Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 169575, 8 pages
http://dx.doi.org/10.1155/2014/169575

Research Article

Least Absolute Deviation Support Vector Regression

Kuaini Wang,[1] Jingjing Zhang,[1] Yanyan Chen,[1,2] and Ping Zhong[1]

[1] *College of Science, China Agricultural University, Beijing 100083, China*
[2] *College of Applied Science and Technology, Beijing Union University, Beijing 102200, China*

Correspondence should be addressed to Ping Zhong; zping@cau.edu.cn

Received 18 November 2013; Revised 9 June 2014; Accepted 10 June 2014; Published 7 July 2014

Academic Editor: Francesco Braghin

Least squares support vector machine (LS-SVM) is a powerful tool for pattern classification and regression estimation. However, LS-SVM is sensitive to large noises and outliers since it employs the squared loss function. To solve the problem, in this paper, we propose an absolute deviation loss function to reduce the effects of outliers and derive a robust regression model termed as least absolute deviation support vector regression (LAD-SVR). The proposed loss function is not differentiable. We approximate it by constructing a smooth function and develop a Newton algorithm to solve the robust model. Numerical experiments on both artificial datasets and benchmark datasets demonstrate the robustness and effectiveness of the proposed method.

1. Introduction

Support vector machine (SVM), introduced by Vapnik [1] and Cristianini and Taylor [2], has been gaining more and more popularity over the past decades as a modern machine learning approach, which has strong theoretical foundation and successes in many real-world applications. However, its training computational load is great, that is, $O(N^3)$, where N is the total size of training samples. In order to reduce the computational effort, many accelerating algorithms have been proposed. Traditionally, SVM is trained by means of decomposition techniques such as SMO [3, 4], chunking [5], SVM$^{\text{light}}$ [6], and LIBSVM [7], which solve the dual problems by optimizing a small subset of the variables during the iteration procedure. Another kind of accelerating algorithm is the least squares SVM introduced by Suykens and Vandewalle [8] which replaces inequality constraints with equality ones, requiring to solve a linear system of equations and results in an extremely fast training speed.

LS-SVM obtains good performance on various classification and regression estimation problems. In LS-SVR, it is optimal when the error variables follow a Gaussian distribution because it tries to minimize the sum of squared errors (SSE) of training samples [9]. However, datasets subject to heavy-tailed errors or outliers are commonly encountered in various applications and the solution of LS-SVR may suffer from lack of robustness. In recent years, much effort has been made to increase the robustness of LS-SVR. The commonly used approach adopts the weight setting strategies to reduce the influence of outliers [9–13]. In these LS-SVR methods, different weight factors are put on the error variables such that the less important samples or outliers have smaller weights. Another approach improves LS-SVR's performances by means of outlier elimination [14–17]. Essentially, LS-SVR is sensitive to outliers since it employs the squared loss function which overemphasizes the impact of outliers.

In this paper, we focus on the situation in which the heavy-tailed errors or outliers are found in the targets. In such a situation, it is well known that the traditional least squares (LS) may fail to produce a reliable regressor, and the least absolute deviation (LAD) can be very useful [18–20]. Therefore, we exploit the absolute deviation loss function to reduce the effects of outliers and derive a robust regression model termed as least absolute deviation SVR (LAD-SVR). Due to the fact that the absolute deviation loss function is not differentiable, the classical optimization method cannot be used directly to solve the LAD-SVR. Recently, some algorithms in the primal space for training SVM have been proposed due to their effective computation. Moreover, it is pointed out that the primal domain methods are superior to the dual domain methods when the goal is to find an approximate solution [21, 22]. Therefore, we approximate

LAD-SVR by constructing a smooth function and develop a Newton algorithm to solve the robust model in the primal space. Numerical experiments on both artificial datasets and benchmark datasets reveal the efficiency of the proposed method.

The paper is organized as follows. In Section 2, we briefly introduce classical LS-SVR and LS-SVR in the primal space. In Section 3, we propose an absolute deviation loss function and derive LAD-SVR. A Newton algorithm for LAD-SVR is given in Section 4. Section 5 performs experiments on artificial datasets and benchmark datasets to investigate the effectiveness of LAD-SVR. In Section 6, some remarkable conclusions are given.

