Int. J. Mach. Learn. & Cyber. (2015) 6:155–166
DOI 10.1007/s13042-013-0200-6

ORIGINAL ARTICLE

A fast algorithm for training support vector regression via smoothed primal function minimization

Songfeng Zheng

Received: 3 July 2013 / Accepted: 10 September 2013 / Published online: 13 November 2013

Abstract The support vector regression (SVR) model is usually fitted by solving a quadratic programming problem, which is computationally expensive. To improve the computational efficiency, we propose to directly minimize the objective function in the primal form. However, the loss function used by SVR is not differentiable, which prevents the well-developed gradient based optimization methods from being applicable. As such, we introduce a smooth function to approximate the original loss function in the primal form of SVR, which transforms the original quadratic programming into a convex unconstrained minimization problem. The properties of the proposed smoothed objective function are discussed and we prove that the solution of the smoothly approximated model converges to the original SVR solution. A conjugate gradient algorithm is designed for minimizing the proposed smoothly approximated objective function in a sequential minimization manner. Extensive experiments on real-world datasets show that, compared to the quadratic programming based SVR, the proposed approach can achieve similar prediction accuracy with significantly improved computational efficiency, specifically, it is hundreds of times faster for linear SVR model and multiple times faster for nonlinear SVR model.

Keywords Support vector regression · Smooth approximation · Quadratic programming · Conjugate gradient · ε-Insensitive loss function

S. Zheng (✉)
Department of Mathematics, Missouri State University, Springfield, MO 65897, USA
e-mail: SongfengZheng@MissouriState.edu

List of symbols

$\mathbf{x}$ and y	The predictor vector and the response variable
$\mathbf{w}$ and b	The weight vector and the intercept in linear regression model
ϵ and C	The sensitive parameter and the penalty parameter in SVR model
ξ and ξ^*	The slack variables in SVR model
α and α^*	The Lagrange multipliers in SVR model
$V_\epsilon(\cdot)$ and $S_{\epsilon,\tau}(\cdot)$	The original and the smoothed version of loss function of SVR
τ	The smoothing parameter
$\Phi(\mathbf{w})$ and $\Phi_\tau(\mathbf{w})$	The original and the smoothed version of the objective function of SVR model in primal form
$\hat{\mathbf{w}}$ and $\hat{\mathbf{w}}_\tau$	The minimum point of $\Phi(\mathbf{w})$ and $\Phi_\tau(\mathbf{w})$, respectively
W	The objective function of SVR model in dual form
$\mathbf{I}$ and $\mathbf{I}^*$	The identity matrix and the augmented identity matrix with the first row and first column being 0's, and the rest is the identity matrix
$\mathbf{H}$	The Hessian matrix of the smoothed objective function $\Phi_\tau(\mathbf{w})$
$K(\cdot,\cdot)$	The kernel function
$\mathcal{H}$	Reproducing kernel Hilbert space
$\langle\cdot,\cdot\rangle_{\mathcal{H}}$	The inner product of two vectors in reproducing kernel Hilbert space
$\|f\|_{\mathcal{H}}$	The function norm associated with the reproducing kernel Hilbert space
β	The coefficients associated with the kernel representation of a function in reproducing kernel Hilbert space

$\mathbf{K}$	The kernel matrix generated from the training set
$\mathbf{K}^+$	An $(n+1) \times n$ matrix with the first row of all 1's, and the rest is the kernel matrix $\mathbf{K}$
$\mathbf{K}^*$	Augmented kernel matrix with the first row and column being 0's and the rest is the kernel matrix $\mathbf{K}$
$\mathbf{A}_{\cdot i}$	The i-th column of matrix $\mathbf{A}$

1 Introduction

Support vector regression (SVR) [23, 26] is a widely used regression technique. The popular SVR toolboxes [5, 9, 13] train an SVR model by solving the dual problem of a quadratic programming. Although the decomposition techniques [19, 20] or sequential minimization methods [21] are employed to speed up the training of SVR, the training procedure has time complexity about $O(n^3)$, where n is the training set size. As such, in general, training an SVR model is time consuming, especially for large training set. Therefore, it is highly desirable to develop time-efficient yet accurate enough training algorithms for SVR.

Gradient based optimization methods [2, 27] are easy to implement, and converge fast to at least a local optimum. Thus, it is attempting to apply gradient base methods to train an SVR model. However, the objective function of SVR is not differentiable, which prevents gradient based optimization methods from being applicable. As such, we introduce a smooth approximation to the original loss function of SVR, and we verify that the smooth approximation converges to the original loss function uniformly as we decrease the controlling smoothing parameter. Further, we prove that the minimum point of the smoothed objective function converges to the solution of the original optimization problem of SVR. To minimize the proposed smoothed objective function, conjugate gradient method is employed, in which we gradually decrease the smoothing parameter to make the solution stable. The proposed model and algorithm could be generalized to nonlinear SVR in reproducing kernel Hilbert space with only slight modification.

Extensive experimental results on publicly available datasets show that the proposed Conjugate Gradient based Smooth SVR (CG-SSVR) often yields testing accuracy very close to that of the traditional Quadratic Programming based SVR (QP-SVR). However, CG-SSVR is far more efficient than QP-SVR: for linear model, CG-SSVR is often hundreds of times faster than QP-SVR, and for nonlinear model, CG-SSVR is about 3–10 times faster than QP-SVR, depending on the size of the training set. We also observe that as the training set size gets larger, the speed advantage of CG-SSVR over QP-SVR becomes greater.

The rest of this paper is organized as follows: Sect. 2 briefly reviews the formulation of linear and nonlinear SVR models; Sect. 3 introduces a smooth function to approximate the ϵ-insensitive loss function used by SVR, and the properties of the smooth approximation are also studied; Sect. 4 proposes the smoothly approximated SVR for linear and nonlinear models, and the convergence properties of the smoothed objective function are discussed; based on the smoothed objective function, Sect. 5 proposes a sequential minimization approach in which we gradually decrease the smoothing parameter and minimize the smoothed objective function by conjugate gradient method; Sect. 6 compares the testing accuracy and training time of the proposed CG-SSVR algorithm to those of the traditional QP-SVR on three publicly available real-world datasets; finally, Sect. 7 summarizes this paper and discusses some future research directions.

2 Support vector regression: a brief review

Given training data $\{(\mathbf{x}_i, y_i), i = 1, \ldots, n\}$, with input $\mathbf{x}_i \in \mathbf{R}^p$ and output $y_i \in \mathbf{R}$, suppose a linear regression function is used

$$y = \mathbf{w}'\mathbf{x} + b, \tag{1}$$

where $\mathbf{w} \in \mathbf{R}^p$ is the regression coefficient vector and $b \in \mathbf{R}$ is the intercept.

Support vector regression (SVR) model employs a loss function which is not sensitive if the difference between the observation (i.e., y_i) and the prediction (i.e., $\mathbf{w}'\mathbf{x}_i + b$) is less than a predefined level ϵ. For the linear model in Eq. (1), the SVR model can be obtained by solving the following constrained minimization problem

$$\begin{cases} \min_{\mathbf{w},\xi^*,\xi} & \Phi(\mathbf{w}, \boldsymbol{\xi}^*, \boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}'\mathbf{w} + C\left[\sum_{i=1}^n \xi_i + \sum_{i=1}^n \xi_i^*\right] \\ \text{s.t.} & y_i - \mathbf{w}'\mathbf{x}_i - b \le \epsilon + \xi_i^* \\ & \mathbf{w}'\mathbf{x}_i + b - y_i \le \epsilon + \xi_i \\ & \xi_i \ge 0 \quad \text{and} \quad \xi_i^* \ge 0 \\ & \text{for} \quad i = 1, \ldots, n. \end{cases} \tag{2}$$

In the above formulation, ϵ is the predefined error tolerance, $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_n)'$ and $\boldsymbol{\xi}^* = (\xi_1^*, \ldots, \xi_n^*)'$, where ξ_i and ξ_i^* are the slack variables, which are the part of error which exceeds the error tolerance ϵ. Alternatively, ξ_i and ξ_i^* could be understood as the effort we should spend in order to bring the prediction $(\mathbf{w}'\mathbf{x}_i + b)$ to an ϵ-neighborhood of the observation y_i, if the distance between the prediction and the observation is above the predefined error tolerance. Similar to support vector machine (SVM), the term $\frac{1}{2}\mathbf{w}'\mathbf{w}$

in Eq. (2) measures the complexity of the regression model, and the parameter $C > 0$ balances the model complexity and the error on the training set made by the model.

