International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.7, No.4 (2014), pp.211-220
http://dx.doi.org/10.14257/ijsip.2014.7.4.21

Robust Support Vector Regression with Flexible Loss Function

Kuaini Wang and Ping Zhong*

(College of science, China Agricultural University, Beijing, 100083, China)
E-mail: wangkuaini1219@sina.com [1]

Abstract

In the interest of deriving regressor that is robust to outliers, we propose a support vector regression (SVR) based on non-convex quadratic insensitive loss function with flexible coefficient and margin. The proposed loss function can be approximated by a difference of convex functions (DC). The resultant optimization is a DC program. We employ Newton's method to solve it. The proposed model can explicitly enhance the robustness and sparseness of SVR. Numerical experiments on six benchmark data sets show that it yields promising results.

Keywords: *Support vector regression, Loss function, Robustness, DC program*

1. Introduction

Support vector machine (SVM), proposed by Vapnik and his group [11, 3, 6] and is popular in the pattern recognition and machine learning community over the past decades. SVM is based on statistical learning theory and structural risk minimization principle [11]. As a state of the art technique, SVM has been applied to various real-life problems, such as text categorization, time series prediction, regression analysis and so on [3, 6]. As we know, loss functions play a key role in supervised learning. Using different loss functions, we can yield different SVMs. In regression estimation, one kind of the important loss functions is quadratic loss functions, and many SVRs are derived by using these loss functions, such as L-2-SVR [6] and least squares SVR (LS-SVR) [9]. Quadratic loss functions are all convex which have the amenability to theoretical analysis as well as computational advantages. In practice, sampling errors, modeling errors and instrument errors may corrupt the training samples with outliers. In the presence of outliers, the classical SVRs with convex loss functions which are unbounded and emphasize the effect of outliers, yield poor generalization performance.

In recent years, non-convex loss functions have shown superiority to convex ones in generalization performance, robustness and sparseness. Xu and Crame [15] studied training algorithms for SVMs with the ramp loss and solved the non-convex optimization by utilizing semi definite program and convex relaxation techniques. Wu and Liu [14] proposed a robust SVM with truncated hinge loss, which was illustrated to be more robust to outliers and derived more accuracy classifiers. Collobert, *et al.,* [5] pointed out the scalability advantages of non-convex approaches and used the Concave-Convex Procedure (CCCP) [16] for non-convex optimization to achieve faster batch SVMs and Transductive SVMs. Motivated by the recent interest in solving SVM in the primal [4, 2], Wang, *et al.,* [13] gave robust support vector machine with smooth ramp loss in the primal. Zhao, *et al.,* [17] extended the similar idea to regression estimation. Wang, *et al.,* [12] and Zhong [18] presented two different non-

[1] * *Corresponding author: zping@cau.edu.cn*

ISSN: 2005-4254 IJSIP

convex loss functions for robust SVR. These studies derived the improvement in generalization performance and running time.

In this paper, based on quadratic insensitive loss function, we propose a flexible loss function which is non-convex with variable coefficient and margin, and derive a robust model, termed as RSVR-FL. We first present two Huber loss functions to approximate the proposed loss function. The resultant optimization problem is a DC program [10, 1]. We utilize a Newton's method to solve the robust model. It reveals that RSVR-FL can explicitly incorporate outlier suppression and sparseness in the training process. Numerical experiments show that the proposed algorithm gives promising results. In the next section, we propose a flexible non-convex loss function and the RSVR-FL. We solve the robust model by DC program in Section 3. Experiments on benchmark data sets are presented in Section 4. The last section concludes the paper.