2. Least Squares Support Vector Regression

2.1. Classical LS-SVR. In this section, we concisely present the basic principles of LS-SVR. For more details, the reader can refer to [8, 9]. Consider a regression problem with a training dataset $\{(x_i, y_i)\}_{i=1}^{n}$, where $x_i \in R^m$ is the input variable and $y_i \in R$ is the corresponding target. To derive a nonlinear regressor, LS-SVR can be obtained through solving the following optimization problem:

$$\begin{aligned} \min_{w,b,\xi_i} \quad & \frac{1}{2}\|w\|^2 + \frac{C}{2}\sum_{i=1}^{n}\xi_i^2 \\ \text{s.t.} \quad & y_i = w^\top \phi(x_i) + b + \xi_i, \quad i = 1,\ldots,n, \end{aligned} \tag{1}$$

where ξ_i represents the error variables, w represents the model complexity, $\phi(\cdot)$ is a nonlinear mapping which maps the input data into a high-dimensional feature space, and $C > 0$ is the regularization parameter that balances the model complexity and empirical risk. To solve (1), we need to introduce Lagrangian multipliers and construct a Lagrangian function. Utilizing the Karush-Kuhn-Tucker (KKT) conditions, we get the dual optimization problem

$$\begin{bmatrix} 0 & e^\top \\ e & K + \frac{1}{C}I_n \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{y} \end{bmatrix}, \tag{2}$$

where $e = (1, 1, \ldots, 1)^\top$, $\mathbf{y} = (y_1, y_2, \ldots, y_n)^\top$, I_n denotes $n\times n$ identity matrix, $K = (K_{ij})_{n\times n}$ is the kernel matrix with $K_{ij} = k(x_i, x_j) = \phi(x_i)^\top\phi(x_j)$, and $k(\cdot,\cdot)$ is the kernel function. By solving (2), the regressor can be gained as

$$f(x) = w^\top \phi(x) + b = \sum_{i=1}^{n}\alpha_i k(x_i, x) + b. \tag{3}$$

2.2. LS-SVR in the Primal Space. In this section, we describe LS-SVR solved in the primal space following the growing interest in training SVMs in the primal space in the last few years [21, 22]. Primal optimization of an SVM has strong similarities with the dual strategy [21] and can be implemented by the widely popular optimization techniques. The optimization problem of LS-SVR (1) can be described as

$$\min_{\mathbf{w},b} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2}\sum_{i=1}^{n} l_1(y_i - f(x_i)), \tag{4}$$

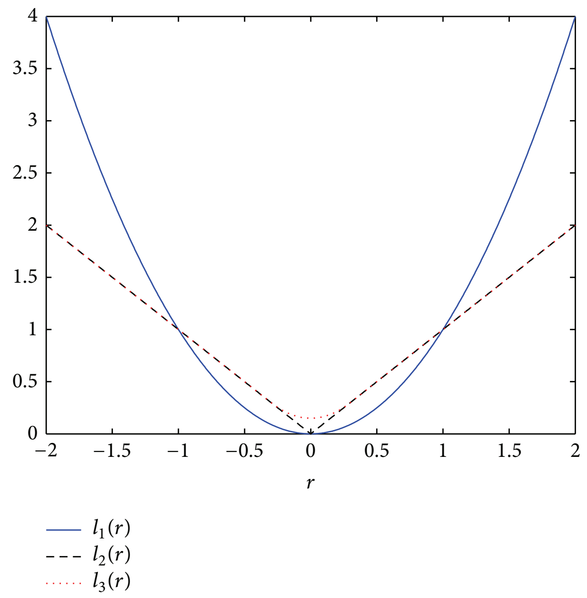

Figure 1: Squared loss function $l_1(r)$, absolute deviation loss function $l_2(r)$, and smooth approximation function $l_3(r)$ with $h = 0.3$.

where $l_1(r) = r^2$ with $r = y - f(x)$, and is a squared loss function, as shown in Figure 1. In the reproducing kernel Hilbert space $\mathscr{H}$, we rewrite the optimization problem (4) as

$$\min_{f} \quad \frac{1}{2}\|f\|_{\mathscr{H}}^2 + \frac{C}{2}\sum_{i=1}^{n} l_1(y_i - f(x_i)). \tag{5}$$

For the sake of simplicity, we can drop the bias b without loss of generalization performance of SVR [21]. According to [21], the optimal function for (5) can be expressed as a linear combination of the kernel functions centering the training samples:

$$f(x) = \sum_{i=1}^{n}\alpha_i k(x, x_i). \tag{6}$$

Substituting (6) into (5), we have

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} + \frac{C}{2}\sum_{i=1}^{n} l_1(y_i - K_i\boldsymbol{\alpha}), \tag{7}$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)^\top$ and K_i is the ith row of kernel matrix K.

3. Least Absolute Deviation SVR

As mentioned, LS-SVR is sensitive to outliers and noises with the squared loss function $l_1(r) = r^2$. When there exist outliers which are far away from the rest of samples, large errors will dominate SSE and the decision hyperplane of LS-SVR will severely deviate from the original position deteriorating the performance of LS-SVR.

In this section, we propose an absolute deviation loss function $l_2(r) = |r|$ to reduce the influence of outliers. This phenomenon is graphically depicted in Figure 1, which shows the squared loss function $l_1(r) = r^2$ and the absolute deviation one $l_2(r) = |r|$, respectively. From the figure, the exaggerative effect of $l_1(r) = r^2$ at points with large errors, as compared with $l_2(r) = |r|$, is evident.

