Available online at www.sciencedirect.com

ScienceDirect

Neurocomputing 70 (2006) 420–429

NEUROCOMPUTING

www.elsevier.com/locate/neucom

Non-flat function estimation with a multi-scale support vector regression

Danian Zheng*, Jiaxin Wang, Yannan Zhao

State Key Laboratory of Intelligent Technology and Systems, Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

Received 28 June 2005; received in revised form 30 November 2005; accepted 26 December 2005
Communicated by J. Tin-Yau Kwok
Available online 16 May 2006

Abstract

Estimating the non-flat function which comprises both the steep variations and the smooth variations is a hard problem. The results achieved by the common support vector methods like SVR, LPR and LS-SVM are often unsatisfactory, because they cannot avoid underfitting and overfitting simultaneously. This paper takes this problem as a linear regression in a combined feature space which is implicitly defined by a set of translation invariant kernels with different scales, and proposes a multi-scale support vector regression (MS-SVR) method. MS-SVR performs better than SVR, LPR and LS-SVM in the experiments tried.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Non-flat function; Combination of feature spaces; Multi-scale support vector regression; Sparse representation; Loss function

1. Introduction

Since support vector machine (SVM) was applied to solve multi-dimensional function estimation problems by Vapnik (also called ε-support vector regression, ε-SVR) [20], many extensions of SVM for solving the regression problems have been developed recently. The ν-SVR is a modification of the ε-SVR, which automatically minimizes the ε-tube to the data for a priori chosen ν [13]. Then a linear programming regression (ε-LPR and ν-LPR), which avoids the quadratic programming and enforces sparseness of the solution, has been proposed in [17]. And a least squares version of SVM (LS-SVM) was also reported for classification and regression by using the least square loss function instead of the ε-insensitive loss function [19]. Furthermore, some important extensions of LS-SVM like weighted LS-SVM [18] and sparse LS-SVM [1,10] have been also researched. The SVRs with a very general class of loss functions are discussed in [14,16].

There are a lot of non-flat functions which comprise both the steep variations and the smooth variations. It is unsuitable to use the standard SVRs (or LPR, LS-SVM) to estimate these non-flat functions. First, SVR requires the estimated function as "flat" as possible for a good generalization ability [16], which may lead to underfit the steep variations. Second, the commonly used kernel functions may be not appropriate for the characteristics of the function so that underfittings and overfittings occur frequently in the estimated function.

This paper proposes a multi-scale support vector regression (MS-SVR) method for the non-flat function estimation, which is motivated by the density estimation using SVM and a dictionary of several kernel functions [21]. MS-SVR overcomes the two above difficulties with SVR: (1) similarly to LPR, a sparse representation is enforced for good generalization instead of the requirement of a "flat" function; (2) a set of kernels with multi scales (resolutions) are employed instead of a single kernel, so the kernels with small scales and the kernels with large scales can deal with the steep variations and the smooth variations, respectively. Experiments demonstrate the great effectiveness of this method for the non-flat function estimation.

*Corresponding author.
E-mail address: zdn02@mails.tsinghua.edu.cn (D. Zheng).

0925-2312/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2005.12.128

Indeed the use of a committee of multi scale or multiple kernels has been investigated by a number of people in the recent years. Lanchriet et al. described a linear combination of the kernel representations $K = \sum_{r=1}^{m} \mu_r K_r$ and encoded it in SVM for yeast protein functional classifications [8,7]. A common coefficient vector α is trained for the m kernel scales, and this greatly limits the representation ability of the decision function. In order to obtain a richer representation, m independent coefficient vectors α_r are assigned to the m kernel scales, respectively in MS-SVR. Guigue et al. introduced the kernel basis pursuit (KBP) algorithm, which enables us to build a l_1-regularized-multiple-kernel estimator for regression [5]. KBP is more similar (also m coefficient vectors) to MS-SVR. However, KBP is prone to overfit the noisy data, which will be illuminated in our experiments. Rakotomamonjy et al. proposed a method to construct a reproducing wavelet kernel Hilbert spaces for non-parametric regression estimation [12,11]. A wavelet decomposition of N levels is performed on the data, then the detail coefficients at each level are used to train a regressor with wavelet kernels, and then the approximate coefficients at the last level are also used to train a regressor. The final estimation is a simple sum of the $N+1$ regressors. Although these regressors are optimal in the $N+1$ local subspaces, the sum of them may be not optimal. MS-SVR estimates the non-flat functions by a global optimizer. An earlier work of generalized additive models is [6], where many additive models on how to learn a model from multi models were discussed.

The rest of this paper is organized as follows. Section 2 explains the regression problem and reviews some previous support vector methods for regression, such as SVR, LPR, LS-SVM and KBP. Section 3 considers the nonlinear regression as a linear regression in a special feature space which is a combination of multi-feature spaces. Then MS-SVR is proposed in Section 4, and the solutions of MS-SVR with three different loss functions are investigated in Section 5. Selections of the kernel scales and the regularization constant are discussed in Section 6. Experiments that verify the prediction performance of MS-SVR are done in Section 7. Finally, some discussion and conclusions are given in Section 8.

2. Previous support vector methods

The problem of regression is to estimate an unknown real-valued function in the relationship

$$y = r(\mathbf{x}) + n, \tag{1}$$

where $\mathbf{x} \in \mathscr{X} \subseteq \mathbb{R}^d$ is a multivariate input, $y \in \mathscr{Y} \subseteq \mathbb{R}$ a scalar output and n an independent identically distributed random noise. And the estimation is made based on a finite number of observations $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l) \in \mathscr{X} \times \mathscr{Y}$.

From the statistical learning theory, one can know that the best estimation $f(\mathbf{x})$ is the one minimizing the expected risk

$$R(f) = \int L(f(\mathbf{x}), y) p(\mathbf{x}, y)\,\mathrm{d}\mathbf{x}\,\mathrm{d}y, \tag{2}$$

where $L(f(\mathbf{x}), y)$ denotes a chosen loss function, and $p(\mathbf{x}, y)$ a joint probability density function. For example, $r(\mathbf{x}) = \int y p(y|\mathbf{x})\,\mathrm{d}y$ is the estimated result of (2) with the squared loss function $L(f(\mathbf{x}), y) = (f(\mathbf{x}) - y)^2$ [2]. Since the underlying $p(\mathbf{x}, y)$ in (2) is unknown, the learning methods often utilize the following feasible risk [15]:

$$R_{\text{reg}}(f) = T(f) + C R_{\text{emp}}(f) \tag{3}$$

instead to obtain some good functions close to the optimal one $r(\mathbf{x})$. Here, the regularization term $T(f)$ characterizes the model complexity, the empirical risk $R_{\text{emp}}(f) = \sum_{i=1}^{l} L(f(\mathbf{x}_i), y)$ controls the training error, and the constant C determines the trade-off between them.

