Neural Comput & Applic
DOI 10.1007/s00521-011-0623-5

ORIGINAL ARTICLE

A flexible support vector machine for regression

Xiaobo Chen · Jian Yang · Jun Liang

Received: 5 December 2010 / Accepted: 3 May 2011

Abstract In this paper, a novel regression algorithm coined flexible support vector regression is proposed. We first model the insensitive zone in classic support vector regression, respectively, by its up- and down-bound functions and then give a kind of generalized parametric insensitive loss function (GPILF). Subsequently, based on GPILF, we propose an optimization criterion such that the unknown regressor and its up- and down-bound functions can be found simultaneously by solving a single quadratic programming problem. Experimental results on both several publicly available benchmark data sets and time series prediction show the feasibility and effectiveness of the proposed method.

Keywords Support vector machine · Function approximation · ε-Insensitive margin · Parametric insensitive loss function

1 Introduction

Support vector machine (SVM) [1–3], as a kind of state-of-the-art techniques, has played an important role in the pattern recognition and machine learning community over the past decades. It is based on the principle of structural risk minimization (SRM) and enjoys successful real-world applications, including text categorization [4], time series prediction [5] and face detection [6]. SVMs were originally developed for the classification problems; however, there exists a corresponding learning algorithm termed support vector regression (SVR) for function approximation, whose computational complexity is as expensive as that of SVM, that is $O(N^3)$, where N is the number of the training samples. This drawback restricts the application of SVM/SVR to large-scale problems. To overcome this problem, so far, many fast training algorithms such as SMO [7], SVMLight [8], SVMTorch [9], LIBSVM [10] have been presented to accelerate the training stage of SVM/SVR.

As we all know, different kinds of loss functions, which are used to measure empirical risk, play an important role in constructing supervised learning algorithms, e.g. hinge loss function for SVM. With regard to SVR, the ε-insensitive loss function (ε-ILF) devised by Vapnik [1], which contains a ε-insensitive margin, is the most commonly used loss function as it not only measures the training error but also leads to the sparse solution. The corresponding SVR is called ε-SVR as the value of ε needs to be predefined and is fixed over all the training samples. For ε-SVR, the determination of ε is an important yet difficult task in terms of its successful application. On the one hand, with the increment of ε-insensitive margin, the solution turns out to be sparser, meaning the number of support vectors reduces, and thus, the performance may decline [11]. On the other, with the decrement of ε-insensitive margin, more training samples tend to be support vectors and the estimated function may become sensitive to the noisy samples [3]. Usually, cross-validation techniques, e.g. [12], are used to

X. Chen (✉) · J. Yang
School of Computer Science and Technology,
Nanjing University of Science and Technology,
Nanjing 210094, People's Republic of China
e-mail: xbchen82@gmail.com

J. Yang
e-mail: csjyang@mail.njust.edu.cn

X. Chen · J. Liang
School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang 212013,
People's Republic of China

Published online: 25 May 2011

set the value of ε; however, it is too expensive in terms of computation.

So far, many algorithms have been proposed to tackle this problem. The v-support vector regression (v-SVR) [13], as a modification of ε-SVR, introduces a new parameter v to control the number of support vectors and training error. v is used to trade-off the width of ε-insensitive tube against empirical risk; meanwhile, the value of ε is optimized as accurate as possible rather than taking it as given a prior. Schölkopf et al. [13] further dropped the assumption that the ε-insensitive zone has a tube shape and thus used a parametric model $\zeta(x)$ of arbitrary shape. The resulting parametric insensitive model is useful in the situations when the noise is heteroscedastic, that is, when it depends on samples. This situation often occurs in time series predictions since the samples are usually non-stationary and volatile in nature. However, the selection of $\zeta(x)$ is still unclear, and the authors selected the function $\sin^2((2\pi/3)x)$ as $\zeta(x)$ in their experiments. Obviously, it may not be optimal for the other cases. To address this problem, Par-v-SVR is presented in the recently work of [14]. A parametric insensitive loss function is first defined by formulating the ε-insensitive zone as a nonlinear model, and then an optimization problem that is similar to that of v-SVR is adopted to learn both the unknown regressor and its parametric insensitive zone. In nature, Par-v-SVR depends on the assumption that the distribution of noise must be symmetrical since its parametric insensitive loss function is symmetrical in terms of the unknown regressor. This drawback may lead to suboptimal solutions in some real-world applications like financial engineering where the noise is highly volatile. In the work of [11], Yang et al. noticed that the margin in conventional ε-insensitive loss function [3] contains two characteristics, i.e. fixed and symmetrical properties. They classified the margin in SVR into four cases depending on whether it is fixed and symmetrical. A general ε-insensitive loss function is then proposed [11, 15] to split the margin into two parts: up-margin and down-margin. However, the determination of both the up- and down-margin is independent of the learning procedure of the regressor itself and thus needs to seek other auxiliary methods to estimate the noise distribution beforehand. Then, they proposed a method to adapt the margin in SVR locally and flexibly to capture the local trend of data [16]. In the work of Chen et al. [17], proposed smooth twin support vector regression (STSVR) that can seek the up- and down-bound functions of the unknown regression by Newton–Armijo Algorithm.

In this paper, inspired by Hao [11] and Chen et al. [14], we first extend the idea of parametric insensitive loss function in Par-v-SVR to an asymmetrical counterpart and then develop a novel regression algorithm coined flexible support vector regression (FSVR), which incorporates the setting of margin into the objective function of v-SVR and thus obtains a simultaneous learning framework for both margin determination and function approximation.

The rest of this paper is organized as follows. In Sect. 2, we give a brief review of Par-v-SVR that motivated our idea. Then, we proposed a novel regression algorithm, called FSVR, which approximates the unknown function and its up- and down-margin functions simultaneously in Sect. 3. Experimental results both on benchmark data sets, and time series are reported in Sect. 4. Finally, we conclude this paper in Sect. 5.