The robust LAD-SVR model can be constructed as

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} + C\sum_{i=1}^{n} l_2\left(y_i - K_i\boldsymbol{\alpha}\right). \tag{8}$$

However, $l_2(r)$ is not differentiable, and the associated optimization problem is difficult to be solved. Inspired by the Huber loss function [23], we propose the following loss function:

$$l_3(r) = \begin{cases} |r|, & \text{if } |r| > h, \\ \dfrac{r^2}{2h} + \dfrac{h}{2}, & \text{otherwise}, \end{cases} \tag{9}$$

where $h > 0$ is the Huber parameter, and its shape is shown in Figure 1. It is verified that $l_3(r)$ is differentiable. For $h \to 0$, $l_3(r)$ approaches $l_2(r)$. Replacing $l_2(r)$ with $l_3(r)$ in (8), we obtain

$$\min_{\boldsymbol{\alpha}} \quad \mathscr{L}(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} + C\sum_{i=1}^{n} l_3\left(y_i - K_i\boldsymbol{\alpha}\right). \tag{10}$$

4. Newton Algorithm for LAD-SVR

Noticing that the objective function $\mathscr{L}(\boldsymbol{\alpha})$ of (10) is continuous and differentiable, (10) can be easily solved by Newton algorithm. At the tth iteration, we divide the training samples into two groups according to $|r_i^t| \le h$ and $|r_i^t| > h$. Let $S_1 = \{i \mid |r_i^t| \le h\}$ denote the index set of samples lying in the quadratic part of $l_3(r)$ and $S_2 = \{i \mid |r_i^t| > h\}$ the index set of samples lying in the linear part of $l_3(r)$. $|S_1|$ and $|S_2|$ represent the number of samples in S_1 and S_2; that is, $|S_1| + |S_2| = n$. For the sake of clarity, we suppose that the two groups are arranged in the order of S_1 and S_2. Furthermore, we define $n \times n$ diagonal matrices I_1 and I_2, where I_1 has the first $|S_1|$ entries being 1 and the others 0, and I_2 has the entries from $|S_1|+1$ to $|S_1|+|S_2|$ being 1 and the others 0. Then, we develop a Newton algorithm for (10). The gradient is

$$\nabla\mathscr{L}\left(\boldsymbol{\alpha}^t\right) = K\left(I_n + \frac{C}{h}I_1K\right)\boldsymbol{\alpha}^t - K\left[\frac{C}{h}I_1\mathbf{y} + CI_2\mathbf{s}\right], \tag{11}$$

where $\mathbf{y} = (y_1, \ldots, y_n)^\top$ and $\mathbf{s} = (s_1, \ldots, s_n)^\top$ with $s_i = \operatorname{sgn}(r_i)$. The Hessian matrix at the tth iteration is

$$H^t = K\left(I_n + \frac{C}{h}I_1K\right). \tag{12}$$

The Newton step at the $(t+1)$th iteration is

$$\begin{aligned} \boldsymbol{\alpha}^{t+1} &= \boldsymbol{\alpha}^t - \left(H^t\right)^{-1}\nabla\mathscr{L}\left(\boldsymbol{\alpha}^t\right) \\ &= \left(I_n + \frac{C}{h}I_1K\right)^{-1}\left(\frac{C}{h}I_1\mathbf{y} + CI_2\mathbf{s}\right). \end{aligned} \tag{13}$$

Denote $A = (I_{|S_1|}+(C/h)K_{S_1,S_1})^{-1}$. The inverse of $I_n+(C/h)I_1K$ can be calculated as follows:

$$\left(I_n + \frac{C}{h}I_1K\right)^{-1} = \begin{pmatrix} A & -\dfrac{C}{h}AK_{S_1,S_2} \\ O & I_{|S_2|} \end{pmatrix}. \tag{14}$$

The computational complexity of A^{-1} is $O((|S_1|)^3)$. Substituting (14) into (13), we obtain

$$\boldsymbol{\alpha}^{t+1} = C\begin{pmatrix} \dfrac{A}{h}\left[\mathbf{y}_{|S_1|} - CK_{S_1,S_2}\mathbf{s}_{|S_2|}\right] \\ \mathbf{s}_{|S_2|} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\alpha}_{S_1}^{t+1} \\ \boldsymbol{\alpha}_{S_2}^{t+1} \end{pmatrix}. \tag{15}$$

Having updated $\boldsymbol{\alpha}^{t+1}$, we get the corresponding regressor

$$f^{t+1}(x) = \sum_{i=1}^{n}\alpha_i^{t+1}k\left(x_i, x\right). \tag{16}$$

The flowchart of implementing LAD-SVR is depicted as follows.

Algorithm 1. LAD-SVR (Newton algorithm for LAD-SVR with absolute deviation loss function).

Input: training set $T = \{x_i, y_i\}_{i=1}^n$, kernel matrix K, and a small real number $\rho > 0$.

