol{\theta} = \boldsymbol{\beta}_k(Z_k) \ . \tag{11}$$

The matrix $\boldsymbol{\Gamma}_k(Z_k)$ is built from rows $\boldsymbol{\gamma}_k(\boldsymbol{x})^T$ as

$$\boldsymbol{\Gamma}_k(Z_k) = \begin{bmatrix} \boldsymbol{\gamma}_k(\boldsymbol{x}_1)^T \\ \vdots \\ \boldsymbol{\gamma}_k(\boldsymbol{x}_p)^T \\ \vdots \\ \boldsymbol{\gamma}_k(\boldsymbol{x}_{|Z_k|})^T \end{bmatrix} \ , \tag{12}$$

where each row corresponds to a point of interest in Z_k. The constraints (11) can easily be introduced into (5) that becomes the following linear programming SVR with equality constraints (LPSVR-EC)

$$\begin{aligned} \min_{(\boldsymbol{\alpha},b,\boldsymbol{\xi},\boldsymbol{a})} \ & \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} \\ \text{s.t.} \quad & \begin{array}{rcl} -\boldsymbol{\xi} \leq \boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} & \leq & \boldsymbol{\xi} \\ 0 \leq \mathbf{1}\varepsilon & \leq & \boldsymbol{\xi} \\ -\boldsymbol{a} \leq \boldsymbol{\alpha} & \leq & \boldsymbol{a} \\ \boldsymbol{\Gamma}_k(Z_k)\boldsymbol{\theta} & = & \boldsymbol{\beta}_k(Z_k), \ k = 1, \ldots, n_{SC} \ . \end{array} \end{aligned} \tag{13}$$

The optimization problem to solve involves $3N + 1$ variables, $\sum_{k=1}^{n_{SC}} |Z_k|$ equality constraints and $5N$ inequality constraints.

Though adding equality constraints does not change the linear programming nature of the optimization problem, this can lead to an infeasible problem. To deal with the case where all the constraints cannot be satisfied simultaneously, the equalities can also be enforced by soft constraints. Introducing a set of n_{SC} vectors $\boldsymbol{z}_k = [z_1^k \ \ldots \ z_p^k \ \ldots \ z_{|Z_k|}^k]^T$ of positive slack variables and a set of trade-off parameters λ_k leads to the linear programming SVR with soft equality constraints (LPSVR-SEC)

$$\begin{aligned} \min_{(\boldsymbol{\alpha},b,\boldsymbol{\xi},\boldsymbol{a},\boldsymbol{z}_k)} \ & \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} + \sum_{k=1}^{n_{SC}} \lambda_k \mathbf{1}^T\boldsymbol{z}_k \\ \text{s.t.} \quad & \begin{array}{rcl} -\boldsymbol{\xi} \leq \boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} & \leq & \boldsymbol{\xi} \\ 0 \leq \mathbf{1}\varepsilon & \leq & \boldsymbol{\xi} \\ -\boldsymbol{a} \leq \boldsymbol{\alpha} & \leq & \boldsymbol{a} \\ -\boldsymbol{z}_k \leq \boldsymbol{\Gamma}_k(Z_k)\boldsymbol{\theta} - \boldsymbol{\beta}_k(Z_k) & \leq & \boldsymbol{z}_k, \ k = 1, \ldots, n_{SC} \ , \end{array} \end{aligned} \tag{14}$$

that involves $3N+1+\sum_{k=1}^{n_{SC}}|Z_k|$ variables and $5N+2\sum_{k=1}^{n_{SC}}|Z_k|$ inequality constraints. This problem formulation also offers the possibility to weight the effect of the prior knowledge on the solution. Large values of λ_k increase the fit to the prior knowledge, whereas small values imply a fit closer to the data.

The problem (14) includes the minimization of the ℓ_1-norm of the errors for the soft equality constraints. Using the ε-insensitive loss function on these errors (with a different ε than the one used for the training set) is also possible and straightforward. It can be used to include almost exact knowledge in an interval by authorizing violations of the equality constraints that are less than ε_{prior}. However, this type of prior knowledge can be equivalently considered with inequality constraints as studied in section 5.

The following presents typical applications of constrained optimization for the introduction of prior knowledge with the corresponding settings of $\boldsymbol{\Gamma}_k(Z_k)$ and $\boldsymbol{\beta}_k(Z_k)$ for (11). Synthetic examples are provided in sections 4 and 5. Since the purpose of these examples is mainly to show the application of the method on simple and easy to visualize problems, the hyperparameters are chosen arbitrarily without fine tuning. The mean square error (MSE) is computed with respect to the true function being approximated and on the same set of points in the x variable than the one used for training.

4.1 Prior knowledge on particular points

Prior knowledge on the function to approximate can sometimes take the form of $|Z_0|$ particular points $(\boldsymbol{x}_p, y_p) \in Z_0$ for which the values are certain (intercept, maximal values, equilibrium points...). For these points, we would like the model to give an exact value and not just an approximation based on noisy data or a dataset that lacks samples around these points. One way to tackle this problem is to apply, for these points, hard constraints of the type

$$f(\boldsymbol{x}_p) = \boldsymbol{K}(\boldsymbol{x}_p, \boldsymbol{X}^T)\boldsymbol{\alpha} + b = y_p, \tag{15}$$

while soft constraints are applied to the training set. Defining the matrix $\boldsymbol{Z}_0 = [\boldsymbol{x}_1 \ \dots \ \boldsymbol{x}_p \ \dots \ \boldsymbol{x}_{|Z_0|}]^T$ and the vector $\boldsymbol{y}^{(0)} = [y_1 \ \dots \ y_p \ \dots \ y_{|Z_0|}]^T$, the following setting

$$\boldsymbol{\Gamma}_0(Z_0) = \left[\boldsymbol{K}(\boldsymbol{Z}_0^T, \boldsymbol{X}^T)\ \mathbf{1}\right], \quad \boldsymbol{\beta}_0(Z_0) = \boldsymbol{y}^{(0)} \tag{16}$$

is used for the problems (13) or (14) to incorporate the prior knowledge.

The points $(\boldsymbol{x}_p, y_p)$ can also be considered as virtual samples with high (or infinite) confidence and thus can be added to the training set as potential SVs. In this case, the linear program with hard constraints on the virtual

samples can be formulated as

$$\begin{aligned} \min_{(\boldsymbol{\alpha},b,\boldsymbol{\xi},\boldsymbol{a})} \quad & \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} \\ \text{s.t.} \quad & -\boldsymbol{\xi} \leq \boldsymbol{K}(\boldsymbol{X}^T, [\boldsymbol{X}^T \ \boldsymbol{Z}_0^T])\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} \leq \boldsymbol{\xi} \\ & 0 \leq \quad \mathbf{1}\varepsilon \quad \leq \boldsymbol{\xi} \\ & -\boldsymbol{a} \leq \quad \boldsymbol{\alpha} \quad \leq \boldsymbol{a} \\ & \boldsymbol{K}(\boldsymbol{Z}_0^T, [\boldsymbol{X}^T \ \boldsymbol{Z}_0^T])\boldsymbol{\alpha} + b\mathbf{1} = \boldsymbol{y}^{(0)} \ . \end{aligned} \tag{17}$$

The corresponding output function is given by

$$f(\boldsymbol{x}) = \boldsymbol{K}(\boldsymbol{x}, [\boldsymbol{X}^T \ \boldsymbol{Z}_0^T])\boldsymbol{\alpha} + b \ , \tag{18}$$

where the vector $\boldsymbol{\alpha}$ contains $N+|Z_0|$ parameters, a certain amount of which will be zero due to the sparsity of LP-SVR.

Example: adding information about one point. In this example, 61 training points are generated in the interval $-3 \leq x \leq 3$ for the approximation of $\mathrm{sinc}(x)$. A Gaussian noise of standard deviation 0.2 and mean 0 is added to the training data. To show the efficiency of the method, one virtual sample $(0, 1)$ is added to the training set. Since this point is known for certain, it is assigned to a hard constraint implemented by equality as in (17). Figure 1 shows the improvement over a simple LP-SVR trained on the training set with the virtual sample considered as a standard sample.

4.2 Prior knowledge on the derivatives

Typically, knowledge on the derivatives may include the general shape of the function, local maxima or minima, saddle-points, high peaks... This type of knowledge is often available but, to our knowledge, there is no general method for its incorporation into SVR.

Consider that prior knowledge on the kth order derivative $f^{(k)}$ of the function f is available as

$$f^{(k)}(\boldsymbol{x}_p) = y_p^{(k)}, \ \forall \boldsymbol{x}_p \in Z_k \ . \tag{19}$$

This prior knowledge can be enforced in the training by using (9) and setting

$$\boldsymbol{\Gamma}_k(Z_k) = \begin{bmatrix} \boldsymbol{r}_k(\boldsymbol{x}_1)^T & 0 \\ \vdots & \\ \boldsymbol{r}_k(\boldsymbol{x}_p)^T & 0 \\ \vdots & \\ \boldsymbol{r}_k(\boldsymbol{x}_{|Z_k|})^T & 0 \end{bmatrix}, \quad \boldsymbol{\beta}_k(Z_k) = \begin{bmatrix} y_1^{(k)} \\ \vdots \\ y_p^{(k)} \\ \vdots \\ y_{|Z_k|}^{(k)} \end{bmatrix} \ , \tag{20}$$

in one of the problems (13) or (14). This method allows to incorporate prior knowledge on any derivative of order k for any set of points possibly different

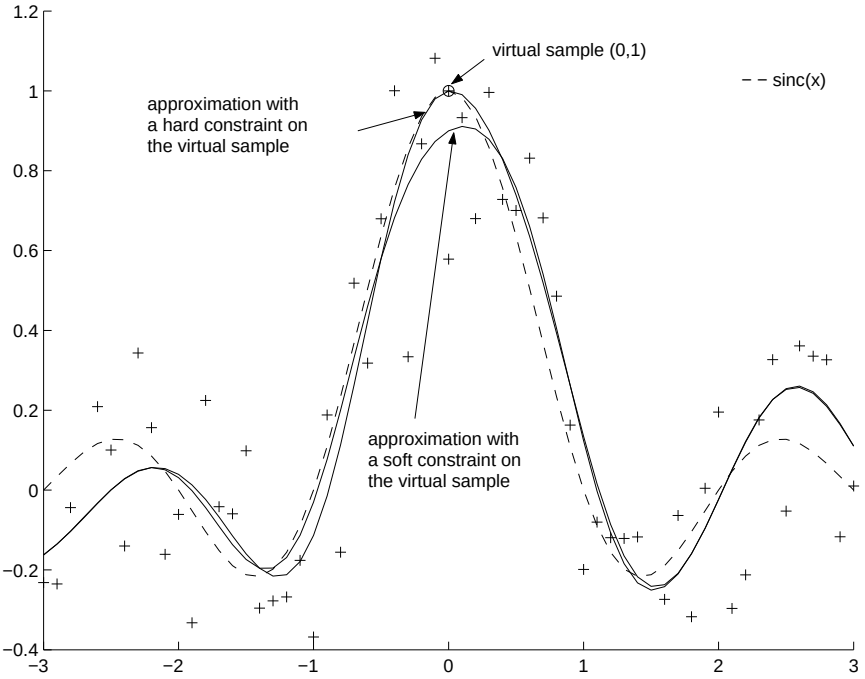

Fig. 1 Approximation of the sinc function with an added virtual sample $(0, 1)$. The values for the different parameters are: $C = 10, \sigma = 0.5, \varepsilon = 0.1$. Applying a hard constraint on the virtual sample (known for certain) improves the accuracy around this point.

for each k, while keeping the model in the form of (1). In comparison, the methods described in [Lázaro et al., 2005b] and [Lázaro et al., 2005a] require the prior derivative values on all the training points and these points only.