2. Robust SVR Model

Given a training set $\{x_i, y_i\}_{i=1}^n$, where $x_i \in R^m$ is the input variable and $y_i \in R$ is the corresponding target. A common used loss function for SVR is the following insensitive quadratic loss function:

$$l_1(z) = \max(0, |z| - \varepsilon)^2 \tag{1}$$

where $\varepsilon > 0$. Its shape is shown in Figure 1. The optimization problem of SVR [11] is usually written as:

$$\min_f \quad \frac{1}{2}\|f\|_A^2 + \upsilon \sum_{i=1}^n l_1(f(x_i) - y_i) \tag{2}$$

where A is the reproducing kernel Hilbert space. According to [7], the optimal function for (2) can be expressed as a linear combination of the training samples in the feature space

$$f(x) = \sum_{i=1}^n \alpha_i k(x, x_i) \tag{3}$$

where $k(\cdot)$ is a kernel function. Substituting (3) into (2), we get

$$\min_\alpha \quad \frac{1}{2}\alpha^T K \alpha + \upsilon \sum_{i=1}^n l_1(K_i\alpha - y_i) \tag{4}$$

where $\alpha = (\alpha_1, \alpha_1, ..., \alpha_n)^T$, and K_i is the ith row of kernel matrix K. (4) is SVR with loss function (1) in the primal.

In this section, based on (4), we propose a robust SVR. First, we generalize the loss function $l_1(z)$ as follows

$$l_2(z) = \begin{cases} d^2(z + \varepsilon_1)^2, & if \ z \le -\varepsilon_1, \\ 0, & if \ -\varepsilon_1 \le z \le \varepsilon_2, \\ c^2(z - \varepsilon_2)^2, & if \ z > \varepsilon_2. \end{cases} \tag{5}$$

The loss function $l_2(z)$ is asymmetry and regularized by the real positive constants c and d. Its shape is shown in Figure 1. The modification of $l_2(\cdot)$, providing both scalable penalties and more flexible insensitive tube, offers a better chance to deal with regression problem. We notice that $l_2(z)$ is unbounded, which indicates that $l_2(\cdot)$ has no limit on the influences of outliers. The outliers keep more influences on the optimal solution of (4), which may lead to the decision hyper plane of SVR deviating from the original position, and thus

deteriorate the generalization performance of SVR. Thus, we set the upper bound of $l_2(\cdot)$ to limit the impact of outliers, and get a flexible loss function

$$l_\theta(z) = \min\{\theta^2, l_2(z)\} \tag{6}$$

where $\theta > 0$ is a constant. Its shape is shown in Figure 1. The loss function in [12] is a special case of $l_\theta(z)$.

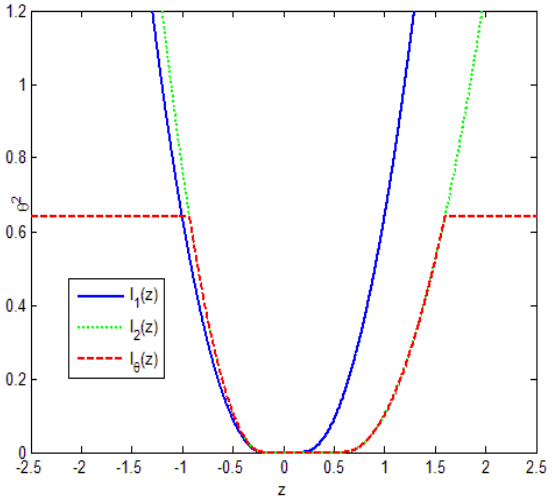

Figure 1. $l_1(z), l_2(z)$ **and** $l_\theta(z)$ **with** $\varepsilon_1 = 0.2, \varepsilon_2 = 0.6, c = 0.8, d = 1.1, \theta = 0.8$

However, $l_\theta(z)$ is not differentiable, and the associated optimization problem is difficult to be solved. We use a differentiable and non-convex function to approximate it. Inspired by the Huber loss function, we propose the following Huber loss functions:

$$l_1^{hu}(z) = \begin{cases} d\theta[-2z - (2\varepsilon_1 + \frac{\theta}{d})], & if \quad z \le -\varepsilon_1 - \frac{\theta}{d}, \\ d^2(z - \varepsilon_1)^2, & if \quad -\varepsilon_1 - \frac{\theta}{d} < z < -\varepsilon_1, \\ 0, & if \quad -\varepsilon_1 \le z \le \varepsilon_2, \\ c^2(z - \varepsilon_2)^2, & if \quad \varepsilon_2 < z < \varepsilon_2 + \frac{\theta}{c}, \\ c\theta[2z - (2\varepsilon_2 + \frac{\theta}{c})], & if \quad z \ge \varepsilon_2 + \frac{\theta}{c}. \end{cases} \tag{7}$$