(1) Let $\boldsymbol{\alpha}^0 \in R^n$ and calculate $r_i^0 = y_i - K_i\boldsymbol{\alpha}^0, i = 1, \ldots, n$. Divide training set into two groups according to $|r_i^0|$. Set $t = 0$.

(2) Rearrange the groups in the order of S_1 and S_2; adjust K and $\mathbf{y}$ correspondingly. Solve $\nabla\mathscr{L}(\boldsymbol{\alpha}^t)$ by (11). If $\|\nabla\mathscr{L}(\boldsymbol{\alpha}^t)\| \le \rho$, stop; or else, go to the next step.

(3) Calculate $\boldsymbol{\alpha}^{t+1}$ by (15) and $f^{t+1}(x)$ by (16).

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

5. Experiments

In order to test the effectiveness of the proposed LAD-SVR, we conduct experiments on several datasets, including six artificial datasets and nine benchmark datasets, and compare it with LS-SVR. Gaussian kernel is selected as the kernel function in the experiments. All the experiments are implemented on Intel Pentium IV 3.00 GHz PC with 2 GB of RAM using Matlab 7.0 under Microsoft Windows XP. The linear system of equations in LS-SVR is realized by Matlab operation "\". Parameters selection is a crucial issue for modeling with the kernel methods, because improper parameters, such as the regularization parameter C and kernel parameter σ, will severely affect the generalization performance of SVR. Grid search [2] is a simple and direct method, which conducts an exhaustive search on the parameters space with the validation minimized. In this paper, we employ grid search for searching their optimal parameters such that they can achieve best performance on the test samples.

To evaluate the performances of the algorithms, we adopt the following four popular regression estimation criterions:

TABLE 1: Experiment results on $y = \sin(3x)/(3x)$.

Noise	Algorithm	RMSE	MAE	SSE/SST	SSR/SST
Type I	LS-SVR	0.0863	0.0676	0.0720	**1.0761**
	LAD-SVR	**0.0455**	**0.0364**	**0.0209**	1.0082
Type II	LS-SVR	0.0979	0.0795	0.0944	0.9374
	LAD-SVR	**0.0727**	**0.0607**	**0.0527**	**1.0224**
Type III	LS-SVR	0.0828	0.0665	0.0664	**1.1137**
	LAD-SVR	**0.0278**	**0.0223**	**0.0078**	1.0044
Type IV	LS-SVR	0.0899	0.0707	0.0772	0.9488
	LAD-SVR	**0.0544**	**0.0439**	**0.0291**	**1.0003**
Type V	LS-SVR	0.2120	0.1749	0.4268	0.7142
	LAD-SVR	**0.1861**	**0.1554**	**0.3372**	**0.7291**
Type VI	LS-SVR	0.1796	0.1498	0.3112	**0.9709**
	LAD-SVR	**0.1738**	**0.1391**	**0.2928**	0.8993

root mean square error (RMSE) [24], mean absolute error (MAE), ratio between the sum squared error SSE and the sum squared deviation testing samples SST (SSE/SST) [25], and ratio between interpretable sum deviation SSR and SST (SSR/SST) [25]. These criterions are defined as follows.

(1) RMSE $= \sqrt{(1/m)\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$.

(2) MAE $= (1/m)\sum_{i=1}^{m}|y_i - \hat{y}_i|$.

(3) SSE/SST $= \sum_{i=1}^{m}(\hat{y}_i - y_i)^2 / \sum_{i=1}^{m}(y_i - \overline{y})^2$.

(4) SSR/SST $= \sum_{i=1}^{m}(\hat{y}_i - \overline{y})^2 / \sum_{i=1}^{m}(y_i - \overline{y})^2$,

where m is the number of testing samples, y_i denotes the target, $\hat{y}_i$ is the corresponding prediction, and $\overline{y} = (1/m)\sum_{i=1}^{m} y_i$. RMSE is commonly used as the deviation measurement between the real and predicted values. It represents the fitting precision. The smaller RMSE is, the better fitting performance is. However, when noises are also used as testing samples, too small value of RMSE probably means overfitting of the regressor. MAE is also a popular deviation measurement between the real and predicted values. In most cases, small SSE/SST indicates good agreement between estimations and real values. Obtaining smaller SSE/SST usually accompanies an increase in SSR/SST. However, the extremely small value of SSE/SST is in fact not good, for it probably means overfitting of the regressor. Therefore, a good estimator should strike balance between SSE/SST and SSR/SST.

