Neurocomputing 72 (2009) 2659–2669

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Localized support vector regression for time series prediction

Haiqin Yang *, Kaizhu Huang, Irwin King, Michael R. Lyu

Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

ARTICLE INFO

Article history:
Received 7 October 2007
Received in revised form
25 September 2008
Accepted 30 September 2008
Communicated by G.P. Zhang
Available online 1 November 2008

Keywords:
Support vector regression
Second order conic programming
Time series prediction

ABSTRACT

Time series prediction, especially financial time series prediction, is a challenging task in machine learning. In this issue, the data are usually non-stationary and volatile in nature. Because of its good generalization power, the support vector regression (SVR) has been widely applied in this application. The standard SVR employs a fixed ε-tube to tolerate noise and adopts the ℓ_p-norm ($p=1$ or 2) to model the functional complexity of the whole data set. One problem of the standard SVR is that it considers data in a global fashion only. Therefore it may lack the flexibility to capture the local trend of data; this is a critical aspect of volatile data, especially financial time series data. Aiming to attack this issue, we propose the localized support vector regression (LSVR) model. This novel model is demonstrated to provide a systematic and automatic scheme to adapt the margin locally and flexibly; while the margin in the standard SVR is fixed globally. Therefore, the LSVR can tolerate noise adaptively. The proposed LSVR is promising in the sense that it not only captures the local information in data, but more importantly, it establishes connection with several models. More specifically: (1) it can be regarded as the regression extension of a recently proposed promising classification model, the Maxi-Min Margin Machine; (2) it incorporates the standard SVR as a special case under certain mild assumptions. We provide both theoretical justifications and empirical evaluations for this novel model. The experimental results on synthetic data and real financial data demonstrate its advantages over the standard SVR.

1. Introduction

Time series analysis is a significant active research topic in machine learning and data engineering [30,33,29,21,34]. We consider the regression or prediction problem for financial time series data in this paper. The objective is to learn a model from a given financial time series data set, $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in \mathbb{R}$, and then use the learned model to make accurate predictions of y's for future values of $\mathbf{x}$'s.

The support vector regression (SVR), a successful method in dealing with this problem, is well suited for generalization [25,31,32]. The standard SVR adopts the ℓ_p-norm ($p=1$ or 2) to control the functional complexity and chooses an ε-insensitive loss function with a fixed tube (margin) to measure the empirical risk. By introducing the ℓ_p-norm, the optimization problem in SVR can be transformed to a tractable programming problem, in particular a quadratic programming problem when $p=2$. Furthermore, the ε-tube has the ability to tolerate noise in data, while fixing the tube confers the advantage of simplicity. Although these settings are effective in common applications, they are designed in a global fashion and lack the flexibility to capture the local trend in some applications, in particular in stock markets data or financial time series. In the context of financial time series prediction, the data are usually highly volatile and the associated variance of noise varies over time. In such domains, fixing the tube cannot capture the local trend of data and cannot tolerate noise adaptively.

One typical illustration can be seen in Fig. 1. In this figure, the data become more noisy as the x value of the data increases. As shown in Fig. 1(a), with a fixed ε-margin (set to 0.04 in this example), SVR considers the data globally and equally: the derived approximating function in SVR deviates from the actual data trend. On the other hand, as illustrated in Fig. 1(b), if we address the local volatility of the data by adaptively and automatically setting a small margin in low-volatile regions and a large margin in high-volatile regions, the resulting approximating function (the blue solid line in Fig. 1(b)) is more appropriate and reasonable.

In order to address this issue, in this paper, we propose the localized support vector regression (LSVR) model [11]. We will show that, by taking the local data trend into consideration, our model provides a systematic and automatic scheme to adapt the margin locally and flexibly. Moreover, we will demonstrate that this novel LSVR model has extensive connections with other models. Specifically, this model can be seen as an extension of a recently proposed general large margin classifier, the Maxi-Min

* Corresponding author.
E-mail address: hqyang@cse.cuhk.edu.hk (H. Yang).

0925-2312/$ - see front matter
doi:10.1016/j.neucom.2008.09.014

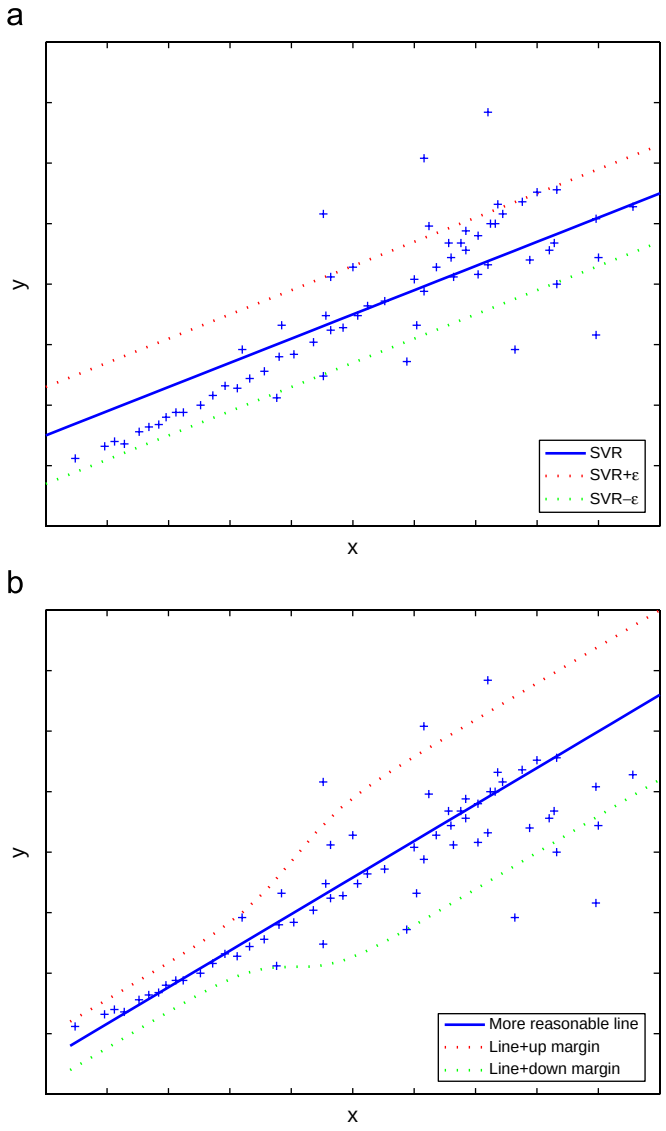


Fig. 1. Illustration of the ε-insensitive loss function with fixed and non-fixed margins in the feature space. In (b) a non-fixed margin setting is more reasonable. It can moderate the effect of the noise by enlarging (shrinking) the margin width in the local area, where contains large (small) variance of noise.

Margin Machine (M^4) [9,13], for regression tasks; it can also yield a special case, which will be proven to be equivalent with the standard SVR under certain mild assumptions. One critical feature of our model is that the associated optimization of LSVR can be relaxed as a second order cone programming (SOCP) problem. This problem can attain global optimal solution and be solved efficiently in polynomial time [19,17,22]. Another appealing feature is that kernelization [5,24] is also applicable to the LSVR model. Therefore, the proposed LSVR can generate non-linear approximating functions and hence can be applied to more general regression tasks, e.g., time series prediction. Specifically, the tube here is adapted directly according to the functional complexity and the local trend of the data. This provides a systematic and rigorous way to moderate the margin automatically.

Currently, there are several other directions to set the ε value. In [23], the standard ε-SVR is transformed to the v-SVR, where the ε parameter is replaced by a new parameter, v. It is stated that v is in the range of [0 1] and it is easier to control the number of errors than ε. However, this parameter v is still global for the whole data set and it does not capture the local change of the data. In [29], the ε is set in a descending order, putting more weights on the recent data. In [33,34], the ε is set by capturing the local variance of the data. Fernandez et al. propose to predict financial time series by using a fixed number of neighbor points close to the input sample [6]. This method is similar to the K-nearest neighbor method [4] and easy to implement. However, all the training samples need to be stored beforehand, and a new regression model needs to be trained again in order to make predictions for a new sample. In [15], the ε in a least square support vector machine is set by three different fuzzy membership functions. However, as shown in the experiments, our proposed LSVR generally performs better than the least square method in the used financial data sets.

The rest of this paper is organized as follows. In Section 2, we review the standard SVR. The linear LSVR model, including its model definition, appealing features, and optimization method, is described in Section 3. In Section 4, we demonstrate how the LSVR model can be linked with other models including M^4 and SVR. The kernelized LSVR is tackled in Section 5 by using Mercer's kernel. In Section 6, we present the result of experiments using both synthetic and real financial data. Finally, we set out the conclusion and propose future work in Section 7.

2. Support vector regression

We define a training data set $\mathscr{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i \in \mathscr{X}$, $y_i \in \mathbb{R}$, N is the number of training data points, and $\mathscr{X}$ denotes the space of the input samples $\mathbb{R}^n$. The aim of regression is to find a function which cannot only approximate these data well, but also can predict the value of y for future data $\mathbf{x}$ accurately.

In general, the approximating function in SVR takes the following linear form:

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

where $\mathbf{w} \in \mathbb{R}^n$ and $b \in R$. Furthermore, the above linear regression model can be extended into the non-linear one by using Mercer's kernel [5,24].

Now the question is to determine $\mathbf{w}$ and b from the training data by minimizing the regression risk, $R_{reg}(f)$, which is defined as

$$R_{reg}(f) = \Omega[f] + C\sum_{i=1}^{N} \Gamma(f(\mathbf{x}_i) - y_i), \tag{2}$$

where $\Omega[f]$ is the structure risk used to control the smoothness or complexity of the function, $\Gamma(\cdot)$ is a loss function that measures the empirical risk, and C is a pre-specified trade-off value. Generally, in SVR, $\Omega[f]$ takes the form of $\|\mathbf{w}\|_1$ in ℓ_1-SVR or $\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$ in ℓ_2-SVR. The empirical loss function adopts the form of an ε-insensitive loss function [25,32], which is defined as follows:

$$\Gamma(f(\mathbf{x}) - y) = \begin{cases} 0 & \text{if } |y - f(\mathbf{x})| < \varepsilon, \\ |y - f(\mathbf{x})| - \varepsilon & \text{otherwise.} \end{cases} \tag{3}$$

In this function, when the data points are in the range of $\pm\varepsilon$, they do not contribute to the empirical error.