$$l_2^{hu}(z) = \begin{cases} d\theta[-2z - (2\varepsilon_1 + \frac{2\theta}{d} + \frac{h}{d})], & if \quad z \le -\varepsilon_1 - \frac{\theta}{d} - \frac{h}{d}, \\ \dfrac{d^2\theta(z + \varepsilon_1 + \frac{\theta}{d})^2}{h}, & if \quad -\varepsilon_1 - \frac{\theta}{d} - \frac{h}{d} < z < -\varepsilon_1 - \frac{\theta}{d}, \\ 0, & if \quad -\varepsilon_1 - \frac{\theta}{d} \le z \le \varepsilon_2 + \frac{\theta}{c}, \\ \dfrac{c^2\theta(z - \varepsilon_2 - \frac{\theta}{c})^2}{h}, & if \quad \varepsilon_2 + \frac{\theta}{c} < z < \varepsilon_2 + \frac{\theta}{c} + \frac{h}{c}, \\ c\theta[2z - (2\varepsilon_2 + \frac{2\theta}{c} + \frac{h}{c})], & if \quad z \ge \varepsilon_2 + \frac{\theta}{c} + \frac{h}{c}. \end{cases} \tag{8}$$

where $h > 0$ is the Huber parameter. We find that $l_1^{hu}(z)$ and $l_2^{hu}(z)$ are convex and differentiable, and the difference of $l_1^{hu}(z)$ and $l_2^{hu}(z)$ is a differentiable approximation

of $l_\theta(z)$. Let $l_{\theta,h}(z) = l_1^{hu}(z) - l_2^{hu}(z)$. Its shape is shown in Figure 2. It is easily verified that $l_{\theta,h}(z)$ is differentiable, and $\lim_{h\to 0} l_{\theta,h}(z) = l_\theta(z)$. Then, we propose the robust SVR with flexible loss function in the primal as

$$\min_{\alpha} \quad L_{\theta,h}(\alpha) = \frac{1}{2}\alpha^{\mathrm{T}} K\alpha + \upsilon\sum_{i=1}^{n} l_1^{hu}(z_i) + \upsilon\sum_{i=1}^{n} l_2^{hu}(z_i) \tag{9}$$

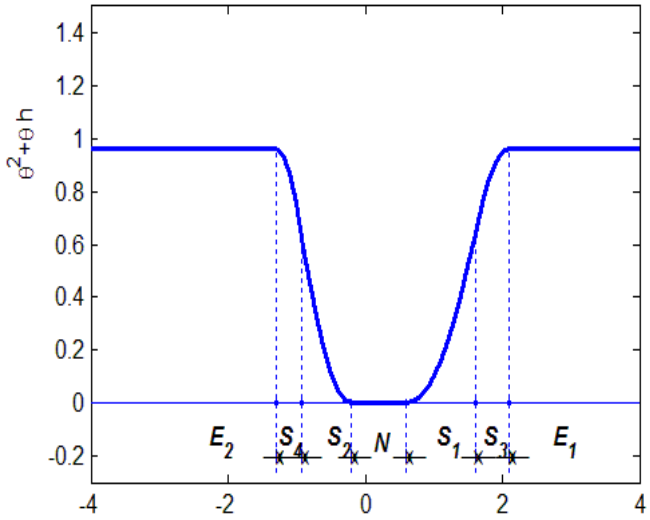

Figure 2. $l_{\theta,h}(z)$ **with** $\varepsilon_1 = 0.2, \varepsilon_2 = 0.6, c = 0.8, d = 1.1, \theta = 0.8, h = 0.4$