2 Brief review of parametric insensitive v-support vector regression

Let the inputs, i.e. $\{x_i\}_{i=1}^n$, are organized in the form of a matrix $A \in R^{n\times m}$, that is, the ith row of A, denoted by A_i, represents the ith training sample. The outputs are filled into a vector Y, i.e. $Y = [y_1, y_2, \ldots, y_n]^T$. The goal of regression is to seek a parametric nonlinear model such that it captures the relevance between the input and output of the training samples, that is, $f : x_i \rightarrow y_i$.

The support vector regression (SVR) aims at finding the following parameterized nonlinear model

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

where $\phi(\cdot)$ is a function, which maps x into a feature space wherein the relationship between $\phi(x)$ and $f(x)$ can be described by a linear model with weights vector w and bias b.

In general, the observation y_i is usually contaminated by different kind of noises. Parametric insensitive v-support vector regression (Par-v-SVR) drops the assumption that the noise level is uniform throughout the domain, or at least, its function dependency is known beforehand [13]. It focuses on the situation when the amount of noise depends on location, that is, heteroscedastic noise. Specifically, Par-v-SVR is constructed by first defining the following parametric insensitive loss function (PILF) $L^g(x, y, f)$ as

$$L^g(x,y,f) = |y - f(x)| = \begin{cases} 0 & \text{if} \quad |y - f(x)| \le g(x) \\ |y - f(x)| - g(x) & \text{otherwise} \end{cases}, \tag{2}$$

where $f(x)$ is the regression function as shown in (1) while $g(x) = c^T\phi(x) + d$ is another nonlinear function with unknown parameters c and d. In fact, it is $g(x)$ that describes the width of the insensitive tube. By investigating PILF (2), we find that it does not penalize the deviation as long as it is inside the insensitive zone $f \pm g$. Only the

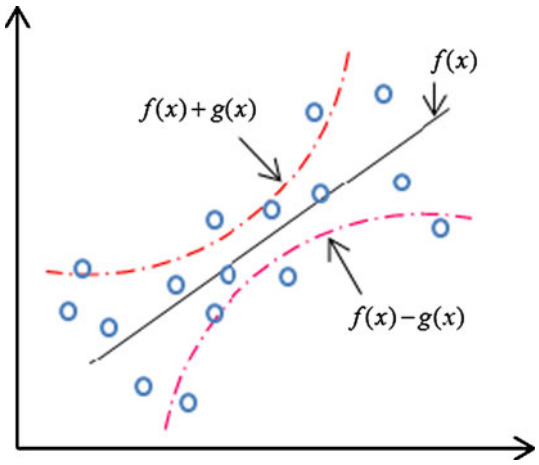

Fig. 1 Illustration of Par-v-SVR

samples that lie outside $f \pm g$ contribute to the cost. Furthermore, the formulation (2) defines a parametric insensitive zone $f \pm g$ of arbitrary shape such that the classical ε-insensitive loss is a special case of (2) when $g(x)$ is constant, i.e. independent of location x. Figure 1 depicts Par-v-SVR graphically. To seek $f(x)$ and $g(x)$ simultaneously, Par-v-SVR minimizes the following objective function

$$\begin{aligned} \min\ & \frac{1}{2}||w||^2 + C\left(v \cdot \left(\frac{1}{2}||c||^2 + d\right) + \frac{1}{N}\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)\right) \\ \text{s.t. } & \left(w^T\phi(x_i) + b\right) + \left(c^T\phi(x_i) + d\right) \geq y_i - \xi_i \\ & \left(w^T\phi(x_i) + b\right) - \left(c^T\phi(x_i) + d\right) \leq y_i + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, i = 1, 2, \ldots, n \end{aligned} \tag{3}$$

The dual problems of (3) is presented as follows using the KKT conditions [18]

$$\begin{aligned} \max\ & -\tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\alpha_i - \alpha_i^*\right)\left(\alpha_j - \alpha_j^*\right)k\left(x_i, x_j\right) \\ & -\tfrac{1}{2Cv}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(\alpha_i + \alpha_i^*\right)\left(\alpha_j + \alpha_j^*\right)k\left(x_i, x_j\right) \\ & + \sum_{i=1}^{n}\left(\alpha_i - \alpha_i^*\right)y_i \\ & \text{s.t.} \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right) = 0, \\ & \sum_{i=1}^{N}\left(\alpha_i + \alpha_i^*\right) = Cv, \quad 0 \leq \alpha_i \leq \tfrac{C}{N}, 0 \leq \alpha_i^* \leq \tfrac{C}{N} \end{aligned} \tag{4}$$

where $k\left(x_i, x_j\right) = \phi(x_i)^T\phi\left(x_j\right)$ is a kernel function. Usually, the feature map $\phi(\cdot)$ may have a very high dimension, and its direct computation is infeasible. However, as we see from (4), the final model only depends on the dot-product of $\phi(x_i)$ and $\phi\left(x_j\right)$, which does not require an explicit expression of $\phi(\cdot)$ itself. It is the so-called kernel trick [2].

After solving the above quadratic programming problem (4), we obtain the regression function $f(x)$ and the corresponding parametric insensitive function $g(x)$ as follows:

$$f(x) = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)k(x_i, x) + b \tag{5}$$

$$g(x) = \frac{1}{Cv}\sum_{i=1}^{N}\left(\alpha_i + \alpha_i^*\right)k(x_i, x) + d \tag{6}$$

though PILF (2) parameterizes the margin model of SVR such that it is able to vary with samples, its symmetrical nature, that is, up-bound function $f(x) + g(x)$ and down-bound function $f(x) - g(x)$ are symmetrical with respect to the regression $f(x)$, restricts its applications in some cases, e.g. in time series. In the context of time series prediction, the data are usually highly volatile and the associated variance of noise varies over time such that the symmetrical margin assumption may fail to capture the trend of data and thus leads to declined performance.