Example: knowledge on the derivatives at particular points. Here, we show the improvement in the quality of the approximation when only partial information about the shape is known at particular points via a mixture of derivatives (first and second order). Consider the problem of approximating the sinc function from noise-free data but without data around zero, that is for x in the interval $-1 < x < +1$. In this setting, the central peak of the sinc function cannot be approximated without prior knowledge. To show the effectiveness of the method, we consider only sparse and grossly approximate prior knowledge: $y^{(1)}(0) = 0,\ y^{(1)}(-0.5) = 1,\ y^{(2)}(0) = -2$. This corresponds to a function with a maximum at $x = 0$, increasing around $x = -0.5$ and with a peak around $x = 0$. Figure 2 shows that incorporating this simple prior knowledge in (14) by (20) allows to recover the shape of the sinc function. Moreover, it must be noticed that the function is approx-

imated without any training samples around 0 and thus without support vectors around 0. However, the number of SVs increases from 8 to 12.

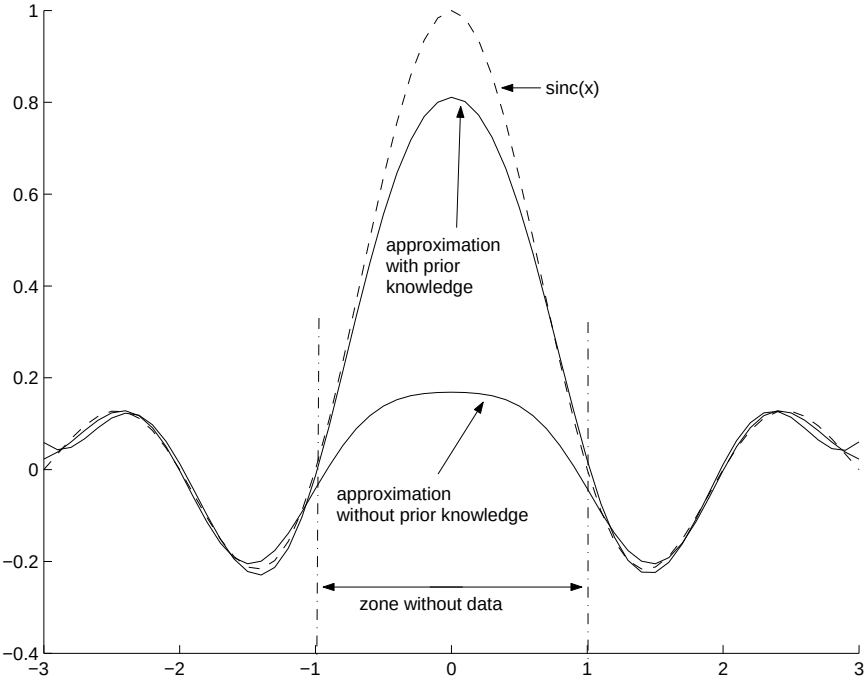

Fig. 2 Approximation of the sinc function without data for x in the interval $-1 < x < +1$ and with prior knowledge on the derivatives at 2 particular points $(y^{(1)}(0) = 0,\ y^{(1)}(-0.5) = 1,\ y^{(2)}(0) = -2)$. The values for the different parameters are: $\lambda = 50, C = 10, \sigma = 0.5, \varepsilon = 0.001$. Knowledge on very few points helps to recover the shape of the overall function.

4.3 Prior knowledge from a prior model

In some applications, one may wish to retain certain properties of a previous model such as the shape or the roughness at some specific points $\boldsymbol{x}_p$ in Z_k. This problem corresponds to the learning of a new model while constraining certain of its derivatives to equal those of the prior model at these particular points.

Assume now that the the prior model is a kernel expansion on N^{pr} samples, then $f^{prior}(\boldsymbol{x}) = \sum_{i=1}^{N^{pr}} \alpha_i^{prior} k(\boldsymbol{x}, \boldsymbol{x}_i) + b^{prior}$. Using the remark of section 3.2, its derivatives can also be expressed in a linear form (9) with respect to the parameters α_i^{prior} as $f^{prior\,(k)}(\boldsymbol{x}_p) = \boldsymbol{r}_k^{pr}(\boldsymbol{x}_p)^T \boldsymbol{\alpha}^{prior}$. Thus, in order to retain the kth order derivative from a prior model in kernel

expansion form, the new model is trained by solving (13) or (14) with the corresponding setting for (11) that is

$$\boldsymbol{\Gamma}_k(Z_k) = \begin{bmatrix} \boldsymbol{r}_k(\boldsymbol{x}_1)^T & 0 \\ \vdots & \\ \boldsymbol{r}_k(\boldsymbol{x}_p)^T & 0 \\ \vdots & \\ \boldsymbol{r}_k(\boldsymbol{x}_{|Z_k|})^T & 0 \end{bmatrix}, \quad \boldsymbol{\beta}_k(Z_k) = \begin{bmatrix} \boldsymbol{r}_k^{pr}(\boldsymbol{x}_1)^T\boldsymbol{\alpha}^{prior} \\ \vdots \\ \boldsymbol{r}_k^{pr}(\boldsymbol{x}_p)^T\boldsymbol{\alpha}^{prior} \\ \vdots \\ \boldsymbol{r}_k^{pr}(\boldsymbol{x}_{|Z_k|})^T\boldsymbol{\alpha}^{prior} \end{bmatrix}. \tag{21}$$

Notice that $\boldsymbol{r}_k(\boldsymbol{x}_p)$ depends only on the kernel and the training set, with one component for each training sample. Thus, if the training set $\boldsymbol{X}$ of the new model includes the N^{pr} points on which the prior model has been trained and if the same kernel is used for both models, then N^{pr} components of $\boldsymbol{r}_k(\boldsymbol{x}_p)$ are equal to the components of $\boldsymbol{r}_k^{pr}(\boldsymbol{x}_p)$. These components can so be computed only once to speed up the process.

Example: knowledge on the derivative from a prior model helps recovering empty regions. This example shows the enhancement of the approximation when the general shape of the function is known via a prior model and incorporated in the learning by the previously described method. The shape is enforced by maximizing the similarity of the first order derivatives of the approximation to the derivatives of the prior model.

Consider the problem of approximating the sinc function based on a set of training points X without noise and a prior model approximating $s(x) = \mathrm{sinc}(x) + \delta$. Only the shape of the function $s(x)$ is retained as prior knowledge, the added constant δ is considered unknown. The prior model is a standard LP-SVR, trained on 31 points of $s(x)$ (with $\delta = 1$) in the interval $-3 \leq x \leq 3$, yielding the parameters $\boldsymbol{\alpha}^{prior}$. On the other hand, the approximation f is trained on 34 points on the intervals $-3 \leq x \leq -1.4$ and $1.4 \leq x \leq 3$. Figure 3 shows the output of the prior model, the true sinc function and two approximations: one that includes the prior shape in the learning (14) by (21) and the other that does not (5). It is clear that the incorporation of prior knowledge on the shape of the function improves considerably the accuracy of the approximation with a small increase of the number of SVs from 8 to 12.

Example: knowledge on the derivative from a prior model helps against the noise. The next example shows how prior knowledge on the shape can reduce the effect of noise. Consider the problem of approximating the sinc function from noisy data. A set of 31 training samples is now generated with additive Gaussian noise of mean 0 and standard deviation σ_{noise}, for x_i in the interval $-3 \leq x_i \leq 3$. Figure 4 shows the results for a large noise level $\sigma_{noise} = 1$ (considering the amplitude of $\mathrm{sinc}(x) \approx 1.2$). It is clear that a standard LP-SVR applied to these data cannot approximate correctly the function. But taking the prior shape into account in the learning (14) by (21) filters the noise and yields a respectable approximation of the sinc though

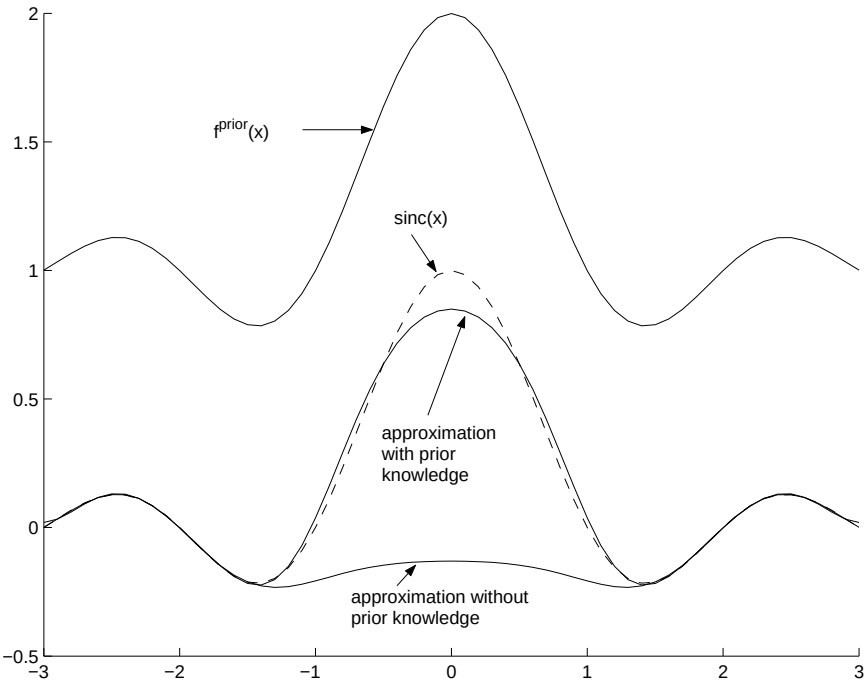

Fig. 3 Approximation of the sinc function with prior knowledge on the shape from a prior model f^{prior}. The values for the different parameters are: $\lambda = 10, C = 13, \sigma = 0.5, \varepsilon = 0.001$.

the prior shape is only given by an approximative prior model issued from a LPSVR training (5) on 31 points of the prior shape $s(x) = \text{sinc}(x) + \delta$, with $\delta = 1$. The number of SVs increases slightly from 7 to 11. These results were obtained with a "handpicked" trade-off parameter $\lambda = 5$ without any tuning. This trade-off parameter appearing in the problems (13) and (14) is used to weight the effect of the prior knowledge on the solution. Based on prior knowledge of the noise level, and so, of the relative relevance of the data, the approximation can still be enhanced by increasing λ. A large λ increases the fit to the prior shape as opposed to the fit to the data implied by a small λ.

This problem is close to the setting of [Lázaro et al., 2005b] and [Lázaro et al., 2005a] as prior knowledge on the derivatives is given at every training point. However, here, the derivatives values are not directly available and are estimated by a prior model trained on another set of samples.