3. DC Program for the Robust SVR Model

Notice that the objective function of (9) is non-convex, and it is difficult to be solved by classical convex optimization techniques. However, from the expression of $l_{\theta,h}(z)$ and optimization problem (9), and by denoting $u(\alpha) = \frac{1}{2}\alpha^{\mathrm{T}} K\alpha + \upsilon\sum_{i=1}^{n} l_1^{hu}(z_i)$, $v(\alpha) = \upsilon\sum_{i=1}^{n} l_2^{hu}(z_i)$, we can see that (9) is a DC program which can be expressed as:

$$\min_{\alpha} \quad u(\alpha) - v(\alpha) \tag{10}$$

In the DC program literature, the DCA [10, 1] was proposed for solving a general DC program of the form $\min\{f(x) = g(x) - h(x) : x \in \mathbb{R}^n\}$ with g and h being proper lower semi-continuous convex functions, which form a large class of functions than the class of differentiable functions. In our program (10), the solution to (10) can be achieved by iteratively solving the following optimization:

$$\alpha^{t+1} = \arg\min_{\alpha} \{u(\alpha) - \alpha^{\mathrm{T}} \nabla v(\alpha^t)\} \tag{11}$$

where $\nabla v(\alpha^t)$ is the derivative of $v(\alpha)$ with respect to α at the tth iteration,

$$\nabla v(\alpha^t) = \frac{\partial v(\alpha^t)}{\partial \alpha} = \upsilon\sum_{i=1}^{n} \frac{\partial l_2^{hu}(z_i^t)}{\partial z_i} \cdot \frac{\partial z_i}{\partial \alpha} = \upsilon\sum_{i=1}^{n} \eta_i^t K_i^{\mathrm{T}} \tag{12}$$

where

$$\eta_i^t = \begin{cases} -2d\theta, & if \quad z_i^t \le -\varepsilon_1 - \frac{\theta}{d} - \frac{h}{d}, \\ \frac{2d^2\theta(z_i^t + \varepsilon_1 + \frac{\theta}{d})}{h}, & if \quad -\varepsilon_1 - \frac{\theta}{d} - \frac{h}{d} < z_i^t < -\varepsilon_1 - \frac{\theta}{d}, \\ 0, & if \quad -\varepsilon_1 - \frac{\theta}{d} \le z_i^t \le \varepsilon_2 + \frac{\theta}{c}, \\ \frac{2c^2\theta(z_i^t - \varepsilon_2 - \frac{\theta}{c})}{h}, & if \quad \varepsilon_2 + \frac{\theta}{c} < z_i^t < \varepsilon_2 + \frac{\theta}{c} + \frac{h}{c}, \\ 2c\theta, & if \quad z_i^t \ge \varepsilon_2 + \frac{\theta}{c} + \frac{h}{c}. \end{cases}$$

In each iteration, we actually only need to minimize the following optimization:

$$\min_{\alpha} \quad L_{\theta,h}(\alpha) = \frac{1}{2}\alpha^{\mathrm{T}} K\alpha + \upsilon\sum_{i=1}^{n} l_1^{hu}(K_i\alpha - y_i) - \upsilon\sum_{i=1}^{n}\eta_i^t K_i\alpha \tag{13}$$

The objective function $L_{\theta,h}(\alpha)$ of (13) is piecewise quadratic and the Newton's method appears a nature choice for an efficient minimization. Each Newton's step consists of the following update

$$\alpha^{t+1} = \alpha^t - (H^t)^{-1}\nabla L_{\theta,h}(\alpha^t) \tag{14}$$

where $\nabla L_{\theta,h}(\alpha^t)$ and H are the gradient and the Hessian of $L_{\theta,h}(\alpha)$, respectively.