In the support vector methods, the nonlinear regression problem in the input space $\mathscr{X}$ is considered as a linear one $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b$ in some feature space $\mathscr{F}$ that induced by the nonlinear mapping $\phi : \mathbf{x} \mapsto \phi(\mathbf{x})$. The $w \in \mathscr{F}$ is a normal vector, and $b \in \mathbb{R}$ a bias term.

SVR [20,16] minimizes the primal problem

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l} |f(\mathbf{x}_i) - y_i|_{\varepsilon}, \tag{4}$$

where an ε-insensitive loss function is often used. The first term $\frac{1}{2}\|w\|^2$ requires the function as flat as possible, and the second term penalizes any deviations larger than the precision ε for all the training data. Its Wolfe dual objective function yields a quadratic programming problem, and a sparse solution (fewer support vectors) can be usually obtained due to the ε-insensitive loss function. A modification of SVR is ν-SVR, which can automatically tune the ε-insensitive tube [13].

Linear programming regression (LPR) [17] seeks a $\mathbf{w}$ that can be represented as the smallest combination of training patterns instead of choosing the flattest function, so it minimizes

$$\min \sum_{i=1}^{l} |\alpha_i| + C\sum_{i=1}^{l} |f(\mathbf{x}_i) - y_i|_{\varepsilon}, \tag{5}$$

where $\mathbf{w} = \sum_{i=1}^{l} \alpha_i \phi(\mathbf{x}_i)$. LPR changes the problem from a quadratic programming to a linear programming problem and generates a very sparse solution. And it also has a corresponding ν-LPR [17].

LS-SVM [19] uses the squared loss function and considers a similar problem

$$\min \frac{1}{2}\|\mathbf{w}\|^2 + \frac{1}{2}C\sum_{i=1}^{l} (f(\mathbf{x}_i) - y_i)^2. \tag{6}$$

LS-SVM finds the solution by solving a set of linear equations instead of quadratic programming, while it losses the sparsity property (all training patterns are support vectors). Recently, there are some extensions for LS-SVM, such as sparse LS-SVM [1,10] and weighted LS-SVM [18].

Kernel basis pursuit (KBP) algorithm consists in solving the following problem:

$$\min_{\alpha} \|y - K\alpha\|^2 \quad \text{s.t.} \sum_{r=1}^{m} |\alpha_r| \leqslant t, \tag{7}$$

where the multiple kernel $K = [K_1, \ldots, K_m] \in \mathbb{R}^{l\times ml}$, $\alpha = (\alpha_1, \ldots, \alpha_m) \in \mathbb{R}^{ml}$ and $\alpha_r = (\alpha_{r1}, \ldots, \alpha_{rl})$. And the solution is $f(\mathbf{x}) = \sum_{r=1}^{m} \sum_{i=1}^{l} \alpha_{ri} k_r(\mathbf{x}_i, \mathbf{x})$ [5].

3. Combination of feature spaces

A linear regression is to be found in a feature space. However, different feature spaces have their respective advantages. It is promising to seek a better solution that fuses their advantages in the feature space extended by them. That is to say, $f(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) + b$, where $\mathbf{w}^{\mathrm{T}} = [\mathbf{w}_1^{\mathrm{T}}, \ldots, \mathbf{w}_m^{\mathrm{T}}]$, $\phi(\mathbf{x})^{\mathrm{T}} = [\phi_1(\mathbf{x})^{\mathrm{T}}, \ldots, \phi_m(\mathbf{x})^{\mathrm{T}}]$, and $\phi_1, \ldots, \phi_m$ are m nonlinear mappings.

The nonlinear mapping ϕ is usually implicitly defined via a kernel function $k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{z})$. Thus, choosing a suitable kernel is a crucial problem for a kernel-based learning method to achieve a good performance. In the widely used kernels, RBF kernels are very popular for their universal approximations. We restrict ourself to such a class of translation invariant kernels $k(\mathbf{x}, \mathbf{z}) = k(\|\mathbf{x} - \mathbf{z}\|/\sigma)$. The parameter σ rescales the input data and determines the kernel capacity. For example, using a Gaussian RBF kernel $k(\mathbf{x}, \mathbf{z}) = \exp(-\|\mathbf{x} - \mathbf{z}\|^2/2\sigma^2)$ with a small σ, SVR may approximate any continuous function at a very high accuracy but overfit the noisy data; conversely, a large σ may result in a low accuracy and underfitting.

For a non-flat function, it is hard to find an appropriate σ for SVR to fit both the steep variations and the smooth variations well, even under the help of the trade-off parameter C. And this will be shown in the experiment section.

Since the non-flat function cannot be described properly in any single feature space, it may be a better choice to consider it in the multi-feature spaces, which are induced by a set of kernels with different scales $k(\|\mathbf{x} - \mathbf{z}\|/\sigma_1), \ldots, k(\|\mathbf{x} - \mathbf{z}\|/\sigma_m)$, $\sigma_1 < \cdots < \sigma_m$. Kernels with the smaller scales meet the steep variations in the function, and kernels with the larger scales meet the smooth variations.

4. Multi-scale support vector regression

Using the dual representation, one may write the normal vector $\mathbf{w}_r$ in the feature space $\mathscr{F}_r$ as a linear combination of the mapped training patterns, i.e. $\mathbf{w}_r = \sum_{i=1}^{l} (\alpha_{ri} - \alpha_{ri}^*)\phi_r(\mathbf{x}_i)$, $r = 1, \ldots, m$, and obtain the following kernel expansion of f as:

$$\begin{aligned} f(\mathbf{x}) &= \sum_{r=1}^{m} \mathbf{w}_r \phi_r(\mathbf{x}) + b \\ &= \sum_{r=1}^{m} \sum_{i=1}^{l} (\alpha_{ri} - \alpha_{ri}^*) k\left(\frac{\|\mathbf{x}_i - \mathbf{x}\|}{\sigma_r}\right) + b, \end{aligned} \tag{8}$$

where the dual variables satisfy $\alpha_{ri}, \alpha_{ri}^* \geqslant 0$ and $\alpha_{ri}\alpha_{ri}^* = 0$. Here, two dual variables $\alpha_{ri}, \alpha_{ri}^*$ are used instead of a single one $\alpha'_{ri} \in \mathbb{R}$, so that $\alpha'_{ri} = \alpha_{ri} - \alpha_{ri}^*$ in (8) and $|\alpha'_{ri}| = \alpha_{ri} + \alpha_{ri}^*$ in (9).