By using Lagrange multiplier method and the KKT conditions, the above optimization can be solved through its dual problem

$$\begin{cases} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} & W(\boldsymbol{\alpha},\boldsymbol{\alpha}^*) = -\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)\mathbf{x}_i'\mathbf{x}_j \\ & \quad -\epsilon\sum_{i=1}^n(\alpha_i+\alpha_i^*)+\sum_{i=1}^n(\alpha_i-\alpha_i^*)y_i \\ \text{s.t.} & \quad \sum_{i=1}^n(\alpha_i-\alpha_i^*)=0 \\ & 0\le\alpha_i\le C \quad \text{and} \quad 0\le\alpha_i^*\le C \quad \text{for} \quad i=1,\ldots,n. \end{cases} \tag{3}$$

In Eq. (3), $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_n)'$ and $\boldsymbol{\alpha}^* = (\alpha_1^*,\ldots,\alpha_n^*)'$, with α_i and α_i^* being the Lagrange multipliers for the corresponding constraints in Eq. (2). Solving the dual problem, which is a quadratic programming, we can get the linear SVR model as

$$y = \sum_{i=1}^n(\alpha_i-\alpha_i^*)\mathbf{x}_i'\mathbf{x}+b,$$

where the bias term b can be calculated from the set of support vectors. Please refer to [23, 26] for more details.

The optimization problem given in Eq. (3) only depends on the inner product of two vectors. For any kernel function $K(\mathbf{u},\mathbf{v})$ which satisfies Mercer's condition [23, 26], it could be written as the inner product of two transformed feature vectors in high dimensional reproducing kernel Hilbert space. This enables us to fit nonlinear SVR by solving the following optimization problem

$$\begin{cases} \max_{\boldsymbol{\alpha},\boldsymbol{\alpha}^*} & W(\boldsymbol{\alpha},\boldsymbol{\alpha}^*) = -\frac{1}{2}\sum_{i=1}^n\sum_{j=1}^n(\alpha_i-\alpha_i^*)(\alpha_j-\alpha_j^*)K(\mathbf{x}_i,\mathbf{x}_j) \\ & \quad -\epsilon\sum_{i=1}^n(\alpha_i+\alpha_i^*)+\sum_{i=1}^n(\alpha_i-\alpha_i^*)y_i \\ \text{s.t.} & \quad \sum_{i=1}^n(\alpha_i-\alpha_i^*)=0 \\ & 0\le\alpha_i\le C \quad \text{and} \quad 0\le\alpha_i^*\le C \quad \text{for} \quad i=1,\ldots,n. \end{cases} \tag{4}$$

The obtained nonlinear model is

$$y = \sum_{i=1}^n(\alpha_i-\alpha_i^*)K(\mathbf{x}_i,\mathbf{x})+b.$$

Similarly, the bias term b could be calculated from support vectors.

As implemented in popular toolboxes [5, 9, 13], the quadratic programming problems in Eqs. (3) and (4) can be solved by the decomposition methods [19, 20] or sequential minimal optimization [21]. However, the aforementioned algorithms are computationally expensive with time complexity roughly $O(n^3)$. Thus, as the training set gets large, it is very expensive to train an SVR model. As such, fast training algorithms for SVR which can achieve similar accuracy as the quadratic programming method are highly appreciated.

3 Smooth approximation to the ϵ-insensitive loss function

To get rid of the constraints in the primal optimization problem for linear SVR in Eq. (2), using implicit constraints, we rewrite the optimization problem as [10, chap. 12]

$$\min_{\mathbf{w},b} \quad \Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}'\mathbf{w}+C\sum_{i=1}^n V_\epsilon(\mathbf{w}'\mathbf{x}_i+b-y_i), \tag{5}$$

where $V_\epsilon(x)$ is called as ϵ-insensitive loss function which is defined explicitly as

$$V_\epsilon(x) = \begin{cases} 0, & \text{if } |x|<\epsilon, \\ |x|-\epsilon, & \text{otherwise.} \end{cases} \tag{6}$$

The ϵ-insensitive loss function $V_\epsilon(x)$ employed by the SVR model in Eq. (5) is not differentiable at points $x = \pm\epsilon$. The non-differentiability of ϵ-insensitive loss function makes it difficult to apply gradient based optimization methods for fitting the SVR model, although gradient based methods are usually time efficient, easy to implement, and yield at least a local optimum.

Chen et al. [7] introduced a class of smooth functions for nonlinear optimization problems and this idea was applied to find an approximate solution to SVM [15, 29]. The idea of using a smooth function to approximate a non-differentiable objective function was also applied to fitting quantile regression model [30].

In [15, 29], the loss function of SVM is written as

$$(x)_+ = \begin{cases} x, & \text{if } x>0 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

and it is approximated by a smooth function

$$p(x,\tau) = \tau\log\left(1+e^{\frac{x}{\tau}}\right), \tag{8}$$

where τ is a small positive number. We notice that, according to Eq. (7),

$$(|x|-\epsilon)_+ = \begin{cases} |x|-\epsilon, & \text{if } |x|>\epsilon \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

which is exactly the expression of $V_\epsilon(x)$ in Eq. (6). Thus, in this paper, we propose the following smooth function to approximate the ϵ-insensitive loss function for SVR

$$S_{\epsilon,\tau}(x) = \tau\log\left(1+e^{\frac{|x|-\epsilon}{\tau}}\right), \tag{10}$$

where $\tau > 0$ is called as the smoothing parameter.

The properties of function $S_{\epsilon,\tau}(x)$ can be summarized as following:

Proposition 1 *As a function of $\tau > 0$, $S_{\epsilon,\tau}(x)$ is monotonically increasing.*

Proof From the definition of $S_{\epsilon,\tau}(x)$ in Eq. (10), we calculate its derivative with respect to τ as

$$\frac{dS_{\epsilon,\tau}(x)}{d\tau} = \log\left(1+e^{\frac{|x|-\epsilon}{\tau}}\right) - \frac{\exp\left\{\frac{|x|-\epsilon}{\tau}\right\}}{1+\exp\left\{\frac{|x|-\epsilon}{\tau}\right\}}\frac{|x|-\epsilon}{\tau}. \quad (11)$$

If $|x|<\epsilon$, the derivative in Eq. (11) is clearly positive since the first term is positive and the second term is negative. If $|x|>\epsilon$, we have

$$\begin{aligned}\frac{dS_{\epsilon,\tau}(x)}{d\tau} &= \log\left(1+e^{\frac{|x|-\epsilon}{\tau}}\right) - \frac{\exp\left\{\frac{|x|-\epsilon}{\tau}\right\}}{1+\exp\left\{\frac{|x|-\epsilon}{\tau}\right\}}\frac{|x|-\epsilon}{\tau} \\ &> \frac{|x|-\epsilon}{\tau} - \frac{\exp\left\{\frac{|x|-\epsilon}{\tau}\right\}}{1+\exp\left\{\frac{|x|-\epsilon}{\tau}\right\}}\frac{|x|-\epsilon}{\tau} \\ &= \frac{1}{1+\exp\left\{\frac{|x|-\epsilon}{\tau}\right\}}\frac{|x|-\epsilon}{\tau} > 0. \end{aligned} \quad (12)$$

Thus, we always have $\frac{dS_{\epsilon,\tau}(x)}{d\tau} > 0$, and the conclusion follows. □

Proposition 2 *For any $\tau > 0$, $S_{\epsilon,\tau}(x)$ is a strictly convex function in x.*

Proof From the definition of $S_{\epsilon,\tau}(x)$ in Eq. (10), we calculate that

$$\frac{dS_{\epsilon,\tau}(x)}{dx} = \frac{\text{sign(x)}}{1+\exp\left\{-\frac{|x|-\epsilon}{\tau}\right\}}, \quad (13)$$

and

$$\frac{d^2S_{\epsilon,\tau}(x)}{dx^2} = \frac{1}{\tau}\frac{\exp\left\{-\frac{|x|-\epsilon}{\tau}\right\}}{\left(1+\exp\left\{-\frac{|x|-\epsilon}{\tau}\right\}\right)^2} > 0, \quad (14)$$

for any $x \in \mathbf{R}$ since $\tau > 0$, which shows that as a function of x, $S_{\epsilon,\tau}(x)$ is strictly convex everywhere. □

Proposition 3 *For any $\tau > 0$ and any $x \in \mathbf{R}$,*

$$0 < S_{\epsilon,\tau}(x) - V_\epsilon(x) \le \tau \log 2 \quad (15)$$

Furthermore, $S_{\epsilon,\tau}(x)$ converges uniformly to $V_\epsilon(x)$ as $\tau \to 0^+$.