4.4 Prior knowledge between outputs

Simultaneous approximation of multiple functions is easily performed by neural networks by simply considering a multidimensional output layer. On

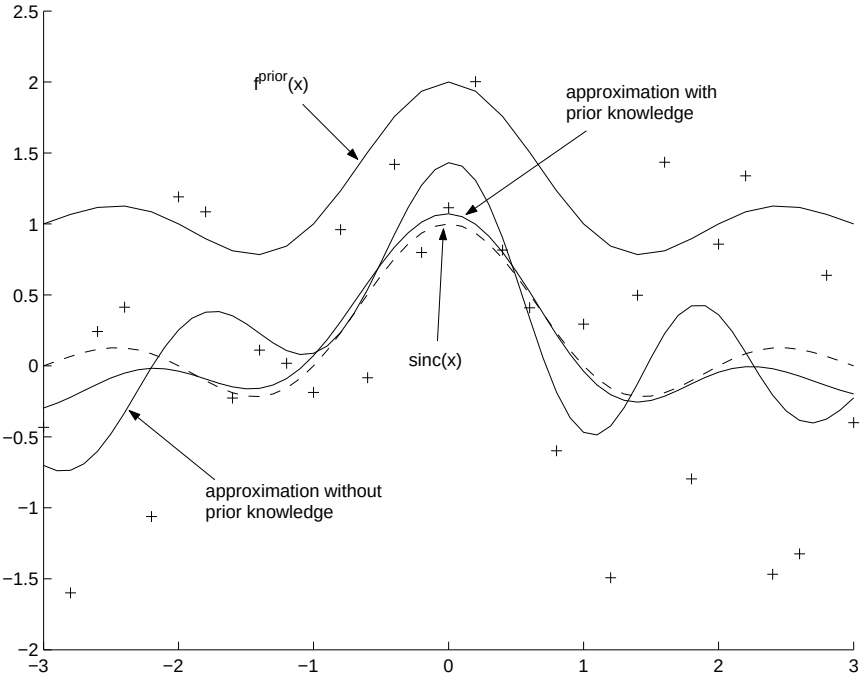

Fig. 4 Approximation of the sinc function with noisy data and prior knowledge on the shape from a prior model f^{prior}. The values for the different parameters are: $\lambda = 5, C = 10, \sigma = 0.5$, $\varepsilon = 1$, $\sigma_{noise} = 1$. The prior shape filters the noise.

the other hand, SVMs are single-output machines. The common approach for multi-outputs problems is thus to train as many independent SVMs as needed, one for each function to approximate. However, in some cases, these functions may be interdependent and the inclusion of these dependencies in the learning cannot be directly handled.

This section presents a multi-outputs SVR approach to approximate multiple functions of the same inputs when some prior knowledge on the dependencies between these functions is available. Section 4.4.1 will show how the proposed method can be mixed with the results on prior knowledge on the derivatives of section 4.2 for the simultaneous approximation of a function and its derivative.

The multi-outputs SVR implements m functions f_j of the same inputs that are learned from the training set $(\boldsymbol{X}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_j, \ldots, \boldsymbol{y}_m)$. The approximated output vector $\boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}) \ \ldots \ f_j(\boldsymbol{x}) \ \ldots \ f_m(\boldsymbol{x})]^T$ is given by

$$\boldsymbol{f}(\boldsymbol{x})^T = \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}^T)\boldsymbol{A} + \boldsymbol{b}^T \ , \tag{22}$$

where the matrix $\boldsymbol{A} = [\boldsymbol{\alpha}_1 \ \ldots \ \boldsymbol{\alpha}_j \ \ldots \ \boldsymbol{\alpha}_m]$ and the vector $\boldsymbol{b} = [b_1 \ \ldots \ b_j \ \ldots \ b_m]^T$ collect the parameters.

It is assumed that some prior knowledge on a linear combination of the m outputs with coefficients μ_j is available in the form

$$\sum_{j=1}^{m} \mu_j f_j(\boldsymbol{x}) = h(\boldsymbol{x}),\ \forall \boldsymbol{x} \in Z\ , \tag{23}$$

for an arbitrary function h. Defining the parameter vector $\tilde{\boldsymbol{\theta}} = [\boldsymbol{\alpha}_1^T\ b_1\ \ldots\ \boldsymbol{\alpha}_m^T\ b_m]^T$ and setting

$$\boldsymbol{\beta}(Z) = \begin{bmatrix} h(\boldsymbol{x}_1) \\ \vdots \\ h(\boldsymbol{x}_p) \\ \vdots \\ h(\boldsymbol{x}_{|Z|}) \end{bmatrix}\ , \tag{24}$$

the matrix $\boldsymbol{\Gamma}(Z)$ is built as in (12) from the rows $\boldsymbol{\gamma}(\boldsymbol{x}_p)^T$ in order to rewrite the prior knowledge (23) in the general form $\boldsymbol{\Gamma}(Z)\tilde{\boldsymbol{\theta}} = \boldsymbol{\beta}(Z)$. For a single point $\boldsymbol{x}_p$ of the set Z, this leads to

$$\boldsymbol{\gamma}(\boldsymbol{x}_p)^T\tilde{\boldsymbol{\theta}} = \mu_1 f_1(\boldsymbol{x}_p) + \cdots + \mu_m f_m(\boldsymbol{x}_p)\ . \tag{25}$$

Writing the output f_j as

$$f_j(\boldsymbol{x}_p) = \left[\boldsymbol{K}(\boldsymbol{x}_p, \boldsymbol{X}^T)\ 1\right] \begin{bmatrix} \boldsymbol{\alpha}_j \\ b_j \end{bmatrix}\ , \tag{26}$$

allows to rewrite (25) as

$$\boldsymbol{\gamma}(\boldsymbol{x}_p)^T\tilde{\boldsymbol{\theta}} = \left[\boldsymbol{K}(\boldsymbol{x}_p, \boldsymbol{X}^T)\ 1\right] \left(\mu_1 \begin{bmatrix} \boldsymbol{\alpha}_1 \\ b_1 \end{bmatrix} + \cdots + \mu_m \begin{bmatrix} \boldsymbol{\alpha}_m \\ b_m \end{bmatrix}\right)\ , \tag{27}$$

in which the parameter vector $\tilde{\boldsymbol{\theta}}$ can be introduced by

$$\boldsymbol{\gamma}(\boldsymbol{x}_p)^T\tilde{\boldsymbol{\theta}} = \left[\boldsymbol{K}(\boldsymbol{x}_p, \boldsymbol{X}^T)\ 1\right] \left[\mu_1\boldsymbol{I}\ \ldots\ \mu_m\boldsymbol{I}\right] \tilde{\boldsymbol{\theta}}\ , \tag{28}$$

where $\boldsymbol{I}$ stands for the identity matrix of size $N+1$. This formulation gives $\boldsymbol{\gamma}(\boldsymbol{x}_p)^T$ and thus yields, for the whole set Z, the matrix

$$\boldsymbol{\Gamma}(Z) = \left[\boldsymbol{K}(\boldsymbol{Z}^T, \boldsymbol{X}^T)\ \mathbf{1}\right] \left[\mu_1\boldsymbol{I}\ \ldots\ \mu_m\boldsymbol{I}\right]\ . \tag{29}$$

The training of a multi-outputs SVR can thus be performed by solving a problem similar to (14) but with multiple outputs as follows

$$\begin{aligned} \min_{(\tilde{\boldsymbol{\theta}}, \boldsymbol{\xi}_j, \tilde{\boldsymbol{a}}, \boldsymbol{z})}\ & \mathbf{1}^T\tilde{\boldsymbol{a}} + \sum_{j=1}^{m} C_j \mathbf{1}^T\boldsymbol{\xi}_j + \lambda \mathbf{1}^T\boldsymbol{z} \\ \text{s.t.}\quad & -\boldsymbol{\xi}_j \leq \boldsymbol{K}\boldsymbol{\alpha}_j + b_j\mathbf{1} - \boldsymbol{y}_j \leq \boldsymbol{\xi}_j,\ j = 1, \ldots, m \\ & 0 \leq \mathbf{1}\varepsilon_j \leq \boldsymbol{\xi}_j,\ j = 1, \ldots, m \\ & -\tilde{\boldsymbol{a}} \leq \tilde{\boldsymbol{\alpha}} \leq \tilde{\boldsymbol{a}} \\ & -\boldsymbol{z} \leq \boldsymbol{\Gamma}(Z)\tilde{\boldsymbol{\theta}} - \boldsymbol{\beta}(Z) \leq \boldsymbol{z}\ , \end{aligned} \tag{30}$$

where $\tilde{\boldsymbol{a}} = [\boldsymbol{a}_1^T \ \dots \ \boldsymbol{a}_m^T]^T$, $\tilde{\boldsymbol{\alpha}} = [\boldsymbol{\alpha}_1^T \ \dots \ \boldsymbol{\alpha}_m^T]^T$, $\boldsymbol{\Gamma}(Z)$ and $\boldsymbol{\beta}(Z)$ are set respectively as in (29) and (24), and where the subscript j indicates the jth output. Different hyperparameters C_j and ε_j are assigned to each output f_j. This problem involves $m(3N+1)+|Z|$ variables and $5Nm+|Z|$ constraints.

The prior knowledge between multiple outputs is thus easily incorporated in a linear program by adding equality constraints (then relaxed by slack variables) in a general form similar to the one used for the other types of prior knowledge. All these types of prior knowledge can so be mixed together. The example in the following section will show how to learn two functions simultaneously while considering that one is the derivative of the other. Then section 4.4.2 will show how the multi-outputs SVR can be used for a simple multi-models problem.

4.4.1 Approximating a function and its derivative by multi-outputs SVR with prior knowledge Consider the problem of approximating two functions knowing that one is the derivative of the other with respect to a particular component x^l. Then, using the results of both sections 4.2 and 4.4, this prior knowledge can be used to enhance the training. In this case, the prior knowledge for a point $\boldsymbol{x}$ is written

$$\frac{\partial f_1(\boldsymbol{x})}{\partial x^l} - f_2(\boldsymbol{x}) = \boldsymbol{r}_1(\boldsymbol{x})^T\boldsymbol{\alpha}_1 - \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}^T)\boldsymbol{\alpha}_2 - b_2 = 0 \tag{31}$$

The points on which this prior knowledge is considered are the points of the training set ($Z = X$). The corresponding setting is thus

$$\boldsymbol{\Gamma}(Z) = \begin{bmatrix} \boldsymbol{r}_1(\boldsymbol{x}_1)^T & 0 & -\boldsymbol{K}(\boldsymbol{x}_1, \boldsymbol{X}^T) & -1 \\ & \vdots & & \\ \boldsymbol{r}_1(\boldsymbol{x}_i)^T & 0 & -\boldsymbol{K}(\boldsymbol{x}_i, \boldsymbol{X}^T) & -1 \\ & \vdots & & \\ \boldsymbol{r}_1(\boldsymbol{x}_N)^T & 0 & -\boldsymbol{K}(\boldsymbol{x}_N, \boldsymbol{X}^T) & -1 \end{bmatrix} , \tag{32}$$

and

$$\boldsymbol{\beta}(Z) = \boldsymbol{0} , \tag{33}$$

for the problem (30).