The complete optimization of SVR (or more precisely, the optimization of ℓ_1-SVR) can be written as follows:

$$\min_{\mathbf{w},b,\xi_i,\xi_i^*} \quad \|\mathbf{w}\|_1 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \tag{4}$$

$$\text{s.t.} \quad y_i - (\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) \leqslant \varepsilon + \xi_i, \tag{5}$$

$$(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i + b) - y_i \leqslant \varepsilon + \xi_i^*, \tag{6}$$

$$\xi_i \geqslant 0, \quad \xi_i^* \geqslant 0, \quad i = 1, \ldots, N, \tag{7}$$

where ξ_i and ξ_i^* are the corresponding positive and negative errors at the i-th point, respectively.

The above optimization problem can be solved by the linear programming method [2]. When the structure risk term $\Omega[f]$ takes the form of $\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$ (as in ℓ_2-SVR), the optimization becomes a quadratic programming problem [1,32].

In the above optimization problem, the standard SVR fixes the margin ε globally for all data points. Although this simple setting achieves great success in many tasks, it lacks the flexibility to capture the data's volatility, which is a typical feature of financial time series data. In order to address this problem, we therefore develop the novel LSVR model.

3. LSVR model

In this section, we first present the definition of the LSVR model. We then detail its interpretation and its appealing characteristics. After that, we state its corresponding optimization method.

3.1. Model definition

The objective of the LSVR model is to learn the linear approximating function in $\mathscr{D}$ by making the function locally as involatile as possible while keeping the error as small as possible. We formulate this objective as follows:

$$\begin{aligned}\min_{\mathbf{w},b,\xi_i,\xi_i^*} \quad & \frac{1}{N}\sum_{i=1}^{N}\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}}+C\sum_{i=1}^{N}(\xi_i+\xi_i^*) \qquad (8)\\ \text{s.t.} \quad & y_i-(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)\leqslant\varepsilon\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}}+\xi_i,\\ & (\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-y_i\leqslant\varepsilon\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}}+\xi_i^*,\\ & \xi_i\geqslant 0,\quad \xi_i^*\geqslant 0,\quad i=1,\ldots,N, \qquad (9)\end{aligned}$$

where ξ_i, ξ_i^*, and ε are defined as in the previous section. Σ_i is the covariance matrix formed by the i-th data point and those data points close to it.

3.2. Model interpretations

In this section, we interpret our novel LSVR model. First, we discuss the physical meaning of the term $\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}$.

Suppose $y_i=\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b$ and $\bar{y}_i=\mathbf{w}^{\mathrm{T}}\bar{\mathbf{x}}_i+b$ ($\bar{\mathbf{x}}_i$ denotes the mean of $\mathbf{x}_i$ and a certain number of points closest to it). We have the variance around the i-th data point as

$$\begin{aligned}\Delta_i &= \frac{1}{2k+1}\sum_{j=-k}^{k}(y_{i+j}-\bar{y}_i)^2\\ &= \frac{1}{2k+1}\sum_{j=-k}^{k}(\mathbf{w}^{\mathrm{T}}(\mathbf{x}_{i+j}-\bar{\mathbf{x}}_i))^2\\ &= \mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w},\end{aligned}$$

where $2k$ is the number of data points closest to the i-th data point. Therefore, $\Delta_i=\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}$ actually captures the volatility in the local region around the i-th data point. In addition, Δ_i can also measure the complexity of the function around the i-th data point, since it reflects the smoothness in the corresponding local region.

By using the first interpretation of $\Delta_i=\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}$ (representing the local volatility), LSVR can systematically and automatically vary the margin: If the i-th data point lies in an area with a larger variance of noise, it will contribute to a larger $\varepsilon\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}}$ or a larger local margin, resulting in a reduction of the impact of the noise around the point. On the other hand, if the i-th data point is in the region with a smaller variance of noise, the local margin, $\varepsilon\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}}$, will be smaller; in this case, the corresponding point will contribute more in the fitting process. By contrast, the standard SVR adopts a fixed margin, which treats each point equally and therefore lacks the ability to tolerate variation of noise.

By applying the second interpretation of $\Delta_i=\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}$, namely, a measure describing the local functional complexity, LSVR controls the overall smoothness of the approximating function by minimizing the average of Δ_i, as seen in (8). In contrast, the standard SVR globally minimizes a complexity term, i.e., $\|\mathbf{w}\|_1$ or $\frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}$, which is insensitive to local changes in the complexity of the function.

3.3. Optimization

In order to solve the optimization problem of (8), we introduce auxiliary variables, $t_1,\ldots,t_N$, and transform the problem as follows:

$$\begin{aligned}\min_{\mathbf{w},b,t_i,\xi_i,\xi_i^*} \quad & \frac{1}{N}\sum_{i=1}^{N}t_i+C\sum_{i=1}^{N}(\xi_i+\xi_i^*)\\ \text{s.t.} \quad & y_i-(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)\leqslant\varepsilon\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}}+\xi_i, \qquad (10)\\ & (\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-y_i\leqslant\varepsilon\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}}+\xi_i^*,\\ & \sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}}\leqslant t_i,\\ & t_i\geqslant 0,\quad \xi_i\geqslant 0,\quad \xi_i^*\geqslant 0,\quad i=1,\ldots,N. \qquad (11)\end{aligned}$$

It is clear that (10) and (11) are non-convex constraints. This may present difficulties in optimizing the LSVR problem. In the following, we relax the optimization to an SOCP problem [17] by replacing $\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}}$ with its upper bound t_i:

$$\begin{aligned}\min_{\mathbf{w},b,t_i,\xi_i,\xi_i^*} \quad & \frac{1}{N}\sum_{i=1}^{N}t_i+C\sum_{i=1}^{N}(\xi_i+\xi_i^*) \qquad (12)\\ \text{s.t.} \quad & y_i-(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)\leqslant\varepsilon t_i+\xi_i, \qquad (13)\\ & (\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-y_i\leqslant\varepsilon t_i+\xi_i^*, \qquad (14)\\ & \sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}}\leqslant t_i, \qquad (15)\\ & t_i\geqslant 0,\quad \xi_i\geqslant 0,\quad \xi_i^*\geqslant 0,\quad i=1,\ldots,N. \qquad (16)\end{aligned}$$

Since t_i is closely related to $\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i\mathbf{w}}$, weighting the margin width with t_i will achieve the original objective, i.e., adapting the margin flexibly. Furthermore, the relaxed form is a linear programming problem under quadratic cone constraints, or more specifically it is an SOCP problem. Therefore, this problem can be solved in polynomial time by using many general optimization packages, e.g., Sedumi [27,28]. Another advantage is that the relaxation also enables the application of kernelization, which can yield more general non-linear approximating functions. This will be demonstrated in Section 5.

We now analyze the time complexity of LSVR. As indicated in [17], if the SOCP is solved based on interior-point methods, it contains a worst-case complexity of $O(n^3)$. Adding the cost of forming the system matrix (constraint matrix), which is $O(Nn^3)$, the total complexity would be $O((N+1)n^3)$, which is in the same order as the M^4 and can be solved in polynomial time. Note that for time series prediction, we do not need to use sorting methods to find the closest points for each data sample, since the series itself provides the order information. For example, $2k$ points closest to the i-th point are simply those data with time values $i-k, i-k+1,\ldots,i-1,i+1,\ldots,i+k$. Therefore no further computation is involved.

4. Theoretical connections

In this section, we establish various connections from our novel model to other models. We first show that the LSVR can be

considered as the extension of the M^4 to regression tasks. We then demonstrate how the standard SVR can be incorporated as a special case of LSVR.

4.1. Connection with M^4

The LSVR model can also be considered as an extension of the general large margin classifier, the M^4 [9]. Within the framework of binary classification, which consists of two classes of data, X and Y, the M^4 model is formulated as follows:

$$\max_{\rho,\mathbf{w}\neq\mathbf{0},b} \quad \rho \tag{17}$$

$$\text{s.t.} \quad \frac{(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)}{\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_{\mathbf{x}}\mathbf{w}}}\geqslant\rho, \quad i=1,2,\ldots,N_{\mathbf{x}}, \tag{18}$$

$$\frac{-(\mathbf{w}^{\mathrm{T}}\mathbf{y}_j+b)}{\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_{\mathbf{y}}\mathbf{w}}}\geqslant\rho, \quad j=1,2,\ldots,N_{\mathbf{y}}, \tag{19}$$

where $\mathbf{\Sigma_x}$ and $\mathbf{\Sigma_y}$ refer to the covariance matrices of the X and the Y data, respectively. The M^4 model seeks to maximize the margin defined as the minimum Mahalanobis distance for all training samples,[1] while simultaneously classifying all the data correctly. This model has been shown to contain the support vector machine, the minimax probability machine [16,8,12,10], and the Fisher discriminant analysis [7] as special cases. Furthermore, it can be linked with the minimum error minimax probability machine known as a worst-case distribution-free classifier [14].

Within the framework of classifications, M^4 considers different data trends for different classes, i.e., it adopts the covariance information of two classes of data, $\Sigma_{\mathbf{x}}$ and $\Sigma_{\mathbf{y}}$. Analogously, in the novel LSVR model, we allow different data trends for different regions, which is more suitable for a regression application.