Proof Since the ϵ-insensitive loss function $V_\epsilon(x)$ and the smoothed function $S_{\epsilon,\tau}(x)$ are both even functions, we only need to prove Eq. (15) for $x > 0$.

For $0 < x \le \epsilon$, it is straightforward that

$$S_{\epsilon,\tau}(x) - V_\epsilon(x) = \tau \log\left(1+e^{\frac{x-\epsilon}{\tau}}\right) > 0.$$

The difference function is clearly monotonically increasing with maximum at $x = \epsilon$. Therefore, in this case

$$S_{\epsilon,\tau}(x) - V_\epsilon(x) \le \tau \log 2.$$

For $x \ge \epsilon$, we have

$$\begin{aligned} S_{\epsilon,\tau}(x) - V_\epsilon(x) &= \tau \log\left(1+e^{\frac{x-\epsilon}{\tau}}\right) - (x-\epsilon) \\ &= \tau \log\left(1+e^{-\frac{x-\epsilon}{\tau}}\right) > 0. \end{aligned}$$

The above difference function is monotonically decreasing in x, with the maximum at $x = \epsilon$. Consequently, it follows that

$$S_{\epsilon,\tau}(x) - V_\epsilon(x) \le \tau \log 2.$$

Thus, Eq. (15) is proved.

It follows from Eq. (15) that for any $\varepsilon > 0$, we can make τ sufficiently small (independent of x) such that $|S_{\epsilon,\tau}(x) - V_\epsilon(x)| < \varepsilon$, and this completes the proof for uniform convergence. □

Figure 1a shows the ϵ-insensitive loss function with $\epsilon = 1$ and the corresponding smoothed versions with smoothing parameter $\tau = 0.4$ and $\tau = 0.5$, respectively. Figure 1a clearly demonstrates that the smoothed function is always positive, smooth, convex, and dominates the original ϵ-insensitive loss function. Figure 1b gives the difference between the smoothed and the original ϵ-insensitive loss function, and it is readily observed that the approximation

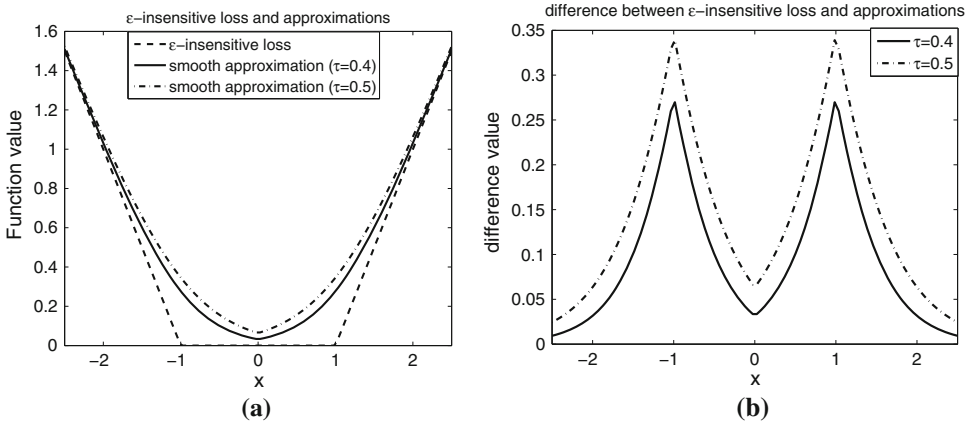

Fig. 1 **a** The ϵ-insensitive loss function (with $\epsilon = 1$) and the smoothed approximations with $\tau = 0.4$ and $\tau = 0.5$; **b** the differences between the smoothed and the original ϵ-insensitive loss function in **a**

is better for small value of τ Fig. 1b also shows that the largest difference occurs at $x = \pm 1$, that is, $x = \pm\epsilon$. These observations are consistent with the properties studied in Propositions 1–3.

4 Smoothly approximated support vector regression

This section first introduces a smoothly approximated model to linear SVR, then generalizes to nonlinear SVR in reproducing kernel Hilbert space.

4.1 Smoothed objective function for linear SVR

For the linear SVR model

$$y = \mathbf{w}'\mathbf{x} + b, \tag{16}$$

for convenience, we rewrite the primal optimization problem given in Eq. (5) as

$$\min_{\mathbf{w}} \quad \Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}'\mathbf{w} + C\sum_{i=1}^{n} V_\epsilon(\mathbf{w}'\mathbf{x}_i + b - y_i), \tag{17}$$

where $V_\epsilon(\cdot)$ is defined in Eq. (6). In order to incorporate the bias term b, we augment the predictor vector by adding 1 as the first component; correspondingly, we augment $\mathbf{w}$ by adding b as the first component. To simplify the notation, we still denote the augmented predictor and regression coefficient vector as $\mathbf{x}$ and $\mathbf{w}$, respectively. With these considerations, the objective function in Eq. (17) can be written as

$$\Phi(\mathbf{w}) = \frac{1}{2}\mathbf{w}'\mathbf{I}^*\mathbf{w} + C\sum_{i=1}^{n} V_\epsilon(\mathbf{w}'\mathbf{x}_i - y_i), \tag{18}$$

where $\mathbf{I}^*$ is the augmented identity matrix with the first row and first column being 0's, and the rest is the $n \times n$ identity matrix $\mathbf{I}$.

The SVR model could be fitted by minimizing the primal objective function in Eq. (18). However, as the ϵ-insensitive loss function in Eq. (18) is not differentiable, the gradient based optimization methods cannot be applied. It is well known that the gradient based methods are easy to implement and converge fast to at least a local optimal point. In order to make use of the advantages of the gradient based optimization methods, we replace the ϵ-insensitive loss function by its smoothed counterpart $S_{\epsilon,\tau}(x)$ defined in Eq. (10), yielding the smoothed objective function

$$\Phi_\tau(\mathbf{w}) = \frac{1}{2}\mathbf{w}'\mathbf{I}^*\mathbf{w} + C\sum_{i=1}^{n} S_{\epsilon,\tau}(\mathbf{w}'\mathbf{x}_i - y_i) \tag{19}$$

We calculate the gradient vector of the smooth approximation as

$$\nabla\Phi_\tau(\mathbf{w}) = \mathbf{I}^*\mathbf{w} + C\sum_{i=1}^{n} \frac{\mathrm{sign}(\mathbf{w}'\mathbf{x}_i - y_i)}{1 + \exp\left\{-\frac{|\mathbf{w}'\mathbf{x}_i - y_i| - \epsilon}{\tau}\right\}}\mathbf{x}_i, \tag{20}$$

and the Hessian matrix of the smooth approximation as

$$\begin{aligned}\mathbf{H}(\mathbf{w}) &= \nabla^2\Phi_\tau(\mathbf{w}) \\ &= \mathbf{I}^* + \frac{C}{\tau}\sum_{i=1}^{n} \frac{\exp\left\{-\frac{|\mathbf{w}'\mathbf{x}_i - y_i| - \epsilon}{\tau}\right\}}{\left(1 + \exp\left\{-\frac{|\mathbf{w}'\mathbf{x}_i - y_i| - \epsilon}{\tau}\right\}\right)^2}\mathbf{x}_i\mathbf{x}_i'\end{aligned} \tag{21}$$

It is easy to see that the second term in Eq. (21) is positive definite, thus the Hessian is positive definite. Consequently, the smoothed objective function $\Phi_\tau(\mathbf{w})$ is convex in $\mathbf{w}$, thus, it has a unique minimum point.

By Proposition 3 and the definitions of $\Phi(\mathbf{w})$ and $\Phi_\tau(\mathbf{w})$ in Eqs. (18) and (19), the following is evident.