At the *t*th iteration, we divided the training samples into 7 groups according to z_i^t, depicted in Figure 2. Let $S_1 = \{i \mid \varepsilon_2 < z_i^t < \varepsilon_2 + \frac{\theta}{c}\}$, $S_2 = \{i \mid -\varepsilon_1 - \frac{\theta}{d} < z_i^t < -\varepsilon_2\}$, $S_3 = \{i \mid \varepsilon_2 + \frac{\theta}{c} < z_i^t < \varepsilon_2 + \frac{\theta}{c} + \frac{h}{c}\}$, $S_4 = \{i \mid \varepsilon_2 + \frac{\theta}{c} < z_i^t < \varepsilon_2 + \frac{\theta}{c} + \frac{h}{c}\}$ denote the index sets of samples lying in the quadratic part of $l_{\theta,h}(z)$, defined as support vectors. Let $N = \{i \mid -\varepsilon_1 < z_i^t < \varepsilon_2\}$ depict the index set of non-support vectors, $E_1 = \{i \mid z_i^t > \varepsilon_2 + \frac{\theta}{c} + \frac{h}{c}\}$, $E_2 = \{i \mid z_i^t < -\varepsilon_1 - \frac{\theta}{d} - \frac{h}{d}\}$ the index sets of error support vectors. We rearrange the 7 groups in the order of $S_1, S_2, S_3, S_4, E_1, E_2$ and N (the numbers are $|S_1|, |S_2|, |S_3|, |S_4|, |E_1|, |E_2|$ and $|N|$), and define $n \times n$ diagonal matrices I_1, I_2, I_3, I_4, where I_1 has the first $|S_1|$ entries being 1 and the others 0, and I_2 only has the middle entries from $|S_1|+1$ to $|S_1|+|S_2|$ entries being 1 and the others 0. I_3 and I_4 are defined similarly.

The gradient and the Hessian of $L_{\theta,h}(\alpha)$ at the *t*th iteration are

$$\begin{aligned} \nabla L_{\theta,h}(\alpha^t) = {} & K\alpha^t + 2\upsilon K(c^2 I_1 + d^2 I_2)K\alpha^t + 2\upsilon\{-c^2 K I_1(\varepsilon_2 e + y) + d^2 K I_2(\varepsilon_1 e - y) \\ & + \frac{c^2\theta}{h} K I_3[(\varepsilon_2 + \frac{\theta}{c} + \frac{h}{c})e - z^t] - \frac{d^2\theta}{h} K I_4[z^t + (\varepsilon_1 + \frac{\theta}{d} + \frac{h}{d})e]\} \end{aligned} \tag{15}$$

$$H = K\left(I_n + 2\upsilon(c^2 I_1 + d^2 I_2)K\right) \tag{16}$$

Where $y = (y_1, y_2, \ldots, y_n)^{\mathrm{T}}$ and $e = (1,1,\ldots,1)^{\mathrm{T}}$. Combining (15) and (16),

$$\begin{aligned} \alpha^{t+1} = {} & 2\upsilon[I_n + 2\upsilon(c^2 I_1 + d^2 I_2)K]^{-1}\{c^2 I_1(\varepsilon_2 e + y) - d^2 I_2(\varepsilon_1 e - y)\} \\ & + \frac{c^2\theta}{h} I_3[z^t - (\varepsilon_2 + \frac{\theta}{c} + \frac{h}{c})e] + \frac{d^2\theta}{h} I_4[z^t + (\varepsilon_1 + \frac{\theta}{d} + \frac{h}{d})e]\} \end{aligned} \tag{17}$$

Denote $U = 2\upsilon(c^2 I_1 + d^2 I_2)K$, $A = I_{|s_1 \cup s_2|} + U_{|s_1 \cup s_2|}$, $U_i = U_{|s_1 \cup s_2 | \times |s_i|}, i = 3, 4$, $U_{E_i} = U_{|s_1 \cup s_2 | \times |E_i|}, i = 1, 2$ and $U_N = U_{|s_1 \cup s_2 | \times |N|}$. We have

$$[I_n + 2\upsilon(c^2 I_1 + d^2 I_2)K]^{-1} = \begin{pmatrix} A^{-1} & -A^{-1}U_3 & -A^{-1}U_4 & -A^{-1}U_{E_1} & -A^{-1}U_{E_2} & -A^{-1}U_N \\ 0 & I_{|S_3|} & 0 & 0 & 0 & 0 \\ 0 & 0 & I_{|S_4|} & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{|E_1|} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{|E_2|} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{|N|} \end{pmatrix} \tag{18}$$