4.2. Connection with SVR

We now analyze the connection of the LSVR model with the standard SVR model. By considering the data trend globally and equally, i.e., setting $\Sigma_i=\Sigma$, for $i=1,\ldots,N$, we can transform the optimization of (8) as follows:

$$\begin{aligned}\min_{\mathbf{w},b,\xi_i,\xi_i^*} \quad & \sqrt{\mathbf{w}^{\mathrm{T}}\Sigma\mathbf{w}}+C\sum_{i=1}^{N}(\xi_i+\xi_i^*)\\ \text{s.t.} \quad & y_i-(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)\leqslant\varepsilon\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma\mathbf{w}}+\xi_i,\\ & (\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-y_i\leqslant\varepsilon\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma\mathbf{w}}+\xi_i^*,\\ & \xi_i\geqslant 0, \quad \xi_i^*\geqslant 0, \quad i=1,\ldots,N.\end{aligned} \tag{20}$$

Further, if $\Sigma=\mathbf{I}$, we obtain

$$\min_{\mathbf{w},b,\xi_i,\xi_i^*} \quad \|\mathbf{w}\|_1+C\sum_{i=1}^{N}(\xi_i+\xi_i^*) \tag{21}$$

$$\begin{aligned}\text{s.t.} \quad & y_i-(\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)\leqslant\|\mathbf{w}\|_1\varepsilon+\xi_i,\\ & (\mathbf{w}^{\mathrm{T}}\mathbf{x}_i+b)-y_i\leqslant\|\mathbf{w}\|_1\varepsilon+\xi_i^*,\\ & \xi_i\geqslant 0, \quad \xi_i^*\geqslant 0, \quad i=1,\ldots,N.\end{aligned} \tag{22}$$

The above optimization problem is very similar to the ℓ_1-norm SVR, except that it has a margin related to the complexity term. In the following, we will prove that the above optimization is actually equivalent to the ℓ_1-norm SVR in the sense that, if one of the models for a given value of the parameter ε produces a solution $\{\mathbf{w},b\}$, then the other method can derive the same solution by adapting its corresponding parameter ε.

[1] This also inspired the name of this model.

Lemma 1. *The LSVR model with setting $\Sigma_i=\mathbf{I}$ is equivalent to the ℓ_1-norm SVR in the sense that*: (1) *Assuming a unique ε_1^* exists for making ℓ_1-norm SVR optimal,[2] if for ε_1^* the ℓ_1-norm SVR achieves a solution $\{\mathbf{w}_1^*,b_1^*\}=\mathrm{SVR}(\varepsilon_1^*)$, then the LSVR can produce the same solution by setting the parameter $\varepsilon=\varepsilon_1^*/\|\mathbf{w}_1^*\|_1$, i.e., $\mathrm{LSVR}(\varepsilon_1^*/\|\mathbf{w}_1^*\|_1)=\mathrm{SVR}(\varepsilon_1^*)$.* (2) *Assuming a unique ε_2^* exists for making the special case of LSVR optimal,[3] if for ε_2^* the special case of LSVR achieves a solution $\{\mathbf{w}_2^*,b_2^*\}=\mathrm{LSVR}(\varepsilon_2^*)$, then the ℓ_1-norm SVR can produce the same solution by setting the parameter $\varepsilon=\varepsilon_2^*\|\mathbf{w}_2^*\|_1$, i.e., $\mathrm{SVR}(\varepsilon_2^*\|\mathbf{w}_2^*\|_1)=\mathrm{LSVR}(\varepsilon_2^*)$.*

Proof. Since (1) and (2) are very similar statements, we only prove (1). When ε is set to $\varepsilon_1^*/\|\mathbf{w}_1^*\|_1$ in the special case of LSVR, the value of the objective function of LSVR will always be smaller than the one obtained by setting $\{\mathbf{w},b\}=\{\mathbf{w}_1^*,b_1^*\}$, since $\{\mathbf{w}_1^*,b_1^*\}$ is easily verified to satisfy the constraints of LSVR. Namely,

$$f_{LSVR}\left(\frac{\varepsilon_1^*}{\|\mathbf{w}_1^*\|_1}\right)\leqslant f_{SVR}(\varepsilon_1^*), \tag{23}$$

where we use $f_{SVR}(\varepsilon_s)$ ($f_{LSVR}(\varepsilon_s)$) to denote the value of the SVR (LSVR) objective function when ε is set to a specific value ε_s.

We assume the solution to be $\{\mathbf{w}_2,b_2\}$ when ε is set to $\varepsilon_1^*/\|\mathbf{w}_1^*\|_1$ in the special case of LSVR. Similarly, by setting $\varepsilon=\varepsilon_1^*\|\mathbf{w}_2\|/\|\mathbf{w}_1^*\|_1$ in SVR, we have

$$f_{SVR}\left(\varepsilon_1^*\frac{\|\mathbf{w}_2\|_1}{\|\mathbf{w}_1^*\|_1}\right)\leqslant f_{LSVR}\left(\frac{\varepsilon_1^*}{\|\mathbf{w}_1^*\|_1}\right). \tag{24}$$

Combining (23) and (24), we have

$$f_{SVR}\left(\varepsilon_1^*\frac{\|\mathbf{w}_2\|_1}{\|\mathbf{w}_1^*\|_1}\right)\leqslant f_{LSVR}\left(\frac{\varepsilon_1^*}{\|\mathbf{w}_1^*\|_1}\right)\leqslant f_{SVR}(\varepsilon_1^*). \tag{25}$$

Since ε_1^* is the unique ε that achieves the objective of minimizing SVR, (25) implies that $\|\mathbf{w}_2\|_1=\|\mathbf{w}_1^*\|_1$. This further implies that $\mathbf{w}_2$ is equal to $\mathbf{w}_1^*$, since, with $\|\mathbf{w}_2\|_1=\|\mathbf{w}_1^*\|_1$, the optimization of LSVR is exactly the same as that of SVR. This will naturally lead to the same solution. □

In addition, if in LSVR we use the item of $\mathbf{w}^{\mathrm{T}}\Sigma\mathbf{w}$ instead of its square root form as the structure risk or complexity risk, a similar proof can also be developed showing that the ℓ_2-norm SVR is equivalent to the special case of LSVR with $\Sigma_i=\Sigma$. In summary, we can see that the LSVR model actually contains the standard SVR model as its special case.

5. Kernelization

In the above discussion, the approximating function derived from LSVR is provided in a linear configuration. In the following, we show that kernelization is also applicable to the relaxed LSVR model. This can therefore generate non-linear approximating functions.

Kernelization [24] is a technique that maps n-dimensional data points into a high-dimensional feature space $\mathbb{R}^f$, where a linear function corresponds to a non-linear function in the original space. The kernel mapping can be formulated as: $\mathbf{x}_i\rightarrow\varphi(\mathbf{x}_i)$, where $i=1,\ldots,N$, and $\varphi:\mathbb{R}^n\rightarrow\mathbb{R}^f$ is a mapping function. The corresponding linear approximating function in $\mathbb{R}^f$ is $\mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x})=b$, where $\mathbf{w}$, $\varphi(\mathbf{x})\in\mathbb{R}^f$, and $b\in\mathbb{R}$.

[2] This means that setting ε to ε_1^* will minimize the objective function of SVR.
[3] This means that setting ε to ε_2^* will minimize the objective function of LSVR.

The optimization of the relaxed LSVR in the feature space can be written as

$$\min_{\mathbf{w},b,t_i,\xi_i,\xi_i^*} \quad \frac{1}{N}\sum_{i=1}^{N} t_i + C\sum_{i=1}^{N}(\xi_i+\xi_i^*) \tag{26}$$

$$\text{s.t.} \quad y_i - (\mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x}_i)+b) \leqslant \varepsilon t_i + \xi_i, \tag{27}$$

$$(\mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x}_i)+b) - y_i \leqslant \varepsilon t_i + \xi_i^*, \tag{28}$$

$$\sqrt{\mathbf{w}^{\mathrm{T}}\Sigma_i^{\varphi}\mathbf{w}} \leqslant t_i, \tag{29}$$

$$t_i \geqslant 0, \quad \xi_i \geqslant 0, \quad \xi_i^* \geqslant 0, \quad i=1,\ldots,N.$$

However, to apply the kernelization, we need to represent the optimization and the final approximating function in a kernel form, $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j)=\varphi(\mathbf{x}_i)^{\mathrm{T}}\varphi(\mathbf{x}_j)$, namely, an inner product form of the mapping data points.

5.1. Basis of kernelization for LSVR

In the following, we demonstrate that kernelization indeed works in LSVR, provided that suitable estimates of means and covariance matrices are applied therein.

Theorem 2. *If the corresponding local covariance Σ_i^{φ} can be estimated by the mapped training data, i.e., $\hat{\varphi}_i$, Σ_i^{φ} can be written as*

$$\Sigma_i^{\varphi} = \frac{1}{2k+1}\sum_{j=-k}^{k}(\varphi(\mathbf{x}_{i+j})-\hat{\varphi}_i)(\varphi(\mathbf{x}_{i+j})-\hat{\varphi}_i)^{\mathrm{T}}, \tag{30}$$

$$\text{with } \hat{\varphi}_i = \frac{1}{2k+1}\sum_{j=-k}^{k}\varphi(\mathbf{x}_{i+j}), \tag{31}$$

where we just consider $2k$ data points which are the closest to the i-th data, then the optimal $\mathbf{w}$ lies in the span of the mapped training data.