Proposition 4 *For any $\tau > 0$, the smoothed objective function $\Phi_\tau(\mathbf{w})$ is an upper bound of the original objective function $\Phi(\mathbf{w})$, i.e., $\Phi_\tau(\mathbf{w}) > \Phi(\mathbf{w})$, for any $\mathbf{w}$.*

We can prove that the smoothed objective function approaches the original objective function as the smoothing parameter decreases. More precisely, we have

Proposition 5 *As $\tau \to 0^+$, the smoothed objective function $\Phi_\tau(\mathbf{w})$ uniformly converges to the original objective function $\Phi(\mathbf{w})$.*

Proof For any $\tau > 0$, from the definitions of $\Phi(\mathbf{w})$ and $\Phi_\tau(\mathbf{w})$, we have

$$\begin{aligned}|\Phi_\tau(\mathbf{w}) - \Phi(\mathbf{w})| &= C\left|\sum_{i=1}^{n} S_{\epsilon,\tau}(\mathbf{w}'\mathbf{x}_i - y_i) - \sum_{i=1}^{n} V_\epsilon(\mathbf{w}'\mathbf{x}_i - y_i)\right| \\ &\le C\sum_{i=1}^{n}\left|S_{\epsilon,\tau}(\mathbf{w}'\mathbf{x}_i - y_i) - V_\epsilon(\mathbf{w}'\mathbf{x}_i - y_i)\right| \\ &\le Cn\tau\log 2,\end{aligned} \tag{22}$$

where the last inequality follows from Eq. (15).

Thus, for any $\varepsilon > 0$, we can make τ sufficiently small (independent of $\mathbf{w}$) such that $|\Phi_\tau(\mathbf{w}) - \Phi(\mathbf{w})| < \varepsilon$, and this finishes the proof for the uniform convergence. □

We further prove that the minimal value of the smoothed objective function $\Phi_\tau(\mathbf{w})$ goes to that of the original objective function $\Phi(\mathbf{w})$ as the smoothing parameter decreases. In detail, we have

Proposition 6 *For any $\tau > 0$, let*

$$\hat{\mathbf{w}}_\tau = \arg\min_{\mathbf{w}} \Phi_\tau(\mathbf{w}) \quad \textit{and} \quad \hat{\mathbf{w}} = \arg\min_{\mathbf{w}} \Phi(\mathbf{w}),$$

then

$$\lim_{\tau\to 0^+} \Phi_\tau(\hat{\mathbf{w}}_\tau) = \Phi(\hat{\mathbf{w}}).$$

Proof From the definitions of $\hat{\mathbf{w}}_\tau$ and $\hat{\mathbf{w}}$, it follows that

$$\Phi_\tau(\hat{\mathbf{w}}) \geq \Phi_\tau(\hat{\mathbf{w}}_\tau) \quad \text{and} \quad \Phi(\hat{\mathbf{w}}_\tau) \geq \Phi(\hat{\mathbf{w}}).$$

Applying the conclusion in Proposition 4, we have

$$\Phi_\tau(\hat{\mathbf{w}}) \geq \Phi_\tau(\hat{\mathbf{w}}_\tau) \geq \Phi(\hat{\mathbf{w}}_\tau) \geq \Phi(\hat{\mathbf{w}}). \tag{23}$$

By Proposition 5, as a function of τ, $\Phi_\tau(\hat{\mathbf{w}})$ converges uniformly to $\Phi(\hat{\mathbf{w}})$, that is,

$$\lim_{\tau \to 0^+} \Phi_\tau(\hat{\mathbf{w}}) = \Phi(\hat{\mathbf{w}}). \tag{24}$$

The conclusion of the Proposition follows by combining Eqs. (24) and (23). □

Finally, we verify that the minimum point of the smoothed objective function approaches to that of the original objective function of linear SVR.

Proposition 7 *With $\hat{\mathbf{w}}_\tau$ and $\hat{\mathbf{w}}$ defined in Proposition 6, let $\hat{\mathbf{w}}_\tau^u$ and $\hat{\mathbf{w}}^u$ be the corresponding unaugmented parts (i.e., with the bias term excluded), then*

$$\lim_{\tau \to 0^+} \hat{\mathbf{w}}_\tau^u = \hat{\mathbf{w}}^u.$$

Proof Since $\hat{\mathbf{w}}_\tau$ is the minimum point of $\Phi_\tau(\mathbf{w})$, by the convexity of $\Phi_\tau(\mathbf{w})$, the gradient $\nabla \Phi_\tau(\hat{\mathbf{w}}_\tau) = \mathbf{0}$. Using the second order Taylor approximation of $\Phi_\tau(\hat{\mathbf{w}})$ at $\hat{\mathbf{w}}_\tau$, there is a $\theta \in [0, 1]$, such that

$$\begin{aligned}\Phi_\tau(\hat{\mathbf{w}}) - \Phi_\tau(\hat{\mathbf{w}}_\tau) &= \frac{1}{2}(\hat{\mathbf{w}} - \hat{\mathbf{w}}_\tau)' \mathbf{H}(\theta \hat{\mathbf{w}} + (1-\theta)\hat{\mathbf{w}}_\tau))(\hat{\mathbf{w}} - \hat{\mathbf{w}}_\tau) \\ &\geq \frac{1}{2}(\hat{\mathbf{w}} - \hat{\mathbf{w}}_\tau)' \mathbf{I}^*(\hat{\mathbf{w}} - \hat{\mathbf{w}}_\tau) = \frac{1}{2}\|\hat{\mathbf{w}}^u - \hat{\mathbf{w}}_\tau^u\|^2.\end{aligned} \tag{25}$$

The " $\geq$ " in Eq. (25) holds because in the expression of the Hessian matrix given by Eq. (21), the second term is semi-positive definite.

From Eq. (25), it follows that

$$\lim_{\tau \to 0^+} \|\hat{\mathbf{w}}^u - \hat{\mathbf{w}}_\tau^u\|^2 \leq 2 \lim_{\tau \to 0^+} [\Phi_\tau(\hat{\mathbf{w}}) - \Phi_\tau(\hat{\mathbf{w}}_\tau)] = 0, \tag{26}$$

since $\lim_{\tau \to 0^+} \Phi_\tau(\hat{\mathbf{w}}_\tau) = \Phi(\hat{\mathbf{w}})$ from Proposition 6 and $\lim_{\tau \to 0^+} \Phi_\tau(\hat{\mathbf{w}}) = \Phi(\hat{\mathbf{w}})$ from Proposition 5. Thus, $\lim_{\tau \to 0^+} \hat{\mathbf{w}}_\tau^u = \hat{\mathbf{w}}^u$ immediately follows from Eq. (26). □

4.2 Generalizing to nonlinear model

For a nonlinear SVR model associated with a kernel function $K(\cdot, \cdot)$ which satisfies Mercer's condition [22, 23, 26], let the associated reproducing kernel Hilbert space be $\mathcal{H}$, and let f be the SVR regression function. The regression function could be obtained by minimizing the following objective function in $\mathcal{H}$ [22, 23, 26]

$$\|f\|_{\mathcal{H}}^2 + C \sum_{i=1}^{n} V_\epsilon(f(\mathbf{x}_i) - y_i), \tag{27}$$

where $\|f\|_{\mathcal{H}}^2$ is the function norm associated with the reproducing kernel Hilbert space $\mathcal{H}$, and it could be understood as the model complexity, similar to the term $\mathbf{w}'\mathbf{w}/2$ in the linear SVR model.

By the representer theorem [14, 22], the regression function could be written as a linear combination of kernel functions evaluated at the training examples. That is, the solution is of the form

$$f(\mathbf{x}) = \sum_{j=1}^{n} \beta_j K(\mathbf{x}_j, \mathbf{x}) + b. \tag{28}$$

In the reproducing kernel Hilbert space, the model complexity term is

$$\begin{aligned}\|f\|_{\mathcal{H}}^2 &= \sum_{i,j=1}^{n} \beta_i \beta_j \langle K(\mathbf{x}_i, \cdot), K(\mathbf{x}_j, \cdot) \rangle_{\mathcal{H}} = \sum_{i,j=1}^{n} \beta_i \beta_j K(\mathbf{x}_i, \mathbf{x}_j) \\ &= \boldsymbol{\beta}' \mathbf{K} \boldsymbol{\beta},\end{aligned}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ is the inner product of two vectors in space $\mathcal{H}$; according to the property of the reproducing space, the inner product could be written as the kernel function; $\mathbf{K}$ is the $n \times n$ kernel matrix with $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$, and $\boldsymbol{\beta} = (\beta_1, \cdots, \beta_n)'$.

