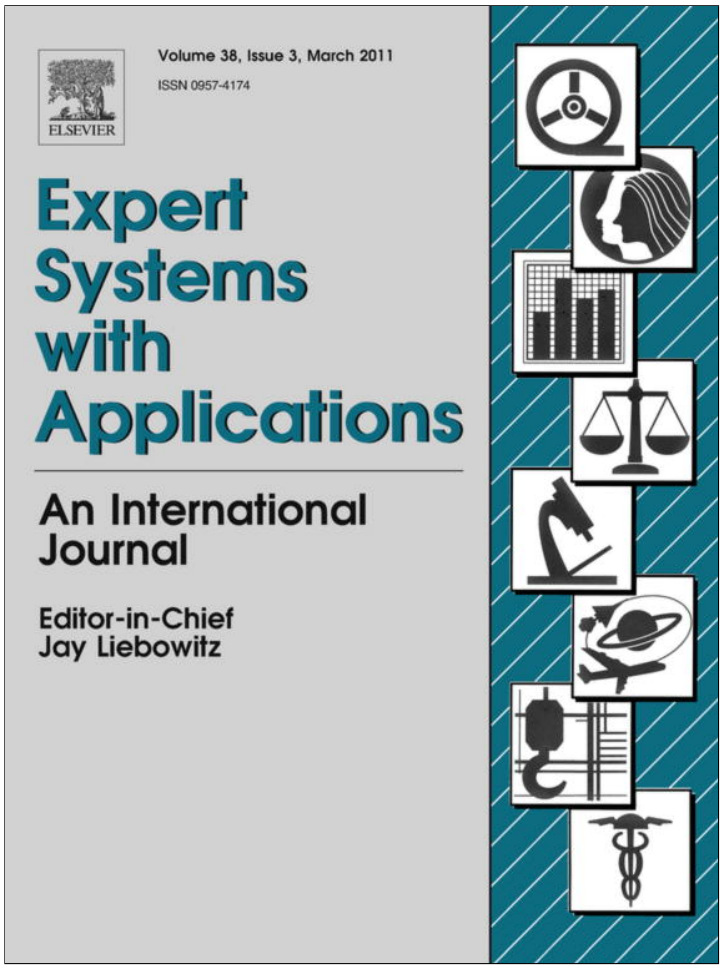

Expert Systems with Applications 38 (2011) 1611–1618

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Multikernel semiparametric linear programming support vector regression

Yong-Ping Zhao [a,*], Jian-Guo Sun [b]

[a] ZNDY of Ministerial Key Laboratory, Nanjing University of Science & Technology, Nanjing 210094, China
[b] Department of Energy and Power Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

ARTICLE INFO

Keywords:
Linear programming support vector regression
Semiparametric technique
Multikernel trick
Classification

ABSTRACT

In many real life realms, many unknown systems own different data trends in different regions, i.e., some parts are steep variations while other parts are smooth variations. If we utilize the conventional kernel learning algorithm, viz. the single kernel linear programming support vector regression, to identify these systems, the identification results are usually not very good. Hence, we exploit the nonlinear mappings induced from the kernel functions as the admissible functions to construct a novel multikernel semiparametric predictor, called as MSLP-SVR, to improve the regression effectiveness. The experimental results on the synthetic and the real-world data sets corroborate the efficacy and validity of our proposed MSLP-SVR. Meantime, compared with other multikernel linear programming support vector algorithm, ours also takes advantages. In addition, although the MSLP-SVR is proposed in the regression domain, it can also be extended to classification problems.