The function (8) can be estimated by minimizing the risk (3) like the previous methods. Here, the regularization term $T(f)$ needs to be well chosen first. Since the target to be estimated is a non-flat function, requiring the flatness of f by minimizing the term $\frac{1}{2}\|\mathbf{w}\|^2$ may produce an underfitting result. Generalization theory shows that the fewer the number of support vectors the better generalization can be expected [3]. So we control the number of support vectors of f like LPR and minimize the following problem:

$$\min \sum_{r=1}^{m} C_r \sum_{i=1}^{l} (\alpha_{ri} + \alpha_{ri}^*) + C \sum_{i=1}^{l} L(f(\mathbf{x}_i), y_i), \tag{9}$$

where the constants C_r penalize the non-zero dual variables, and C determines the trade-off between the model complexity and the training error. To avoid overfitting the smooth variations by the kernels with the smaller scales, more penalties are put on them. That is to say, a decreasing penalty sequence $(C_1 > \cdots > C_m)$ should be given for the kernel sequence with the increasing scales $(\sigma_1 < \cdots < \sigma_m)$. In this paper, we choose $C_1 = 1/\sigma_1, \ldots, C_m = 1/\sigma_m$, and call this learning method (8)–(9) a MS-SVR.

5. Optimizations with different loss functions

Given the density model of the noise n, a robust loss function $L(f(\mathbf{x}), y)$ can be designed for it. The following three loss functions $L(f(\mathbf{x}) - y)$ and their corresponding density models [14,16] $P(f(\mathbf{x}) - y) \propto \exp(-L(f(\mathbf{x}) - y))$ are usually used.

5.1. Vapnik's ε-insensitive loss function

The ε-insensitive loss function only penalizes any training errors larger than ε,

$$L(x) = \begin{cases} |x| - \varepsilon & \text{if } |x| > \varepsilon, \\ 0 & \text{if} |x| \leqslant \varepsilon, \end{cases} \tag{10}$$

where $x = f(\mathbf{x}) - y$. Fig. 1(a) shows it and its density model [14].

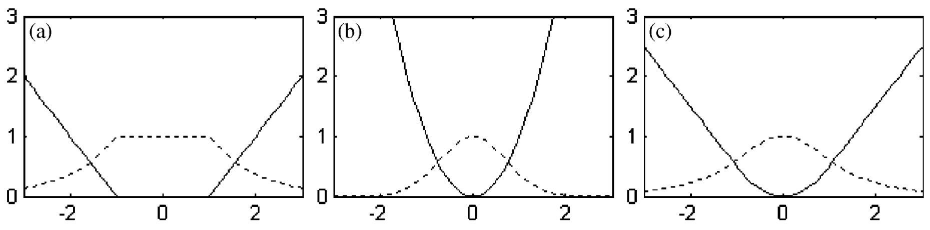

Fig. 1. Loss functions (solid lines) and their density models (dotted lines): (a) Vapnik's ε-insensitive loss function (ε = 1), (b) quadratic loss function and (c) Huber's robust loss function (ε = 1).

With the loss function (10), MS-SVR solves the following optimization problem:

$$\min \quad \sum_{r=1}^{m} C_r \sum_{i=1}^{l} (\alpha_{ri} + \alpha_{ri}^*) + C \sum_{i=1}^{l} (\xi_i + \xi_i^*)$$

$$\text{s.t.} \quad \begin{cases} y_i - \sum_{r=1}^{m}\sum_{j=1}^{l} (\alpha_{rj} - \alpha_{rj}^*) k_{rij} - b \leqslant \varepsilon + \xi_i, \\ \sum_{r=1}^{m}\sum_{j=1}^{l} (\alpha_{rj} - \alpha_{rj}^*) k_{rij} + b - y_i \leqslant \varepsilon + \xi_i^*, \\ \alpha_{ri}, \alpha_{ri}^*, \xi_i, \xi_i^* \geqslant 0, \end{cases} \tag{11}$$

where ξ_i, ξ_i^* denote the slack variables, and the kernel value $k_{rij} = k(\|\mathbf{x}_i - \mathbf{x}_j\|/\sigma_r)$. Here, the constraints $\alpha_{ri}\alpha_{ri}^* = 0$ are guaranteed implicitly in the objective function of (11). If $\alpha_{ri} > 0, \alpha_{ri}^* > 0$ and $\alpha_{ri} \geqslant \alpha_{ri}^*$ (similar for $\alpha_{ri} < \alpha_{ri}^*$) arise in the current solution during the iterations, minimizing the objective function will result in a better solution $\alpha_{ri}' = \alpha_{ri} - \alpha_{ri}^*, \alpha_{ri}^{*\prime} = 0$. Note that $\alpha_{ri}' - \alpha_{ri}^{*\prime} = \alpha_{ri} - \alpha_{ri}^*$ and the other constraints in (11) are still satisfied. So the constraints $\alpha_{ri}\alpha_{ri}^* = 0$ are redundant and not required in (11).

This linear programming problem can be written in matrix form

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{s} \\ \text{s.t.} \quad & A\mathbf{s} \geqslant \mathbf{b}, \mathbf{s} \geqslant [\vec{0}, \ldots, \vec{0}, \vec{0}, \ldots, \vec{0}, \vec{0}, \vec{0}, -\infty] \\ & \in \mathbb{R}^{2(m+1)l+1}, \end{aligned}$$

where

$$\mathbf{s} = [\vec{\alpha}_1, \ldots, \vec{\alpha}_m, \vec{\alpha}_1^*, \ldots, \vec{\alpha}_m^*, \vec{\xi}, \vec{\xi}^*, b] \in \mathbb{R}^{2(m+1)l+1},$$

$$\mathbf{c} = [C_1\vec{1}, \ldots, C_m\vec{1}, C_1\vec{1}, \ldots, C_m\vec{1}, C\vec{1}, C\vec{1}, 0] \in \mathbb{R}^{2(m+1)l+1},$$

$$A = \begin{bmatrix} -K_1 & \cdots & -K_m & K_1 & \cdots & K_m & -I & 0 & -\vec{1} \\ K_1 & \cdots & K_m & -K_1 & \cdots & -K_m & 0 & -I & \vec{1} \end{bmatrix},$$

$$\mathbf{b} = [\varepsilon\vec{1} - \vec{y}, \varepsilon\vec{1} + \vec{y}] \in \mathbb{R}^{2l}. \tag{12}$$

Here, $\vec{\alpha}_r$, $\vec{\alpha}_r^*$, $\vec{\xi}$, $\vec{\xi}^*$, $\vec{y}$, $\vec{0}$ and $\vec{1}$ are all $l \times 1$ column vectors, and $K_r = (k_{rij})_{l \times l}$, I and 0 are all $l \times l$ matrixes, $r = 1, \ldots, m$. The solution $\mathbf{s}$ can be found by the Simplex method or the interior-point method.

This linear programming problem in $(2m+2)l+1$ variables and $(2m+4)l$ constraints can be solved in $O(4ml^2)$ space and $O((2m+4)l)$ time when m is fixed. The algorithm with ε-insensitive loss function only requires linear programming and may yield a sparse but biased solution.