We write the estimated function value at $\mathbf{x}_i$ as

$$f(\mathbf{x}_i) = \sum_{j=1}^{n} \beta_j K(\mathbf{x}_j, \mathbf{x}_i) + b = \boldsymbol{\beta}^{+\prime} \mathbf{K}_{\cdot i}^{+},$$

where $\mathbf{K}^+$ is an $(n+1) \times n$ matrix with the first row of all 1's, and the other part of $\mathbf{K}^+$ is the original kernel matrix $\mathbf{K}$; we use the notation $\mathbf{A}_{\cdot i}$ to represent the i-th column of matrix $\mathbf{A}$; and $\boldsymbol{\beta}^+ = (b, \beta_1, \cdots, \beta_n)'$. Let $\mathbf{K}^*$ be the augmented kernel matrix of size $(n+1) \times (n+1)$, with the first row and first column being 0's, and the other part being the original kernel matrix $\mathbf{K}$. It is clear that $\boldsymbol{\beta}^{+\prime} \mathbf{K}^* \boldsymbol{\beta}^+ = \boldsymbol{\beta}' \mathbf{K} \boldsymbol{\beta}$.

With these notations, the objective function of the primal problem in Eq. (27) for nonlinear SVR is

$$\Phi(\boldsymbol{\beta}^+) = \frac{1}{2} \boldsymbol{\beta}^{+\prime} \mathbf{K}^* \boldsymbol{\beta}^+ + C \sum_{i=1}^{n} V_\epsilon(\boldsymbol{\beta}^{+\prime} \mathbf{K}_{\cdot i}^{+} - y_i). \tag{29}$$

Introducing the smooth approximation to the ϵ-insensitive loss function, yielding the smoothed objective function for nonlinear SVR as

$$\Phi_\tau(\boldsymbol{\beta}^+) = \frac{1}{2} \boldsymbol{\beta}^{+\prime} \mathbf{K}^* \boldsymbol{\beta}^+ + C \sum_{i=1}^{n} S_{\epsilon,\tau}(\boldsymbol{\beta}^{+\prime} \mathbf{K}_{\cdot i}^{+} - y_i). \tag{30}$$

We calculate the gradient vector as

$$\nabla \Phi_\tau(\boldsymbol{\beta}^+) = \mathbf{K}^* \boldsymbol{\beta}^+ + C \sum_{i=1}^{n} \frac{\text{sign}(\boldsymbol{\beta}^{+\prime} \mathbf{K}_{\cdot i}^{+} - y_i)}{1 + \exp\left\{ -\frac{|\boldsymbol{\beta}^{+\prime} \mathbf{K}_{\cdot i}^{+} - y_i| - \epsilon}{\tau} \right\}} \mathbf{K}_{\cdot i}^{+}, \tag{31}$$

and the Hessian matrix is

$$\mathbf{H}(\boldsymbol{\beta}^+) = \nabla^2 \Phi_\tau(\boldsymbol{\beta}^+) = \mathbf{K}^* + \frac{C}{\tau}\sum_{i=1}^{n} \frac{\exp\left\{-\frac{|\boldsymbol{\beta}^{+\prime}\mathbf{K}_i^+ - y_i| - \epsilon}{\tau}\right\}}{\left(1+\exp\left\{-\frac{|\boldsymbol{\beta}^{+\prime}\mathbf{K}_i^+ - y_i| - \epsilon}{\tau}\right\}\right)^2} \mathbf{K}_{:i}^+ \mathbf{K}_{:i}^{+\prime}. \tag{32}$$

It is easy to verify that the Hessian matrix is positive definite, thus the function $\Phi_\tau(\boldsymbol{\beta}^+)$ is convex. Consequently, $\Phi_\tau(\boldsymbol{\beta}^+)$ has a unique minimum point.

Comparing the optimization problems in Eqs. (19) and (30), we clearly observe that they share the same formulation. Thus, the properties stated in Propositions 4–7 also hold for $\Phi_\tau(\boldsymbol{\beta}^+)$. The similar formulations and properties of the optimization problems shared by linear and nonlinear smooth SVR models indicate that they can be solved by the same algorithm with only slight modifications.

5 The conjugate gradient algorithm

In applying the idea of smoothly approximating a non-smooth objective function, some works [6, 15] used the Newton's method to minimize the smoothed objective function. However, the Newton's method involves estimating and inverting the Hessian matrix, which is time consuming (with time complexity about $O(n^3)$) and prone to errors, especially in high dimensional spaces. Compared to the Newton's method, conjugate gradient method avoids using the second order derivative information and inverting a matrix, and it only has a simple formula to determine the new search direction. This simplicity makes the method very easy to implement, only slightly more complicated than steepest descent. Other advantages of the conjugate gradient method include its low memory requirements and its convergence speed. Please refer to [2, chap. 1] for more details.

There are many choices of conjugate directions for conjugate gradient methods, for example, Fletcher–Reeves, Polak–Rbiere, and Hestenes–Stiefel. In our experiments, we empirically found these methods perform similarly. Algorithm 1 gives the Fletcher–Reeves conjugate gradient method for minimizing a general function f over $\mathbf{R}^n$.

0. Initialization: choose a starting point $\mathbf{x}_0 \in \mathbf{R}^n$, compute $\mathbf{g}_0 = \nabla f(\mathbf{x}_0)$ and $\mathbf{d}_0 = -\mathbf{g}_0$. Denote the iteration number as m.
1. **for** $t = 1$ to m **do**:
2. If $\mathbf{g}_{t-1} = 0$, terminate and return $\mathbf{x}_{t-1}$ as the minimum point of f.
3. Set $\mathbf{x}_t = \mathbf{x}_{t-1} + \gamma_t \mathbf{d}_{t-1}$, where γ_t is the step-size at iteration t.
4. Compute $\mathbf{g}_t = \nabla f(\mathbf{x}_t)$, and set $\mathbf{d}_t = -\mathbf{g}_t + \delta_t \mathbf{d}_{t-1}$, with $\delta_t = \frac{\mathbf{g}_t' \mathbf{g}_t}{\mathbf{g}_{t-1}' \mathbf{g}_{t-1}}$.
5. **end for**
6. Return $\mathbf{x}_m$ as the minimum point of f.

Algorithm 1 Fletcher-Reeves Conjugate Gradient Method for Minimizing aFunction f over $\mathbf{R}^n$

In the 3rd step, the step-size γ_t could be chosen as

$$\gamma_t = \arg\min_{\gamma > 0} f(\mathbf{x}_{t-1} + \gamma \mathbf{d}_{t-1}). \tag{33}$$

The minimization problem in Eq. (33) can be solved by backtracking line search algorithms [3, chap. 9]. We choose to use Armijo rule [1] for its simplicity, which is given in Algorithm 2 for completeness.

0. Given the objective function $f(\mathbf{x})$, the current estimation $\mathbf{x}_c$, the current gradient vector $\nabla f(\mathbf{x}_c)$, and the search direction $\mathbf{d}$.
1. Initialize $\eta = 1$.
2. Calculate $D = f(\mathbf{x}_c + \eta \mathbf{d}) - f(\mathbf{x}_c)$.
3. If $D \leq \frac{\eta}{4} \nabla f(\mathbf{x}_c)' \mathbf{d}$, return the current η; otherwise, set $\eta = \eta/2$, and go back to step 2.

Algorithm 2 Armijo Rule to Determine a Step-size

When applying the conjugate gradient method to minimize the objective function in Eq. (19) or Eq. (30), according to the properties of the smooth approximation, we should have a small τ value. However, as studied in [29], if we start with a very small τ, the algorithm will be unstable. Starting from a relatively large τ and decreasing it gradually will stablize the solution. In practice, we do not decrease τ after each conjugate gradient step, because we should let the conjugate gradient algorithm run several iterations to fully utilize its power in minimizing the objective function at the current τ value. As such, we adopt a sequential minimization strategy for solving the smoothly approximated SVR. Algorithm 3 sketches the procedure for minimizing the objective function of the smoothly approximated linear SVR presented in Eq. (19). The corresponding algorithm for the smoothly approximated nonlinear SVR could be developed similarly, which is omitted.