Proof. Suppose $\mathbf{w}=\mathbf{w}_p+\mathbf{w}_o$, where $\mathbf{w}_p$ is the projection of $\mathbf{w}$ in the span of the mapped training data, $\mathbf{w}_o$ is the orthogonal component to the span. Since $\mathbf{w}_o^{\mathrm{T}}\varphi(\mathbf{x}_i)=0$ for $i=1,\ldots,N$, we can easily know that

$$\mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x}_i) = \mathbf{w}_p^{\mathrm{T}}\varphi(\mathbf{x}_i),$$
$$\mathbf{w}^{\mathrm{T}}\Sigma_i^{\varphi}\mathbf{w} = \mathbf{w}_p^{\mathrm{T}}\Sigma_i^{\varphi}\mathbf{w}_p.$$

Therefore, we can omit $\mathbf{w}_o$ since it disappears in the optimization. We then set it to $\mathbf{0}$ and obtain $\mathbf{w}=\mathbf{w}_p$, i.e., the optimal $\mathbf{w}$ lies in the span of the mapped training data. □

According to Theorem 2, we can write $\mathbf{w}$ as a linear combination of training data points:

$$\mathbf{w} = \sum_{i=1}^{N}\mu_i\varphi(\mathbf{x}_i), \tag{32}$$

where the coefficients $\mu_i \in \mathbb{R},\ i=1,\ldots,N$.

5.2. The kernelization result

Before we present the main kernelization result, we first introduce the notation. The element of the Gram matrix $\mathbf{K}$ in the position of (i,j) is defined as $\mathbf{K}_{i,j}=\varphi(\mathbf{x}_i)^{\mathrm{T}}\varphi(\mathbf{x}_j)$ for $i,j=1,2,\ldots,N$. We further define $\mathbf{K}_{\cdot i}$ as the i-th column of the Gram matrix. Namely,

$$\mathbf{K}_{\cdot i} := [\mathbf{K}_{1,i},\mathbf{K}_{2,i},\ldots,\mathbf{K}_{N,i}]^{\mathrm{T}}.$$

We define the block of the Gram matrix $\mathbf{K}_{[i-k:i+k,N]}$ as follows:

$$\mathbf{K}_{[i-k:i+k,N]} := \begin{pmatrix} \mathbf{K}_{i-k,1} & \cdots & \mathbf{K}_{i-k,N} \\ \vdots & \ddots & \vdots \\ \mathbf{K}_{i+k,1} & \cdots & \mathbf{K}_{i+k,N} \end{pmatrix}.$$

Furthermore, the matrix $\mathbf{L}_i$ is denoted as

$$\mathbf{L}_i = \frac{1}{\sqrt{2k+1}}(\mathbf{K}_{[i-k:i+k,N]} - \mathbf{1}_{2k+1}\mathbf{I}_i^{\mathrm{T}}).$$

In the above, the t-th element of $\mathbf{I}_i^{\mathrm{T}}$ is defined as $(\mathbf{I}_i^{\mathrm{T}})_t = (1/(2k+1))\sum_{j=-k}^{k}\mathbf{K}(\mathbf{x}_{i+j},\mathbf{x}_t)$, and $\mathbf{1}_{2k+1}$ is a column vector of dimension $2k+1$ with each element equal to 1.

We present the kernelization result as the following theorem.

Theorem 3 (*Kernelization theorem of LSVR*). *Finding the optimal approximating hyperplane for LSVR involves solving the following optimization problem*:

$$\min_{\mu,b,t_i,\xi_i,\xi_i^*} \quad \frac{1}{N}\sum_{i=1}^{N} t_i + C\sum_{i=1}^{N}(\xi_i+\xi_i^*)$$

$$\text{s.t.} \quad y_i - (\mu^{\mathrm{T}}\mathbf{K}_{\cdot i}+b) \leqslant \varepsilon t_i + \xi_i,$$

$$(\mu^{\mathrm{T}}\mathbf{K}_{\cdot i}+b) - y_i \leqslant \varepsilon t_i + \xi_i^*,$$

$$\sqrt{\mu^{\mathrm{T}}\mathbf{L}_i^{\mathrm{T}}\mathbf{L}_i\mu} \leqslant t_i,$$

$$t_i \geqslant 0, \quad \xi_i \geqslant 0, \quad \xi_i^* \geqslant 0, \quad i=1,\ldots,N.$$

Proof. By using Theorem 2, we write $\mathbf{w}$ as $\sum_{j=1}^{N}\mu_j\varphi(\mathbf{x}_j)$ and substitute it into (27)–(29). By rewriting (27)–(29) in the kernel form using a kernel function $\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j)=\varphi(\mathbf{x}_i)^{\mathrm{T}}\varphi(\mathbf{x}_j)$ and conducting some simple manipulations, we then obtain

$$\mathbf{w}^{\mathrm{T}}\varphi(\mathbf{x}_i) = \sum_{j=1}^{N}\mu_j\mathbf{K}(\mathbf{x}_i,\mathbf{x}_j) = \boldsymbol{\mu}^{\mathrm{T}}\mathbf{K}_{\cdot i},$$
$$\mathbf{w}^{\mathrm{T}}\Sigma_i^{\varphi}\mathbf{w} = \boldsymbol{\mu}^{\mathrm{T}}\mathbf{L}_i^{\mathrm{T}}\mathbf{L}_i\boldsymbol{\mu}.$$

If we simply substitute the above into the optimization of LSVR, we obtain the kernelized optimization form. □

Hence, the kernelized LSVR is transformed into an SOCP problem which is a convex programming problem [17,3].

Note that, in the above, the matrix $\mathbf{L}_i$ is not necessarily a square matrix, and hence it is not necessarily positive definite. Instead, $\mathbf{L}_i^{\mathrm{T}}\mathbf{L}_i$ is guaranteed to be at least positive semi-definite. This can be clearly observed and verified since $\mathbf{L}_i^{\mathrm{T}}\mathbf{L}_i$ describes the covariance around the i-th data point in the feature space. In practice, we still need to make this matrix positive definite. This can be achieved by adding a small positive regularization term in the diagonal elements of $\mathbf{L}_i^{\mathrm{T}}\mathbf{L}_i$.

6. Experiments

In this section, we compare the three algorithms, our LSVR model, the ε-SVR, and the regularized least squares fuzzy SVR (RLFSVR) [15], on both the synthetic *Sinc* data and the real world financial time series data. The experiments are performed on a PC with a 2.13 GHz Intel Core 2 CPU and 1 G RAM.

The SOCP problem associated with our LSVR model is solved using a general software package, Sedumi [27,28]. The SVR algorithm is performed by the toolbox, Spider.[4] The RLFSVR is implemented in Matlab 7.1.

6.1. Evaluations on synthetic Sinc data

The synthetic data are tested in the function approximation scheme to examine the performance of different models. Fifty examples (x_i,y_i) are generated from a *Sinc* function [23], where x_i's are drawn uniformly from $[-3.0,3.0]$, and $y_i=\sin(\pi x_i)/(\pi x_i)+\tau_i$, with τ_i being drawn from a Gaussian with zero mean and variance σ^2.

[4] http://www.kyb.tuebingen.mpg.de/bs/people/spider/

Two cases are evaluated. In the first case, the standard deviation (STD) is set to zero, i.e., $\sigma = 0.0$; in the second case, the STD of the data is symmetric with respect to the origin, while increasing linearly from 0.1 at $x = 0.0$ to 0.5 at $x = 3.0$. Hence, in this case, the variance of noise is different in different regions. We use the default parameters $C = 100.0$ and the RBF kernel $\mathbf{K}(\mathbf{u}, \mathbf{v}) = \exp(-\|\mathbf{u} - \mathbf{v}\|^2)$.

We first compare the performance between the ε-SVR and the LSVR with respect to different ε's in order to investigate their difference clearly. Since the fuzzy parameters in the RLFSVR are not directly related to the ε parameter in the ε-SVR and the LSVR, it might be confusing to include the results of RLFSVR in Table 1. For this reason, we intentionally report the results of SVRs (including ε-SVR and LSVR) and RLFSVR in separate tables. We then further report the best results given by these three algorithms in Table 3.

Table 1 reports the results in terms of the mean square error (MSE) and the associated STD of the LSVR and the ε-SVR against different ε values in the abovementioned two cases. All the results are averaged over 100 random trails. For both cases, the MSE incurred by the ε-SVR generally increases as ε increases (the only exception occurs in the case of $\varepsilon = 0.2$ for the *Sinc* data with varying σ). Moreover, in the first case (with zero noise), when the ε value is too large, there are no support vectors in the ε-SVR and the MSE reaches a maximum. In contrast, the MSE incurred by the LSVR always maintains small even if the margin width ε increases to 2.0. In the second case (with a varying σ), the LSVR model also outputs smaller MSEs and smaller STDs than the ε-SVR for all ε's. This result shows how the proposed LSVR can improve the performance of the standard ε-SVR.

Table 2 reports the results of the RLFSVR with the effect of fuzzy parameter values. Here, we show the results of the RLFSVR on different values, ranging from 0 to 1, in the linear membership function. Since the RLFSVR requires the positive definite of the membership matrix, the first value we set is 1e−6 which is close to 0. We can see that for non-noise *Sinc* data, the RLFSVR achieves the best result when the parameter value is 1; while for the noise *Sinc* data, the best result is obtained when the parameter value is 0.4.

Table 3 reports the best results of the ε-SVR, the LSVR and the RLFSVR. In the first test case, i.e., $\sigma = 0$, the RLFSVR model cannot exactly fit the data because of the fuzzy membership. That is, the MSE given by the RLFSVR model is approximating to 0, but not exactly equal to 0. In contrast, in this case, both the ε-SVR and the LSVR can exactly generate the same curve as the true *Sinc* function, provided that a good ε is set. In the second test case (with varying σ), the LSVR model also demonstrates the best performance. It has the smallest MSEs. Moreover, the associated STDs are also significantly smaller than those of the other two algorithms.

Table 1
Experimental results of the LSVR model and the ε-SVR algorithm on the *Sinc* data with different ε values.