5.2. *Quadratic loss function*

The quadratic loss function (13) corresponds to the density model of Gaussian noise, and Fig. 1(b) show them [14]

$$L(x) = x^2. \tag{13}$$

Let us consider MS-SVR with the loss function (13). The problem can be defined as follows:

$$\min \sum_{r=1}^{m} C_r \sum_{i=1}^{l} (\alpha_{ri} + \alpha_{ri}^*) + \frac{C}{2} \sum_{i=1}^{l} e_i^2, \tag{14}$$

where $e_i = \sum_{r=1}^{m} \sum_{j=1}^{l} (\alpha_{rj} - \alpha_{rj}^*) k_{rij} + b - y_i$ denote the output errors, and $\alpha_{ri}, \alpha_{ri}^* \geqslant 0$. The constraints $\alpha_{ri}\alpha_{ri}^* = 0$ are guaranteed implicitly. And the Lagrangian for this problem is

$$\begin{aligned} \min\ & L(\alpha, \alpha^*, b, e) \\ &= \sum_{r=1}^{m} C_r \sum_{i=1}^{l} (\alpha_{ri} + \alpha_{ri}^*) + \frac{C}{2} \sum_{i=1}^{l} e_i^2 \\ &\quad + \sum_{i=1}^{l} \beta_i \left(\sum_{r=1}^{m} \sum_{j=1}^{l} (\alpha_{rj} - \alpha_{rj}^*) k_{rij} + b - y_i - e_i \right) \\ &\quad - \sum_{r=1}^{m} \sum_{i=1}^{l} (\gamma_{ri}\alpha_{ri} + \gamma_{ri}^*\alpha_{ri}^*), \end{aligned} \tag{15}$$

where $\beta_i, \gamma_{ri}, \gamma_{ri}^* \geqslant 0$ are the Lagrange multipliers. Differentiating with respect to $\vec{\alpha}_r$, $\vec{\alpha}_r^*$, b and $\vec{e}$ and imposing stationarity yield

$$\frac{\partial L}{\partial \vec{\alpha}_r} = C_r\vec{1} + K_r\vec{\beta} - \vec{\gamma}_r = \vec{0},$$

$$\frac{\partial L}{\partial \vec{\alpha}_r^*} = C_r\vec{1} - K_r\vec{\beta} - \vec{\gamma}_r^* = \vec{0},$$

$$\frac{\partial L}{\partial b} = \vec{\beta}^{\mathrm{T}}\vec{1} = 0,$$

$$\frac{\partial L}{\partial \vec{e}} = C\vec{e} - \vec{\beta} = \vec{0}. \tag{16}$$

Resubstitute these relations back into (15), and we obtain the dual problem

$$\begin{aligned} \max \quad & -\frac{1}{2C}\vec{\beta}^{\mathrm{T}}\vec{\beta} - \vec{y}^{\mathrm{T}}\vec{\beta} \\ \text{s.t.} \quad & -C_r\vec{1} \leqslant K_r\vec{\beta} \leqslant C_r\vec{1}, \quad r = 1,\ldots,m, \\ & \vec{\beta}^{\mathrm{T}}\vec{1} = 0, \end{aligned} \tag{17}$$

which can be solved by the quadratic programming method.

The first two equations in (16) imply that non-zero coefficients α_{ri} (α^*_{ri}) can only occur when $K^{\mathrm{T}}_{ri}\vec{\beta} = -C_r$ ($K^{\mathrm{T}}_{ri}\vec{\beta} = C_r$), where K_{ri} denotes the ith column of the kernel matrix K_r. Let $sv_r = \{i : |K^{\mathrm{T}}_{ri}\vec{\beta}| = C_r\}$ be a subindex set corresponding to those support vectors, K^{sv}_r be a matrix with all columns $K_{ri}, i \in sv_r$, and α^{sv}_r be a vector with elements $\alpha_{ri} - \alpha^*_{ri}, i \in sv_r$. It should point out that sv_r will be a very small portion of the whole set $\{1,\ldots,l\}$ because of enforcing sparsity in (14). From the relations $\sum_{r=1}^m K^{sv}_r\vec{\alpha}^{sv}_r + b\vec{1} - \vec{y} - \vec{e} = \vec{0}$ and $C\vec{e} - \vec{\beta} = 0$, we can derive the final solution $\mathbf{s}$ by

$$A\mathbf{s} = \vec{y} + \frac{1}{C}\vec{\beta},$$

$$\mathbf{s} = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}\left(\vec{y} + \frac{1}{C}\vec{\beta}\right), \tag{18}$$

where $A = [K^{sv}_1 \ \cdots \ K^{sv}_m \ \vec{1}]$, and $\mathbf{s} = [\vec{\alpha}^{sv}_1,\ldots,\vec{\alpha}^{sv}_m, b]$. In numerical solution, some small changes $|K^{\mathrm{T}}_{ri}\vec{\beta}| \geqslant C_r - 10^{-4}$ and $A^{\mathrm{T}}A + 10^{-4}I$ can be done in finding sv_r and $\mathbf{s}$.

A post-processing (18) is required for the final solution of (14), because $\alpha_{ri}, \alpha^*_{ri}$ are not the Lagrange multipliers and not solved directly. The quadratic programming (17) in l variables and $2ml + 1$ constraints can be solved in $O((2m+3)l)$ space and $O(nml^2)$ time for n iterations, and the post-processing in less than $O(2l^2)$ space and $O((vl)^3)$ time $(0 < v < 1)$. The algorithm with quadratic loss function can yield a sparse and unbiased solution, while it requires quadratic programming and is not robust enough to the long tail noise.

5.3. *Huber's robust loss function*

The Huber's robust loss function in (19) is a mixture of ε-insensitive loss function and quadratic loss function. Therefore, this loss function does not introduce additional bias. Fig. 1(c) shows it and its density model [14]

$$L(x) = \begin{cases} |x| - \frac{\varepsilon}{2} & \text{if } |x| > \varepsilon, \\ \frac{1}{2\varepsilon}x^2 & \text{if } |x| \leqslant \varepsilon. \end{cases} \tag{19}$$

Similarly to the two cases above, MS-SVR with the loss function (19) optimizes the problem

$$\min \quad \sum_{r=1}^m C_r \sum_{i=1}^l (\alpha_{ri} + \alpha^*_{ri}) + C\sum_{i=1}^l (L(\xi_i) + L(\xi^*_i))$$

$$\text{s.t.} \quad \begin{cases} y_i - \sum_{r=1}^m\sum_{j=1}^l (\alpha_{rj} - \alpha^*_{rj})k_{rij} - b \leqslant \xi_i, \\ \sum_{r=1}^m\sum_{j=1}^l (\alpha_{rj} - \alpha^*_{rj})k_{rij} + b - y_i \leqslant \xi^*_i, \\ \alpha_{ri}, \alpha^*_{ri}, \xi_i, \xi^*_i \geqslant 0, \end{cases} \tag{20}$$

where ξ_i, ξ^*_i are the slack variables. The constraints $\alpha_{ri}\alpha^*_{ri} = 0$ are guaranteed implicitly. From the Lagrangian of this problem, a dual problem similar to the one in the last subsection will be deduced

$$\begin{aligned} \max \quad & -\frac{\varepsilon}{2C}(\vec{\beta}^{\mathrm{T}}\vec{\beta} + \vec{\beta}^{*\mathrm{T}}\vec{\beta}^*) + \vec{y}^{\mathrm{T}}(\vec{\beta} - \vec{\beta}^*) \\ \text{s.t.} \quad & -C_r\vec{1} \leqslant K_r(\vec{\beta} - \vec{\beta}^*) \leqslant C_r\vec{1}, \quad r = 1,\ldots,m \\ & (\vec{\beta} - \vec{\beta}^*)^{\mathrm{T}}\vec{1} = 0. \end{aligned} \tag{21}$$