1. Motivation

During the past decade, known as an excellent kernel modeling technique, support vector machine (SVM) (Burges, 1998; Cristianini & Shawe-Taylor, 2000; Schölkopf & Smola, 2002; Vapnik, 1995) has been becoming of popularity in the realm of machine learning and has been referred to as the state-of-the-art strategy for classification and regression applications like image segmentation (Chen & Wang, 2005), time series prediction (Lau & Wu, 2008), and face recognition (Guo, Li, & Chan, 2001). Its basis is the so-called structural risk minimization principle which consists of two parts, one of which is empirical risk also owned in the cost function of the artificial neural networks (ANNs) such as the well-known back propagation (BP) networks, and another is the confidence interval which is capable of controlling the model complexity through the Vapnik–Chervonenkis (VC) dimension. Initially, the SVM is constructed to solve the binary classification problem using a decision hyperplane with as soon as large margin, which is expected to obtain good generalization performance. As a consequence, a small subset of the training samples, termed as support vectors, is chosen to support the optimal hyperplane. From the inception of the SVM, it is broadly extended to various fields like density estimation, regression and linear operator equation. When it is exploited to cope with the regression estimation and function approximation, a variable, viz. support vector regression (SVR), is born. Like the hinge loss function in SVM, an innovative loss function, namely ε-insensitive loss function, is proposed by Vapnik (1995) to construct a tolerance band for the purpose of realizing the sparse solution for the SVR. Although it is very effective for the function estimation, in particular for the problems involving the high-dimensional input space, there exist two limitations more or less. One is the computational complexity, that is to say, the computational burden of the conventional quadratic programming support vector regression (QP-SVR) scales cubically with the number of the training samples. Although the existing algorithms such as Osuna, Freund, and Girosi (1997), SMO (Platt, 1998; Shevade, Keerthi, Bhattacharyya, & Murthy, 2000), SVM^{light} (Joachims, 1999), SVMTorch (Collobert & Bengio, 2001), LIBSVM (Chang & Lin, 2001), and so forth alleviate the computational cost, this problem, that is, $R \cdot O(Nq + q)$, where R is the iterative number and q is the scale of working set, survives to a certain extent. Besides, the solution of QP-SVR is not sparse enough, even hardly controllable. For example, in system identification, the QP-SVR is not always able to construct sparse models (Drezet & Harrison, 1998). As an extreme, if error insensitivity is not contained, it will select the ensemble training set as support vectors (Drezet & Harrison, 2001). Lee and Billings (2002) compared the conventional SVM with uniformly regularized orthogonal least squares algorithm on time series prediction problems, and showed that both algorithms own similar excellent generalization performance but the resultant model from SVM is not sparse enough. Actually, the conventional QP-SVR can only give an upper bound on the number of necessary and sufficient support vectors because of the linear dependencies of support vectors in the high-dimensional feature space. Thus, some efforts (Drezet & Harrison, 2001; Li, Jiao, & Hao, 2007; Nguyen & Ho, 2006) have been made to let it gain

* Corresponding author.
E-mail addresses: y.p.zhao@nuaa.edu.cn, zhaoyongping_007@163.com (Y.-P. Zhao), jgspe@nuaa.edu.cn (J.-G. Sun).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.07.082

sparser representation due to the Occam's razor, that is, "plurality should not be posited without necessity".