Example: position and speed of a synthetic mechanical system. Let two output variables y_1 and y_2 be the position and speed of a mechanical system with one input variable (the time). The particular example studied here uses $y_1(x) = \sin(x)$ and $y_2(x) = \cos(x)$. The training data is composed of 31 samples in the interval $0 \leq x \leq 3$ with an additive centered Gaussian noise of standard deviation 0.2. Figure 5 shows the resulting approximations on both functions y_1 and y_2 with and without prior knowledge between these outputs considered. The prior knowledge $f_1'(x) = f_2(x)$ is included in (30) by (32) and (33) with a trade-off parameter set at $\lambda = 5$. The other hyperparameters are set at the same values for both methods and for

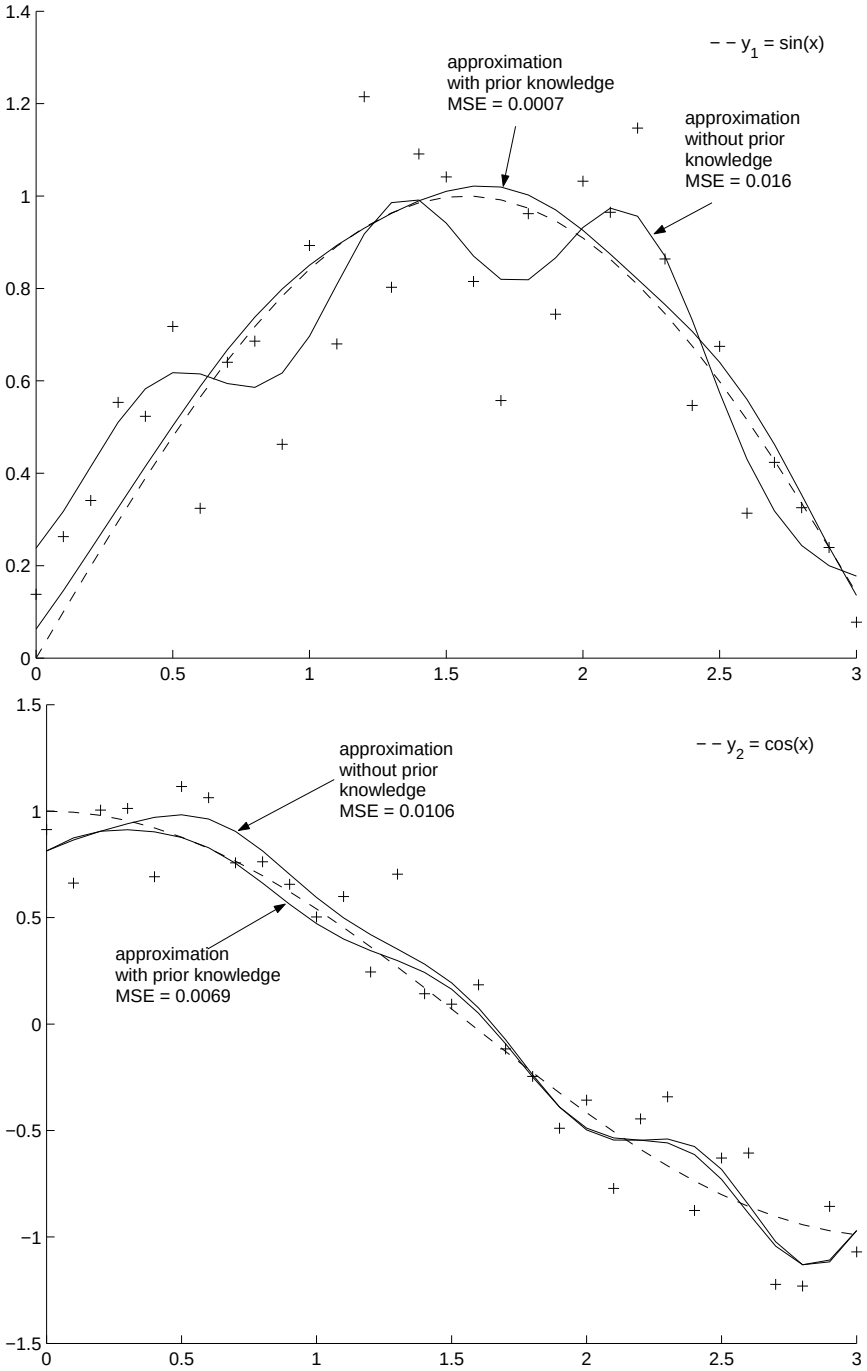

Fig. 5 Simultaneous approximation of two functions $y_1 = \sin(x)$ (top) and $y_2 = \cos(x)$ (bottom) with prior knowledge between outputs: $f_1'(x) = f_2(x)$. The values for the different parameters are: $C_1 = C_2 = 10, \sigma = 0.3, \varepsilon_1 = \varepsilon_2 = 0.1, \lambda = 5$.

both outputs: $C_1 = C_2 = 10, \sigma = 0.3, \varepsilon_1 = \varepsilon_2 = 0.1$. The MSE decreases dramatically when adding the prior knowledge above: from 0.016 to 0.0007 for y_1, and from 0.0106 to 0.0069 for y_2. Moreover the general shapes of the functions are recovered without being much affected by the noise.

4.4.2 Multi-models for continuous function approximation The following assumes that the function to approximate is composed of different subfunctions in different known regions of the input space. Though the method can be applied to multidimensional inputs, only the one-dimensional case is presented here for the sake of clarity. In this case, the prior knowledge leads to consider the approximation function $f(x)$ computed from a multitude of submodels $f_j(x)$ as

$$f(x) = \begin{cases} f_1(x) & , \text{ if } l_1 \leq x \leq u_1 \\ f_2(x) & , \text{ if } l_2 \leq x \leq u_2 \\ \vdots & \\ f_n(x) & , \text{ if } l_n \leq x \leq u_n \end{cases} . \tag{34}$$

Moreover, if the function to approximate is continuous, then a coupling between the submodels can be expressed as

$$f_j(x) = f_{j+1}(x), \text{ for } x = u_j = l_{j+1} , \tag{35}$$

where f_j is the submodel for the input region $l_j \leq x \leq u_j$, trained only on this region (training set X_j).

The simultaneous training of the submodels with continuity prior enforced by hard constraints is, in this case, expressed as

$$\begin{aligned} \min_{(\tilde{\boldsymbol{\alpha}}, \boldsymbol{b}, \boldsymbol{\xi}_j, \tilde{\boldsymbol{a}})} \ & \mathbf{1}^T \tilde{\boldsymbol{a}} + \sum_{j=1}^{m} C_j \mathbf{1}^T \boldsymbol{\xi}_j \\ \text{s.t.} \quad & \begin{array}{rcll} -\boldsymbol{\xi}_j \leq & \boldsymbol{K}_j \boldsymbol{\alpha}_j + b_j \mathbf{1} - \boldsymbol{y}_j & \leq \boldsymbol{\xi}_j, & j = 1, \ldots, m \\ 0 \leq & \mathbf{1}\varepsilon_j & \leq \boldsymbol{\xi}_j, & j = 1, \ldots, m \\ -\tilde{\boldsymbol{a}} \leq & \tilde{\boldsymbol{\alpha}} & \leq \tilde{\boldsymbol{a}} & \\ & \boldsymbol{\Gamma}(Z)\tilde{\boldsymbol{\theta}} & = \boldsymbol{\beta}(Z) , & \end{array} \end{aligned} \tag{36}$$

where $\boldsymbol{K}_j = \boldsymbol{K}(\boldsymbol{X}_j^T, \boldsymbol{X}_j^T)$,

$$\boldsymbol{\Gamma}(Z) = \begin{bmatrix} \boldsymbol{K}(u_1, \boldsymbol{X}_1^T) & \mathbf{1} & -\boldsymbol{K}(u_1, \boldsymbol{X}_2^T) & -\mathbf{1} & 0 & \ldots \\ \mathbf{0} & 0 & \boldsymbol{K}(u_2, \boldsymbol{X}_2^T) & \mathbf{1} & -\boldsymbol{K}(u_2, \boldsymbol{X}_3^T) & -\mathbf{1}\ 0\ \ldots \\ & & \ddots & & & \end{bmatrix} , \tag{37}$$

and

$$\boldsymbol{\beta}(Z) = \mathbf{0} . \tag{38}$$

Depending on the partitioning of the training set, the parameter vectors $\boldsymbol{\alpha}_j$ may be of different sizes. Thus, the formulation of the output (22) is not valid anymore. In this case, the outputs are computed separately by

$$f_j(x) = \boldsymbol{K}(x, \boldsymbol{X}_j^T)\boldsymbol{\alpha}_j + b_j \tag{39}$$

Example: piecewise affine (PWA) function approximation. In this example, the continuous function to approximate is composed of three affine parts: $y = 0.5x$, for $0 \leq x \leq 3$; $y = 2x - 4.5$, for $3 \leq x \leq 6$; $y = -2x + 19.5$, for $6 \leq x \leq 10$. The training data contains 101 points with additive Gaussian noise of standard deviation 1 and mean 0. Figure 6 shows the training data, the function, its approximation with a single SVR using RBF kernels and an approximation using the previously described method (36) with linear kernels and the settings (37) and (38). The test error decreases when considering a multitude of linear submodels (MSE= 0.078) instead of using a single RBF model (MSE= 0.117), whereas the number of SVs is equivalent and equals 3 for both methods. This number is highly correlated to the number of affine pieces in the global function. Comparing to independent linear models using only the prior knowledge on the piecewise affine form of the function, the overall MSE is also reduced thanks to the prior knowledge on the continuity. Moreover, the continuity ensured by the proposed method cannot be obtained by independent linear models.

5 Adding inequality constraints

As seen in the previous section, equality constraints allow the inclusion of a large variety of prior knowledge. However, some interesting types of knowledge cannot be written in equalities and requires the use of inequalities such as, for instance, the monotonicity of the function: $f^{(1)}(x) \leq 0$ or $f^{(1)}(x) \geq 0$ for all x. This section proposes to include these types of knowledge on the function by considering inequality constraints such as

$$g_k(\boldsymbol{x}) \leq h_k(\boldsymbol{x}),\ \forall \boldsymbol{x} \in Z_k\ , \tag{40}$$

where the set Z_k contains the $|Z_k|$ points of interest for the kth set of constraints. As for equalities, these inequalities can be reformulated as inequality constraints that are linear in the parameters $\boldsymbol{\theta} = [\boldsymbol{\alpha}\ b]^T$, of the general form

$$\boldsymbol{\Gamma}_k(Z_k)\boldsymbol{\theta} \leq \boldsymbol{\beta}_k(Z_k)\ , \tag{41}$$

that can be introduced into (5) without changing the linear programming nature of the optimization problem that becomes the following linear pro-

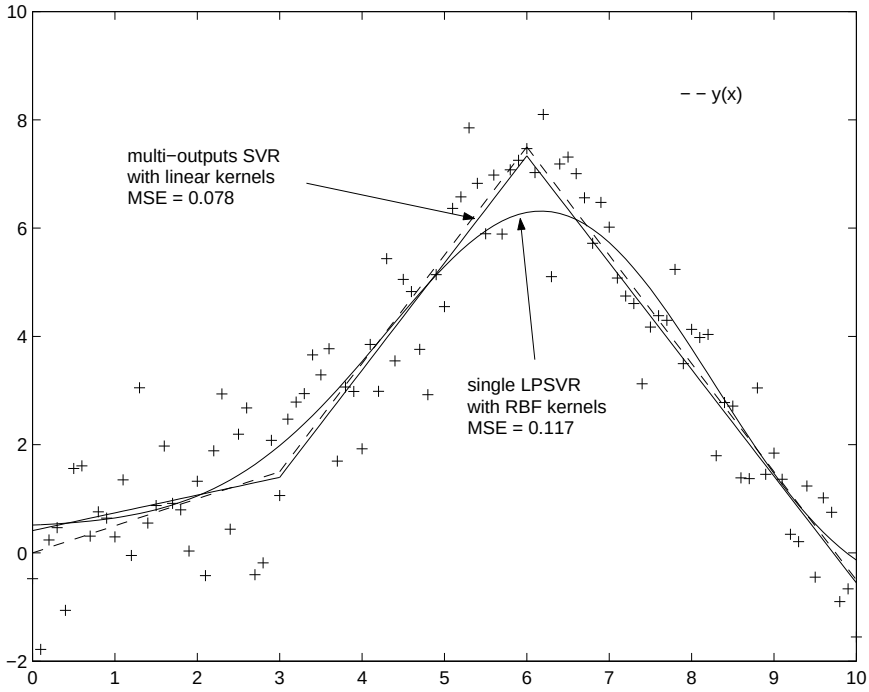

Fig. 6 Approximation of a piecewise affine function. The values for the different parameters of the multi-outputs SVR are $C_j = 0.2, \varepsilon_j = 1, \forall j$, and for the single SVR with RBF kernels: $C = 1, \varepsilon = 1, \sigma = 2$. Continuity is enforced by the prior knowledge $f_1(x) = f_2(x)$ for $x = 3$ and $f_2(x) = f_3(x)$ for $x = 6$.