ε	Case I: $\sigma = 0.0$ (MSE)		Case II: varying σ (MSE ± STD)	
	LSVR	SVR	LSVR	SVR
0.0	0	0	0.0742 ± 0.0281	0.0909 ± 0.0293
0.2	0.0004	0.0160	0.0384 ± 0.0184	0.0852 ± 0.0265
0.4	0.0016	0.0722	0.0281 ± 0.0114	0.1008 ± 0.0206
0.6	0.0044	0.1695	0.0263 ± 0.0096	0.1471 ± 0.0264
0.8	0.0082	0.1748	0.0262 ± 0.0125	0.2204 ± 0.0330
1.0	0.0125	0.1748	0.0250 ± 0.0116	0.2619 ± 0.0266
2.0	0.0452	0.1748	0.0240 ± 0.0113	0.2670 ± 0.0330

Table 2
Experimental results of RLFSVR model on the *Sinc* data with different fuzzy parameter values.

Parameters	Case I: $\sigma = 0.0$ (MSE)	Case II: varying σ (MSE ± STD)
1e−6	9.2e−4	0.0415 ± 0.0172
0.2	3.7e−5	0.0398 ± 0.0174
0.4	8.2e−6	0.0397 ± 0.0170
0.6	3.3e−6	0.0400 ± 0.0169
0.8	1.8e−6	0.0402 ± 0.0170
1.0	1.3e−6	0.0406 ± 0.0170

Table 3
Comparison of the best results given by the ε-SVR, the LSVR, and the RLFSVR algorithm on the *Sinc* data.

Algorithms	Case I: non-noise (MSE)	Case II: noise (MSE ± STD)
SVR	0	0.0852 ± 0.0265
LSVR	0	0.0240 ± 0.0113
RLFSVR	1.3e−6	0.0397 ± 0.0170

The disadvantage of the LSVR is its training computational time. In the first case, the average training time is 0.5611, 0.0035, and 0.0050 s for the LSVR, the ε-SVR, and the RLFSVR, respectively. In the second case, the training time is 0.5342, 0.0029, and 0.0022 s for the LSVR, the ε-SVR, and the RLFSVR, respectively. All the report computational times are the average values when the best results are achieved for each algorithm. How to reduce the time complexity of the LSVR is an interesting and important open problem. We will leave this as future work.

To visually examine the performance of the three algorithms, we plot in Fig. 2 the approximation results with $\varepsilon = 0.2$ for the ε-SVR and the LSVR model[5] and the best curve given by the RLFSVR. Fig. 2(a) shows the curves in the first test case. In this case, compared with the ε-SVR, the LSVR model can adjust the margin automatically to fit the data with a smaller MSE. However, since it uses a fixed margin, the ε-SVR algorithm models the data poorly when an unsuitable margin is chosen. Moreover, the curve of the LSVR and the RLFSVR can almost perfectly (nearly) fit the *Sinc* function when there is no noise. Fig. 2(b) shows the fitting curves of three algorithms in the noisy case with varying σ. We can see that the resulting approximating function in the LSVR is smoother than that in the ε-SVR, also in the RLFSVR. More specifically, although $\varepsilon = 0.2$ is not the best parameter value for the margin of the LSVR, the associated MSE is still smaller than that of the best fitting of the RLFSVR.

Finally, we also evaluate in Fig. 3 how the parameter k affects the performance of the LSVR in the noisy case. We vary k from 1 to $N/2$, where $N = 50$ in this case. It can be clearly observed that the LSVR achieves the best MSE when $k = 1$. As k increases, the MSE is generally getting larger and larger, though the maximum MSE is achieved when k is set to around 14. Note that $k = N/2$ means that our proposed LSVR reduces to the standard SVR, which considers the data trend only globally. The curve shows that it is worthwhile considering the data trend locally as addressed by the LSVR.

6.2. Evaluations on real financial data

We evaluate our model on financial time series data which are highly volatile in nature and compare the performance of the

[5] We intentionally show the results of the ε-SVR and the LSVR with $\varepsilon = 0.2$ (not the best parameter for the LSVR) in order to see how our proposed LSVR can adapt the margin against the noise especially for the first test case.

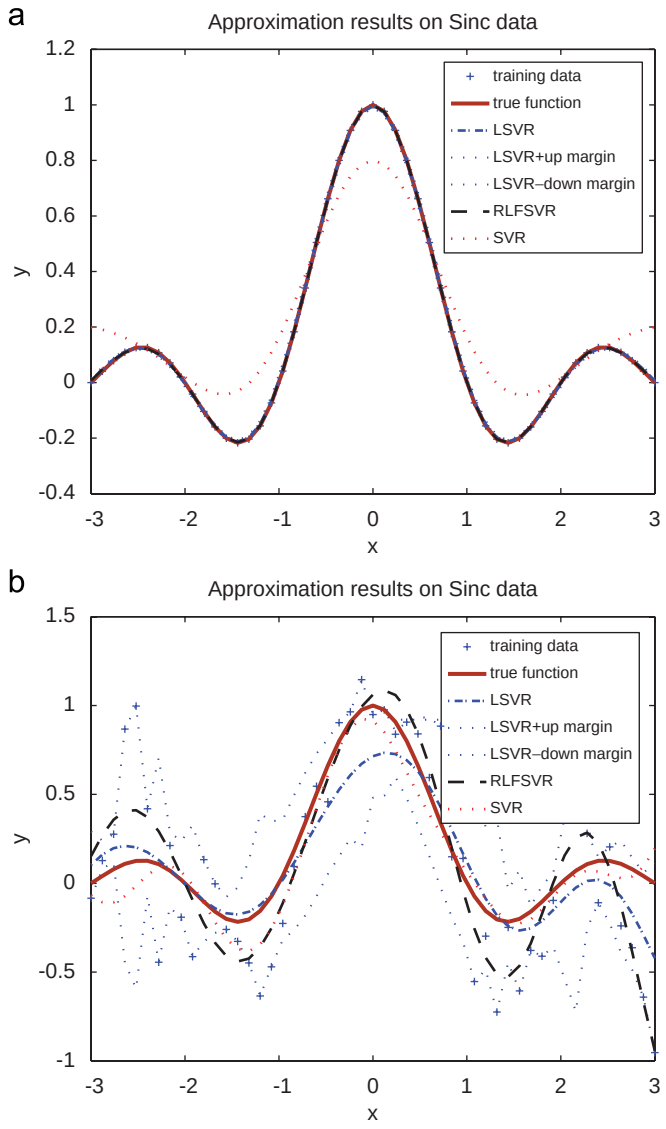


Fig. 2. Experimental results on synthetic *Sinc* data with $\varepsilon = 0.2$. (a) $\sigma = 0.0$, (b) varying σ.

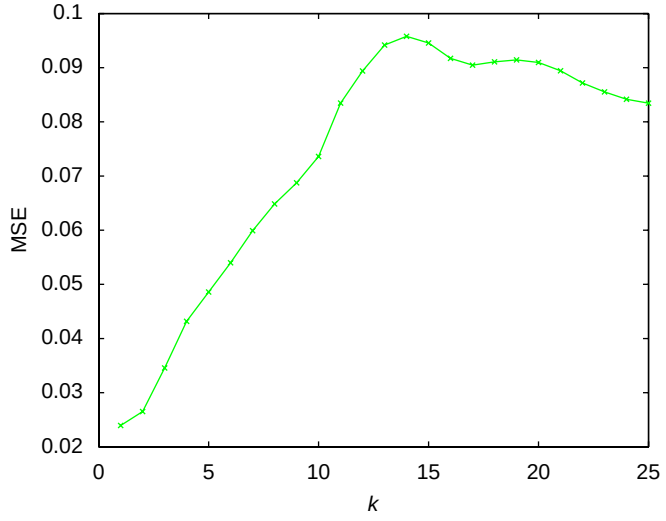


Fig. 3. Experimental results of the LSVR model on *Sinc* data with ε from 0.0 to 2.0 and k for 1 to $N/2$, where $N = 50$.

LSVR model against the ε-SVR and the RLFSVR. The experimental data used are drawn from three major indices from the period of January 2, 2004 to April 30, 2004: (1) the Dow Jones Industrial Average (DJIA), (2) the NASDAQ, and (3) the Standard & Poor 500 index (S&P500).

Data preprocessing: The daily closing prices (d_t) of the above three indices are converted to continuously compounded returns as $r_t = \log(d_{t+1}/d_t)$ first, and then they are normalized by $R_t = (r_t - Mean(r_t))/STD(r_t)$, where the means and STDs are computed for each individual index in the training period. The statistical properties of these data are reported in Table 4. One may note that the data in this period for three indices contain substantially different skewness. In the evaluation, the ratio of the number of the training return series to the number of test return series is set to 5:1. Therefore, we obtain the training and test periods and the number of data as reported in Table 5.

Model selection: The predicted system is modeled as $\hat{R}_t = f(\mathbf{x}_t)$, where $\mathbf{x}_t$ takes the previous p days' normalized returns as indicators, i.e., $\mathbf{x}_t = (R_{t-p}, \ldots, R_{t-1})$, where the parameter p, called *lag*, is picked from 1 to 6 as [30]. The trade-off parameter C and the parameter β of the RBF kernel ($\mathbf{K}(\mathbf{u}, \mathbf{v}) = \exp(-\beta\|\mathbf{u} - \mathbf{v}\|^2)$) are obtained by 10-fold cross validation on the following paired points: $[2^{-5}, 2^{-4}, \ldots, 2^{10}] \times [2^{-5}, 2^{-4}, \ldots, 2^{10}]$. Since when $\varepsilon \geqslant 2.0$, there are no support vectors in the SVR, we just restrict the ε values in the range of $0.0, 0.2, \ldots, 1.0$ to 2.0 for both the LSVR model and the ε-SVR algorithm. For the parameter k in the LSVR, we test it from 1 to 20. The fuzzy parameters used in the RLFSVR are also obtained by 10-fold cross validation.

Results analysis: Similar to the *Sinc* example, we first compare the performances of the proposed LSVR with the ε-SVR against different ε's in order to have a closer examination on the LSVR. Then, we report the best results obtained by the ε-SVR, the LSVR, and the RLFSVR.