As a modification, the linear programming support vector regression (LP-SVR) (Weston et al., 1999; Smola, Schölkopf, & Rätsch, 1999) is developed to change the support vector problem from a quadratic programming to a linear programming problem. And from Kecman (2001) and Hadzic and Kecman (2000), we know that the linear programming support vector regression holds the superiorities of model sparse representation and computational efficiency to QP-SVR. The motivation of linear programming support vector machines is to utilize the linear kernel combination as an ansatz for the solution, and to employ a different regularizer, viz. the ℓ_1 norm of the coefficient vector, in the cost function. That is to say, the LP-SVR is treated as a linear one in the kernel space as a surrogate of the case of QP-SVR in the feature space. As we know, the choice of kernel function plays a paramount role in the modeling process of the kernel methods. When we face with the systems owning different data trends in different regions, the single kernel to construct model commonly does not achieve the satisfying result, i.e., the model does not globally fit the system. Furthermore, recently multikernel learning algorithms (Bi, Zhang, & Bennett, 2004; Lanckriet, Cristianini, Bartlett, Ghaoui, & Joran, 2004; Ong, Smola, & Williamson, 2005) have demonstrated the necessity to consider multiple kernels or the combination of kernels rather than a single fixed kernel. Hence, it is necessary to realize the multikernel learning for LP-SVR. Both the conventional QP-SVR and LP-SVR are implemented using the nonparametric technique. However, when one happens to have additional knowledge about the learning system, it is unwise not to take advantage of it. In this situation, the semiparametric technique (Smola, Frie, & Schölkopf, 1998) is a good selection to utilize the prior knowledge. If we are able to combine the semiparametric technique with multikernel trick, the identification results of the systems owning different data trends in different regions will be improved more or less (Nguyen & Tay, 2008). When data are provided us to identify the system, usually there is no additional knowledge to utilize, so some strategies may be used to mine the additional knowledge for the data-driven modeling. In general, the data selected by the support vector learning algorithms as support vectors are commonly significant for system identification. Here, we treat the pre-selected data as the additional knowledge to utilize the semiparametric technique to model the identification system, and after combining with the multikernel trick, a novel linear programming support vector learning algorithm, called as multikernel semiparametric linear programming support vector regression and MSLP-SVR for short, is proposed in this letter. The experimental results on the synthetic and real-world data sets corroborate the efficacy and validity of the presented MSLP-SVR.

The remainder of this letter is organized as follows. In the following section, the conventional linear programming support vector regression is dwelled on. Subsequently, after combining the semiparametric technique with the multikernel trick, we give the multikernel semiparametric linear programming support vector regression. To show the effectiveness of the MSLP-SVR, in Section 4 we do experiments on the synthetic and real-world data sets, and we compare MSLP-SVR with the other multikernel linear programming algorithm. Finally, conclusions follow.

2. Linear programming support vector regression

In this section, we will firstly introduce the conventional QP-SVR, and then LP-SVR is derived, because both algorithms have some similarities like the ε-insensitive loss function and kernel functions in the feature space. Given the training data set $\{(\boldsymbol{x}_i, d_i)\}_{i=1}^N$ of the size N, where $\boldsymbol{x}_i \in \Re^n$ is the input and d_i is the corresponding output, we construct a linear predictor $f(\cdot)$ in the high-dimensional (even infinite-dimensional) feature space with ε-insensitive loss function in the following:

$$\begin{aligned} \min_{w,b} \quad & \left\{ \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C'\sum_{i=1}^{N}\left(\xi_i^* + \xi_i\right) \right\} \\ \text{s.t.} \quad & d_i - \left(\boldsymbol{w}^T \cdot \varphi(\boldsymbol{x}_i) + b\right) \leqslant \varepsilon + \xi_i^* \\ & \boldsymbol{w}^T \cdot \varphi(\boldsymbol{x}_i) + b - d_i \leqslant \varepsilon + \xi_i \\ & \xi_i,\ \xi_i^* \geqslant 0, \quad i = 1, \ldots, N \end{aligned} \tag{1}$$

where ξ_i and ξ_i^* are the slack variables, $\varphi(\cdot)$ is a nonlinear mapping which can transform the input data x_is in the input space into $\varphi(\boldsymbol{x}_i)$s in the feature space, $C' > 0$ is the regularization parameter which can control the tradeoff between the flatness of f and closeness to the training data. By defining the following ε-insensitive loss function,

$$H_\varepsilon(d_i - f(\boldsymbol{x}_i)) = \max\{0, |d_i - f(\boldsymbol{x}_i)| - \varepsilon\} \tag{2}$$

and letting $\lambda = (2C')^{-1}$, the optimization problem (1) can be reformulated as the following regularization problem:

$$\min_f \left\{ \lambda\|\boldsymbol{w}\|^2 + \sum_{i=1}^{N} H_\varepsilon(d_i - f(\boldsymbol{x}_i)) \right\} \tag{3}$$

Eq. (3) is completely equivalent to (1). From the representer theorem (Kimeldorf & Wahba, 1970), the optimal function of (3) can be represented as a linear combination of the kernel functions centering the training examples,