gramming SVR with inequality constraints (LPSVR-IC)

$$\begin{aligned} \min_{(\boldsymbol{\alpha},b,\boldsymbol{\xi},\boldsymbol{a})} \quad & \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} \\ \text{s.t.} \quad & -\boldsymbol{\xi} \leq \boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} \leq \boldsymbol{\xi} \\ & 0 \leq \quad \mathbf{1}\varepsilon \quad \leq \boldsymbol{\xi} \\ & -\boldsymbol{a} \leq \quad \boldsymbol{\alpha} \quad \leq \boldsymbol{a} \\ & \quad \boldsymbol{\Gamma}_k(Z_k)\boldsymbol{\theta} \quad \leq \boldsymbol{\beta}_k(Z_k), \ k = 1, \ldots, n_{SC} \ . \end{aligned} \tag{42}$$

To deal with the case where all the constraints cannot be satisfied simultaneously, the inequalities can also be enforced by soft constraints. Introducing a set of n_{SC} vectors $\boldsymbol{z}_k = [z_1^k \ \ldots \ z_p^k \ \ldots \ z_{|Z_k|}^k]^T$ of positive slack variables and a set of trade-off parameters λ_k leads to the linear program

SVR with soft inequality constraints (LPSVR-SIC)

$$\min_{(\boldsymbol{\alpha},b,\boldsymbol{\xi},\boldsymbol{a},\boldsymbol{z}_k\geq\mathbf{0})} \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} + \sum_{k=1}^{n_{SC}} \lambda_k \mathbf{1}^T\boldsymbol{z}_k$$

$$\begin{array}{rccl} \text{s.t.} & -\boldsymbol{\xi} \leq & \boldsymbol{K}\boldsymbol{\alpha} + b\mathbf{1} - \boldsymbol{y} & \leq \boldsymbol{\xi} \\ & 0 \leq & \mathbf{1}\varepsilon & \leq \boldsymbol{\xi} \\ & -\boldsymbol{a} \leq & \boldsymbol{\alpha} & \leq \boldsymbol{a} \\ & & \boldsymbol{\Gamma}_k(Z_k)\boldsymbol{\theta} - \boldsymbol{\beta}_k(Z_k) & \leq \boldsymbol{z}_k,\ k = 1,\ldots,n_{SC}\ . \end{array} \tag{43}$$

5.1 Prior on the function, the derivatives and from a prior model

All the forms of prior knowledge described in section 4 can be considered with inequalities instead of equalities by using (42) or (43) and setting $\boldsymbol{\Gamma}_k$ and $\boldsymbol{\beta}_k$ as depicted for the equalities. This corresponds to the inclusion of bounds on the function or any derivative. For instance the following prior knowledge on the derivative $f^{(k)}(\boldsymbol{x}_p) \leq y_p^{(k)}$, for all $\boldsymbol{x}_p$ in Z_k, can be included in the learning by setting

$$\boldsymbol{\Gamma}_k(Z_k) = \begin{bmatrix} \boldsymbol{r}_k(\boldsymbol{x}_1)^T & 0 \\ \vdots & \\ \boldsymbol{r}_k(\boldsymbol{x}_p)^T & 0 \\ \vdots & \\ \boldsymbol{r}_k(\boldsymbol{x}_{|Z_k|})^T & 0 \end{bmatrix}, \quad \boldsymbol{\beta}_k(Z_k) = \begin{bmatrix} y_1^{(k)} \\ \vdots \\ y_p^{(k)} \\ \vdots \\ y_{|Z_k|}^{(k)} \end{bmatrix}, \tag{44}$$

for the linear program (42) or (43). A practical use of inequalities occurs for instance when the function to estimate corresponds to a physical variable known to be positive. In this absurd approximations that yield negative values can be avoided by forcing bounds on the function as prior knowledge in the learning process.

Example: lower bound on the function. This example is taken from [Mangasarian et al., 2004] and shows the benefit of including a lower bound on the function in a corrupted region of input space. The data to be modeled are generated by the sinc function with an additional Gaussian noise of mean 0 and standard deviation 0.5 for 32 points in the intervals $-3 \leq x \leq -1.43$ and $1.43 \leq x \leq 3$. Three points at $x = 0$ with output values $y = -1$, 0 and $+1$ are added to the training set in order to mislead the approximation. Three models are trained on these data: the standard LP-SVR without prior knowledge (5), the Knowledge-Based Kernel Approximation (KBKA) [Mangasarian et al., 2004] and the proposed LP-SVR with prior knowledge (43). The knowledge originally used for KBKA is $f(x) \geq \text{sinc}(0.25)$ in the interval $-0.25 \leq x \leq 0.25$. In order to include this information in our

method, three points $x = -0.25$, 0 and 0.25 are considered with the output value $y = \mathrm{sinc}(0.25)$. The settings for constraints (41) are thus given by (16) with $\boldsymbol{Z}_0 = [-0.25\ 0\ 0.25]^T$ and $\boldsymbol{y}^0 = [\mathrm{sinc}(0.25)\ \mathrm{sinc}(0.25)\ \mathrm{sinc}(0.25)]^T$. The hyperparameters of the KBKA are chosen as in [Mangasarian et al., 2004]: $C = 13$, $\sigma = 0.7071$, $\mu_1 = 5$, $\mu_2 = 450$. The other models use the same values for C and σ. λ is arbitrarily set to 10. The resulting approximations for the three models appear in Figure 7. This example shows that even in the case of knowledge defined over polyhedral sets, our method, though requiring discretization of these sets, can yield comparable results to the ones obtained by Mangasarian's method. The MSE computed on 101 equally spaced points in the interval $[-3, 3]$ is respectively 0.176, 0.104 and 0.032 for the LP-SVR without knowledge, the KBKA and the proposed method.

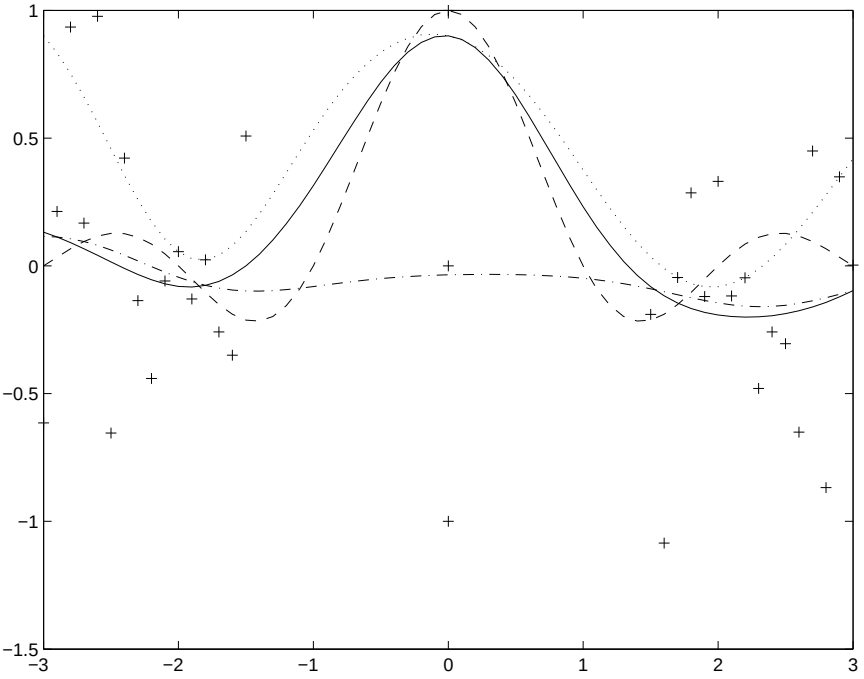

Fig. 7 Approximation of the sinc function (dashed line) with additional wrong data at $x = 0$. The mode without knowledge (dash-dotted line) cannot recover the true shape around $x = 0$. Prior knowledge stating that $f(x) \geq \mathrm{sinc}(0.25)$ in the interval $-0.25 \leq x \leq 0.25$ is included either by the method of [Mangasarian et al., 2004] (dotted line) or by the proposed method (solid line). The values for the different parameters are: $\lambda = 10, C = 13, \sigma = 0.7071, \varepsilon = 0.01$.

Example: knowledge on the derivatives at particular points by inequalities. Taking the example of Sect. 4.2 and using the same prior knowledge, in-

equality constraints might help to approximate the infinite second order derivative at $x = 0$. In the setting of section 4.2, an equality constraint enforces $f^{(2)}(0) = -2$, which penalizes notably the values less than -2. At the contrary, the constraint $f^{(2)}(0) \leq -2$ is less restrictive and can let the derivative go to minus infinite. Figure 8 shows the approximations provided by (13) and (43) respectively using equality and inequality constraints. The two methods gave very comparable results. Actually, the inequality constraints bounding the magnitude of the derivatives from below are active and thus become equalities. This is explained by the use of the RBF kernel and the minimization of the parameters α_i that lead to a smooth function. Thus the algorithm looks for the smoothest function that satisfies the constraints, i.e. the function with minimum derivatives in magnitude, which corresponds in this case to the equality constraints $y^{(1)}(0) = 0$, $y^{(1)}(-0.5) = 1$ and $f^{(2)}(0) = -2$.

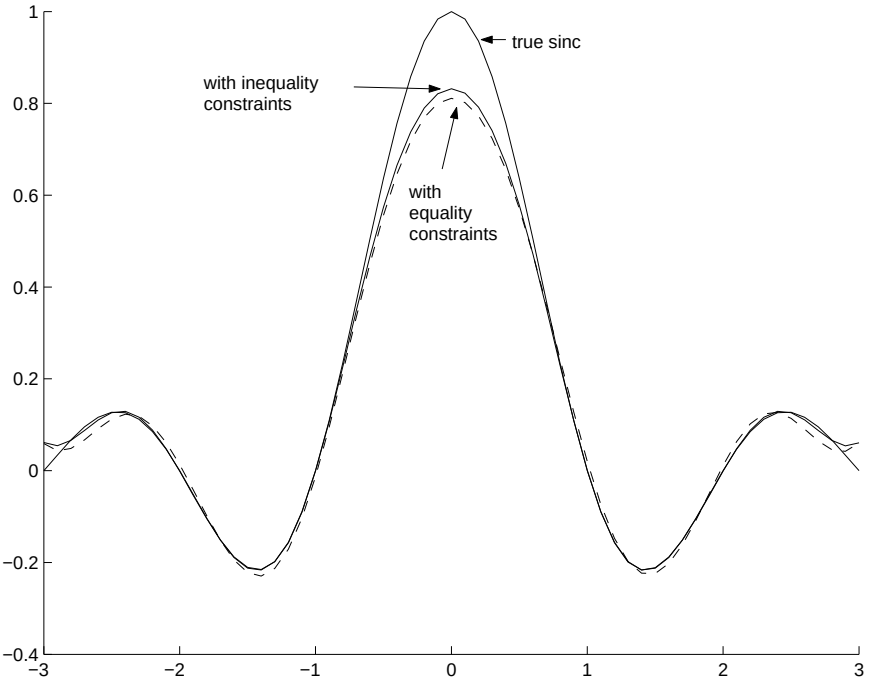

Fig. 8 Approximation of the sinc function without data for x in the interval $-1 < x < +1$ and with prior knowledge on the derivatives at 2 particular points ($y^{(1)}(0) \leq 0.1,\ y^{(1)}(-0.5) \geq 1,\ y^{(2)}(0) \leq -2$) incorporated via inequality constraints (solid line) and equalities (dash line). The values for the different parameters are: $\lambda = 50, C = 10, \sigma = 0.5, \varepsilon = 0.001$.