3 Flexible support vector regression

3.1 Generalized parametric insensitive loss function (GPILF)

To introduce asymmetrical margin into SVR, we first extend the PILF (2) as follows.

Definition 1 Generalized parametric insensitive loss function (GPILF).

$$\begin{aligned} L_d^u(x, y, f) &= |y - f(x)|_d^u \\ &= \begin{cases} y - f(x) - u(x), & \text{if } y - f(x) > u(x) \\ 0, & \text{if } u(x) \geq y - f(x) \geq -d(x), \\ f(x) - y - d(x), & \text{if } f(x) - y > d(x) \end{cases} \end{aligned} \tag{7}$$

where $u(x)$ and $d(x)$ are, respectively, called parametric up- and down-margin functions, which determine the up-margin and down-margin at sample x. It should be emphasized that we further assume the regressor $f(x)$ and its up- and down-margin functions are all parameterized. The idea behind (7) is that we do not penalize the deviation if it lies between $u(x)$ and $-d(x)$, and all further deviation should incur a linear penalty. In the case of $u(x) = d(x)$, GPILF reduces to the original PILF (2). Furthermore, when $u(x)$ and $d(x)$ are equal to an identical constant ε, GPILF amounts to the classical ε-ILF. Therefore, the proposed GPILF generalizes the existing loss functions, including ε-ILF and PILF. In this paper, we will discuss the general case, that is, $u(x) \neq d(x)$ and both vary with sample x. Thus, the margin in our case is asymmetrical (since

$u(x) \neq d(x)$) and heteroscedastic (since both $u(x)$ and $d(x)$ vary with different samples).

Rearranging the formulation of (7) yields the following expression

$$L_l^h(x,y,f) = |y - f(x)|_l^h = \begin{cases} y - h(x), & \text{if} \quad y > h(x) \\ 0, & \text{if} \quad h(x) \geq y \geq l(x) \\ l(x) - y, & \text{if} \quad y < l(x) \end{cases}, \tag{8}$$

where $h(x) = f(x) + u(x)$, $l(x) = f(x) - d(x)$ are, respectively, called parametric up- and down-bound functions at point x. An intuitive geometric interpretation for (8) is shown in Fig. 2. Based on the GPILF, we devise a novel regression algorithm, which learns the estimated function $f(x)$ and corresponding up- and down-bound functions, i.e. $h(x)$ and $l(x)$, simultaneously. Once $h(x)$ and $l(x)$ are estimated, it is easy to obtain the up- and down-margin functions from the following equations:

$$u(x) = h(x) - f(x) \tag{9}$$

$$d(x) = f(x) - l(x). \tag{10}$$

3.2 Flexible support vector regression based on GPILF

We express the regression function $f(x)$ and its up- and down-bound functions $h(x)$ and $l(x)$ by the following nonlinear models:

$$h(x) = w_h^T \phi(x) + b_h \tag{11}$$

$$f(x) = w^T \phi(x) + b \tag{12}$$

$$l(x) = w_l^T \phi(x) + b_l, \tag{13}$$

where $[w_h, b_h]$, $[w, b]$ and $[w_l, b_l]$ are the coefficients of $h(x)$, $f(x)$ and $l(x)$, respectively. Our goal is to maximize the margin while softly penalize points that lie outside the parametric insensitive zone depicted by its up- and down-bound functions. Therefore, FSVR is obtained by solving the following quadratic programming problem:

$$\min \frac{1}{2}\left(||w||^2 + b^2\right) + C\left(\frac{v_1}{2}\left|\left|\begin{bmatrix} w \\ b \end{bmatrix} - \begin{bmatrix} w_h \\ b_h \end{bmatrix}\right|\right|^2 + \frac{v_2}{2}\left|\left|\begin{bmatrix} w \\ b \end{bmatrix} - \begin{bmatrix} w_l \\ b_l \end{bmatrix}\right|\right|^2 + \frac{1}{N} e^T(\xi + \xi^*)\right), \tag{14}$$

$$\text{s.t.} \quad \begin{aligned} & Aw_h + eb_h + \xi \geq Y, \\ & Aw_l + eb_l - \xi^* \leq Y, \quad \xi \geq 0, \xi^* \geq 0 \end{aligned} \tag{15}$$

where $C \geq 0$, $v_1 \geq 0$, $v_2 \geq 0$ are trade-off parameters, ξ and $\xi^*$are slack vectors, $A = \left[\phi(x_1)^T; \phi(x_2)^T; \ldots \phi(x_n)^T\right]$ is the sample matrix in feature space.

The FSVR algorithm seeks three functions $f(x)$, $h(x)$ and $l(x)$, i.e. the unknown regressor, its parametric up- and down-bound functions, in a unified framework. The first item of the objective function (14) is used to penalize model complexity to alleviate overfitting. Note that we penalize the weight vector and threshold simultaneously as in [19, 20] to simplify the dual model shown in the next section. The second and third items measure the sizes of the parametric up- and down-margins. Therefore, minimizing them leads the up- and down-margins as small as possible. In other words, $h(x)$ and $l(x)$ are as close as possible to $f(x)$. The slack vectors ξ and ξ^* are introduced to measure the error whenever the data lies outside the asymmetrical parametric zone. The constant C determines the trade-off between the flatness of the function and the amount of larger deviations of tolerance. With respect to the constraints, the first one of (15) is used to force the parametric up-bound function $h(x)$ lie above the whole training data with slack error. The meaning of the other constraint is similar to that of the first one. In a word, considering GPILF provides a systematic and automatic scheme to adapt the margin locally and flexibly, the proposed FSVR can tolerate noise adaptively.