Note that $\beta_i\beta^*_i = 0$ and $\vec{\beta}^{\mathrm{T}}\vec{\beta} + \vec{\beta}^{*\mathrm{T}}\vec{\beta}^*$ can be substituted by $(\vec{\beta} - \vec{\beta}^*)^{\mathrm{T}}(\vec{\beta} - \vec{\beta}^*)$. So the problem (21) is also solved by the quadratic programming method. With $sv_r = \{i : |K_r(\vec{\beta} - \vec{\beta}^*)| = C_r\}$, the final solution $\mathbf{s}$ is stated as follows:

$$A\mathbf{s} = \vec{y} - \frac{\varepsilon}{C}(\vec{\beta} - \vec{\beta}^*),$$

$$\mathbf{s} = (A^{\mathrm{T}}A)^{-1}A^{\mathrm{T}}\left(\vec{y} - \frac{\varepsilon}{C}(\vec{\beta} - \vec{\beta}^*)\right), \tag{22}$$

where $A = [K^{sv}_1 \ \cdots \ K^{sv}_m \ \vec{1}]$ and $\mathbf{s} = [\vec{\alpha}^{sv}_1,\ldots,\vec{\alpha}^{sv}_m, b]$.

The algorithm with Huber's loss function can yield a sparse, unbiased and robust solution, while it requires quadratic programming too.

6. Practical parameter selections

The m kernel scales can be set as a geometric sequence $\sigma_1, \sigma_1 q, \ldots, \sigma_1 q^{m-1}$ or an arithmetic sequence $\sigma_1, \sigma_1 + d, \ldots, \sigma_1 + (m-1)d$, and the former is employed in this paper. Since the penalty coefficients C_i for different scales are determined by $C_i = 1/\sigma_i$, there are only four parameters m, σ_1, σ_m and C to be selected in MS-SVR with quadratic loss function (MS-SVR(Q)).

Suppose that the number of kernel scales m is given beforehand. Generally, a small $m = 2, 3$ may be sufficient for dealing with some easy problems and a large $m = 4, 5, \ldots$ may be necessary for some hard problems.

For the simplicity and speediness, a line search strategy is considered for the selections of σ_1, σ_m and C. The selection procedure is described as below:

1. initialize C and σ_m by two roughly evaluated values $\hat{C}$ and $\hat{\sigma}_m$;
2. let $C = \hat{C}$ and $\sigma_m = \hat{\sigma}_m$ be fixed, and search an optimal $\sigma_1 = \sigma^*_1$ to minimize a predefined criterion (generalization error, cross-validation error, test error, etc.);
3. let $C = \hat{C}$ and $\sigma_1 = \sigma^*_1$ be fixed, and search an optimal $\sigma_m = \sigma^*_m$;

4. let $\sigma_1 = \sigma_1^*$ and $\sigma_m = \sigma_m^*$ be fixed, and search an optimal $C = C^*$.

The value of C can be related to the range on target values y_i of the training data and the noise variance σ^2_{noise}. A result of qualitative analysis is $\hat{C} \propto \max(|\bar{y} + 3\sigma_y|, |\bar{y} - 3\sigma_y|)/l\sigma^2_{\text{noise}}$, where $\bar{y}$ and σ_y are the mean and the standard deviation of y. The values of σ_1 and σ_m can be related to the range and the dimension of the inputs $\mathbf{x}_i$, but they are tuned according to the experimental results: (1) σ_1 moves from smaller to larger values, and σ_m moves in reverse order; (2) the numbers of support vectors corresponding to them, SV(σ_1) and SV(σ_m), increase from zero to non-zero. The initial value $\hat{\sigma}_m$ should make SV($\hat{\sigma}_m$) not too small.

The selection procedure can be carried out twice, and at the second time, the optimal values in step 4 are used to initialize the values in step 1. However, good results have usually been achieved after one time execution.

For MS-SVR with the ε-insensitive loss function (MS-SVR(E)) or the Huber's loss function (MS-SVR(H)), the parameter ε is also to be selected. It is well known that the value of ε should be proportional to the noise level $\hat{\sigma}_{\text{noise}}$. Based on the empirical tuning, a practical prescription $\varepsilon = 3\hat{\sigma}_{\text{noise}}\sqrt{\ln l/l}$ was suggested in [2]. The noise variance can be readily estimated from the squared sum of residuals of the training data. In this paper, it is estimated from the training result of MS-SVR with the quadratic loss function, $\hat{\sigma}^2_{\text{noise}} = (1/(l-1))\sum_{i=1}^{l}(f(\mathbf{x}_i) - y_i)^2$.

7. Experiments

To demonstrate the practicability and the good performance of MS-SVR for non-flat function estimations, we applied it to some simulated data and real data. The Gaussian kernels were used in our experiments.

7.1. Example 1: mixture function

A simple mixture of two Gaussian distributions

$$r(x) = \frac{1}{\sqrt{2\pi} \times 0.3}\exp\left(-\frac{(x-2)^2}{2 \times 0.3^2}\right) + \frac{1}{\sqrt{2\pi} \times 1.2}\exp\left(-\frac{(x-7)^2}{2 \times 1.2^2}\right)$$

was studied. We generated 30 training sets of size $l = 100$ by an additive noise process $y_i = r(x_i) + n_i$, where the inputs x_i were uniformly sampled from the domain [0,10] and the noise $n_i \sim \mathcal{N}(0, 0.05^2)$.