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \beta_i k(\boldsymbol{x}, \boldsymbol{x}_i) \tag{4}$$

where $k(\boldsymbol{x}_i, \boldsymbol{x}) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x})$ is the kernel function. $k(\cdot,\cdot)$ can be chosen from Gaussian, polynomial, or MLP, etc. Commonly, the Gaussian is the choice. Through changing the norm used in (3), i.e., ℓ_2 norm being replaced by the ℓ_1 norm (3) can be altered as

$$\min_f \left\{ \lambda\|\boldsymbol{\beta}\|_1 + \sum_{i=1}^{N} H_\varepsilon(d_i - f(\boldsymbol{x}_i)) \right\} \tag{5}$$

where $\boldsymbol{\beta} = [\beta_1, \beta_2, \ldots, \beta_N]^T$. Hence, the corresponding form of Eq. (1) is stated in the following:

$$\begin{aligned} \min_{\beta,b} \quad & \left\{ \frac{1}{2}\|\boldsymbol{\beta}\|_1 + C'\sum_{i=1}^{N}\left(\xi_i^* + \xi_i\right) \right\} \\ \text{s.t.} \quad & d_i - \left(\sum_{j=1}^{N}\beta_j k(\boldsymbol{x}_j, \boldsymbol{x}_i) + b\right) \leqslant \varepsilon + \xi_i^* \\ & \sum_{j=1}^{N}\beta_j k(\boldsymbol{x}_j, \boldsymbol{x}_i) + b - d_i \leqslant \varepsilon + \xi_i \\ & \xi_i,\ \xi_i^* \geqslant 0, \quad i = 1, \ldots, N \end{aligned} \tag{6}$$

In order to transform the above optimization problem into a linear programming problem, here we use a variable replacement trick to decompose β_j and $|\beta_j|$ as

$$\beta_j = \alpha_j^* - \alpha_j, \quad |\beta_j| = \alpha_j^* + \alpha_j \tag{7}$$

where $\alpha_j^*,\ \alpha_j \geqslant 0$. It is worth noting that the decompositions of (7) are unique and $\alpha_j^* \cdot \alpha_j = 0 \quad (j = 1, \ldots, N)$ are guaranteed implicitly. If $\alpha_j^* > 0,\ \alpha_j > 0$ and $\alpha_j^* > \alpha_j$ (similar for $\alpha_j^* < \alpha_j$) arise, then we will obtain a better solution $\alpha_j'^* = \alpha_j^* - \alpha_j,\ \alpha_j' = 0\ (i = 1, 2, \ldots, N)$ which satisfies the constraints and makes the cost function of (6) much smaller. Thus, the implicit constraints $\alpha_i^* \cdot \alpha_i = 0$ are guaranteed in the final solution of (6), that is to say, α_i^* or α_i must equal zero. If both α_i^* and α_i are equal to zeros, the corresponding sample $\boldsymbol{x}_i$ is non-support vector. After plugging (7) into (6), we obtain

$$\begin{aligned}
\min_{\boldsymbol{\alpha}^*,\boldsymbol{\alpha},b} \quad & \left\{\sum_{i=1}^{N}(\alpha_i^*+\alpha_i)+C\sum_{i=1}^{N}(\xi_i^*+\xi_i)\right\} \\
\text{s.t.} \quad & d_i-\left(\sum_{j=1}^{N}(\alpha_j^*-\alpha_j)k(\boldsymbol{x}_j,\boldsymbol{x}_i)+b\right)\leqslant\varepsilon+\xi_i^* \\
& \sum_{j=1}^{N}(\alpha_j^*-\alpha_j)k(\boldsymbol{x}_j,\mathbf{x}_i)+b-d_i\leqslant\varepsilon+\xi_i \\
& \xi_i,\ \xi_i^*\geqslant 0,\quad \alpha_i^*,\ \alpha_i\geqslant 0,\quad i=1,\ldots,N
\end{aligned} \tag{8}$$