Substituting (18) into (17), we get the optimal solution at the (t + 1)th iteration

$$\alpha^{t+1} = 2\upsilon \begin{pmatrix} \beta^t \\ \frac{c^2\theta}{h}[z^t_{|S_3|} - (\varepsilon_2 + \frac{\theta}{c} + \frac{h}{c})e_{|S_3|}] \\ \frac{d^2\theta}{h}[z^t_{|S_4|} + (\varepsilon_1 + \frac{\theta}{d} + \frac{h}{d})e_{|S_4|}] \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \alpha^{t+1}_{|S_1|} \\ \alpha^{t+1}_{|S_2|} \\ \alpha^{t+1}_{|S_3|} \\ \alpha^{t+1}_{|S_4|} \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{19}$$

where

$$\beta^t = A^{-1}\left[\begin{pmatrix} c^2(\varepsilon_2 e_{|S_1|} + y_{|S_1|}) \\ -d^2(\varepsilon_1 e_{|S_2|} - y_{|S_2|}) \end{pmatrix} - \frac{c^2\theta U_3(z^t_{|S_3|} - (\varepsilon_2 + \frac{\theta}{c} + \frac{h}{c})e_{|S_3|})}{h} - \frac{d^2\theta U_4(z^t_{S_4} + (\varepsilon_1 + \frac{\theta}{d} + \frac{h}{d})e_{|S_4|})}{h}\right] \tag{20}$$

It shows that the samples belonging to E_1, E_2 and N groups have no contribution to the optimal solution because their corresponding elements in α^{t+1} are fixed at 0. Noticing that the N region is equivalent to the ε-insensitive zone, while outliers usually lie in the E_1 and E_2 regions, our proposed model not only keeps the sparseness like the classical SVRs but also to a certain extent discards outliers as support vectors. Having updated α^{t+1}, we get the corresponding predictor

$$f^{t+1}(\mathrm{x}) = \sum_{i=1}^{n} \alpha_i^{t+1} k(\mathrm{x}_i, \mathrm{x}) \tag{21}$$

The flowchart of implementing our proposed model is described as follows.

RSVR-FL (Newton's method for RSVR with flexible loss function)

Input: Training set $T = \{\mathrm{x}_i, y_i\}_{i=1}^n$, kernel matrix K, and a small real $\rho > 0$.

1. Let $\alpha^0 \in R^n$ and calculate $z_i^0 = K_i\alpha^0 - y_i$, $i = 1, 2, ..., n$. Divide training set into 7 groups according to z_i^0. Set t = 0.

2. Rearrange the groups in the order of $S_1, S_2, S_3, S_4, E_1, E_2$ and N , and adjust K and y correspondingly. Solve $\nabla L_{\theta,h}(\alpha^t)$ by (15). Check whether $\| \nabla L_{\theta,h}(\alpha^t) \| \leq \rho$. If so, stop; else, go to next step.

3. Calculate α^{t+1} by (19) and $f^{t+1}(x)$ by (21).

4. Divide training set into 7 groups according to $| z_i^{t+1} | = | K_i \alpha^{t+1} - y_i |$. Let $t = t + 1$, and go to step 2.

4. Experiments

To examine the efficiency of the RSVR-FL, we compare it with LS-SVR [9], classical L-2-SVR [6] and robust support vector regression with insensitive quadratic loss function (RSVR-QL) [12] on benchmark data sets. All these algorithms employ quadratic loss functions. Gaussian kernel is chosen in the experiments. We adopt five popular regression criterions, root mean square error (RMSE), mean absolute error (MAE), mean relative error (MRE), ratio between the sum squared error SSE and the sum squared deviation testing samples SST (SSE/SST), and ratio between interpretable sum deviation SSR and SST (SSR/SST) [8], to evaluate these algorithms. All the experiments are implemented in Matlab 7.0 in Core 3.00GHz, 2GB RAM.

We select six popular benchmark data sets for our experiments, including Pyrim, Triazines and Servo which are from UCI[2], Body fat, Pollution from StatLib[3], and Diabetes from the web page[4]. The data set is randomly divided into two parts, while one is used for training and another for testing. In this paper, we focus on outliers and hence we contaminate the training samples by adding large noise on their targets. The testing samples are not added noise. We repeat the above process 10 times with different partition of training and testing samples.