0. Initialize $\mathbf{w}$ as a random vector, set the maximum outer iteration number M, and the conjugate gradient iteration m.
1. **for** $i = 1$ to M **do**:
2. Set $\tau = 1/i$.
3. Minimize $\Phi_\tau(\mathbf{w})$ with m iterations of conjugate gradient algorithm, with the current estimated $\mathbf{w}$ as the initial point.
4. **end for**

Algorithm 3 Conjugate Gradient Method for Smoothly Approximated LinearSVR

According to Proposition 3, when $\tau = 0.04$, the difference between the smoothed ϵ-insensitive loss function and the original version is less than 0.028, which is small enough for most applications. As such, in the above algorithm, we set the outer iteration number M to be 25 for linear model, and set M to be 30 for nonlinear model. In our experiments, we set the inner iteration number m to be 20 for linear model, and set m to be 25 for nonlinear model. Thus, the algorithm will run totally 500 iterations of conjugate gradient for linear model, and will run 750 iterations

Table 1 The performance of the proposed CG-SSVR and the quadratic programming based SVR (QP-SVR) on the "cpuSmall Prototask" using comp-activ dataset

N_{tr}	Kernel	Testing Error		Training Time	
		CG-SSVR	QP-SVR	CG-SSVR	QP-SVR
300	Linear	0.0779 (0.0114)	0.0785 (0.0111)	0.6955 (0.0157)	84.4166 (1.7787)
	Gaussian	0.0393 (0.0071)	0.0350 (0.0074)	10.9986 (0.0350)	84.6061 (0.9189)
600	Linear	0.0723 (0.0062)	0.0723 (0.0060)	2.0186 (0.0366)	664.2234 (9.9379)
	Gaussian	0.0331 (0.0047)	0.0276 (0.0053)	68.3052 (0.8401)	668.6166 (6.4621)
900	Linear	0.0701 (0.0036)	0.0700 (0.0035)	4.422 (0.0619)	2305 (26.44)
	Gaussian	0.0310 (0.0026)	0.0264 (0.0035)	230.8 (0.7892)	2317 (23.47)

For different training set sizes (N_{tr}), we list the mean values of the ϵ-insensitive testing errors of 100 runs with the standard deviations listed in parentheses. The average training time and standard deviation (in seconds) for different algorithms and models are also given

for nonlinear model. Of course, setting the parameters M and m to be larger values will make the algorithm more precise but the algorithm will spend more time as well.

6 Experimental results

On three publicly available datasets, this section compares the performance of the proposed algorithm, Conjugate Gradient based Smooth SVR (CG-SSVR), to that of the Quadratic Programming based SVR (QP-SVR). The CG-SSVR algorithm is developed using MATLAB[1], without particular code optimization; QP-SVR was implemented based on the MATLAB SVM toolbox of [9] with the quadratic programming solver from the C++ version of LIBSVM [5]. All the experiments were conducted on a personal computer with Pentium IV CPU 3.00 GHz and 3.25 GB memory, with WinXP operating system and MATLAB® R2007b as the platform.

We compared CG-SSVR and QP-SVR with linear kernel and with the Gaussian kernel

$$K(\mathbf{u},\mathbf{v}) = \exp\left\{-\frac{\|\mathbf{u}-\mathbf{v}\|^2}{2\sigma^2}\right\}$$

with $\sigma = 5$. We set the SVR parameters $C = 2{,}000$ and $\epsilon = 0.5$ in the algorithms. The parameter setting in our experiment might not be the optimal to achieve the smallest testing error. However, our purpose is not to achieve the optimal testing error, but to compare the performances of CG-SSVR and QP-SVR; therefore, the comparison is fair as long as the parameter settings are the same for the two algorithms.

For each algorithm and each of the linear and nonlinear models, we partition the whole dataset as training subset and testing subset, and use the average of ϵ-insensitive error on the testing set to evaluate the prediction accuracy, that is

$$\text{Error} = \frac{1}{N_{\text{test}}}\sum_{i=1}^{N_{\text{test}}} V_\epsilon(y_i - f(\mathbf{x}_i)), \tag{34}$$

where $V_\epsilon(\cdot)$ is defined as in Eq. (6), $f(\cdot)$ is the fitted function, and N_{test} is the size of testing set. We use the training time as the measure of time complexity for each algorithm.

6.1 The comp-activ dataset

The comp-activ dataset is a collection of a computer system's activity measures, please refer to http://www.cs.toronto.edu/%7Edelve/data/comp-activ/compActivDetail.html for detailed description about this dataset. The dataset contains 8,192 data points and 25 numerical attributes. We normalize the predictors and response to have zero mean and unit standard deviations.

We first implement the "cpuSmall Prototask" which involves using 12 of the attributes to predict what fraction of a CPU's processing time is devoted to a specific mode ("user mode"). For different training set sizes (300, 600, and 900), we randomly partition the whole dataset into training set and testing set, and evaluate the prediction accuracy of the trained model on the testing set. The partition-training-evaluating process is repeated 100 times for each algorithm. Table 1 gives the average testing errors and the average training time for different algorithms and models along with corresponding standard deviations. It is evident from Table 1 that when using linear kernel, the CG-SSVR algorithm and QP-SVR have essentially the same testing errors; when using the Gaussian kernel, CG-SSVR only has slightly larger testing errors (by about 0.005 in average).

We repeat the same experiment on the "cpu Prototask" which involves using 21 predictors. Table 2 gives the average testing error and the average training time for

[1] The source code is available upon request.

Table 2 The performance measures of CG-SSVR and QP-SVR on "cpu Prototask", using the comp-activ dataset

N_{tr}	Kernel	Testing Error		Training Time	
		CG-SSVR	QP-SVR	CG-SSVR	QP-SVR
300	Linear	0.0835 (0.0136)	0.0860 (0.0141)	0.7864 (0.0129)	85.7356 (2.0314)
	Gaussian	0.0507 (0.0081)	0.0523 (0.0084)	10.9355 (0.0469)	84.4198 (0.9393)
600	Linear	0.0731 (0.0094)	0.0733 (0.0091)	2.3234 (0.0214)	673.9525 (12.4385)
	Gaussian	0.0426 (0.0045)	0.0407 (0.0047)	67.8063 (0.1395)	664.8266 (6.2866)
900	Linear	0.0694 (0.0053)	0.0695 (0.0051)	5.160 (0.0315)	2331.1 (39.0994)
	Gaussian	0.0405 (0.0036)	0.0374 (0.0042)	229.64 (0.7705)	2308.2 (16.4527)

See the footer of Table 1 for more information

different algorithms and different training set sizes, along with the corresponding standard deviation. From Table 2, we observe that, when using linear kernel, the proposed CG-SSVR yields slightly smaller testing error than QP-SVR, although they are on the same level; for the model with Gaussian kernel, the testing error of CG-SSVR is only slightly larger (by about 0.002 in average) than that of QP-SVR.

In order to clearly illustrate the speed advantage of CG-SSVR, we calculate the ratio between the training time of QP-SVR and that of CG-SSVR for different models and different training set sizes, and list the results in Table 3. On the two experiments, we observe that CG-SSVR with linear kernel runs hundreds of times faster than the corresponding QP-SVR, and CG-SSVR with Gaussian kernel runs about 8–10 times faster, depending on the training set size. For linear model, the number of parameters which CG-SSVR needs to estimate is the same as the dimensionality of the data, as indicated by the expression of the fitted model given in Eq. (16). Since the "cpuSmall Prototask" uses fewer predictors than the "cpu Prototask", the speed advantage of CG-SSVR on "cpuSmall Prototask" would be more significant, as observed from the first row of Table 3. On the other hand, for nonlinear model, the speed advantage of CG-SSVR is not sensitive to the dimensionality of data, only depending on the training set size, as indicated by the second row of Table 3. This is because for nonlinear model, the number of parameters that CG-SSVR needs to estimate is the same as the training set size, as seen from the expression of the desired function given by Eq. (28). It is also clear that as the training set size grows, the speed advantage of CG-SSVR increases, and this pattern is especially prominent for linear model. More importantly, we should mention that in our implementation, the core quadratic programming code for QP-SVR was developed in C++ which is much more computationally efficient than MATLAB, in which CG-SSVR was implemented. Taking this factor into account, the proposed CG-SSVR would be far more efficient than QP-SVR, if they were implemented in the same programming language and ran on the same platform.