5.1. Experiments on Artificial Datasets. In artificial experiments, we generate the artificial datasets $\{(x_i, y_i)\}_{i=1}^{n}$ by the following Sinc function which is widely used in regression estimation [17, 24]. Consider

$$\begin{aligned}
&\text{Type I: } y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4,4], \\
&\qquad \gamma_i \sim N\left(0, 0.15^2\right). \\
&\text{Type II: } y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4,4], \\
&\qquad \gamma_i \sim N\left(0, 0.3^2\right). \\
&\text{Type III: } y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4,4], \\
&\qquad \gamma_i \sim U[-0.15, 0.15]. \\
&\text{Type IV: } y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4,4], \\
&\qquad \gamma_i \sim U[-0.3, 0.3]. \\
&\text{Type V: } y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4,4], \\
&\qquad \gamma_i \sim T(4). \\
&\text{Type VI: } y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4,4], \\
&\qquad \gamma_i \sim T(8),
\end{aligned} \tag{17}$$

where $N(0, d^2)$ represents the Gaussian random variable with zero means and variance d^2, $U[a,b]$ denotes the uniformly random variable in $[a,b]$, and $T(c)$ depicts the student random variable with freedom degree c.

In order to avoid biased comparisons, for each kind of noises, we randomly generate ten independent groups of noisy samples which, respectively, consist of 350 training samples and 500 test samples. For each training dataset, we randomly choose 1/5 samples and add large noise on their

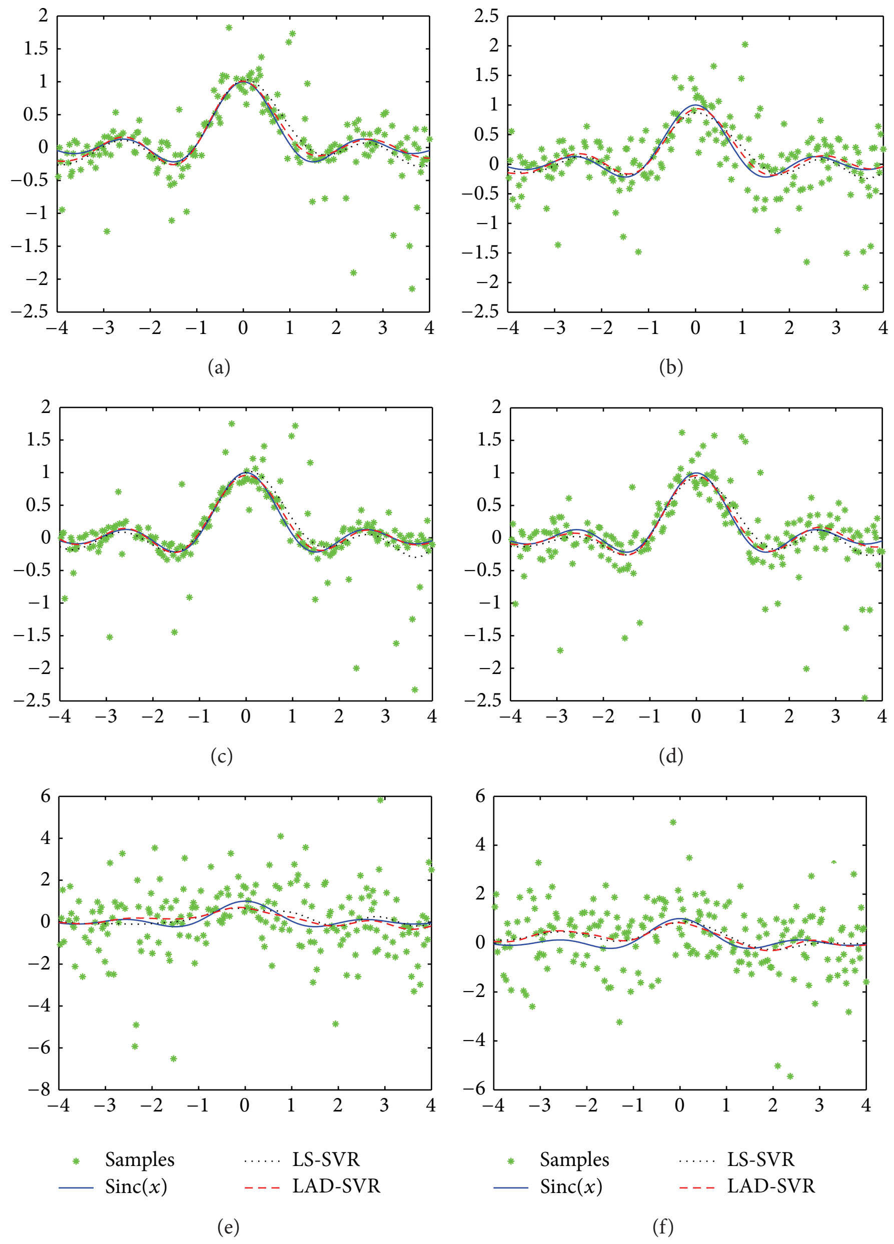

Figure 2: Experiment results on $y = \sin(3x)/(3x)$ with noise of Type I(a), II(b), . . ., VI(f).

targets to simulate outliers. The testing samples are uniformly from the objective Sinc function without any noise. Table 1 shows the average accuracies of LS-SVR and LAD-SVR with ten independent runs. From Table 1, we can see that LAD-SVR has advantages over LS-SVR for all types of noises in terms of RMSE, MAE, and SSE/SST. Hence, LAD-SVR is robust to noises and outliers. Moreover, LAD-SVR derives larger SSR/SST value for three types of noises (types II, IV, and V). From Figure 2, we can see that the LAD-SVR follows the actual data more closely than LS-SVR for most of the test samples. The main reason is that LAD-SVR employs an absolute deviation loss function which reduces the penalty of outliers in the training process. The histograms of LS-SVR and LAD-SVR for distribution of the error variables ξ_i for these different types of noises are shown in Figure 3. We notice that the histograms of LAD-SVR for all types of noises are closer to Gaussian distribution, compared with LS-SVR. Therefore, our proposed LAD-SVR derives better approximation than LS-SVR.