5.2 Prior knowledge between outputs

The multi-outputs SVR of section 4.4 can also consider prior knowledge on a linear combination of the m outputs as inequalities

$$\sum_{j=1}^{m} \mu_j f_j(\boldsymbol{x}) \leq h(\boldsymbol{x}),\ \forall \boldsymbol{x} \in Z\ , \tag{45}$$

for an arbitrary function h. In this case, the training of a multi-outputs SVR is performed by solving, as in (30), the following problem

$$\begin{aligned} \min_{(\tilde{\boldsymbol{\alpha}},\boldsymbol{b},\boldsymbol{\xi}_j,\tilde{\boldsymbol{a}},\boldsymbol{z}\geq\boldsymbol{0})} \ & \mathbf{1}^T\tilde{\boldsymbol{a}} + \sum_{j=1}^{m} C_j \mathbf{1}^T\boldsymbol{\xi}_j + \lambda \mathbf{1}^T\boldsymbol{z} \\ \text{s.t.} \quad -\boldsymbol{\xi}_j \leq\ & \boldsymbol{K}\boldsymbol{\alpha}_j + b_j\mathbf{1} - \boldsymbol{y}_j \leq \boldsymbol{\xi}_j,\ j=1,\dots,m \\ 0 \leq\ & \mathbf{1}\varepsilon_j \leq \boldsymbol{\xi}_j,\ j=1,\dots,m \\ -\tilde{\boldsymbol{a}} \leq\ & \tilde{\boldsymbol{\alpha}} \leq \tilde{\boldsymbol{a}} \\ & \boldsymbol{\Gamma}(Z)\tilde{\boldsymbol{\theta}} - \boldsymbol{\beta}(Z) \leq \boldsymbol{z}\ , \end{aligned} \tag{46}$$

with the same setup (29) and (24) as for equalities. Of course, hard constraints can also be considered here, leading to

$$\begin{aligned} \min_{(\tilde{\boldsymbol{\alpha}},\boldsymbol{b},\boldsymbol{\xi}_j,\tilde{\boldsymbol{a}})} \ & \mathbf{1}^T\tilde{\boldsymbol{a}} + \sum_{j=1}^{m} C_j \mathbf{1}^T\boldsymbol{\xi}_j \\ \text{s.t.} \quad -\boldsymbol{\xi}_j \leq\ & \boldsymbol{K}\boldsymbol{\alpha}_j + b_j\mathbf{1} - \boldsymbol{y}_j \leq \boldsymbol{\xi}_j,\ j=1,\dots,m \\ 0 \leq\ & \mathbf{1}\varepsilon_j \leq \boldsymbol{\xi}_j,\ j=1,\dots,m \\ -\tilde{\boldsymbol{a}} \leq\ & \tilde{\boldsymbol{\alpha}} \leq \tilde{\boldsymbol{a}} \\ & \boldsymbol{\Gamma}(Z)\tilde{\boldsymbol{\theta}} \leq \boldsymbol{\beta}(Z)\ . \end{aligned} \tag{47}$$

The prior knowledge as inequalities between multiple outputs is thus easily incorporated in a linear program by adding inequalities in a general form similar to the one used for the other types of prior knowledge.

Example: ordered functions. Consider the problem of simultaneously approximating two scalar functions while knowing that one is greater than the other. In this case, the prior knowledge $f_1(x) \leq f_2(x)$ may help to enhance the quality of the model. In this example, the functions $y_1 = \sin(x)$ and $y_2 = \sin(x) + 0.1$ are approximated based on noisy data. A Gaussian noise of mean zero and standard deviation 0.2 has been added to both functions for 71 samples in the interval $0 \leq x \leq 7$. Two outliers are added to the training set of y_1: $(3.7, 0)$ and $(3.8, 0)$, as shown on Figure 9. On this problem, training two independent models by (11), with the parameters set to $C = 10, \varepsilon = 0.1$ and $\sigma = 0.5$, yields approximations that are significantly in violation of the prior knowledge. At the contrary, using the proposed method with hard constraints (47) allows to obtain better approximations

that respect the constraints, as shown on Fig. 9. Moreover, the effect of the outliers on y_1 is reduced thanks to the relation imposed between f_1 and f_2 and the fact that the training data for y_2 do not contain outliers. The MSE for y_1 is reduced from 0.0102 to 0.0034, and for y_2 from 0.0072 to 0.0035.

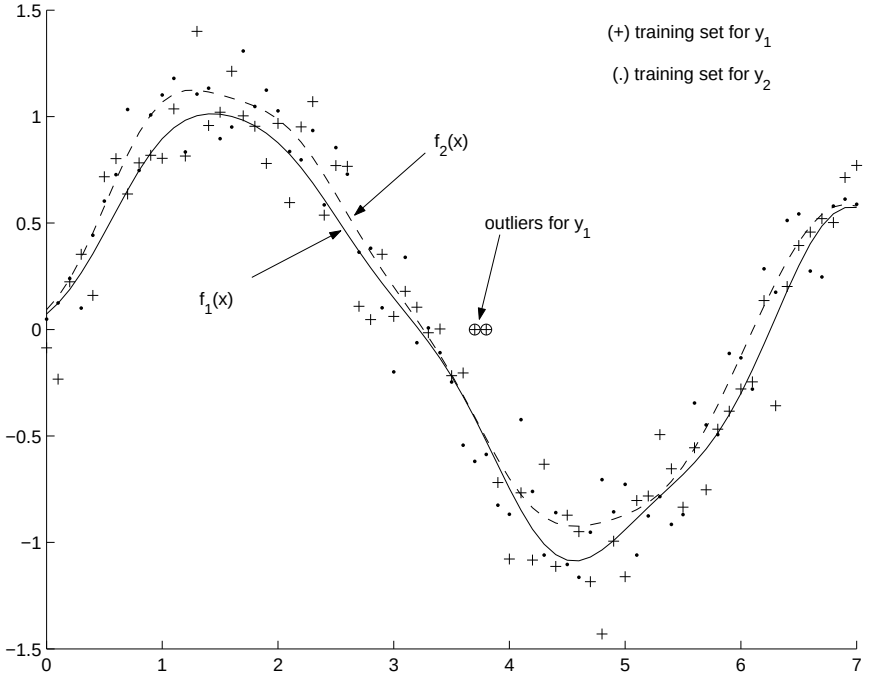

Fig. 9 Approximation of two sine functions $y_1 = \sin(x)$ and $y_2 = \sin(x) + 0.1$ with the corresponding training sets. The approximation uses the prior knowledge $f_1(x) \leq f_2(x)$. The values for the different parameters are: $C_1 = C_2 = 1, \sigma = 0.5, \varepsilon_1 = \varepsilon_2 = 0.1$.

6 Numerical examples

This section provides three examples of practical use for the proposed method. The two first examples deal with the identification of simulated nonlinear dynamical systems. The last example shows the benefit of the method on a real-life application, namely, the estimation of in-cylinder residual gas fraction in spark ignition engines.

6.1 Nonlinear dynamical system with known equilibrium points

Consider the discrete-time nonlinear dynamical system taken from [Narendra and Parthasarathy, 1990] and described by

$$y(k+1) = \frac{y(k)y(k-1)[y(k)+2.5]}{1+y^2(k)+y^2(k-1)} + u(k) \ , \tag{48}$$

where $y(k)$ and $u(k)$ are respectively the output and the input of the system at step k. The aim of this example is to find a Nonlinear AutoRegressive model with eXogenous input (NARX) $\hat{y}(k+1) = f(y(k), y(k-1), u(k))$ that minimizes the mean square error $MSE = \sum_{k=1}^{N_t}(y(k) - \hat{y}(k))^2$ over a trajectory of N_t test points, for which the NARX model is used as a simulation model $\hat{y}(k+1) = f(\hat{y}(k), \hat{y}(k-1), u(k))$. In [Narendra and Parthasarathy, 1990], the two equilibrium points in state space $(y(k) = 0, y(k-1) = 0)$ and $(y(k) = 2, y(k-1) = 2)$ of the unforced system (for $u(k) = 0$) are considered to be known. These points are the solutions in y of the equation (48), with $y(k+1) = y(k) = y(k-1) = y$ and for $u(k) = 0$.

A trajectory is generated from the initial condition $(y(1) = 1, y(0) = 1)$ by 100 iterations of the system (48) with additional Gaussian noise of mean 0 and standard deviation 0.1 for a uniformly distributed random input sequence in the interval $[-2, 2]$. This trajectory is used to build the training set as $\boldsymbol{x}_i = [y(i)\ y(i-1)\ u(i)]^T$ and $y_i = y(i+1)$, for $i = 1, \ldots, 100$. Two models are trained on these data: a standard LP-SVR without prior knowledge (5) and a LP-SVR with hard constraints (13) on the equilibrium points such that $f([0\ 0\ 0]^T) = 0$ and $f([2\ 2\ 0]^T) = 0$. For both algorithms RBF kernels are used and the hyperparameters are arbitrarily chosen to be $C = 10$, $\sigma = 1$ and $\varepsilon = 0.01$. The MSE is evaluated on the test set built from a different trajectory of 100 points generated by (48) from the initial condition $(y(1) = 0.5, y(0) = 0.5)$ and a sinusoidal input $u(k) = \sin(2\pi k/25)$. Table 1 shows the average and standard deviation of the test MSE computed over 100 runs. A gain in performance is observed when prior knowledge on equilibrium points is used as the average MSE decreases by 20.4 %.

Table 1 Average and standard deviation of the test MSE over 100 runs for the modeling of the nonlinear dynamical system (48). The average MSE decreases when using prior knowledge on the equilibrium points of the system.

Model	error
with prior knowledge	0.354 ± 0.356
without prior knowledge	0.445 ± 0.457

6.2 Nonlinear dynamical system with prior knowledge on the input function

Consider the discrete-time nonlinear dynamical system also taken from [Narendra and Parthasarathy, 1990] and described by

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \ , \tag{49}$$

where $y(k)$ and $u(k)$ are respectively the output and the input of the system at step k. In this example, the predictions are based on two variables only as $\hat{y}(k+1) = f(y(k), u(k))$. The prior knowledge extracted from the expression of the system is that the partial derivative of f with respect to the input must be an even function.