Before deriving specific algorithm for FSVR, we give the following proposition implying that the minimization of (14) under constraints (15) is convex with respect to optimization variables simultaneously.

Proposition 1 The minimization problem (14) under constraints (15) is a convex quadratic optimization problem.

Proof Obviously, the constraints (15) only include linear inequalities in terms of primal variables w_h, b_h, w_l, b_l, ξ and ξ^*. The last term of objective function (14) is also linear in ξ and ξ^*. By introducing auxiliary vector

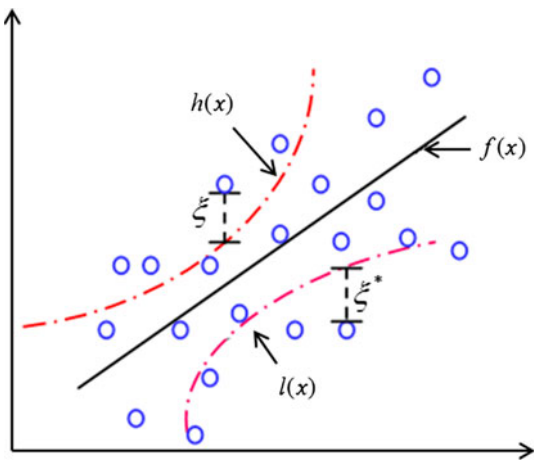

Fig. 2 Illustration of flexible support vector regression (FSVR)

$r = [w; b; w_h; b_h; w_l; b_l]$, where $[v_1; v_2]$ is defined as $\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$, we re-express the vectors $[w; b]$, $[w_h; b_h]$ and $[w_l; b_l]$ as follows

$$[w; b] = [\mathrm{I}\ 0\ 0]r, \quad [w_h; b_h] = [0\ \mathrm{I}\ 0]r, \quad [w_l; b_l] = [0\ 0\ \mathrm{I}]r,$$

where I is an identity matrix and 0 is a matrix of zeros with appropriate dimensionality.

Considering the previous relationship, we can re-write the first three terms of (14), i.e. quadratic terms, as

$$\begin{aligned}
&\tfrac{1}{2}\left(||w||^2 + b^2\right) + C\left(\tfrac{v_1}{2}||\begin{bmatrix} w \\ b \end{bmatrix} - \begin{bmatrix} w_h \\ b_h \end{bmatrix}||^2 + \tfrac{v_2}{2}||\begin{bmatrix} w \\ b \end{bmatrix} - \begin{bmatrix} w_l \\ b_l \end{bmatrix}||^2\right) \\
&= \tfrac{1}{2}||[\mathrm{I}00]r||^2 + \tfrac{Cv_1}{2}||[\mathrm{I} - \mathrm{I}0]r||^2 + \tfrac{Cv_2}{2}||[\mathrm{I}0 - \mathrm{I}]r||^2 \\
&= \tfrac{1}{2} r^T \left(\mathrm{I}_1^T \mathrm{I} + \tfrac{Cv_1}{2}\mathrm{I}_2^T \mathrm{I} + \tfrac{Cv_2}{2}\mathrm{I}_3^T \mathrm{I}\right) r \\
&= r^T \begin{bmatrix} \left(1 + \frac{Cv_1 + Cv_2}{2}\right)\mathrm{I} & -\frac{Cv_1}{2}\mathrm{I} & -\frac{Cv_2}{2}\mathrm{I} \\ -\frac{Cv_1}{2}\mathrm{I} & \frac{Cv_1}{2}\mathrm{I} & 0 \\ -\frac{Cv_2}{2}\mathrm{I} & 0 & \frac{Cv_2}{2}\mathrm{I} \end{bmatrix} r
\end{aligned}$$

where $\mathrm{I}_1 = [array{*}20c\mathrm{I} - \mathrm{I}0]$ and $\mathrm{I}_3 = [\mathrm{I}\ \ 0\ \ -\mathrm{I}]$. We immediately notice that the above expression is symmetrical positive semi-definite with respect to r, i.e. primal variables $[w; b; w_h; b_h; w_l; b_l]$. In conclusion, the minimization of (14) under (15) is a convex quadratic problem.

3.3 Algorithm derivation

According to proposition 1, the formulation of FSVR has global optimal solution. For the sake of simplicity, let us define $G = [A\ \ e]$, $u = [w^T, b]^T$, $u_h = \left[w_h^T, b_h\right]^T$, $u_l = \left[w_l^T, b_l\right]^T$ To derive the dual formulation for (14), we introduce the following lagrangian function

$$\begin{aligned}
L = {} & \tfrac{1}{2}||u||^2 + \tfrac{Cv_1}{2}||u - u_h||^2 + \tfrac{Cv_2}{2}||u - u_l||^2 + \tfrac{C}{N}e^T(\xi + \xi^*) \\
& - \alpha_1^T(Gu_h + \xi - Y) - \alpha_2^T(-Gu_l + \xi^* + Y) - \alpha_3^T\xi - \alpha_4^T\xi^*
\end{aligned} \tag{16}$$

where $\alpha_1 = [\alpha_{11}, \alpha_{12}, \ldots, \alpha_{1n}]^T \in R^n$ is the vector of nonnegative lagrangian multipliers, the meaning of α_2, α_3, α_4 are similar to that of α_1. To minimize (14), we have to find the saddle point of L, meaning that we minimize over the primal variables u, u_h, u_l, ξ, ξ^* and maximize over the dual variables α_1, α_2, α_3, α_4. Setting the derivatives with respect to the primal variables equal to zero yields the following equations

$$\frac{\partial L}{\partial u} = u + Cv_1(u - u_h) + Cv_2(u - u_l) = 0 \tag{17}$$

$$\frac{\partial L}{\partial u_h} = Cv_1(u_h - u) - G^T\alpha_1 = 0 \tag{18}$$

$$\frac{\partial L}{\partial u_l} = Cv_2(u_l - u) + G^T\alpha_2 = 0 \tag{19}$$

$$\frac{\partial L}{\partial \xi} = \frac{C}{N}e - \alpha_1 - \alpha_3 = 0 \tag{20}$$

$$\frac{\partial L}{\partial \xi^*} = \frac{C}{N}e - \alpha_2 - \alpha_4 = 0. \tag{21}$$