We apply the obtained function f to test the performance by one-step ahead prediction. Tables 6, 7 and 8 report the corresponding MSEs in the DJIA index, NASDAQ index, and S&P500 index for the LSVR and the ε-SVR, respectively. In these tables, we list the results as well as the optimal parameters against the different ε's, ranging from 0.0 to 2.0; these parameters are the lag p and k for the LSVR model, and the lag for the ε-SVR algorithm. From these tables, the LSVR usually demonstrates better performance than the ε-SVR when an ε is specified. The LSVR is only slightly worse than the SVR in very few cases. These observations once again validate that considering data in a local fashion can indeed boost the prediction performance.

Table 4
Summary of statistics of normalized returns of DJIA, NASDAQ and S&P500 in the experiments.

Moments	DJIA		NASDAQ		S&P500	
	Train	Test	Train	Test	Train	Test
Mean	0.0000	−0.2850	−0.0000	−0.4819	0.0000	−0.3858
SD	1.0000	0.9957	1.0000	1.1312	1.0000	1.1298
Skew	−0.0678	0.1684	0.0928	0.3256	−0.1298	−0.0102
Kurt	2.5437	2.7706	2.6600	1.8631	2.5308	2.4124

Table 5
Periods and number of the financial data.

Item	Total	Training	Test
Period	January 2–April 30	January 2–April 12	April 13–30
#	82	69	13

Table 6
Experimental results (MSE) of the LSVR model and the ε-SVR algorithm on the DJIA index with different ε values.

ε	LSVR		k	SVR	
	Error	Lag		Error	Lag
0.0	0.9086	1	1	0.9199	1
0.2	0.9177	1	10	0.9199	1
0.4	0.8982	1	1	0.8695	1
0.6	0.8690	1	9	0.9096	1
0.8	0.8501	1	13	0.9353	1
1.0	0.8471	1	11	0.9347	1
2.0	**0.8388**	1	2	0.9018	1

Table 7
Experimental results (MSE) of the LSVR model and the ε-SVR algorithm on the NASDAQ index with different ε values.

ε	LSVR		k	SVR	
	Error	Lag		Error	Lag
0.0	1.3105	1	1	1.2683	1
0.2	1.2959	1	4	1.2998	1
0.4	1.3229	1	9	1.3100	1
0.6	1.2720	1	1	1.3109	1
0.8	1.2315	1	1	1.3344	1
1.0	1.2232	1	1	1.3448	1
2.0	**1.2115**	1	1	1.2758	1

Table 8
Experimental results (MSE) of the LSVR model and the ε-SVR algorithm on the S&P500 index with different ε values.

ε	LSVR		k	SVR	
	Error	Lag		Error	Lag
0.0	1.2184	6	1	1.2247	6
0.2	0.9534	2	1	1.2394	6
0.4	**0.9234**	2	8	1.2194	2
0.6	0.9544	2	12	1.1593	2
0.8	0.9657	2	6	1.0624	1
1.0	1.0232	2	5	1.0077	1
2.0	1.1589	2	1	1.1601	1

Table 9
Experimental results (MSE) of the RLFSVR algorithm on the financial indices with different lags.

Lag	1	2	3	4	5	6
DJIA	**0.8530**	0.9205	1.0527	1.3363	1.4766	1.3716
NASDAQ	**1.2768**	1.6566	1.5781	1.6781	1.7630	1.5434
S&P500	**1.0838**	1.1293	1.2255	1.4155	1.6014	1.2248

In Table 9, we report the best results achieved by the RLFSVR with different lags. To make it clear, we plot the best results given by the ε-SVR, the LSVR, and the RLFSVR in Fig. 4. Again, the proposed LSVR demonstrates the smallest MSEs in all the three indices. More specifically, the smallest MSEs of the LSVR are 0.8388 in DJIA, 1.2115 in NASDAQ, and 0.9234 in S&P500, while the smallest MSEs of the ε-SVR are 0.8695, 1.2758, and 1.0077, and the smallest MSEs of the RLFSVR are 0.8530, 1.2768 and 1.0838, respectively. A paired t-test [18] performed on the best results of three models show that the LSVR model outperforms the ε-SVR and the RLFSVR with the $\alpha = 5\%$ significance level for a one-tailed test, while the RLFSVR algorithm is slightly better than the ε-SVR algorithm in the DJIA index, but worse in the NASDAQ index and S&P500 index.

As mentioned before, the disadvantage of the proposed LSVR is its relatively long training time, since it needs to solve an SOCP problem which is conducted by using the general-purpose optimization package Sedumi in our implementation. In Table 10, we list the training time when the best MSE is achieved in the three indices. Although the LSVR costs less than 1 s in training, it spends much longer time than the other two algorithms. The RLFSVR needs to solve a linear equation system only, while

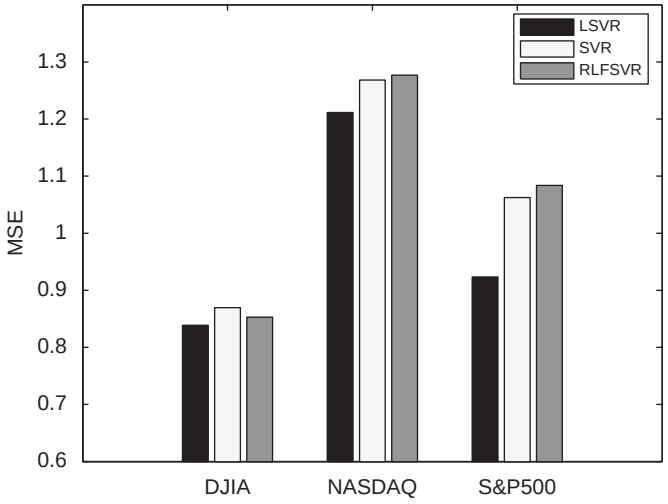


Fig. 4. MSE comparison among the LSVR, the ε-SVR, and the RLFSVR.

Table 10
Training time (second) for the LSVR, the ε-SVR, and the RLFSVR on the three indices.

Methods	LSVR	ε-SVR	RLFSVR
DJIA	0.8839	0.0021	0.0024
NASDAQ	0.8639	0.0009	0.0023
S&P500	0.9624	0.0095	0.0024

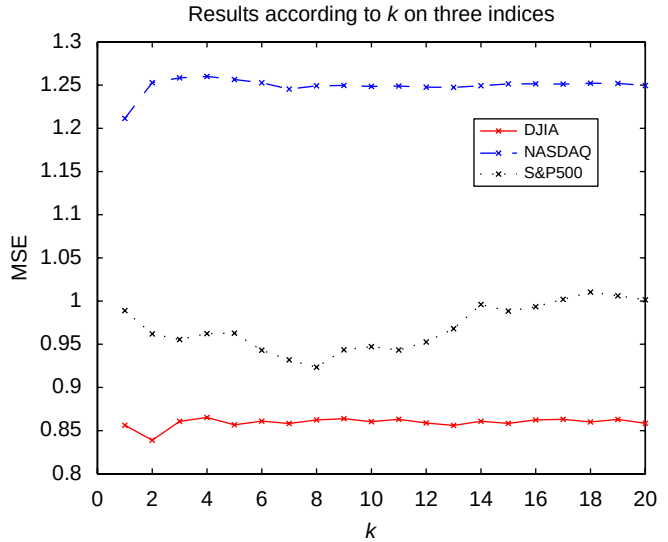


Fig. 5. Results of LSVR model with different k values on the best results of lag and ε in the three indices.

the ε-SVR can reduce the training time by exploiting the sequential minimal optimization technique [20]. It is very interesting to investigate whether some decomposable algorithms can be used in the LSVR in order to reduce its time complexity. We leave this topic as future work.

How parameters affect the performance of LSVR? In this section, we discuss how the parameters involved in the LSVR influence its prediction performance.

We first examine how the parameter k affects the prediction performance. From Tables 6–8, we notice that the best values for k's are usually relatively small. In more detail, in DJIA, the largest k among all the best MSEs is 13 when $\varepsilon = 0.8$; in NASDAQ, the largest k among the best MSEs is 9 when $\varepsilon = 0.4$, while most of the k's associated with other ε's are optimal at 1; in S&P500, the largest optimal k is 12 with $\varepsilon = 0.6$. To clearly see the influence of k, we plot in Fig. 5 the MSE against different k's when the best results are attained (i.e., other parameters except k are fixed to the values associated with the best results given by LSVR in the three indices). It is evident that a smaller k generally achieves better performance in NASDAQ, while the best k is 2 and 8 for DJIA and S&P500, respectively. When k is set to some large value (say a value larger than 14), the MSEs in the three financial data sets are usually relatively large. Note that the LSVR tends to use the global variance and will perform similar to the ε-SVR as k increases. This shows that appropriately considering the data trend in the local way as emphasized in the LSVR can usually outperform the ε-SVR model with the global setting.

We next analyze the influence of the lag parameter on the prediction performance. From Tables 6–9, one can observe that the best results for the LSVR model in the DJIA and NASDAQ indices and the best results for the ε-SVR and RLFSVR algorithms in the three indices are obtained when the lag is selected as 1, while the best result of the LSVR model on the S&P500 is obtained when the lag is 2. Although the best lag might be possibly data-dependent, for these three data sets, it seems that we have more chances to get better results if the lag is equal to 1.