A MS-SVR(Q) with $m = 3$ kernel scales was trained for this easy problem. Tables 1 and 2 show the parameter selection procedure that begins from the two rough values $\hat{C} = 20$ and $\hat{\sigma}_m = 2.0$. The averaged root mean square error, RMSE $= ((1/l)\sum_{i=1}^{l}(f(x_i) - r(x_i))^2)^{1/2}$, between the estimated functions $f(x)$ and the true function $r(x)$ on five random training sets was used to measure the optimality of the parameters. Table 1 gives us some characteristics of the appropriate kernel scales: (1) the support vectors are well assigned to all the kernel scales, i.e. make

Table 1
Selection procedure of σ_1 and σ_m for Example 1 while fixing $C = 20$

σ_1	σ_3	SV(σ_1)	SV(σ_2)	SV(σ_3)	SV	RMSE
0.1	2.0	0.2	8.6	3.2	12.0	0.0479
0.2	2.0	4.8	6.8	3.4	15.0	0.0314
0.3	2.0	2.6	4.6	3.4	10.6	0.0182
0.4	2.0	4.4	3.6	3.2	11.2	0.0382
0.5	2.0	6.2	2.4	1.8	10.4	0.0578
0.3	1.8	2.8	4.4	3.6	10.8	0.0182
0.3	1.6	2.6	3.4	3.8	9.8	0.0182
0.3	1.4	2.6	2.8	3.4	8.8	0.0179
0.3	1.2	2.6	2.4	4.0	9.0	0.0176
0.3	1.0	2.4	1.8	5.6	9.8	0.0187
0.3	0.8	2.6	2.2	6.2	11.0	0.0199

Table 2
Selection procedure of C for Example 1 while fixing $\sigma_1 = 0.3$ and $\sigma_m = 1.2$

C	SV(σ_1)	SV(σ_2)	SV(σ_3)	SV	RMSE
10	1.8	2.0	2.8	6.6	0.0260
20	2.6	2.4	4.0	9.0	0.0176
25	3.8	2.6	3.4	9.8	0.0167
30	4.2	2.8	4.0	11.0	0.0164
35	5.6	2.4	3.8	11.8	0.0165
40	6.4	2.0	4.0	12.4	0.0167

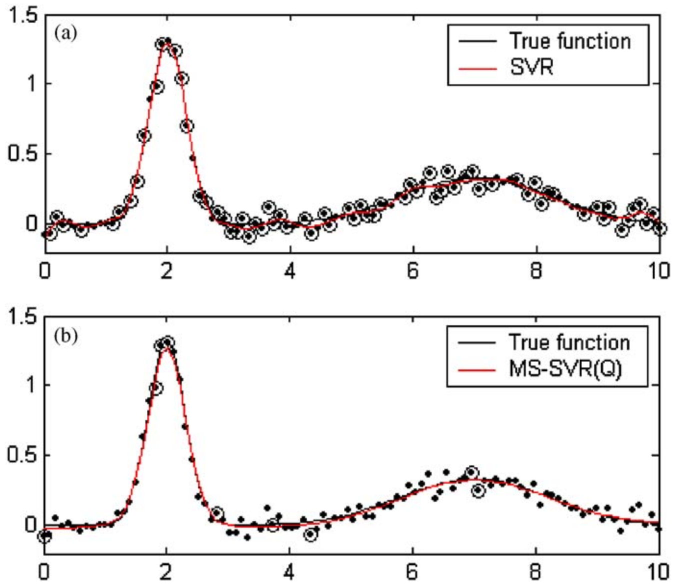

Fig. 2. Estimating the mixture function: (a) SVR and (b) MS-SVR(Q).

Table 3
The optimal parameters selected for the seven methods on Example 1

	SVR	LPR	LS-SVM	KBP	MS-SVR(E)	MS-SVR(Q)	MS-SVR(H)
ε	0.0296	0.0296	—	—	0.0296	—	0.0296
σ	0.5	0.4	0.4			$m=3, \sigma_1=0.3, \sigma_m=1.2$	
C	15	25	30	15	5	30	1

Table 4
The averaged experimental results over 30 training sets for Example 1

	SV(σ_1)	SV(σ_2)	SV(σ_3)	SV	RMSE
SVR	—	—	—	63.9	0.0274 ± 0.0041
LPR	—	—	—	22.3	0.0268 ± 0.0037
LS-SVM	—	—	—	100.0	0.0250 ± 0.0034
KBP	10.3	1.7	2.7	14.7	0.0177 ± 0.0032
MS-SVR(E)	8.9	3.0	3.9	15.8	0.0202 ± 0.0035
MS-SVR(Q)	6.1	3.3	4.0	13.4	0.0171 ± 0.0031
MS-SVR(H)	3.8	3.1	4.0	10.9	0.0169 ± 0.0031

SV(σ_1), …, SV(σ_m) look as uniform as possible; (2) the total number of support vectors (SV) is small; (3) the generalization error, cross-validation error or test error is minimized.

The training results of SVR and MS-SVR(Q) are given in Fig. 2, where circle points denote support vectors. It is obvious that SVR overfits the noisy data. From Fig. 2(b), the evaluated noise level is $\hat{\sigma}_{\text{noise}} = 0.046$ (true 0.050), and a consequent parameter is $\varepsilon = 0.0296$. Then there is only one parameter C to be selected in MS-SVR(E) and MS-SVR(H). Table 3 specifies all the parameters selected for SVR, LPR, LS-SVM, KBP and MS-SVRs via using such a line search strategy.

We tested the performance of these methods on this simple problem, and reported the averaged results over 30 training sets in Table 4. The methods with multi-kernel scales do much better than those with a single kernel scale, and furthermore they need fewer support vectors. KBP can be regarded as a special case of MS-SVR(Q) with $C_1 = \cdots = C_m = 1$ and $b = 0$. Because the equivalent penalties are given to different kernel scales, more support vectors will be assigned to the smaller kernel scales. KBP is quite prone to overfit the noisy data.

7.2. Example 2: oscillating function

The second example is a highly oscillating function $r(x) = \sin(2\pi(0.35 \times 10 + 1)/(0.35x + 1))$ in the fixed domain $x \in [0,10]$. We generated 30 training sets of size $l = 100$ from this function by adding an independent Gaussian noise $n_i \sim \mathcal{N}(0, 0.1^2)$ too.

Since it is a hard problem, a MS-SVR(Q) with $m = 4$ kernel scales was first trained to estimate it. The parameter selection procedure was performed on five random training sets again, and it began from the two rough values $\hat{C} = 30$ and $\hat{\sigma}_m = 3.0$. And the optimal parameters minimized the averaged RMSE between the five estimated functions and the true function. Finally, we obtained the optimal parameters for MS-SVR(Q) $m = 4, \sigma_1 = 0.15, \sigma_m = 1.4$ and $C = 60$.

