Research Article

Least Absolute Deviation Support Vector Regression

Kuaini Wang,1 Jingjing Zhang,1 Yanyan Chen,1,2 and Ping Zhong1

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

Copyright © 2014 Kuaini Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

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], SVMlight [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 performance 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 reveal the efficiency of the proposed method. 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 \( x_i \in \mathbb{R}^m \) is the input variable and \( y_i \in \mathbb{R} \) is the corresponding target. To derive a nonlinear regressor, LS-SVR can be obtained through solving the following optimization problem:

\[
\min_{\omega, b, \xi} \frac{1}{2} \| \omega \|^2 + \frac{C}{2} \sum_{i=1}^n \xi_i^2 \\
\text{s.t. } y_i = \omega^T \phi(x_i) + b + \xi_i, \quad i = 1, \ldots, n,
\]

where \( \xi_i \) represents the error variables, \( \omega \) 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

\[
\begin{bmatrix} 0 & e^T \\ e & K + \frac{1}{C} I_n \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ y \end{bmatrix},
\]

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

\[
f(x) = w^T \phi(x) + b = \sum_{i=1}^n \alpha_i k(x_i, x) + b.
\]

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

\[
\min_{\omega, b} \frac{1}{2} \| \omega \|^2 + \frac{C}{2} \sum_{i=1}^n l_i(y_i - f(x_i)),
\]

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

\[
\min_{f} \frac{1}{2} \| f \|^2_{\mathcal{H}} + \frac{C}{2} \sum_{i=1}^n l_i(y_i - f(x_i)).
\]

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:

\[
f(x) = \sum_{i=1}^n \alpha_i k(x_i, x).
\]

Substituting (6) into (5), we have

\[
\min_{\alpha} \frac{1}{2} \alpha^T K \alpha + \frac{C}{2} \sum_{i=1}^n l_i(y_i - K_i \alpha),
\]

where \( \alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T \) 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 exagerative 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

$$\min_{\alpha} \frac{1}{2} \alpha^T K \alpha + C \sum_{i=1}^{n} l_3(y_i - K_i \alpha).$$

(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:

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

(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

$$\min_{\alpha} \mathcal{L}(\alpha) = \frac{1}{2} \alpha^T K \alpha + C \sum_{i=1}^{n} l_3(y_i - K_i \alpha).$$

(10)

### 4. Newton Algorithm for LAD-SVR

Noticing that the objective function $\mathcal{L}(\alpha)$ of (10) is continuous and differentiable, (10) can be easily solved by Newton algorithm. At the $t$th iteration, we divide the training samples into two groups according to $|r_i^t| \leq h$ and $|r_i^t| > h$. Let $S_1 = \{i \mid |r_i^t| \leq h\}$ and $S_2 = \{i \mid |r_i^t| > h\}$, then the index set of samples lying in the quadratic part of $l_3(r)$ and $S_2$ is 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

$$\nabla \mathcal{L}(\alpha^t) = K\left(I_n + \frac{C}{h}I_1K\right)\alpha^t - K\left[\frac{C}{h}I_1y + CI_2s\right],$$

(11)

where $y = (y_1, \ldots, y_n)^T$ and $s = (s_1, \ldots, s_n)^T$ with $s_i = \text{sgn}(r_i)$. The Hessian matrix at the $t$th iteration is

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

(12)

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

$$\alpha^{t+1} = \alpha^t - \left(H^t\right)^{-1}\nabla \mathcal{L}(\alpha^t)$$

$$= \left(I_n + \frac{C}{h}I_1K\right)^{-1}\left[\frac{C}{h}I_1y + CI_2s\right].$$

(13)

The computational complexity of $A^{-1}$ is $O(|S_1|^2)$. Substituting (14) into (13), we obtain

$$\alpha^{t+1} = \left(I_n + \frac{C}{h}I_1K\right)^{-1}\left[\frac{C}{h}I_1y + CI_2s\right] = \left(A + \frac{C}{h}K\right)^{-1}\left[\frac{C}{h}I_1y + CI_2s\right].$$

(15)

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

$$f^{t+1}(x) = \sum_{i=1}^{n} a_i^{t+1} k(x_i, x).$$

(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 $\alpha^0 \in \mathbb{R}^n$ and calculate $r_i^0 = y_i - K_i \alpha^0$, $i = 1, \ldots, n$.

   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 $y$ correspondingly. Solve $\nabla \mathcal{L}(\alpha^t)$ by (11). If $\|\nabla \mathcal{L}(\alpha^t)\|_2^2 \leq \rho$, stop; or else, go to the next step.

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

4. Divide training set into two groups according to $|r_i^{t+1}| = |y_i - K_i \alpha^{t+1}|$. 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 $\sigma$, 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:

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

$$\left(I_n + \frac{C}{h}I_1K\right)^{-1} = \left(A - \frac{C}{h}AK_{S_1}\right).$$

(14)

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

$$\alpha^{t+1} = \mathcal{C}\left(\frac{A^{t+1}}{h} \left[y_{S_1} - CK_{S_1}s_{S_1}\right]\right) = \left[\alpha_{S_1}^{t+1}\right]_{S_1}.$$
Table 1: Experiment results on $y = \sin(3x)/(3x)$.

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

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

$$y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4, 4],$$

$$\gamma_i \sim N(0, 0.15^2).$$

Type II: $$y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4, 4],$$

$$\gamma_i \sim N(0, 0.3^2).$$

Type III: $$y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4, 4],$$

$$\gamma_i \sim U[-0.15, 0.15].$$

Type IV: $$y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4, 4],$$

$$\gamma_i \sim U[-0.3, 0.3].$$

Type V: $$y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4, 4],$$

$$\gamma_i \sim T(4).$$

Type VI: $$y_i = \frac{\sin(3x_i)}{3x_i} + \gamma_i, \quad x_i \in [-4, 4],$$

$$\gamma_i \sim T(8),$$

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...
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 $\xi$, 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...
(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.

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

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