where $\boldsymbol{\alpha}=[\alpha_1,\alpha_2,\ldots,\alpha_N]^T$, $\boldsymbol{\alpha}^*=[\alpha_1^*,\alpha_2^*,\ldots,\alpha_N^*]^T$, $C=2C'$. To mitigate the computational complexity and memory space of (8), we modify (8) subtly and rewrite it as

$$\begin{aligned}
\min_{\boldsymbol{\alpha}^*,\boldsymbol{\alpha},b} \quad & \left\{\sum_{i=1}^{N}(\alpha_i^*+\alpha_i)+C\sum_{i=1}^{N}\tilde{\xi}_i\right\} \\
\text{s.t.} \quad & d_i-\left(\sum_{j=1}^{N}(\alpha_j^*-\alpha_j)k(\boldsymbol{x}_j,\boldsymbol{x}_i)+b\right)\leqslant\varepsilon+\tilde{\xi}_i \\
& \sum_{j=1}^{N}(\alpha_j^*-\alpha_j)k(\boldsymbol{x}_j,\boldsymbol{x}_i)+b-d_i\leqslant\varepsilon+\tilde{\xi}_i \\
& \tilde{\xi}_i\geqslant 0,\quad \alpha_i^*,\ \alpha_i\geqslant 0,\quad i=1,\ldots,N
\end{aligned} \tag{9}$$

In the following, we will give Proposition 1 to demonstrate the relation between (8) and (9).

Proposition 1. *For LP-SVR,* Eq. (9) *is completely equivalent to* (8).

Proof. Since Eq. (8) is a linear programming problem, we can use the sophisticated solver like the interior point method or the simplex method to obtain the global unique optimization solution. Assume that $\bar{\boldsymbol{s}}=[\bar{\alpha}_1^*,\ldots,\bar{\alpha}_N^*,\bar{\alpha}_1,\ldots,\bar{\alpha}_N,\bar{\xi}_1^*,\ldots,\bar{\xi}_N^*,\bar{\xi}_1,\ldots,\bar{\xi}_N,\bar{b}]$ is the optimal solution of (8), hence, $\bar{\xi}_i^*\cdot\bar{\xi}_i$ must be equal to zero. Here, we will proof this conclusion using three cases.

(1) If $\bar{\xi}_i^*\cdot\bar{\xi}_i\neq 0$ and $d_i-\left(\sum_{j=1}^{N}(\bar{\alpha}_j^*-\bar{\alpha}_j)k(\mathbf{x}_j,\mathbf{x}_i)+\bar{b}\right)>\varepsilon$, then $\hat{\xi}_i^*=\bar{\xi}_i^*$ and $\hat{\xi}_i=0\quad(i=1,\ldots,N)$ can let the cost function of (8) much smaller than $\bar{\xi}_i^*$ and $\bar{\xi}_i\quad(i=1,\ldots,N)$ meantime subjecting to the constraints of (8), that is to say, $\bar{\boldsymbol{s}}$ is not the optimal solution of (8), which is in contradiction to the assumption.

(2) If $\bar{\xi}_i^*\cdot\bar{\xi}_i\neq 0$ and $\sum_{j=1}^{N}(\bar{\alpha}_j^*-\bar{\alpha}_j)k(\mathbf{x}_j,\mathbf{x}_i)+\bar{b}-d_i>\varepsilon$, similarly, then $\hat{\xi}_i^*=0$ and $\hat{\xi}_i=\bar{\xi}_i\quad(i=1,\ldots,N)$ can make the cost function of (8) much smaller than $\bar{\xi}_i^*$ and $\bar{\xi}_i\quad(i=1,\ldots,N)$ meantime satisfying the constraints of (8), which can also lead to the contradiction to the assumption.