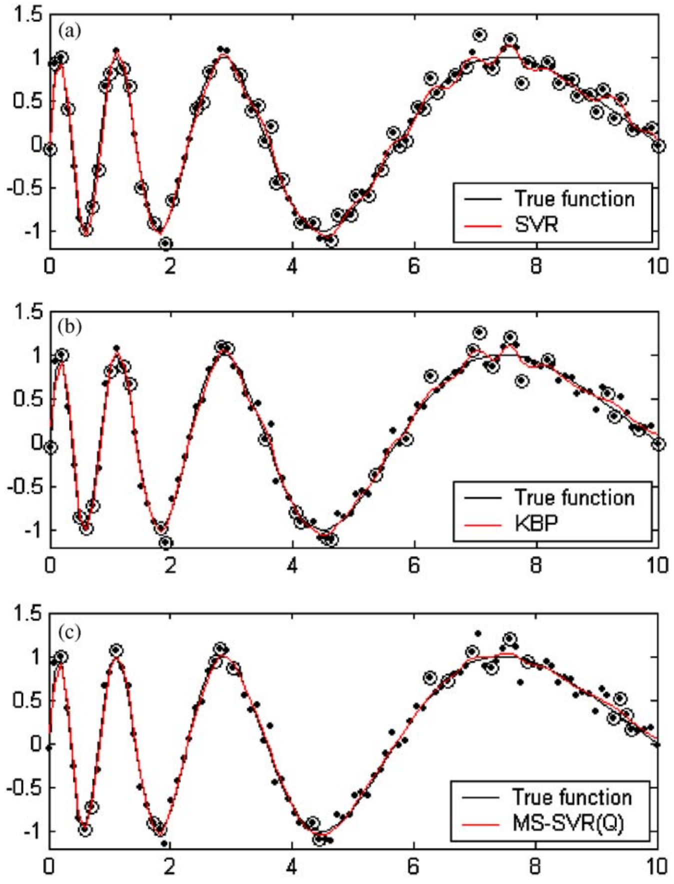

Fig. 3. Estimating the oscillating function: (a) SVR, (b) KBP and (c) MS-SVR(Q).

Fig. 3 shows the training results of SVR, KBP and MS-SVR(Q) on a random training set. Both SVR and KBP overfit the noisy data, but MS-SVR(Q) does better. The noise level estimated by MS-SVR(Q) is $\hat{\sigma}_{\text{noise}} = 0.1037$ (true 0.10), and a consequent parameter is $\varepsilon = 0.0668$, which was used in other methods. All the parameters selected for this problem are specified in Table 5.

Table 5
The optimal parameters selected for the seven methods on Example 2

	SVR	LPR	LS-SVM	KBP	MS-SVR(E)	MS-SVR(Q)	MS-SVR(H)
ε	0.0668	0.0668	—	—	0.0668	—	0.0668
σ	0.35	0.30	0.30		$m=4$, $\sigma_1=0.15$, $\sigma_m=1.4$		
C	30	70	75	15	10	60	5

Table 6
The averaged experimental results over 30 training sets for Example 2

	SV(σ_1)	SV(σ_2)	SV(σ_3)	SV(σ_4)	SV	RMSE
SVR	—	—	—	—	61.5	0.0637 ± 0.0100
LPR	—	—	—	—	33.6	0.0634 ± 0.0098
LS-SVM	—	—	—	—	100.0	0.0613 ± 0.0075
KBP	21.5	7.0	5.4	1.6	35.5	0.0608 ± 0.0070
MS-SVR(E)	10.5	8.1	4.6	3.8	27.0	0.0581 ± 0.0087
MS-SVR(Q)	8.8	8.3	4.6	3.8	25.4	0.0515 ± 0.0072
MS-SVR(H)	11.9	8.4	4.2	3.9	28.4	0.0516 ± 0.0073

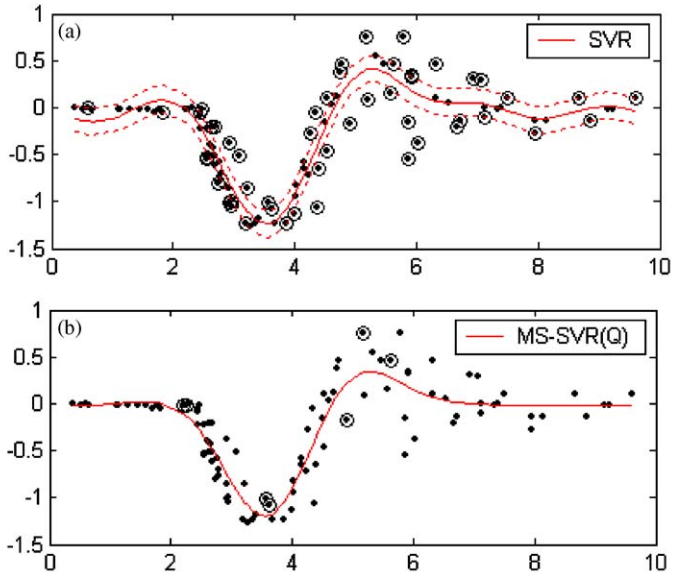

Fig. 4. Estimating the motorcycle data: (a) SVR and (b) MS-SVR(Q).

We tested MS-SVRs and the previous methods on this hard problem, and reported averaged results over 30 training sets in Table 6. As one would expect, the methods with multi-kernel scales are more competent for fitting the non-flat function than those with a single kernel scale. In KBP, SV(σ_1) is much larger than SV(σ_2), SV(σ_3) and SV(σ_4), which leads to the overfitting results.

7.3. Example 3: motorcycle data

The motorcycle data[1] is a classic non-stationary dataset that used to demonstrate the success of non-stationary models. It consists of a sequence of accelerometer readings through time following a simulated motorcycle crash during an experiment to determine the efficacy of crash-helmets. There are a total of 133 samples in this dataset. And these samples were preprocessed $x_i' = x_i/6$ and $y_i' = y_i/100$ so that the ranges on the inputs and targets are similar to Example 1 and 2.

For this real problem, MS-SVR(Q) used $m = 3$ kernel scales. The inputs were resorted into increasing order and then were partitioned into four equal-size disjoint sets by uniformly sampling. The four-fold cross-validation error was used to measure the optimality of the parameters. The parameter selection procedure began with the two rough values $\hat{C} = 10$ and $\hat{\sigma}_m = 3.0$, and arrived at the optimal parameters $m = 3, \sigma_1 = 0.7, \sigma_m = 2.8$ and $C = 5$.

Fig. 4 plots the training results of SVR and MS-SVR(Q). SVR misfits the beginning of the training data, whereas MS-SVR(Q) produces some very smooth results. Methods with the ε-insensitive loss functions, such as SVR, LPR, and MS-SVR(E), are biased estimators. The noise in the motorcycle data is heteroscedastic, and the noise level at the beginning phase is very slight, which results in the biasness of SVR. The noise level estimated by MS-SVR(Q) is $\hat{\sigma}_{\text{noise}} = 0.2230$, and a consequent $\varepsilon = 0.1436$ was used for other methods. Parameter specifications are given in Table 7.

We reported the cross-validation results of all the methods in Table 8, where RMSEs were evaluated between the predictions $f(x_i)$ and the targets y_i on the validation sets. Methods with multi-kernel scales perform slightly better than others, and their prediction functions need fewer support vectors.

8. Discussion and conclusions

A MS-SVR method has been proposed and applied to the non-flat function estimation problem in this paper. Like most support vector methods (SVR, LPR and LS-SVM), MS-SVR finds a good function by minimizing

[1]Available at "http://theoval.cmp.uea.ac.uk/~gcc/matlab/default.html#bench-marks", in file "hkrr_demo.m".