Table 1. Experimental Results on Benchmark Data Sets

Dataset	Algorithm	RMSE	MAE	MRE	SSE/ SST	SSR/ SST	#SV	Time(s)
Servo	LS-SVR	0.7054	0.4194	0.4553	0.2126	**0.8353**	100	**0.0342**
(167× 4)	L-2-SVR	0.7105	0.4150	0.4417	0.2157	0.7785	98.4	57.6764
	RSVR-QL	0.6920	0.3865	0.4088	0.2061	0.7567	86.7	0.0540
	RSVR-FL	**0.6695**	**0.3739**	**0.4011**	**0.1962**	0.8075	**80.7**	0.0806
Diabetes	LS-SVR	0.5816	0.4734	0.1055	0.9009	0.4568	30	**0.0070**
(43× 2)	L-2-SVR	0.5829	0.4731	0.1057	0.9062	0.3811	23.1	2.8326
	RSVR-QL	0.5795	0.4687	0.1008	0.9461	**0.8529**	14.3	0.0177
	RSVR-FL	**0.5721**	**0.4559**	**0.1026**	**0.8983**	0.6754	**8.6**	0.0298
Pollution	LS-SVR	39.8592	31.0642	0.0325	0.5358	**0.8910**	40	**0.0083**
(60× 16)	L-2-SVR	39.8785	31.0743	0.0325	0.5361	0.8894	39.9	5.3624
	RSVR-QL	37.9818	29.3810	**0.0309**	0.4795	0.7261	35	0.0187
	RSVR-FL	**37.7460**	**29.2860**	**0.0309**	**0.4730**	0.6998	**34.1**	0.0252
Pyrim	LS-SVR	0.1056	0.0649	0.1959	0.5925	**0.5729**	50	**0.0137**
(74× 27)	L-2-SVR	0.1047	0.0652	**0.1952**	0.5826	0.5715	**31.9**	9.0532
	RSVR-QL	0.1045	0.0645	0.1966	0.5813	0.4058	46.8	0.0322
	RSVR-FL	**0.1013**	**0.0616**	0.1971	**0.5462**	0.5498	32	0.0370
Triazines	LS-SVR	0.1481	0.1126	0.2884	0.9295	**0.3356**	150	**0.0759**
(186× 60)	L-2-SVR	0.1486	0.1141	0.2903	0.9368	0.2966	149.1	168.8824

[2] http://www.ics.uci.edu/~mlearn/MLRepository.html
[3] http://lib.stat.cmu.edu/datasets/
[4] http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html

	RSVR-QL	0.1420	0.1088	0.2845	0.8403	0.2211	**103.9**	0.1529
	RSVR-FL	**0.1403**	**0.1039**	**0.2825**	**0.8233**	0.2331	121.6	0.1252
Bodyfat	LS-SVR	0.0075	0.0056	0.0053	0.1868	0.7799	200	**0.0854**
(252× 14)	L-2-SVR	0.0076	0.0057	0.0054	0.1931	0.7785	191.9	393.0776
	RSVR-QL	0.0030	0.0015	0.0014	0.0357	0.9408	131.9	0.1286
	RSVR-FL	**0.0020**	**6.2054E-4**	**5.8098 E-4**	**0.0224**	**0.9768**	**55.1**	0.1689

The average testing results of these algorithms are reported in Table 1. The results indicate that the RSVR-QL and our proposed RSVR-FL enhance testing accuracies when compared with LS-SVR and L-2-SVR. In addition, the numbers of support vectors (#SV) obtained by the RSVR-QL and RSVR-FL are also less than those of SVR for most of the data sets. The main reason is that RSVR-QL and RSVR-FL employ non-convex loss function which can discard outliers as support vectors in the training process. As mentioned above, our RSVR-FL uses the loss function with more flexible coefficient and margin and gives a better chance to learn regression estimation. It outperforms RSVR-QL, which is verified in Table 1. Furthermore, it is shown by Table 1 that our RSVR-FL outperforms the other regressors in generalization. These results imply the RSVR-FL obtains a sparser regressor with good generalization. As for the running time (Time), Table 1 shows that our RSVR-FL requires less running time when compared with the L-2-SVR. But our RSVR-FL needs a little more running time than these of the LS-SVR and RSVR-QL.