As in the previous example, the training set is built from a trajectory that is generated from the initial condition ($y(1) = 0$) by 100 iterations of the system (49) with additional Gaussian noise of mean 0 and standard deviation 0.1 for a uniformly distributed random input sequence in the interval $[-2, 2]$. Two models with RBF kernels are trained on these data: a standard LP-SVR without prior knowledge (5) and a LP-SVR (14) with prior knowledge on derivatives such that $\partial f(y(k), u(k))/\partial u = \partial f(y(k), -u(k))/\partial u$. To include this information, the partial derivative of f is computed by (55) at every point $(y(k), u(k))$ of the training set to build the matrix $\boldsymbol{R} = [\boldsymbol{r}_1(y(1), u(1)) \ \ldots \ \boldsymbol{r}_1(y(k), u(k)) \ \ldots \ \boldsymbol{r}_1(y(N), u(N))]^T$ and also at all N points $(y(k), -u(k))$ to build a similar matrix $\tilde{\boldsymbol{R}}$. The prior knowledge $\boldsymbol{R}\boldsymbol{\alpha} = \tilde{\boldsymbol{R}}\boldsymbol{\alpha}$ is then included in the learning (14) by setting, as in (20), $\boldsymbol{\Gamma} = [(\boldsymbol{R} - \tilde{\boldsymbol{R}}) \ \boldsymbol{0}]$ and $\boldsymbol{\beta} = \boldsymbol{0}$. The hyperparameters are arbitrarily chosen to be $C = 10$, $\sigma = 1$ and $\varepsilon = 0.01$. The prior knowledge parameter λ is set to 1. The MSE is evaluated on the test set built from a different trajectory generated from the initial condition ($y(0) = 0$) and the input $u(k) = \sin(2\pi k/25) + \sin(2\pi k/10)$, as in [Narendra and Parthasarathy, 1990]. Table 2 shows the average and standard deviation of the test MSE computed over 100 runs. A gain in performance is observed when prior knowledge is used as the average MSE decreases by 25.7 %.

Table 2 Average and standard deviation of the test MSE over 100 runs for the modeling of the nonlinear dynamical system (49). The average MSE decreases when forcing the partial derivative of the model with respect to the input to be even.

Model	error
with prior knowledge	1.158 ± 0.755
without prior knowledge	1.560 ± 0.667

6.3 Estimation of in-cylinder residual gas fraction

The application deals with the estimation of residual gases in the cylinders of Spark Ignition (SI) engines with Variable Camshaft Timing (VCT). More precisely, we are interested in estimating the residual gas mass fraction as studied in [Bloch et al., 2007]. Knowing this fraction allows to control torque as well as pollutant emissions. The residual gas mass fraction χ_{res} can be expressed as a function of the engine speed N_e, the ratio p_{man}/p_{exh}, where p_{man} and p_{exh} are respectively the (intake) manifold pressure and the exhaust pressure, and an overlapping factor OF, which is an image of the time during which the valves are opened together.

The available data are provided, on one hand, from the modeling and simulation environment Amesim [Imagine, 2006], which uses a high frequency zero-dimensional thermodynamic model and, on the other hand, from off line measurements, which are accurate, but complex and costly to obtain, by direct in-cylinder sampling [Giansetti et al., 2007]. The problem is thus as follows. How to obtain a simple, embeddable, black box model with a good accuracy and a large validity range for the real engine, from precise real measurements as less numerous as possible and a representative, but possibly biased, prior simulation model? The problem thus posed, although particular, is very representative of numerous situations met in engine control, and more generally in engineering, where complex models, more or less accurate, exist and where the experimental data which can be used for calibration are difficult or expensive to obtain.

Three datasets are built from the available data composed of 26 experimental samples plus 26 simulation samples:

- the training set $(\boldsymbol{X}, \boldsymbol{y})$ composed of a limited amount of real data (N samples),
- the test set composed of independent real data ($26 - N$ samples),
- the simulation set $(\boldsymbol{Z}, \boldsymbol{y}_p)$ composed of data provided by the simulator ($N^{pr} = 26$ samples).

The test samples are assumed to be unknown during the training and are retained for testing only. It must be noted that the inputs of the simulation data do not exactly coincide with the inputs of the experimental data.

The simulator being biased but approximating rather well the overall shape of the function, the prior knowledge will be incorporated on the derivatives. In order to be able to evaluate the prior derivatives a *prior model*, $f^{prior}(\boldsymbol{x}) = \sum_{i=1}^{N^{pr}} \alpha_i^{prior} k(\boldsymbol{x}, \boldsymbol{x}_i) + b^{prior}$, is first trained on the simulation data only. This prior model is then used to provide the values $y'_p = \boldsymbol{r}_1^{pr}(\boldsymbol{x}_p)^T \boldsymbol{\alpha}^{prior}$ of the derivative with respect to the input p_{man}/p_{exh}, at the points $\boldsymbol{x}_p$ of the simulation set. The *proposed model* is then trained by a variant of algorithm (14) with derivative constraints (21), where the points $\boldsymbol{x}_p$ are added as potential SVs. Defining the matrix $\boldsymbol{R}(\boldsymbol{Z}^T, [\boldsymbol{X}^T \ \boldsymbol{Z}^T]) = [\boldsymbol{r}_1(\boldsymbol{x}_1) \ \dots \ \boldsymbol{r}_1(\boldsymbol{x}_p) \ \dots \ \boldsymbol{r}_1(\boldsymbol{x}_{N^{pr}})]^T$, where $\boldsymbol{r}_1(\boldsymbol{x})$ corresponds to the derivative of $f(\boldsymbol{x}) = \boldsymbol{K}(\boldsymbol{x}, [\boldsymbol{X}^T \ \boldsymbol{Z}^T])\boldsymbol{\alpha} + b$ with respect to

the input p_{man}/p_{exh}, this algorithm reads

$$\min_{(\boldsymbol{\alpha},b,\boldsymbol{\xi},\boldsymbol{a},\boldsymbol{z})} \frac{1}{N+N^{pr}}\mathbf{1}^T\boldsymbol{a} + \frac{C}{N}\mathbf{1}^T\boldsymbol{\xi} + \frac{\lambda}{N^{pr}}\mathbf{1}^T\boldsymbol{z}$$

$$\text{s.t.} \quad \begin{array}{rcc} -\boldsymbol{\xi} \leq & \boldsymbol{K}(\boldsymbol{X}^T,[\boldsymbol{X}^T\ \boldsymbol{Z}^T])\boldsymbol{\alpha}+b\mathbf{1}-\boldsymbol{y} & \leq \boldsymbol{\xi} \\ \mathbf{0} \leq & \mathbf{1}\varepsilon & \leq \boldsymbol{\xi} \\ -\boldsymbol{a} \leq & \boldsymbol{\alpha} & \leq \boldsymbol{a} \\ -\boldsymbol{z} \leq & \boldsymbol{R}(\boldsymbol{Z}^T,[\boldsymbol{X}^T\ \boldsymbol{Z}^T])\boldsymbol{\alpha}-\boldsymbol{y}'_p & \leq \boldsymbol{z}\ . \end{array} \tag{50}$$

The hyperparameters have also been changed to C/N and λ/N^{pr} to maintain the same order of magnitude between the regularization, error and prior knowledge terms in the cost function. This allows to ease the choice of their value based on the application goals and confidence on the prior knowledge. Hence, the hyperparameters become problem-size independent. The inclusion of the points of Z as potential SVs supports the model in the regions of input space not covered by the training data but where prior knowledge is given.

As the points of interest $\boldsymbol{x}_p$ are different of the inputs $\boldsymbol{x}$ found in the training set, the methods of [Lázaro et al., 2005b] or [Lázaro et al., 2005a] cannot be used.

In this experiment, the *proposed model* is compared to an *experimental model* trained by (5) on the training set only and a *mixed model* trained by (5) on the training set simply extended with the simulation data. The training of this latter model is close in spirit to the virtual sample approach, where additional data are added to the training set. These models are evaluated on the basis of three indicators: the root mean square error (RMSE) on all experimental data (*RMSE total*), the RMSE on the test set (*RMSE test*) and the maximum absolute error on all experimental data (*MAE*).

Before training, the variables are normalized with respect to their mean and standard deviation. When both experimental and simulation data are available, the simulation data are preferred since they are supposed to cover a wider region of the input space. Thus, the mean and standard deviation are determined on the simulation data for all the models except for the *experimental model*, in which case the training set must be used to determine the normalization parameters. The different hyperparameters are set according to the following heuristics. One goal of the application is to obtain a model that is accurate on both the training and test samples (the training points are part of the performance index *RMSE total*). Thus C is set to a large value ($C = 100$) in order to ensure a good approximation of the training points. Accordingly, ε is set to 0.001 in order to approximate the real data well. The trade-off parameter λ of the proposed method is set to 100, which gives as much weight to both the training data and the prior knowledge. Since all standard deviations of the inputs equal 1 after normalization, the RBF kernel width σ is set to 1.

Two sets of experiments are performed for very low numbers of training samples $N = 6$ and $N = 3$. The results in Table 3 show that both the

Table 3 Errors on the residual gas mass fraction for a training set sizes $N = 6$ and $N = 3$. '–' appears when the result is irrelevant (model mostly constant).

N	Model	RMSE total	RMSE test	MAE
6	1 (experimental model)	6.00	6.84	15.83
	2 (prior model)	4.93	4.86	9.74
	3 (mixed model)	4.88	4.85	9.75
	4 (proposed-model)	**2.15**	**2.44**	**5.94**
3	1 (experimental model)	–	–	–
	2 (prior model)	4.93	4.93	9.74
	3 (mixed model)	4.86	4.89	9.75
	4 (proposed model)	**2.79**	**2.97**	**5.78**

experimental and the *mixed* models cannot yield a better approximation than the *prior model* with so few training data. Moreover, for $N = 3$, the *experimental model* yields a quasi-constant function due to the fact that the model has not enough free parameters (only 3 plus a bias term) and thus cannot model the data. In this case, the RMSE is irrelevant. On the contrary, the *proposed model* do not suffer from this problem and yields good results from very few training samples. Moreover, the performance decreases only slightly when reducing the training set size from 6 to 3. Thus, the proposed method seems to be a promising alternative to obtain a simple black box model with a good accuracy from a limited number of experimental data and a prior simulation model.

7 Conclusion

This paper reviewed the possibilities allowed by adding constraints to the optimization problem for the incorporation of prior knowledge into LPSVR learning. Equality and inequality constraints were studied with the corresponding applications illustrated by examples. Many types of prior knowledge can be taken into account by the proposed method such as particular points with known values, prior knowledge on any derivative either provided by a prior model or available only at some points, bounds on the function or a derivative... This extends and regroups the previous works of [Mangasarian and Wild, 2007] and [Lázaro et al., 2005b], which considered particular forms of prior knowledge. Moreover, a new method for the simultaneous approximation of multiple outputs linked by some prior knowledge has been proposed. This method uses the general framework for the incorporation of prior knowledge by the addition of constraints and thus allows consideration of different types of prior knowledge on single outputs while training on multiple outputs. The methods were applied to nonlinear system identification problems and promising results were obtained on real-life data for the estimation of in-cylinder residual gas fraction in spark ignition

engines. On this application, the addition of virtual samples with derivative values, not handled in [Lázaro et al., 2005b], efficiently improved the model.

A Derivatives of f with the RBF kernel

Considering the RBF kernel $k(\boldsymbol{x}, \boldsymbol{x}_i) = \exp\left(-\|\boldsymbol{x} - \boldsymbol{x}_i\|^2/2\sigma^2\right)$, gives the following derivative of f with respect to x^j, the jth component of $\boldsymbol{x}$,

$$\begin{aligned}\frac{\partial f(\boldsymbol{x})}{\partial x^j} &= \frac{1}{\sigma^2}\sum_{i=1}^{N}\alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)(x_i^j - x^j) \\ &= \frac{1}{\sigma^2}(\boldsymbol{X}_j - \mathbf{1}_N x^j)^T \boldsymbol{D}_{\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}^T)}\,\boldsymbol{\alpha} = \boldsymbol{r}_1(\boldsymbol{x})^T\boldsymbol{\alpha}\ ,\end{aligned} \tag{51}$$

where $\boldsymbol{D}_{\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}^T)} = \operatorname{diag}\left(\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}^T)\right)$ and $\boldsymbol{X}_j$ represents the jth column of $\boldsymbol{X}$.