Table 7
The optimal parameters selected for the seven methods on Example 3

	SVR	LPR	LS-SVM	KBP	MS-SVR(E)	MS-SVR(Q)	MS-SVR(H)
ε	0.1436	0.1436	—	—	0.1436	—	0.1436
σ	1.0	1.0	1.1			$m=3, \sigma_1=0.7, \sigma_m=2.8$	
C	10	10	15	5	5	5	1

Table 8
The results of four-fold cross-validation for Example 3

	SV(σ_1)	SV(σ_2)	SV(σ_3)	SV	RMSE
SVR	—	—	—	49.3	0.2334
LPR	—	—	—	12.0	0.2343
LS-SVM	—	—	—	100.0	0.2330
KBP	9.0	0.0	0.0	9.0	0.2324
MS-SVR(E)	7.25	0.75	0.0	8.0	0.2322
MS-SVR(Q)	5.75	0.75	1.0	7.5	0.2329
MS-SVR(H)	6.75	0.75	1.5	9.0	0.2329

the framework (3), which controls the model complexity and the training error simultaneously. Unlike them, MS-SVR considers the regression problem in some multi-scale feature spaces instead of a single feature space. In this sense, it can be also viewed as a multi-resolution analysis like wavelet transform.

The solutions of MS-SVRs with three different loss functions have been discussed in Section 5. These training algorithms may deal with small size problems well, but not suitable for large size problems. To process a large number of points, some fast training algorithms are still to be found.

The experiments in Section 7 have demonstrated that MS-SVR is more robust and effective than SVR, LPR and LS-SVM in estimating the non-flat functions. It may be considerable to apply MS-SVR to some applications like financial data fitting, time series prediction [9], image denoising or compression [4].

References

[1] G.C. Cawley, N.L.C. Talbot, Improved sparse least-squares support vector machines, Neurocomputing 48 (2002) 1025–1031.

[2] V. Cherkassky, Y.Q. Ma, Practical selection of SVM parameters and noise estimation for SVM regression, Neural Networks 17 (2004) 113–126.

[3] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, Cambridge, MA, 2000.

[4] S. Ernesto, G. Adolf, M. Danny, Support vector machines and quad-trees applied to image compression, Proceedings of the Sixth Noric Signal Processing Symposium, Espoo, Finland, 2004, pp. 105–108.

[5] V. Guigue, A. Rakotomamonjy, S. Canu, Kernel Basis Pursuit, Proceedings of the 16th European Conference on Machine Learning, Porto, 2005.

[6] T. Hastie, R. Tibshirani, Generalized Additive Models, Chapman & Hall, New York, 1990.

[7] G.R.G. Lanckriet, N. Cristianini, P. Bartlett, L.E. Ghaoui, M.I. Jordan, Learning the Kernel matrix with semidefinite programming, J. Mach. Learning Res. 5 (2004) 27–72.

[8] G.R.G. Lanckriet, M. Deng, N. Cristianini, M.I. Jordan, W.S. Noble, Kernel-based data fusion and its application to protein function prediction in yeast, Pacific Symposium on Biocomputing, 2004, pp. 300–311.

[9] K.R. Müller, A.J. Smola, G. Rätsch, B. Schölkopf, J. Kohlmogen, V. Vapnik, Predicting time series with support vector machines, in: Advances in Kernel Methods—Support Vector Learning, Cambridge, 1999, pp. 243–254.

[10] K. Pelckmans, J.A.K. Suykens, B.D. Moor, Building sparse representations and structure determination on LS-SVM substrates, Neurocomputing 64 (2005) 137–159.

[11] A. Rakotomamonjy, S. Canu, Frame, reproducing kernel, regularization and learning, J. Mach. Learning Res. 6 (2005) 1485–1515.

[12] A. Rakotomamonjy, X. Mary, S. Canu, Non-parametric regression with wavelet kernels, Appl. Stochastic Models Business Industry 21 (2005) 153–163.

[13] B. Schölkopf, P. Bartlett, A. Smola, R. Williamson, Support vector regression with automatic accuracy control, Proceedings of the Eighth International Conference on Artificial Neural Networks, Berlin, 1998, pp. 111–116.

[14] A. Smola, Regression estimation with support vector learning machines, Master's Thesis, Technische University München, 1996. Available at ⟨http:// www.kernel-machines.org/publications.html⟩.

[15] A. Smola, T.T. Frieß, B. Schölkopf, Semiparametric support vector and linear programming machines, NeuroCOLT2 Technical Report Series NC2-TR-1998-024, Berlin, Germany, 1998.

[16] A. Smola, B. Schölkopf, A tutorial on support vector regression, Stat. Comput. 14 (2004) 199–222.

[17] A. Smola, B. Schölkopf, G. Rätsch, Linear programs for automatic accuracy control in regression, Proceedings of the Ninth International Conference on Artificial Neural Networks, London, 1999, pp. 575–580.

[18] J.A.K. Suykens, J.D. Brabanter, L. Lukas, J. Vandewalle, Weighted least squares support vector machines: robustness and sparse approximation, Neurocomputing 48 (2002) 85–105.

[19] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (3) (1999) 293–300.

[20] V. Vapnik, S.E. Golowich, A. Smola, Support vector method for function approximation, regression estimation, and signal processing, Adv. Neural Inform. Process. Syst. 9 (1997) 281–287.

[21] J. Weston, A. Gammerman, M. Stitson, V. Vapnik, V. Vovk, C. Watkins, Density estimation using support vector machines, Technical Report CSD-TR-97-23, England, 1998.

Danian Zheng received his Bachelor degree in Computer Science and Technology in 2002 from Tsinghua University, Beijing, China. Since 2002 he has been pursuing the Master degree and the Doctoral degree at the Department of Computer Science and Technology at Tsinghua University. His research interests are mainly in the areas of support vector machines, kernel methods and their applications.

Jiaxin Wang received his Bachelor degree in Automatic Control in 1965 from Beijing University of Aeronautics and Astronautics, his Master degree in Computer Science and Technology in 1981 from Tsinghua University, Beijing, China, and his Doctoral degree in 1996 from Engineering Faculty of Vrije Universiteit Brussel, Belgium. He is currently a professor of Department of Computer Science and Technology, Tsinghua University. His research interests are in the areas of artificial intelligence, intelligent control and robotics, machine learning, pattern recognition, image processing and virtual reality.

Yannan Zhao received her Bachelor degree in Automatic Control in 1970 from Tsinghua University, Beijing, China. From 1992 to 1993, she was Visiting Scholar at Vrije Universiteit Brussel, Belgium. He is currently a professor of Department of Computer Science and Technology, Tsinghua University. Her research interests are in the areas of neural networks, intelligent robotics, pattern recognition and data mining.