Furthermore, the Karush–Kuhn–Tucker (KKT) complementarity conditions are given by

$$\alpha_{1i}(G_i u_h + \xi_i - y_i) = 0 \tag{22}$$

$$\alpha_{2i}\left(-G_i u_l + \xi_i^* + y_i\right) = 0 \tag{23}$$

$$\left(\frac{C}{N} - \alpha_{1i}\right)\xi_i = 0 \tag{24}$$

$$\left(\frac{C}{N} - \alpha_{2i}\right)\xi_i^* = 0. \tag{25}$$

Substituting (18) and (19) into (17), we obtain the following explicit expression for u

$$u = G^T(\alpha_1 - \alpha_2). \tag{26}$$

Taking (18) (19) and the above expression (26) into account, we have

$$u_h = G^T(\alpha_1 - \alpha_2) + \frac{1}{Cv_1}G^T\alpha_1 \tag{27}$$

$$u_l = G^T(\alpha_1 - \alpha_2) - \frac{1}{Cv_2}G^T\alpha_2 \tag{28}$$

substituting (26), (27) and (28) into the lagrangian function (16) yields the following wolfe dual problem for (14):

$$\begin{aligned}
\max L(\alpha_1, \alpha_2) = {} & -\tfrac{1}{2}(\alpha_1 - \alpha_2)^T GG^T(\alpha_1 - \alpha_2) - \tfrac{1}{2Cv_1}\alpha_1^T GG^T\alpha_1 \\
& - \tfrac{1}{2Cv_2}\alpha_2^T GG^T\alpha_2 + Y^T(\alpha_1 - \alpha_2)
\end{aligned} \tag{29}$$

$$\text{s.t.} \qquad \frac{C}{N}e \geq \alpha_1 \geq 0, \quad \frac{C}{N}e \geq \alpha_2 \geq 0. \tag{30}$$

The dual problem (29) is similar to that of Par-v-SVR (4) and v-SVR [11] and is also convex with respect to dual variables. This fact verifies the convexity of primal formulation (14), i.e. Proposition 1, from another perspective.

After solving the above QP problem, we obtain the regression function and corresponding parametric up- and down-bound functions as:

$$f(x) = w^T\phi(x) + b = (K(A, x) + e)^T(\alpha_1 - \alpha_2) \tag{31}$$

$$h(x) = w_h^T \phi(x) + b = (K(A,x) + e)^T \left(\alpha_1 - \alpha_2 + \frac{1}{Cv_1}\alpha_1\right) \tag{32}$$

$$l(x) = w_l^T \phi(x) + b = (K(A,x) + e)^T \left(\alpha_1 - \alpha_2 - \frac{1}{Cv_2}\alpha_2\right) \tag{33}$$

the KKT conditions (22)–(25) provide us several useful conclusions. The training sample x_i for which $\alpha_{1i} > 0$ or $\alpha_{2i} > 0$ is termed support vector (SV). Specifically, the sample x_i satisfying $\alpha_{1i} > 0$ can be viewed as up-SV, while x_i satisfying $\alpha_{2i} > 0$ is a down-SV. Specifically, taking α_{1i} for example, we need to distinguish the difference between sample $\frac{C}{N} > \alpha_{1i} > 0$ and $\alpha_{1i} = \frac{C}{N}$. In the first case, from (22) and (24), it follows that $\xi_i = 0$ and $w_1^T \phi(x_i) + b_1 = y_i$. In other words, the sample $(\phi(x_i), y_i)$ with corresponding $\frac{C}{N} > \alpha_{1i} > 0$ lies exactly on the up-bound function $h(x)$. In the second case, it follows that $\xi_i > 0$ and $w_1^T \phi(x_i) + b_1 < y_i$. In other words, the corresponding sample lies upon $h(x)$. The case for α_{2i} is similar to that of α_{1i} expect that we focus on down-bound rather than up-bound function. Comparing the dual problem in the proposed FSVR (given in (29) and (30)) to the dual problem in Par-v-SVR (given in (4)), we notice that there is no equality constraints in FSVR. On the one hand, this fact simplifies the solution of QPP, and on the other hand, the physical meaning of v_1 and v_2 is loss. Thus, the values of v_1 and v_2 have nothing to do with the number of support vectors. With respect to time complexity, FSVR, Par-v-SVR and v-SVR all need to solve a QPP whose size is two times of the number of training samples. Thus, we expect they have similar training cost.

4 Experimental results

To check the validity of the proposed FSVR, we compare it with the classical v-SVR and Par-v-SVR on several UCI benchmark data sets [21]. Then, we apply the proposed method to time series prediction, which is highly volatile in nature.