5.2. Experiments on Benchmark Datasets. In this section, we test nine benchmark datasets to further illustrate the effectiveness of the LAD-SVR, including Pyrimidines (Pyrim), Triazines, AutoMPG, Boston Housing (BH) and Servo from UCI datasets [26], Bodyfat, Pollution, Concrete Compressive Strength (Concrete) from StatLib database (Available from http://lib.stat.cmu.edu/datasets/), and Machine CPU

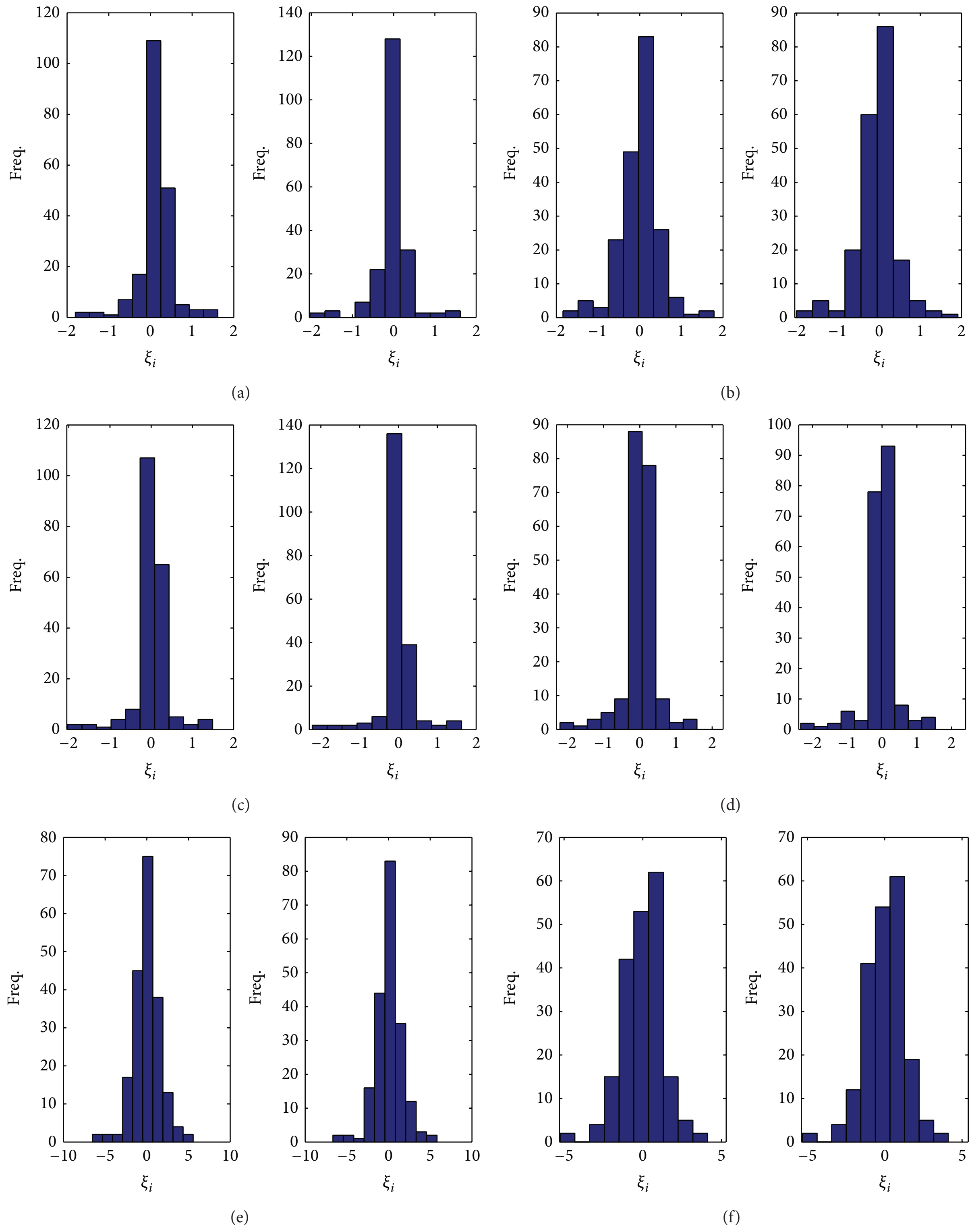

FIGURE 3: Histograms of distribution of the error variables ξ_i for noise of Type I ((a) LS-SVR (Left), LAD-SVR (Right)), II(b), . . ., VI(f).