Table 3 For the two prototasks on comp-activ dataset, the ratio between the training time spent by QP-SVR and CG-SSVR for different models and different training set sizes

Experiment	"cpuSmall Prototask"			"cpu Prototask"		
N_{tr}	300	600	900	300	600	900
Linear Kernel	121.38	329.05	521.26	109.02	290.07	451.76
Gaussian Kernel	7.69	9.79	10.04	7.72	9.80	10.05

For one of the 100 runs of the algorithms on "cpuSmall Prototask", Fig. 2 presents the evolution of the objective functions (calculated according to Eq. (19) or Eq. (30)), the training and testing errors (calculated as in Eq. (34)) for CG-SSVR. Figure 2a and b also plot the final objective function values of QP-SVR (evaluated by Eq. (18) or Eq. (29)). Figure 2a shows that with linear model, the CG-SSVR algorithm converges in about 300 iterations; while Fig. 2b indicates that with nonlinear kernel, CG-SSVR needs about 600 iterations to converge due to a larger number of parameters to estimate. Figure 2c and d demonstrate that as the training proceeds, both the training error and testing error monotonically decrease, and the nonlinear model has lower training/testing error, compared to the linear model.

6.2 Concrete compressive strength data

Concrete is the most important material in civil engineering. The concrete compressive strength is a function of Age (1–365 days) and certain ingredients, which include Cement, Blast Furnace Slag, Fly Ash, Water, Superplasticizer, Coarse Aggregate, and Fine Aggregate. The actual concrete compressive strength for a given mixture under a specific age (days) was determined from laboratory. The dataset along with a brief description can be downloaded from UCI machine learning repository at http://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength. The

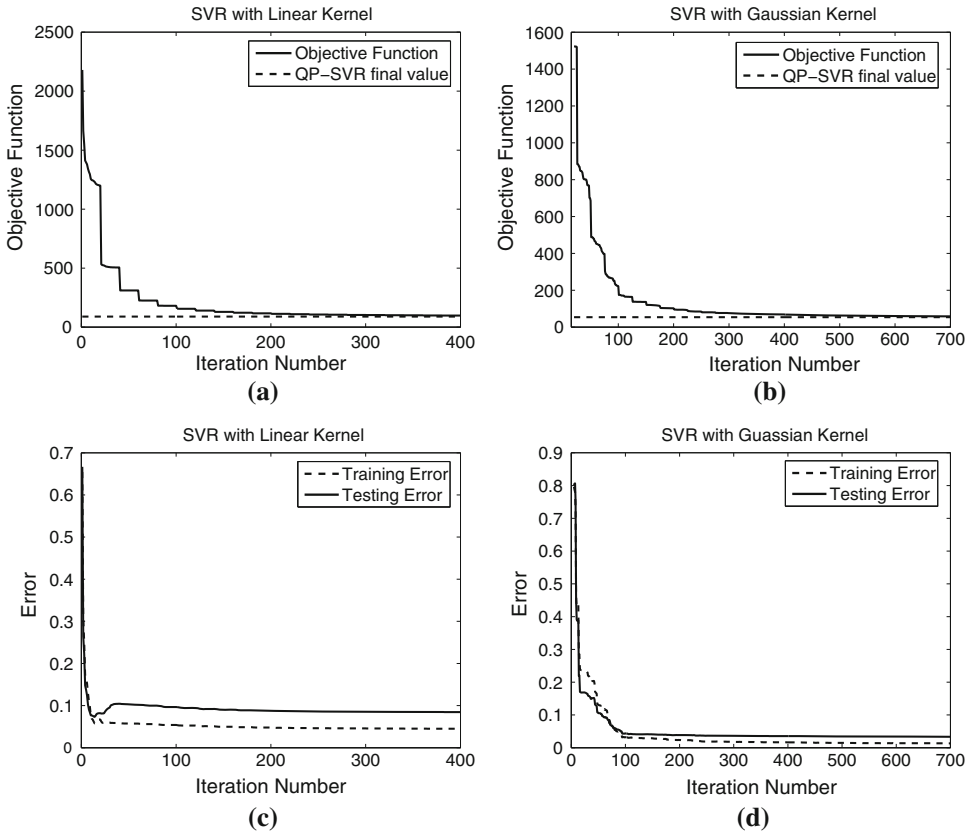

Fig. 2 The evolution curves for CG-SSVR algorithm with training set size 300, on the "cpuSmall Prototask": **a** the objective function of CG-SSVR with linear model; **b** the objective function of CG-SSVR with Gaussian kernel; **c** the training/testing errors of linear CG-SSVR; **d** the training/testing errors of Gaussian CG-SSVR

Table 4 The performance measures of the proposed CG-SSVR and QP-SVR on the concrete strength dataset

N_{tr}	Kernel	Testing Error		Training Time	
		CG-SSVR	QP-SVR	CG-SSVR	QP-SVR
200	Linear	0.1599 (0.0096)	0.1607 (0.0098)	0.3567 (0.0106)	24.2981 (0.2378)
	Gaussian	0.0916 (0.0081)	0.0855 (0.0083)	4.1997 (0.0197)	24.4755 (0.1674)
400	Linear	0.1535 (0.0074)	0.1539 (0.0076)	0.9463 (0.0117)	187.3117 (1.6807)
	Gaussian	0.0835 (0.0060)	0.0753 (0.0059)	22.9003 (0.0486)	188.8853 (0.9353)
600	Linear	0.1517 (0.0084)	0.1518 (0.0084)	1.8055 (0.0237)	629.0577 (5.3291)
	Gaussian	0.0813 (0.0067)	0.0727 (0.0066)	68.3958 (0.2583)	634.5108 (2.6732)
800	Linear	0.1485 (0.0169)	0.1484 (0.0168)	3.124 (0.0455)	1,510.3 (11.8654)
	Gaussian	0.0780 (0.0096)	0.0702 (0.0090)	169.0244 (0.3874)	1,525.1 (6.2058)

See the footer of Table 1 for more information

dataset includes 1,030 observations, with 8 quantitative predictor variables (the 8 factors mentioned above), and 1 quantitative response variable (concrete compressive strength). Please refer to [28] for more details about the dataset. We normalize the predictor variables and the response variable to have zero mean and unit standard deviation.

We repeat the experiment in Sect. 6.1 with training set sizes 200, 400, 600, and 800. Table 4 reports the testing accuracy and the training time of the proposed CG-SSVR and QP-SVR for different training set sizes.

Table 5 On the concrete strength dataset, the ratio between the training time spent by QP-SVR and CG-SSVR for different models and different training set sizes

N_{tr}	200	400	600	800
Linear Kernel	68.12	197.94	348.41	483.49
Gaussian Kernel	5.83	8.25	9.28	9.02

We clearly observe that the testing accuracy of CG-SSVR linear model is at the same level as that of QP-SVR with linear kernel; for the Gaussian model,

Table 6 The performance measures of the proposed CG-SSVR and QP-SVR on the Boston house dataset

N_{tr}	Kernel	Testing error		Training time	
		CG-SSVR	QP-SVR	CG-SSVR	QP-SVR
100	Linear	0.1128 (0.0158)	0.1170 (0.0170)	0.2453 (0.0081)	3.5147 (0.0639)
	Gaussian	0.0733 (0.0130)	0.0773 (0.0161)	1.2233 (0.0154)	3.5081 (0.0670)
200	Linear	0.0979 (0.0119)	0.0998 (0.0117)	0.3891 (0.0087)	25.1167 (0.3063)
	Gaussian	0.0580 (0.0104)	0.0541 (0.0113)	4.1536 (0.0235)	25.1569 (0.3294)
300	Linear	0.0924 (0.0170)	0.0936 (0.0171)	0.6933 (0.0090)	82.5833 (0.8637)
	Gaussian	0.0512 (0.0144)	0.0457 (0.0134)	11.0211 (0.0656)	82.6587 (0.7135)
400	Linear	0.0860 (0.0282)	0.0869 (0.0284)	1.0216 (0.0191)	193.1191 (1.9420)
	Gaussian	0.0456 (0.0226)	0.0399 (0.0204)	22.7733 (0.0408)	193.7238 (0.9853)

See the footer of Table 1 for more information

Table 7 On the Boston house dataset, the ratio between the training time spent by QP-SVR and CG-SSVR for different models and different training set sizes

N_{tr}	100	200	300	400
Linear Kernel	14.33	64.55	119.12	189.04
Gaussian Kernel	2.87	6.06	7.50	8.51

QP-SVR performs slightly better than CG-SSVR, with advantage only about 0.008 in average. Table 5 presents the ratio of the training times of QP-SVR and CG-SSVR for different models and different training set sizes. We again observe that CG-SSVR with linear kernel is hundreds of time faster than the QP-SVR counterpart, and CG-SSVR is multiple times faster than QP-SVR for the nonlinear SVR model.