4.1 Experimental specification

All the regression methods are implemented in MATLAB (7.8.0) R2009a environment on a PC with Intel Core 2 Duo processor (2.4 GHz), 3 GB RAM. Mosek optimization toolbox [22] for MATLAB was used to solve all the dual QPPs. In all our experiments, Gaussian kernel $k(x_i, x_j) = \exp\left(-||x_i - x_j||^2 \Big/ (2\sigma^2)\right)$ is used to construct nonlinear regressor. We first rescale the input and response variables for each data set such that the means and standard deviation of input variables are 0 and 1. Fivefold cross-validation is used to evaluate the performance of these methods. For the three methods, the penalty parameter C is selected from the set $\{10^{-5}, 10^{-4}, \ldots, 10^4, 10^5\}$ while v is selected from $\{0.1, 0.2, \ldots, 1\}$. In addition, performance indices including root mean square error (RMSE) test error $\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2}$ [14], standard deviation of

Table 1 Comparison among FSVR, Par-v-SVR and v-SVR on benchmark data sets

Data sets	Algorithm	RMSE	Std	SVs (%)	CPU (s)
Boston housing 506 × 14	FSVR	0.3822	0.0863	38.22	0.9661
	Par-v-SVR	0.3782	0.0854	26.04	1.0384
	v-SVR	0.3720	0.0871	28.61	1.0125
AutoMPG 398 × 8	FSVR	0.3665	0.0379	33.61	0.5057
	Par-v-SVR	0.3740	0.0502	19.48	0.5450
	v-SVR	0.3684	0.0363	28.90	0.5270
Automobile 205 × 26	FSVR	0.4113	0.0743	29.39	0.0998
	Par-v-SVR	0.4423	0.0535	19.34	0.1024
	v-SVR	0.4227	0.0823	23.61	0.0962
Servo 164 × 12	FSVR	0.5383	0.0639	53.95	0.0694
	Par-v-SVR	0.5616	0.0990	42.83	0.0728
	v-SVR	0.5691	0.0938	40.39	0.0731
Machine CPU 209 × 8	FSVR	0.0650	0.0391	17.46	0.1694
	Par-v-SVR	0.0704	0.0433	15.31	0.1639
	v-SVR	0.0664	0.0385	16.86	0.1612
Bodyfat 252 × 14	FSVR	0.1523	0.1042	23.41	0.1934
	Par-v-SVR	0.1658	0.0960	20.24	0.1950
	v-SVR	0.1532	0.1038	23.41	0.1965
Diabetes 43 × 2	FSVR	0.6450	0.1846	57.34	0.0132
	Par-v-SVR	0.6185	0.1774	47.34	0.0156
	v-SVR	0.6317	0.1731	55.60	0.0126
Triazine 186 × 61	FSVR	0.9069	0.1550	36.94	0.0841
	Par-v-SVR	0.9427	0.1450	27.93	0.1006
	v-SVR	0.9365	0.1263	40.96	0.0917
AutoPrice 159 × 16	FSVR	0.4159	0.0865	33.79	0.0682
	Par-v-SVR	0.4875	0.0661	14.76	0.0721
	v-SVR	0.4469	0.0860	26.25	0.0684
Wpbc 194 × 33	FSVR	0.9662	0.1433	62.14	0.0893
	Par-v-SVR	1.0531	0.2508	48.72	0.1213
	v-SVR	1.0448	0.2498	59.44	0.1217
ConcreteCS 1,030 × 9	FSVR	0.3778	0.0209	33.01	6.2719
	Par-v-SVR	0.3719	0.0251	24.88	7.0064
	v-SVR	0.3813	0.0205	33.16	6.3979
Motorcycle 133 × 2	FSVR	0.4719	0.0885	47.19	0.0524
	Par-v-SVR	0.5184	0.1575	34.03	0.0530
	v-SVR	0.4912	0.0691	39.09	0.0475

Table 2 Summary of statistics of the time series data sets

	Train				Test			
	Mean	Std	Skew	Kurt	Mean	Std	Skew	Kurt
MG17	−0.0000	1.0000	−0.4473	2.5031	0.0232	1.0429	−0.4728	2.5220
MG30	−0.0000	1.0000	−0.4543	2.5588	0.01084	0.9449	−0.6124	2.8078
DJ	−0.0000	1.0000	0.5827	2.7186	1.2505	1.0787	−0.0248	1.9089
AO	−0.0000	1.0000	0.2842	2.6516	0.2847	0.9546	0.9867	3.6713
Sunspot	−0.0000	1.0000	0.8646	3.0032	0.1699	1.1776	1.0134	3.4448

test error, mean training time (s) and the percent of support vectors (%) across five tests are adopted to measure the performance of each algorithm.

4.2 Evaluations on benchmark data sets

We test eleven UCI benchmark data sets, which are shown in Table 1. To accelerate model selection, we tune a set comprising of randomly 20% of the training samples [17] to select optimal parameters. Table 1 also reports the testing results of the proposed FSVR, Par-v-SVR and v-SVR on these data sets. From Table 2, we observe that FSVR outperforms the other two methods in most cases. For instance, FSVR obtains notable improvement on Servo, triazine, Wpbc and Motorcycle data sets. On the remaining data sets, FSVR is as competitive as Par-v-SVR and v-SVR. The improvements may be due to the fact that FSVR can, respectively, adapt its up- and down-margin functions, which leads to a more flexible model than its competitors. We also report the percent of SVs (%) in Table 2. It can be seen that FSVR is also sparse, although it needs slightly more support vectors than the other two methods. In terms of computational time, from Table 1, we find that FSVR is similar to Par-v-SVR and v-SVR. We further conduct experiments with different v_1 and v_2 on Motorcycle data set to show the influence of these two parameters. The result is show in Fig. 3. We observe that different v_1 and v_2 have severely impact the performance.