Finally, we investigate how the cost parameter C influences the performance of the proposed algorithm. We plot the MSEs of the ε-SVR, the RLFSVR, and the LSVR against C in Fig. 6, when the best results for the three algorithms are achieved (i.e., other

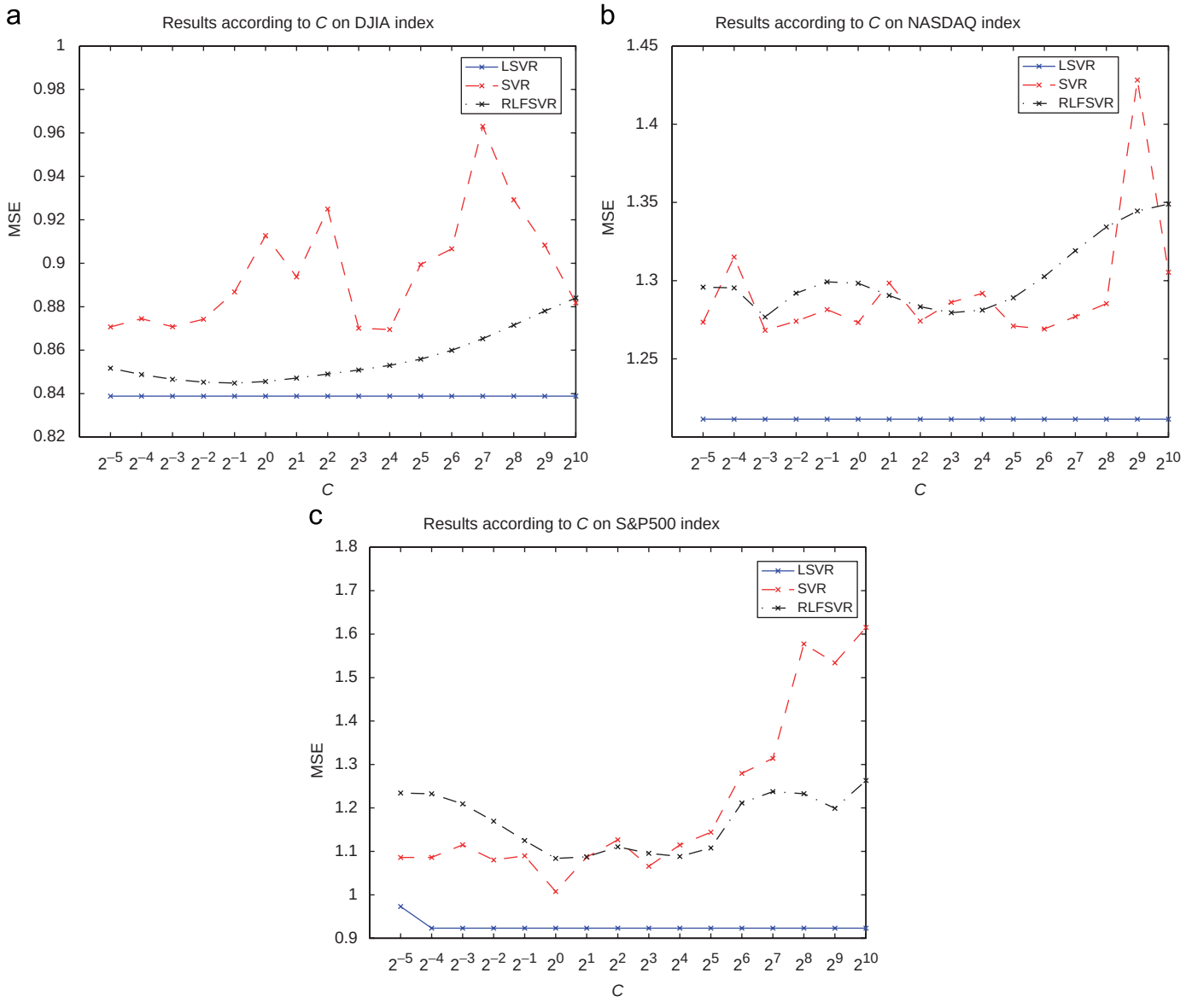


Fig. 6. The influence of C on the performance. (a) DJIA, (b) NASDAQ, (c) S&P500.

parameters except C are, respectively, fixed to the values associated with the best results given by the three algorithms in the three indices).

Two observations deserve our attentions. First, for the ε-SVR and the RLFSVR, the best C is generally data-dependent. Moreover, the performance of the ε-SVR is very sensitive to the choice of C in these high-volatile data, while the RLFSVR generates a relatively flatter curve than the ε-SVR. This shows that the RLFSVR might also be able to mitigate the influence caused by the high volatility.

Second, it is surprising that the performance of the LSVR almost remains unchanged against different C's. If one looks back into the optimization problem (12)–(15) in the LSVR, the margin ε is weighted by the upper bound t_i around each data point $\mathbf{x}_i$. The non-fixed margin would "absorb" the influence caused by the noise points. Namely, when the points are located in a noisy area, the margin will be flexibly expanded so as to reduce the impact caused by the noise points. In comparison, in the ε-SVR, the impact caused by the noisy points are only mitigated by the slack variable ξ_i. In other words, one can regard that each ξ_i, associated with ε, is also able to adapt the margin around each $\mathbf{x}_i$ slightly. In this sense, the non-fixed adaptive margin functions similar to the slack variable ξ in the ε-SVR. In fact, as we find from the implementation of the LSVR, ξ just fine-tunes the margin, while the approximating function is mainly decided by the adaptive margin introduced by εt_i. It should be noticed that this phenomenon actually makes our model more appealing: we can even remove the term $C\sum_i \xi_i$ from (12). Meanwhile, the varying margin in our LSVR model benefits for removing the effect of outliers.

7. Conclusion and future work

In this paper, we have proposed the localized support vector regression (LSVR) model in order to improve the performance of the standard support vector regression (SVR) model for time series prediction. In contrast to the standard SVR model, our novel model offers a systematic and automatic scheme to adapt the margin locally and flexibly. Therefore, it can tolerate noise adaptively. We have demonstrated that this promising model not only captures the local information of data in approximating functions, but also incorporates the standard SVR as a special case. Moreover, kernelization can also be applied to this novel model. Therefore it can generate non-linear approximating functions and can be applied to general regression tasks. The experiments conducted on synthetic *Sinc* data and three series from real financial time series indices show that our model outperforms the standard SVR and the RLFSVR in modeling the data.

Four main issues need to be investigated in future work. First, we have relaxed the difficult non-convex optimization of LSVR to the tractable SOCP problem. Although this relaxation is acceptable both theoretically and empirically, two interesting questions arise. First, can some new technique be used to solve the optimization of LSVR directly or to establish how well the relaxed LSVR performs relative to the unrelaxed one? Second, although it is polynomial, the time complexity of LSVR may cause difficulties in very large data regression tasks. In particular, LSVRs time complexity of $O((N+1)n^3)$ is larger than that of $O(n^3+Nn^2)$ involved in solving the quadratic programs of the standard SVR. To solve this problem, one possible direction is to exploit novel techniques to decompose the Gram matrix and to develop some specialized optimization procedures for LSVR.

Second, we have shown how the associated parameters of the proposed LSVR affect its prediction performance empirically. The results show that, to generate better performance, k needs to be chosen carefully, and the performance of LSVR is actually insensitive to the trade-off parameter C. Nevertheless, how to automatically set the best parameters may still be data-dependent and it remains to be a highly challenging yet interesting topic.

Third, although it is specially designed for highly volatile data, i.e., the time series financial data, the proposed LSVR can still be applicable for regular function approximation tasks. We need to perform extensive investigations on our method in the regular function approximation tasks. Specifically, we would like to examine whether the possible accuracy improvement is worthwhile especially when we take into account the long training time incurred by LSVR. We leave this as future work.

Finally, as the regression counterpart of the Maxi-Min Margin Machine (M^4), LSVR may contain further connections with those regression models that are directly extended from special cases of M^4. In particular, it is worth investigating whether LSVR can be linked with the minimax probability machine regression [26]. Exploring these connections is therefore one of our intended research topics in the future.

Acknowledgments

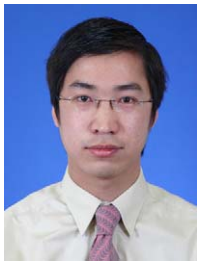
The work described in this paper was fully supported by two grants from the Research Grants Council of the Hong Kong SAR, China (Project nos. CUHK 4125/07E and CUHK 4150/07E).

References

[1] D.P. Bertsekas, Nonlinear Programming, second ed., Athena Scientific, Belmont, MA, 1999.
[2] D. Bertsimas, J.N. Tsitsiklis, Induction of Linear Optimization, Athena Scientific, Belmont, MA, 1997.
[3] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, Cambridge, 2004.
[4] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory IT-13 (1) (1967) 21–27.
[5] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, Cambridge, 2000.
[6] R. Fernandez, Predicting time series with a local support vector regression machine, in: Proceedings of the ECCAI Advanced Course on Artificial Intelligence (ACAI-1999), Chania, Greece, 1999.
[7] K. Fukunaga, Introduction to Statistical Pattern Recognition, second ed., Academic Press, San Diego, 1990.
[8] K. Huang, H. Yang, I. King, M.R. Lyu, Learning classifiers from imbalanced data based on biased minimax probability machine, in: Proceedings of 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR-2004), vol. 2, 2004, pp. 558–563.
[9] K. Huang, H. Yang, I. King, M.R. Lyu, Learning large margin classifiers locally and globally, in: R. Greiner, D. Schuurmans (Eds.), The 21st International Conference on Machine Learning (ICML-2004), 2004, pp. 401–408.
[10] K. Huang, H. Yang, I. King, M.R. Lyu, Imbalanced learning with biased minimax probability machine, IEEE Trans. System Man Cybern. Part B 36 (2006) 913–923.
[11] K. Huang, H. Yang, I. King, M.R. Lyu, Local support vector regression for time series prediction, in: Proceedings of International Joint Conference on Neural Network (IJCNN-2006), Vancouver, BC, Canada, 2006.
[12] K. Huang, H. Yang, I. King, M.R. Lyu, Maximizing sensitivity in medical diagnosis using biased minimax probability machine, IEEE Trans. Biomed. Eng. 53 (2006) 821–831.
[13] K. Huang, H. Yang, I. King, M.R. Lyu, Maxi-Min Margin Machine: learning large margin classifiers locally and globally, IEEE Trans. Neural Networks 19 (2) (2008) 260–272.
[14] K. Huang, H. Yang, I. King, M.R. Lyu, L. Chan, The minimum error minimax probability machine, J. Mach. Learn. Res. 5 (2004) 1253–1286.
[15] R. Khemchandani, Jayadeva, S. Chandra, Regularized least squares fuzzy support vector regression for financial time series forecasting, Expert Systems Appl. 36 (1) (2007) 132–138.
[16] G.R.G. Lanckriet, L.E. Ghaoui, C. Bhattacharyya, M.I. Jordan, A robust minimax approach to classification, J. Mach. Learn. Res. 3 (2002) 555–582.
[17] M. Lobo, L. Vandenberghe, S. Boyd, H. Lebret, Applications of second order cone programming, Linear Algebra Appl. 284 (1998) 193–228.
[18] D.C. Montgomery, G.C. Runger, Applied Statistics and Probability for Engineers, second ed., Wiley, New York, 1999.
[19] Y. Nesterov, A. Nemirovsky, Interior Point Polynomial Methods in Convex Programming: Theory and Applications, Studies in Applied Mathematics, Philadelphia, 1994.