(MCPU) from the web page (http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html), which are widely used in evaluating various regression algorithms. The detailed descriptions of datasets are presented in Table 2, where #train and #test denote the number of training and testing samples, respectively. In experiments, each dataset is randomly split into training and testing samples. For each training dataset, we randomly choose 1/5 samples and add large noise on their

Table 2: Experiment results on Benchmark datasets.

Number	Dataset	Algorithm	RMSE	MAE	SSE/SST	SSR/SST	Time (s)	Iter	#train	#test
1	Bodyfat	LS-SVR	0.0075	0.0056	0.1868	0.7799	**0.0854**	/	200	52
	(252 × 14)	LAD-SVR	**0.0026**	**0.0012**	**0.0277**	**0.9686**	0.0988	4.4	200	52
2	Pyrim	LS-SVR	0.1056	0.0649	0.5925	**0.5729**	**0.0137**	/	50	24
	(74 × 27)	LAD-SVR	**0.1047**	**0.0641**	**0.5832**	0.5342	0.0385	9.8	50	24
3	Pollution	LS-SVR	39.8592	31.0642	0.5358	**0.8910**	**0.0083**	/	40	20
	(60 × 16)	LAD-SVR	**36.1070**	**28.2570**	**0.4379**	0.7312	0.0202	5	40	20
4	Triazines	LS-SVR	0.1481	0.1126	0.9295	**0.3356**	**0.0759**	/	150	36
	(186 × 60)	LAD-SVR	**0.1415**	**0.1066**	**0.8420**	0.3026	0.0958	6	150	36
5	MCPU	LS-SVR	53.6345	27.5341	0.1567	0.9485	**0.0509**	/	150	59
	(209 × 6)	LAD-SVR	**50.2263**	**26.5241**	**0.1364**	**0.9666**	0.0883	5.4	150	59
6	AutoMPG	LS-SVR	2.8980	2.1601	0.1308	0.8421	**0.2916**	/	300	92
	(392 × 7)	LAD-SVR	**2.6630**	**1.9208**	**0.1102**	**0.8502**	0.3815	6.4	300	92
7	BH	LS-SVR	4.3860	3.2163	0.2402	0.8563	**0.2993**	/	300	206
	(506 × 13)	LAD-SVR	**3.6304**	**2.5349**	**0.1648**	**0.8593**	0.4971	8.8	300	206
8	Servo	LS-SVR	0.7054	0.4194	0.2126	**0.8353**	**0.0342**	/	100	67
	(167 × 4)	LAD-SVR	**0.6830**	**0.3767**	**0.2010**	0.8147	0.0561	4.9	100	67
9	Concrete	LS-SVR	7.0293	5.1977	0.1740	**0.9005**	**1.1719**	/	500	530
	(1030 × 8)	LAD-SVR	**6.8708**	**5.0748**	**0.1662**	0.8936	2.0737	8.7	500	530

targets to simulate outliers. Similar to the experiments on artificial datasets, the testing datasets are not added any noise on their targets. All the regression methods are repeated ten times with different partition of training and testing dataset.

Table 2 displays the testing results of LS-SVR and the proposed LAD-SVR. We observe that the three criterions (RMSE, MAE, and SSE/SST) of LAD-SVR are obviously better than LS-SVR on all datasets, which shows that the robust algorithm achieves better generalization performance and has good stability as well. Moreover, our LAD-SVR algorithm outperforms LS-SVR. For instance, LAD-SVR obtains the smaller RMSE, MAE, and SSE/SST on the Bodyfat dataset; meanwhile, it keeps larger SSR/SST than LS-SVR. The proposed algorithm derives the similar results on the MCPU, AutoMPG, and BH datasets.

To obtain the final regressor of LAD-SVR, the resultant model is implemented in the primal space by classical Newton algorithm iteratively. The number of iterations (Iter) and the running time (Time) including the training and testing time are listed in Table 2. Iter shows the average number of iterations of ten independent runs. Compared with LS-SVR, LAD-SVR requires more running time. The main reason is that the running time of LAD-SVR is affected by the selection approach of the starting point α^0, the value of $|S_1|$, and the number of iterations. In the experiments, the starting point α^0 is derived by LS-SVR on a small number of training samples. It can be observed that the average number of iterations does not exceed 10, which implies that LAD-SVR is suitable enough for medium and large scale problems. We notice that LAD-SVR does not burden the running time severely. A worse case is that the maximum ratio of their speeds is no more than 3 times on Pyrim dataset. These experimental results conclude that the proposed LAD-SVR is effective in dealing with robust regression problems.