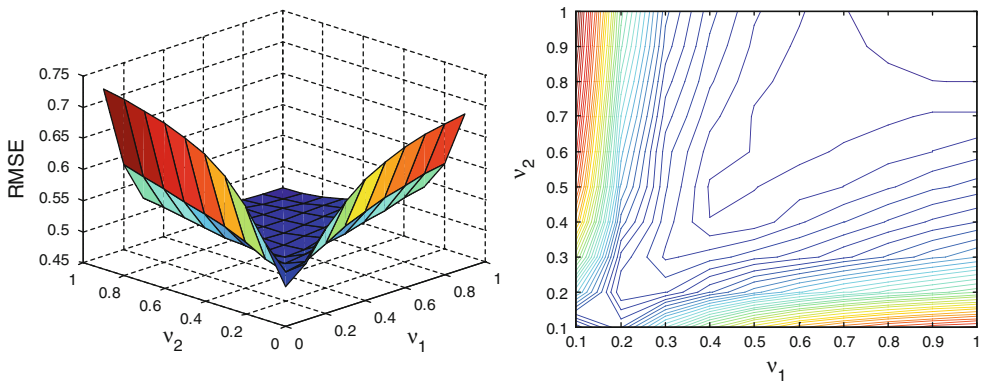

Fig. 3 The influence of v_1 and v_2 on motorcycle data set

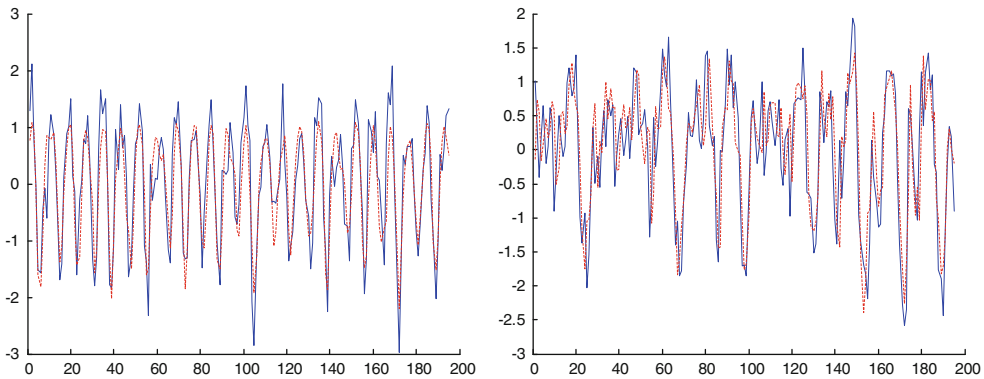

Fig. 4 Prediction on MG17 on MG30

Table 3 Comparison of regression performance on time series data sets

Data sets	Lag	RMSE	SVs (%)	CPU (s)
MG_{17}	1	0.8544	56.57	0.1539
	3	0.6095	42.27	0.1233
	5	0.5267	50.53	0.1004
	7	0.5398	47.31	0.1147
MG_{30}	1	0.6867	31.31	0.1112
	3	0.6760	37.11	0.1208
	5	0.6052	34.74	0.1150
	7	0.5238	59.14	0.3120
DJ	1	0.3233	12.12	0.1130
	3	0.3074	22.68	0.1016
	5	0.3166	28.42	0.1078
	7	0.4167	25.81	0.1058
AO	1	0.2375	15.15	0.1152
	3	0.2321	19.59	0.1377
	5	0.2450	8.42	0.1371
	7	0.4812	44.09	0.1139
Sunspots	1	0.6712	32.32	0.1108
	3	0.4382	37.11	0.1409
	5	0.4949	41.05	0.1041
	7	0.4635	47.31	0.1060

Therefore, selecting an appropriate value for v_1 and v_2 is very important for FSVR. In conclusion, FSVR may derive better generalization performance by utilizing the flexible margin configuration without increasing any training time.

4.3 Applications to time series forecasting

In this section, we apply the proposed method to do time series data sets. The involved time series includes the series generated by the Mackey–Glass delay differential equation, Dowjones (DJ), Australian all ordinaries index (AO) and sunspot data set. Mackey–Glass data sets are taken from the website http://www.cse.ogi.edu/~ericwan. All other data sets are taken from the website http://www.robjhyndman.com/TSDL/. The DJ and AO are financial data sets, while sunspot data describes the number of spots on the sun's surface from time to time. Since different lags have different impact on the predication accuracy, we test the best results achieved by FSVR with different lags [16, 23, 24]. We report the statistical properties of these data sets in Table 2.

The Mackey–Glass time delay differential equation is originally introduced as a model of blood cell regulation and has become quite common as an artificial forecasting benchmark since it can generate high-dimensional chaotic system. The equation is given by:

$$\frac{\mathrm{d}x(t)}{\mathrm{d}t} = -bx(t) + a\frac{x(t-\tau)}{1+x(t-\tau)^{10}}, \tag{34}$$

where a, b are parameters and τ is the time delay. We test our algorithm on two time series generated by the above differential equation by first fixing $a = 0.2$, $b = 0.1$ and then let $\tau = 17$ and $\tau = 30$ respectively. The time series corresponding to $\tau = 17$ and $\tau = 30$ are, respectively, called as MG_{17} and MG_{30}. In addition, Gaussian noise with mean value 0 and variance 0.1 is added to the data sets. We use 300 samples in our experiments. The first 100 points are used for training set, while the remaining is for testing. The real values (blue curve) and their predictions (red curve) on the testing data sets for MG_{17} and MG_{30} are shown in Fig. 4a, b, respectively. The RMSE, SVs and CPU time with different lags for MG_{17} and MG_{30} are reported in Table 3.

For DJ, AO and sunspot data sets, the initial 100 samples are assumed for training and the remaining reserved for testing. The RMSE, SVs and CPU time with different lags for these data sets are also reported in Table 3. As seen from Table 3, the proposed FSVR is a suitable method for regression problems. We also show the influence of v_1 and v_2 on sunspots data set in Fig. 5 and observe a similar result as in the above section.