6.3 Boston house dataset

The Boston house dataset is available online at http://lib.stat.cmu.edu/datasets/boston_corrected.txt. This dataset concerns the median house price in suburb of Boston area, and there are 12 non-constant continuous predictor variables. We normalize the predictor variables and the response variable to have zero mean and unit standard deviation.

We repeat the experiment in Sect. 6.1 with training set sizes 100, 200, 300, and 400, and Table 6 reports the testing accuracy and training time for different models and different training set sizes. We observe that for linear model, the proposed CG-SSVR algorithm has slightly smaller testing errors, and for the nonlinear model with Gaussian kernel, the testing error of CG-SSVR is slightly larger (by only about 0.005) than QP-SVR. Table 7 presents the comparison of the training time of QP-SVR and CG-SSVR, which demonstrates the same pattern as in Tables 3 and 5.

7 Conclusion and future works

In literature, the support vector regression (SVR) model is often fitted by solving the dual of the original constrained optimization problem, resulting in a quadratic programming problem, which is computationally expensive to solve, with time complexity about $O(n^3)$, where n is the training set size. As an alternative, this paper attempts to fit the SVR model by directly minimizing the primal form of the optimization problem. However, the primal objective function is not differentiable, which makes the well-developed gradient based optimization methods inapplicable, although the gradient based methods are easy to implement, converge fast to at least a local optimum. As such, we introduce a smooth approximation to the original primal objective function of the SVR model. We prove that as the smoothing parameter becomes smaller, the approximation has better quality, and the solution to the approximated problem converges to that of the original SVR model. We propose to use conjugate gradient method to minimize the smoothed objective function, in which we gradually decrease the smoothing parameter, in order to stablize the solution.

Extensive experiments were conducted on various publicly available real-world datasets, and we compared the performance of the proposed Conjugate Gradient based Smooth SVR (CG-SSVR) to that of the Quadratic Programming based SVR (QP-SVR), in terms of testing accuracy and training time. All our results show that CG-SSVR has very similar testing accuracy as QP-SVR, for either linear or nonlinear model. Although not implemented in a computationally efficient programming language, CG-SSVR was shown to be much faster than QP-SVR. Specifically, for linear model, CG-SSVR is often hundreds of times faster than QP-SVR, and for nonlinear model, CG-SSVR is about 3–10 times faster, depending on the training set size. We also observe that as the training set

size gets larger, the speed advantage of CG-SSVR over QP-SVR becomes greater.

Although the presented experimental results are sufficient to support the superiority of CG-SSVR, next, we would like to test the proposed CG-SSVR algorithm on some very large datasets, as those used in [11], and compare the performance with the alternatives [11, 17, 18]. Currently, we employ the conjugate gradient algorithm for minimizing the smoothed objective function. It will be interesting to compare other gradient based optimization methods, for example, quasi-Newton method [2, 27], coordinate gradient descent [4, 12], or blockwise coordinate descent [16].

Besides minimizing the regression error, the linear SVR model studied in this paper essentially imposes an L_2 constraint on the regression coefficients. When the predictor vector is in a very high dimensional space, however, an L_1 constraint on the regression coefficients is often preferred because of the variable selection ability associated with the L_1 norm [25]. In literature, this idea was applied to linear support vector machine [8, 31] and linear SVR [17, 18, 24]. Thus, it would be interesting to apply the smoothing idea to L_1 constrained linear SVR with a large number of predictors. This is another potential extension to the current work.

Acknowledgments The author would like to extend his sincere gratitude to the associate editor and two anonymous reviewers for their constructive suggestions and comments, which have greatly helped improve the quality of this paper. This work was supported by a Faculty Research Grant from Missouri State University.

References

1. Armijo L (1966) Minimization of functions having Lipschitz-continuous first partial derivatives. Pac J Math 16:1-3
2. Bertsekas DP (1999) Nonlinear programming, 2nd edn. Athena Scientific, Belmont
3. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge
4. Chang K-W, Hsieh C-J, Lin C-J (2008) Coordinate descent method for large-scale L_2-loss linear support vector machines. J Mach Learn Res 9:1369–1398
5. Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2(3):27:1–27:27
6. Chapelle O (2007) Training a support vector machine in the primal. Neural Comput 19:1155–1178
7. Chen C, Mangasarian OL (1996) A class of smoothing functions for nonlinear and mixed complementarity problems. Comput Optim Appl 5:97–138
8. Fung G, Mangasarian OL (2004) A feature selection Newton method for support vector machine classification. Comput Optim Appl 28(2):185–202
9. Gunn SR (1997) Support vector machines for classification and regression. Technical Report, Image Speech and Intelligent Systems Research Group, University of Southampton. http://users.ecs.soton.ac.uk/srg/publications/pdf/SVM.pdf
10. Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: prediction, inference and data mining, 2nd edn. Springer, New York
11. Ho C-H, Lin C-J (2012) Large-scale linear support vector regression. Technical report of Department of Computer Science and Information Engineering, National Taiwan University. http://www.csie.ntu.edu.tw/cjlin/papers/linear-svr.pdf
12. Hsieh C-J, Chang K-W, Lin C-J, Keerthi SS, Sundararajan S (2008) A dual coordinate descent method for large-scale linear SVM. In: Proceedings of the 25th international conference on machine learning, pp 408–415
13. Joachims J (1999) Making large-scale SVM learning practical. In: Schölkopf B, Burges C, Smola A (eds) Advances in kernel methods—support vector learning. MIT-Press, London
14. Kimeldorf GS, Wahba G (1971) Some results on Tchebycheffian spline functions. J Math Anal Appl 33(1):82–95
15. Lee Y-J, Mangasarian OL (2001) SSVM: a smooth support vector machine for classification. Comput Optim Appl 20(1):5–22
16. Liu H, Palatucci M, Zhang J (2009) Blockwise coordinate descent procedures for the multi-task Lasso, with applications to neural semantic basis discovery. In: Proceedings of the 26th international conference on machine learning, pp 649–656
17. Mangasarian OL, Musicant DR (2002) Large scale kernel regression via linear programming. Mach Learn 46(1/3):255-269
18. Musicant DR, Mangasarian OL (1999) Massive support vector regression. In: Proceedings of NIPS workshop on learning with support vectors: theory and applications
19. Osuna E, Freund R, Girosi F (1997a) An improved training algorithm for support vector machines. In: Proceedings of IEEE workshop on neural networks for signal processing, pp 276–285
20. Osuna E, Freund R, Girosi F (1997b). Training support vector machines: an application to face detection. In: Proceedings of IEEE conference on computer vision and pattern recognition
21. Platt J (1998) Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B, Burges C, Smola A (eds) Advances in kernel methods—support vector learning. MIT-Press, London
22. Schölkopf B, Smola A (2002) Learning with kernels. MIT Press, Cambridge
23. Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Stat Comput 14(3):199–222
24. Smola AJ, Schölkopf B, Rätsch G (1999) Linear programs for automatic accuracy control in regression. In: Proceedings of ninth international conference on artificial neural networks, pp 575–580
25. Tibshirani R (1996) Regression shrinkage and selection via the Lasso. J R Stat Soc Ser B 58(1): 267-288
26. Vapnik V (1998) Statistical learning theory. Wiley, NY
27. Walsh GR (1975) Methods of optimization. Wiley, NY
28. Yeh I-C (1998) Modeling of strength of high performance concrete using artificial neural networks. Cement Concrete Res 28(12):1797–1808
29. Zhang J, Jin R, Yang Y, Hauptmann AG (2003) Modified logistic regression: an approximation to SVM and its applications in large-scale text categorization. In: Proceedings of the 20th international conference on machine learning, pp 888–895
30. Zheng S (2011) Gradient descent algorithms for quantile regression with smooth approximation. Int J Mach Learn Cybern 2(3):191–207
31. Zhu J, Rosset S, Hastie T, Tibshirani R (2004) 1-norm support vector machines. In: Proceedings of neural information processing systems