6. Conclusion

In this paper, we propose LAD-SVR, a novel robust least squares support vector regression algorithm on dataset with outliers. Compared with the classical LS-SVR which is based on squared loss function, LAD-SVR employs an absolute deviation loss function to reduce the influence of outliers. To solve the resultant model, we smooth the proposed loss function by a Huber loss function and develop a Newton algorithm. Experimental results on both artificial datasets and benchmark datasets confirm that LAD-SVR owns better robustness compared with LS-SVR. However, LAD-SVR still loses sparseness as LS-SVR. In the future, we plan to develop more efficient LAD-SVR to improve the sparseness and robustness.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

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

References

[1] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.

[2] N. Cristianini and J. S. Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, Cambridge, UK, 2000.

[3] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design," *Neural Computation*, vol. 13, no. 3, pp. 637–649, 2001.

[4] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds., pp. 185–208, MIT Press, Cambridge, UK, 1999.

[5] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Proceedings of the 7th IEEE Workshop on Neural Networks for Signal Processing (NNSP '97)*, J. Principe, L. Gile, N. Morgan, and E. Wilson, Eds., pp. 276–285, September 1997.

[6] T. Joachims, "Making large-scale SVM learning practical," in *Advances in Kernel Methods-Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds., pp. 169–184, MIT Press, Cambridge, Mass, USA, 1999.

[7] C. Chang and C. Lin, "LIBSVM: a library for support vector machines," 2001, http://www.csie.ntu.edu.tw/~cjlin/.

[8] J. A. K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[9] J. A. K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: robustness and sparce approximation," *Neurocomputing*, vol. 48, pp. 85–105, 2002.

[10] W. Wen, Z. Hao, and X. Yang, "A heuristic weight-setting strategy and iteratively updating algorithm for weighted least-squares support vector regression," *Neurocomputing*, vol. 71, no. 16-18, pp. 3096–3103, 2008.

[11] K. De Brabanter, K. Pelckmans, J. De Brabanter et al., "Robustness of kernel based regression: acomparison of iterative weighting schemes," in *Proceedings of the 19th International Conference on Artificial Neural Networks (ICANN '09)*, 2009.

[12] J. Liu, J. Li, W. Xu, and Y. Shi, "A weighted *Lq* adaptive least squares support vector machine classifiers-Robust and sparse approximation," *Expert Systems with Applications*, vol. 38, no. 3, pp. 2253–2259, 2011.

[13] X. Chen, J. Yang, J. Liang, and Q. Ye, "Recursive robust least squares support vector regression based on maximum correntropy criterion," *Neurocomputing*, vol. 97, pp. 63–73, 2012.

[14] L. Xu, K. Crammer, and D. Schuurmans, "Robust support vector machine training via convex outlier ablation," in *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI '06)*, pp. 536–542, July 2006.

[15] P. J. Rousseeuw and K. van Driessen, "Computing LTS regression for large data sets," *Data Mining and Knowledge Discovery*, vol. 12, no. 1, pp. 29–45, 2006.

[16] W. Wen, Z. Hao, and X. Yang, "Robust least squares support vector machine based on recursive outlier elimination," *Soft Computing*, vol. 14, no. 11, pp. 1241–1251, 2010.

[17] C. Chuang and Z. Lee, "Hybrid robust support vector machines for regression with outliers," *Applied Soft Computing Journal*, vol. 11, no. 1, pp. 64–72, 2011.

[18] G. Bassett Jr. and R. Koenker, "Asymptotic theory of least absolute error regression," *Journal of the American Statistical Association*, vol. 73, no. 363, pp. 618–622, 1978.

[19] P. Bloomfield and W. L. Steiger, *Least Absolute Deviation: Theory, Applications and Algorithms*, Birkhauser, Boston, Mass, USA, 1983.

[20] H. Wang, G. Li, and G. Jiang, "Robust regression shrinkage and consistent variable selection through the LAD-Lasso," *Journal of Business & Economic Statistics*, vol. 25, no. 3, pp. 347–355, 2007.

[21] O. Chapelle, "Training a support vector machine in the primal," *Neural Computation*, vol. 19, no. 5, pp. 1155–1178, 2007.

[22] L. Bo, L. Wang, and L. Jiao, "Recursive finite Newton algorithm for support vector regression in the primal," *Neural Computation*, vol. 19, no. 4, pp. 1082–1096, 2007.

[23] P. J. Huber, *Robust Statistics*, Springer, Berlin, Germany, 2011.

[24] P. Zhong, Y. Xu, and Y. Zhao, "Training twin support vector regression via linear programming," *Neural Computing and Applications*, vol. 21, no. 2, pp. 399–407, 2012.

[25] X. Peng, "TSVR: an efficient Twin Support Vector Machine for regression," *Neural Networks*, vol. 23, no. 3, pp. 365–372, 2010.

[26] C. Blake and C. J. Merz, "UCI repository for machine learning databases," 1998, http://www.ics.uci.edu/~mlearn/MLRepository.html.