Looking at the second order derivatives

$$\begin{aligned}\frac{\partial^2 f(\boldsymbol{x})}{\partial {x^j}^2} &= \frac{1}{\sigma^2}\sum_{i=1}^{N}\alpha_i\left[\frac{k(\boldsymbol{x}, \boldsymbol{x}_i)(x_i^j - x^j)^2}{\sigma^2} - k(\boldsymbol{x}, \boldsymbol{x}_i)\right] \\ &= \frac{1}{\sigma^2}\sum_{i=1}^{N}\alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)\left(\frac{(x_i^j - x^j)^2}{\sigma^2} - 1\right)\ ,\end{aligned} \tag{52}$$

gives

$$\nabla^2 f(\boldsymbol{x}) = \frac{1}{\sigma^2}\sum_{j=1}^{d}\sum_{i=1}^{N}\alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i)\left(\frac{(x_i^j - x^j)^2}{\sigma^2} - 1\right)\ , \tag{53}$$

which becomes in matrix form

$$\begin{aligned}\nabla^2 f(\boldsymbol{x}) &= \frac{1}{\sigma^4}\sum_{j=1}^{d}\left[((\boldsymbol{X}_j - \mathbf{1}_N x^j)^2)^T \boldsymbol{D}_{\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}^T)}\,\boldsymbol{\alpha}\right] - \frac{d}{\sigma^2}\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}^T)\boldsymbol{\alpha} \\ &= \left[\frac{1}{\sigma^4}\sum_{j=1}^{d}\left[((\boldsymbol{X}_j - \mathbf{1}_N x^j)^2)^T \boldsymbol{D}_{\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}^T)}\right] - \frac{d}{\sigma^2}\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}^T)\right]\boldsymbol{\alpha} \\ &= \boldsymbol{r}_2(\boldsymbol{x})^T\boldsymbol{\alpha}\ ,\end{aligned} \tag{54}$$

where $(\boldsymbol{X}_j - \mathbf{1_N} x^j)^2$ is a vector with all components equals to the squares of the components of $(\boldsymbol{X}_j - \mathbf{1_N} x^j)$.

To summarize the results, the following parameters can be used to constrain the first order derivative with respect to x^j and the Laplacian:

$$\boldsymbol{r}_1(\boldsymbol{x})^T = \frac{1}{\sigma^2}(\boldsymbol{X}_j - \mathbf{1}_N x^j)^T \boldsymbol{D}_{\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}^T)}\ , \tag{55}$$

$$\boldsymbol{r}_2(\boldsymbol{x})^T = \frac{1}{\sigma^4}\sum_{j=1}^{d}\left[((\boldsymbol{X}_j - \mathbf{1}_N x^j)^2)^T \boldsymbol{D}_{\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}^T)}\right] - \frac{d}{\sigma^2}\boldsymbol{k}(\boldsymbol{x}, \boldsymbol{X}^T)\ . \tag{56}$$

References

[Andrews and Geva, 1999] Andrews, R. and Geva, S. (1999). On the effects of initializing a neural network with prior knowledge. In *Proc. of the Int. Conf. on Neural Information Processing, Perth, Western Australia*, pages 251–256.

[Bennett, 1999] Bennett, K. P. (1999). Combining support vector and mathematical programming methods for classification. In [Schölkopf et al., 1999], pages 307–326.

[Bloch et al., 2007] Bloch, G., Lauer, F., Colin, G., and Chamaillard, Y. (2007). Combining experimental data and physical simulation models in support vector learning. In *Proc. of the 10th Int. Conf. on Engineering Applications of Neural Networks, Thessaloniki, Greece*, pages 284–295.

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press.

[Evgeniou et al., 2000] Evgeniou, T., Pontil, M., and Poggio, T. (2000). Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13:1–50.

[Fung et al., 2002] Fung, G., Mangasarian, O. L., and Shavlik, J. W. (2002). Knowledge-based support vector machine classifiers. In Becker, S., Thrun, S., and Obermayer, K., editors, *NIPS*, pages 521–528, Cambridge, MA, USA. MIT Press.

[Fung et al., 2003] Fung, G., Mangasarian, O. L., and Shavlik, J. W. (2003). Knowledge-based nonlinear kernel classifiers. In Schölkopf, B. and Warmuth, M. K., editors, *COLT*, volume 2777 of *Lecture Notes in Computer Science*, pages 102–113. Springer.

[Giansetti et al., 2007] Giansetti, P., Colin, G., Higelin, P., and Chamaillard, Y. (2007). Residual gas fraction measurement and computation. *International Journal of Engine Research*, 8(4):347–364.

[Hastie et al., 2001] Hastie, T., Tibshirani, R., Friedman, J., et al. (2001). *The elements of statistical learning: data mining, inference, and prediction*. Springer.

[Imagine, 2006] Imagine (2006). Amesim web site. www.amesim.com.

[Joachims, 2002] Joachims, T. (2002). *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms*. Kluwer Academic Publishers.

[Johansen, 1996] Johansen, T. (1996). Identification of non-linear systems using empirical data and prior knowledge-an optimization approach. *Automatica*, 32(3):337–356.

[Lauer and Bloch, 2007] Lauer, F. and Bloch, G. (2007). Incorporating prior knowledge in support vector machines for classification: a review. *to appear in Neurocomputing*.

[Lawson and Hanson, 1995] Lawson, C. L. and Hanson, R. J. (1995). *Solving Least Squares Problems*. Classics in Applied Mathematics. SIAM.

[Lázaro et al., 2005a] Lázaro, M., Pérez-Cruz, F., and Artés-Rodriguez, A. (2005a). Learning a function and its derivative forcing the support vector expansion. *IEEE Signal Processing Letters*, 12:194–197.

[Lázaro et al., 2005b] Lázaro, M., Santamaria, I., Pérez-Cruz, F., and Artés-Rodriguez, A. (2005b). Support vector regression for the simultaneous learning of a multivariate function and its derivatives. *Neurocomputing*, 69:42–61.

[Maclin et al., 2005] Maclin, R., Shavlik, J., Torrey, L., Walker, T., and Wild, E. (2005). Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. *Proc. of the 20th National Conf. on Artificial Intelligence, Pittsburgh, PA, US*.

[Mangasarian, 2000] Mangasarian, O. (2000). Generalized support vector machines. In Smola, A., Bartlett, P., Schölkopf, B., and Schuurmans, D., editors, *Advances in Large Margin Classifiers*, pages 135–146, Cambridge, MA, USA. MIT Press.

[Mangasarian and Musicant, 2002] Mangasarian, O. L. and Musicant, D. R. (2002). Large scale kernel regression via linear programming. *Machine Learning*, 46(1-3):255–269.

[Mangasarian et al., 2004] Mangasarian, O. L., Shavlik, J. W., and Wild, E. W. (2004). Knowledge-based kernel approximation. *Journal of Machine Learning Research*, 5:1127–1141.

[Mangasarian and Wild, 2007] Mangasarian, O. L. and Wild, E. W. (2007). Nonlinear knowledge in kernel approximation. *IEEE Trans. on Neural Networks*, 18:300–306.

[Mattera and Haykin, 1999] Mattera, D. and Haykin, S. (1999). Support vector machines for dynamic reconstruction of a chaotic system. In [Schölkopf et al., 1999], pages 211–241.

[Micchelli and Utreras, 1988] Micchelli, C. and Utreras, F. (1988). Smoothing and interpolation in a convex subset of a hilbert space. *SIAM Journal on Scientific and Statistical Computing*, 9:728.

[Müller et al., 1997] Müller, K., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., and Vapnik, V. (1997). Predicting time series with support vector machines. In *Proc. of the Int. Conf. on Artificial Neural Networks*, pages 999–1004.

[Narendra and Parthasarathy, 1990] Narendra, K. S. and Parthasarathy, K. (1990). Identification and control of dynamical systems using neural networks. *IEEE Trans. on Neural Networks*, 1(1):4–27.

[Poggio and Vetter, 1992] Poggio, T. and Vetter, T. (1992). Recognition and structure from one 2D model view: Observations on prototypes, object classes and symmetries. Technical Report AIM-1347, Massachusetts Institute of Technology, Cambridge, MA, USA.

[Sánchez-Fernández et al., 2004] Sánchez-Fernández, M., De Prado-Cumplido, M., Arenas-García, J., and Pérez-Cruz, F. (2004). SVM multiregression for nonlinear channel estimation in multiple-input multiple-output systems. *IEEE Trans. on Signal Processing*, 52(8):2298–2307.

[Schölkopf et al., 1996] Schölkopf, B., Burges, C., and Vapnik, V. (1996). Incorporating invariances in support vector learning machines. In von der Malsburg, C., von Seelen, W., Vorbrüggen, J. C., and Sendhoff, B., editors, *ICANN*, volume 1112 of *Lecture Notes in Computer Science*, pages 47–52. Springer.

[Schölkopf et al., 1999] Schölkopf, B., Burges, C. J., and Smola, A. J., editors (1999). *Advances in kernel methods: Support vector learning*, Cambridge, MA, USA. MIT Press.

[Smola et al., 1999a] Smola, A. J., Friess, T., and Schölkopf, B. (1999a). Semiparametric support vector and linear programming machines. In *Advances in Neural Information Processing Systems*, volume 11, pages 585–591. MIT Press.

[Smola and Schölkopf, 2004] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

[Smola et al., 1998] Smola, A. J., Schölkopf, B., and Müller, K. R. (1998). The connection between regularization operators and support vector kernels. *Neural Networks*, 11(4):637–649.

[Smola et al., 1999b] Smola, A. J., Schölkopf, B., and Rätsch, G. (1999b). Linear programs for automatic accuracy control in regression. In *Proc. of the 9th Int. Conf. on Artificial Neural Networks, Edinburgh, UK*, volume 2, pages 575–580.

[Söderström and Stoica, 1988] Söderström, T. and Stoica, P. (1988). *System Identification*. Prentice-Hall Inc., Upper Saddle River, NJ, USA.

[Stitson et al., 1999] Stitson, M. O., Gammerman, A., Vapnik, V., Vovk, V., Watkins, C., and Weston, J. (1999). Support vector regression with ANOVA decomposition kernels. In [Schölkopf et al., 1999], pages 285–291.

[Tay and Cao, 2002] Tay, F. and Cao, L. (2002). Modified support vector machines in financial time series forecasting. *Neurocomputing*, 48:847–861.

[Towell and Shavlik, 1994] Towell, G. G. and Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artif. Intell.*, 70(1-2):119–165.

[Vapnik, 1995] Vapnik, V. N. (1995). *The nature of statistical learning theory*. Springer-Verlag, New York, NY, USA.

[Villalobos and Wahba, 1987] Villalobos, M. and Wahba, G. (1987). Inequality-constrained multivariate smoothing splines with application to the estimation of posterior probabilities. *Journal of the American Statistical Association*, 82(397):239–248.

[Weston et al., 2003] Weston, J., Chapelle, O., Elisseeff, A., Scholkopf, B., and Vapnik, V. (2003). Kernel dependency estimation. *Advances in Neural Information Processing Systems*, volume 15, pages 873–880.

[Weston et al., 1999] Weston, J., Gammerman, A., Stitson, M. O., Vapnik, V., Vovk, V., and Watkins, C. (1999). Support vector density estimation. In [Schölkopf et al., 1999], pages 293–305.

[Wu and Srihari, 2004] Wu, X. and Srihari, R. (2004). Incorporating prior knowledge with weighted margin support vector machines. In *Proc. of the tenth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Seattle, WA, USA*, pages 326–333, New York, NY, USA. ACM Press.