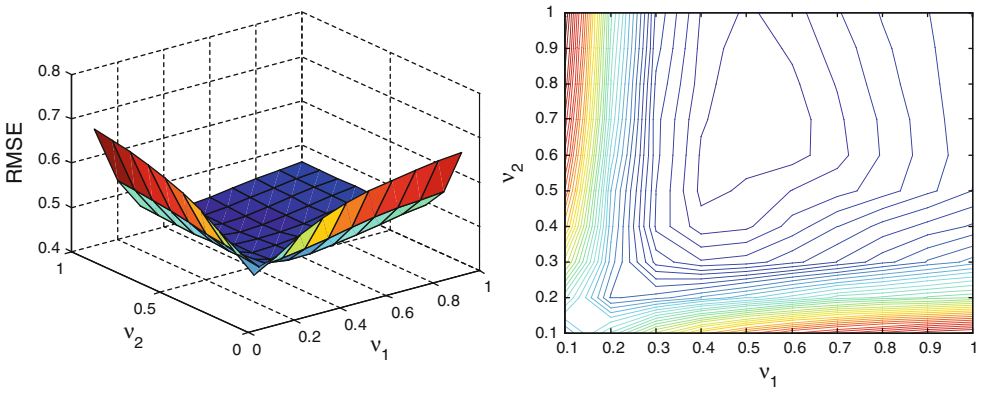

Fig. 5 The influence of v_1 and v_2 on sunspots data set

5 Conclusions and future work

In this paper, we developed a novel regression technique, coined flexible support vector regression (FSVR). The idea behind FSVR is to learn the unknown function and its up- and down-bound functions simultaneously based on the generalized parametric insensitive loss function (GPILF). The up- and down-bound functions are both nonfixed and asymmetrical. As a result, FSVR can adjust its margins locally and automatically to explore the trend of samples. Experiments on both benchmark data sets and time series show that the proposed algorithm may obtain better performance by adjusting its margins flexibly. A drawback of FSVR is that the meaning of v_1 and v_2 cannot be explained intuitively as that in v-SVR and Par-v-SVR. How to improve FSVR to introduce certain physical meaning for v_1 and v_2 is one of our intended research topic. Another disadvantage of FSVR is that the number of hyper-parameters is more than that of v-SVR and Par-v-SVR, which leads to slow model selection procedure. How to develop efficient parameters selection strategy, e.g. by particle swarm optimization [24], is an interesting problem in the future.

Acknowledgments The authors would like to thank the anonymous reviewers for their critical and constructive comments and suggestions. The authors are also thankful to Scientific Foundation of Jiangsu province (BK2010339) and Natural Science Fund for Colleges and Universities in Jiangsu Province (10KJD580001).

References

1. Vapnik VN (1998) Statistical learning theory. Wiley, New York
2. Schölkopf B, Smola AJ (2002) Learning with kernels. MIT Press, Cambridge
3. Vapnik VN (1999) The nature of statistical learning theory, 2nd edn. Springer, New York
4. Joachims T (1998) Text categorization with support vector machines: learning with many relevant features [A]. In: European conference on machine learning no. 10[C] 1398. Chemnitz, Springer, pp 137–142
5. Cao L, Tay FEH (2001) Financial forecasting using support vector machines. Neural Comput Appl 10(2):184–192
6. Osuna E, Freund R, Girosi F (1997) Training support vector machines: an application to face detection [A]. In: Proceedings of the 1997 conference computer vision and pattern recognition[C]. IEEE Computer Society, Washington, pp 130–136
7. Platt JC (1998) Fast training of support vector machines using sequential minimal optimization. In: Advances in Kernel methods—support vector machines. Cambridge
8. Joachims T (1999) Making large-scale SVM learning practical. In: Advances in Kernel methods support vector machine. Cambridge
9. Collobert R, Bengio S (2001) SVMTorch: support vector machines for large-scale regression problems. J Mach Learn 1(2):143–160
10. Chang CC, Lin CJ, LIBSVM: a library for support vector machines. Available from. http://www.csie.ntu.edu.tw/~cjlin
11. Yang HQ, Chan LW, King I (2002) Support vector machine regression for volatile stock market prediction. In Intelligent data engineering and automated learning (IDEAL 2002). Springer, NewYork, 2412 of LNCS, pp 391–396
12. Cao LJ, Chua KS, Guan LK (2003) Ascending support vector machines for financial time series forecasting. In: International conference on computational intelligence for financial engineering (CIFEr2003). pp 329–335
13. Schölkopf B, Smola AJ, Williamson R, Bartlett PL (2000) New support vector algorithms. Neural Comput 12(5):1207–1245
14. Hao PY (2010) New support vector algorithms with parametric insensitive/margin model. Neural Netw 23:60–73
15. Huang KZ, Yang HQ, King I, Lyu M (2008) Machine learning: modeling data locally and globally. In: Advanced topics in science and tecnology in China, 1st edn. Springer, Berlin, ISBN-13: 978-3540794516. Zhejiang University Press, Hangzhou, ISBN-10: 540794514
16. Yang HQ, Huang KZ, King I, Lyu MR (2009) Localized support vector regression for time series prediction. Neurocomputing 72:2659–2669
17. Chen XB, Yang J, Liang J, Ye QL (2010) Smooth twin support vector regression. Neural Comput Appl. doi:10.1007/s00521-010-0454-9
18. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge Univ. Press, Cambridge
19. Fung G, Mangasarian OL (2001) Proximal support vector machine classifiers. In: Proceedings KDD-2001: knowledge discovery and data mining. San Francisco, pp 77–86
20. Mangasarian OL, Musicant DR (1999) Successive overrelaxation for support vector machines. IEEE Trans Neural Netw 10(5): 1032–1037
21. Muphy PM, Aha DW (1992) UCI repository of machine learning databases
22. The MOSEK Optimization Tools Version 5.0, Denmark. [Online]. Available: http://www.mosek.com (2008)
23. Balasundaram S, Kapil (2010) On Lagrangian support vector regression. Expert Syst Appl 37:8784–8792
24. Guo XC, Yang JH, Wu CG, Wang CY, Liang YC (2008) A novel LS-SVMs hyper-parameter selection based on particle swarm optimization. Neurocomputing 71:3211–3215