[20] J. Platt, Sequential minimal optimization: a fast algorithm for training support vector machines, Technical Report MSR-TR-98-14, 1998.
[21] R.J Povinelli, X. Feng, A new temporal pattern identification method for characterization and prediction, IEEE Trans. Knowledge Data Eng. 15 (2) (2003) 339–352.
[22] A. Pruessner, Conic programming in GAMS, in: Optimization Software—The State of the Art, INFORMS Atlanta, 2003 ⟨http://www.gamsworld.org/cone/links.htm⟩.
[23] B. Schölkopf, P. Bartlett, A. Smola, R. Williamson, Shrinking the tube: a new support vector regression algorithm, in: M.S. Kearns, S.A. Solla, D.A. Cohn (Eds.), Advances in Neural Information Processing Systems, vol. 11, Cambridge, MA, 1999, pp. 330–336.
[24] B. Scholkopf, A. Smola, Learning with Kernels, MIT Press, Cambridge, MA, 2002.
[25] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Technical Report NeuroCOLT NC-TR-98-030, Royal Holloway College, University of London, UK, 1998. URL: ⟨http://citeseer.ist.psu.edu/smola98tutorial.html⟩.
[26] T.R. Strohmann, G.Z. Grudic, Minimax probability machine regression, in: Advances in Neural Information Processing Systems (NIPS) 15, 2003.
[27] J.F Sturm, Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones, Optim. Methods Software 11 (1999) 625–653.
[28] J.F. Sturm, Central region method, in: J.B.G. Frenk, C. Roos, T. Terlaky, S. Zhang (Eds.), High Performance Optimization, Kluwer Academic Publishers, Dordrecht, 2000, pp. 157–194.
[29] F.E.H. Tay, L. Cao, Epsilon descending support vector machines for financial time series forecasting, Neural Process. Lett. 15 (2) (2002) 179–195.
[30] T. Van Gestel, J. Suykens, D. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, J. Vandewalle, Financial time series prediction using least squares support vector machines within the evidence framework, IEEE Trans. Neural Networks 12 (4) (2001) 809–821 (special issue on Neural Networks in Financial Engineering).
[31] V.N. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[32] V.N. Vapnik, The Nature of Statistical Learning Theory, second ed., Springer, New York, 1999.
[33] H. Yang, L. Chan, I. King, Support vector machine regression for volatile stock market prediction, in: IDEAL 2002, 2002, pp. 391–396.
[34] H. Yang, I. King, L. Chan, K. Huang, Financial time series prediction using non-fixed and asymmetrical margin setting with momentum in support vector regression, in: J.C. Rajapakse, L. Wang (Eds.), Neural Information Processing: Research and Development, Studies in Fuzziness and Soft Computing, vol. 152, Springer, Berlin, 2004, pp. 334–350.

Haiqin Yang received his B.S. degree in Computer Science and Technology from Nanjing University, Nanjing, China, in 2001, the M.Phil. degree in Computer Science and Engineering from the Chinese University of Hong Kong, in 2003. He is currently a Ph.D. student in the Chinese University of Hong Kong. His research interests include machine learning, data mining, pattern recognition, image processing and computer vision.



Kaizhu Huang received the B.E. (1997) in Automation from Xi'an Jiaotong University, the M.E. (2000) in Pattern Recognition and Intelligent Systems from Institute of Automation, the Chinese Academy of Sciences, and the Ph.D. (2004) in Computer Science and Engineering from the Chinese University of Hong Kong. After that, he worked as a researcher in Information Technology Laboratory, Fujitsu Research Center Co. Ltd. He is currently a research fellow in the Chinese University of Hong Kong. His research interests include machine learning, pattern recognition, image processing, and information retrieval.

Irwin King received the B.Sc. degree in Engineering and Applied Science from California Institute of Technology, Pasadena, in 1984. He received his M.Sc. and Ph.D. degree in Computer Science from the University of Southern California, Los Angeles, in 1988 and 1993, respectively. He joined the Chinese University of Hong Kong in 1993.

His research interests include web intelligence and social computing, machine learning, and multimedia processing. In these research areas, Dr. King has published over 140 refereed journal (JMLR, ACM TOIS, IEEE TNN, IEEE BME, PR, IEEE SMC, JAMC, JASIST, IJPRAI, NN, etc.) and conference manuscripts (NIPS, CIKM, SIGIR, IJCNN, ICONIP, ICDAR,WWW, etc.). In addition, he has contributed over 20 book chapters and edited volumes. Moreover, Dr. King has over 30 research and applied grants. One notable system Dr. King has developed is the CUPIDE (Chinese University Plagiarism IDentification Engine) system, which detects similar sentences and performs readability analysis of text-based documents.

Dr. King is an Associate Editor of the IEEE Transactions on Neural Networks (TNN). He is a member of the Editorial Board of the Open Information Systems Journal, Journal of Nonlinear Analysis and Applied Mathematics, and Neural Information Processing-Letters and Reviews Journal (NIP-LR). He has also served as Special Issue Guest Editor for Neurocomputing, International Journal of Intelligent Computing and Cybernetics (IJICC), Journal of Intelligent Information Systems (JIIS), and International Journal of Computational Intelligent Research (IJCIR). He is a member of ACM, IEEE Computer Society, International Neural Network Society (INNS), and Asian Pacific Neural Network Assembly (APNNA). Currently, he is serving the Neural Network Technical Committee (NNTC) and the Data Mining Technical Committee under the IEEE Computational Intelligence Society (formerly the IEEE Neural Network Society). He is also a Vice-President and Governing Board Member of the Asian Pacific Neural Network Assembly (APNNA).

He is serving or has served as program and/or organizing member in numerous top international conferences and workshops, e.g., WWW, ACM MM, CIKM, ICME, ICASSP, IJCNN, ICONIP, ICPR, etc. He has also served as reviewer for international conferences as well as journals, e.g., Information Fusion, IEEE TCAS, SIGMOD, IEEE Transactions on Neural Networks, IEEE Pattern Analysis and Machine Intelligence, IEEE Transactions on Multimedia, IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on System, Man, and Cybernetics, Machine Vision and Applications, International Journal of Computer Vision, Real-Time Imaging, SPIE Journal of Electronic Imaging, International Journal of Pattern Recognition and Artificial Intelligence, etc.

Dr. King has received several exemplary teaching and service awards from the Department as well as from the Faculty of Engineering. He serves as a member of the Engineering Panel with the Research Grants Council (RGC), Hong Kong SAR Government. He is also the Director of the International Programmes in the Engineering Faculty. In addition, he is a member of the Faculty Curriculum Committee and serves as the Chair of the Curriculum Committee for the department. Dr. King is also actively involved in education of students outside of the classroom. For example, he has led several ACM Programming Contest Teams to the ACM ICPC World Finals since 2000. Moreover, he actively promotes the use of technologies in education.

Michael Rung-Tsong Lyu received his B.S. Degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan, China, in 1981; his M.S. Degree in Computer Engineering from University of California, Santa Barbara, in 1985; and his Ph.D. Degree in Computer Science from University of California, Los Angeles, in 1988. He was with the Jet Propulsion Laboratory as a Technical Staff Member from 1988 to 1990. From 1990 to 1992, he was with the Department of Electrical and Computer Engineering, The University of Iowa, Iowa City, as an Assistant Professor. From 1992 to 1995, he was a Member of the Technical Staff in the applied research area of Bell Communications Research (Bellcore), Morristown, New Jersey. From 1995 to 1997, he was a Research Member of the Technical Staff at Bell Laboratories, Murray Hill, New Jersey. In 1998 he joined the Chinese University of Hong Kong, where he is now a Professor in the Department of Computer Science and Engineering. He is also Founder and Director of the Video over InternEt and Wireless (VIEW) Technologies Laboratory.

Dr. Lyu's research interests include software reliability engineering, distributed systems, fault-tolerant computing, mobile and sensor networks, Web technologies, multimedia information processing and retrieval, and machine learning. He has published 300 refereed journal and conference papers in these areas. He was the editor of two book volumes: Software Fault Tolerance (New York: Wiley, 1995) and The Handbook of Software Reliability Engineering (Piscataway, NJ: IEEE and New York: McGraw-Hill, 1996). He initiated the First International Symposium on Software Reliability Engineering (ISSRE) in 1990. He was the Program Chair for ISSRE'96 and General Chair for ISSRE'2001. He was also PRDC'99 Program Co-Chair, WWW10 Program Co-Chair, SRDS'2005 Program Co-Chair, PRDC'2005 General Co-Chair, and ICEBE'2007 Program Co-Chair. He will be the General Chair for DSN'2011 in Hong Kong. He was on the Editorial Board of IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Reliability, Journal of Information Science and Engineering, and Wiley Software Testing, Verification and Reliability Journal. Dr. Lyu is an IEEE Fellow, an AAAS Fellow, and a Croucher Senior Research Fellow.