5. Conclusion

In this paper, we propose a robust regression model, called RSVR-FL, which adopts a non-convex quadratic loss function with more flexible coefficient and margin to suppress the influence of outliers. The robustness comes from the property of non-convex loss function, which is bounded. Since the proposed loss function is a non-convex one, we approximate it by a difference of two huber loss functions. Newton's method is employed to solve the corresponding DC program. The RSVR-FL has shown better robustness and sparseness in numerical experiments.

Acknowledgments

The work is supported by National Natural Science Foundation of China Grant No.11171346 and Chinese Universities Scientific Fund No. 2013YJ010.

References

[1] L. T. H. An and P. D. Tao, "The DC (difference of convex functions) programming and DCA revisited with DC models of real world non convex optimization problems", Annals of Operations Research, vol. 133**, (2005)**.
[2] L. Bo, L. Wang and L. Jiao, "Recursive finite Newton algorithm for support vector regression in the primal", Neural Computation, vol. 4, no. 19, **(2007)**.
[3] C. J. Burges, "A tutorial on support vector machines for pattern recognition", Data mining and knowledge discovery, vol. 2, no. 2, **(1998)**.
[4] O. Chapelle, "Training a support vector machine in the primal", Neural Computation, vol. 5, no. 19, **(2007)**.
[5] R. Collobert, F. Sinz, J. Weston and L. Bottou, "Trading convexity for scalability", In Proceeding of the 23rd International Conference on Machine Learning (ICML), **(2006)**.
[6] N. Cristianini and J. Shawe-Taylor, "An introduction to support vector machines and other kernel-based learning methods", Cambridge University press, **(2000)**.
[7] G. S. Kimeldo, G. Wahba, "A correspondence between Bayesian estimation on stochastic processes and smoothing by splines", The Annals of Mathematical Statistics, vol. 2, no. 41, **(1970)**.

[8] X. Peng, "TSVR: An efficient twin support vector machine for regression", Neural Networks, vol. 3, no. 23, **(2009).**
[9] J. Suykens, J. De Brabanter and L. Lukas, "Weighted least squares support vector machines: robustness and sparse approximation", Neuro computing, vol. 48, **(2002)**.
[10] P. Tao and L. T. H. An, "A DC optimization algorithms for solving the trust region sub problem", SIAM Journal of Optimization, vol. 8, **(1998).**
[11] V. N. Vapnik, "The Nature of Statistical Learning Theory", Springer, New York, **(1995)**.
[12] K. Wang, P. Zhong and Y. Zhao, "Training robust support vector regression via D.C. program", Journal of Information and Computational Science, vol. 12, no. 7, **(2010)**.
[13] L. Wang, H. Jia and J. Li, "Training robust support vector machine with smooth ramp loss in the primal space", Neurocomputing, vol. 13–15, no. 71, **(2008)**.
[14] Y. Wu and Y. Liu, "Robust truncated hinge loss support vector machines", Journal of the American Statistical Association, vol. 479, no. 102, **(2007)**.
[15] L. Xu, K. Crammer and D. Schuurmans, "Robust support vector machine training via convex outlier ablation", In Proceedings of the 21st National Conference on Artifical Intelligence, AAAI'06, **(2006)**.
[16] A. L. Yuille and A. Rangarajan, "The concave-convex procedure", Neural Computation, vol. 4, no. 15, **(2003)**.
[17] Y. Zhao and J. Sun, "Robust support vector regression in the primal", Neural Networks, vol. 10, no. 21 **(2008).**
[18] P. Zhong, "Training robust support vector regression with smooth non-convex loss function", Optimization Methods and Software, vol. 6, no. 27, **(2012)**.

Authors

Kuaini Wang, has received the B.S. degree in information and computation science from Jilin University, China, 2005, the M.S. degree in mathematics from China Agricultural University, 2009. She is currently pursuing the Ph.D. degree in operations research and management science at China Agricultural University. Her research interest includes machine learning and support vector machines.

Ping Zhong, has received the B.S degree and Master degree in mathematics in 1994 and 1997 both from Qufu Normal University, China, and her Ph. D degree in 2002 from China Agricultural University. Now she is a professor of China Agricultural University. She has published more than 40 papers cited by SCI and EI. Her research interest includes optimization theory, machine learning and support vector machines.