(3) If $\bar{\xi}_i^*\cdot\bar{\xi}_i\neq 0$ and $\left|\sum_{j=1}^{N}(\bar{\alpha}_j^*-\bar{\alpha}_j)k(\mathbf{x}_j,\mathbf{x}_i)+\bar{b}-d_i\right|\leqslant\varepsilon$, then we let $\hat{\xi}_i^*=0$ and $\hat{\xi}_i=0$, which can get the same contradictory conclusion analog to (1) and (2).

From the three cases above, we can get the conclusion, viz. $\bar{\xi}_i^*\ldots\bar{\xi}_i=0$. Here, we let $\bar{\tilde{\xi}}_i=\bar{\xi}_i^*+\bar{\xi}_i$, and then $\bar{\tilde{\boldsymbol{s}}}=[\bar{\alpha}_1^*,\ldots,\bar{\alpha}_N^*,\bar{\alpha}_1,\ldots,\bar{\alpha}_N,\bar{\tilde{\xi}}_1,\ldots,\bar{\tilde{\xi}}_N,\bar{b}]$ is the optimal solution of (9). If not, we assume that $\bar{\tilde{\boldsymbol{s}}}'=\left[\bar{\alpha}_1^{*\prime},\ldots,\bar{\alpha}_N^{*\prime},\bar{\alpha}_1',\ldots,\bar{\alpha}_N',\bar{\tilde{\xi}}_1',\ldots,\bar{\tilde{\xi}}_N',\bar{b}'\right]$ is the optimal solution of (9).

Moreover, we suppose $d_i-\left(\sum_{j=1}^{N}\left(\bar{\tilde{\alpha}}_j^{*\prime}-\bar{\tilde{\alpha}}_j'\right)k(\mathbf{x}_j,\mathbf{x}_i)+\bar{b}'\right)>\varepsilon$ (similar for $\left(\sum_{j=1}^{N}\left(\bar{\tilde{\alpha}}_j^{*\prime}-\bar{\tilde{\alpha}}_j'\right)k(\mathbf{x}_j,\mathbf{x}_i)+\bar{b}'\right)-d_i>\varepsilon$ or $\left|\left(\sum_{j=1}^{N}\left(\bar{\tilde{\alpha}}_j^{*\prime}-\bar{\tilde{\alpha}}_j'\right)k(\mathbf{x}_j,\mathbf{x}_i)+\bar{b}'\right)-d_i\right|\leqslant\varepsilon$), and then we let $\alpha_i^*=\bar{\tilde{\alpha}}_i^{*\prime}$, $\alpha_i=\bar{\tilde{\alpha}}_i'$, $\xi_i^*=\bar{\tilde{\xi}}_i'$, $\xi_i=0$ ($i=1,\ldots,N$), and $b=\bar{b}'$. Naturally, $\boldsymbol{s}'=\left[\bar{\tilde{\alpha}}_1^{*\prime},\ldots,\bar{\tilde{\alpha}}_N^{*\prime},\bar{\tilde{\alpha}}_1',\ldots,\bar{\tilde{\alpha}}_N',\bar{\tilde{\xi}}_1',\ldots,\bar{\tilde{\xi}}_N',0,\ldots,0,\bar{b}'\right]$ can make the cost function of (8) be smaller than $\bar{\boldsymbol{s}}$, which is in contradiction to the assumption, i.e. $\bar{\boldsymbol{s}}$ is the optimal solution of (8). Hence, $\boldsymbol{s}'=\bar{\boldsymbol{s}}$ must hold, namely, Eq. (8) is completely equivalent to (9). Here, the proof is completed. □

Compared with (8) and (9) eliminate N slack variables, which can reduce the computational burden and memory space. For the sake of simplicity, we utilize ξ_i as a replacement of $\tilde{\xi}_i$ in (9) and rewrite it in economy size as follows:

$$\begin{aligned}
\min \quad & \boldsymbol{c}^T\boldsymbol{s} \\
\text{s.t.} \quad & A\boldsymbol{s}\leqslant\boldsymbol{v} \\
& \boldsymbol{s}\geqslant\left[\underbrace{0,\ldots,0}_{N},\underbrace{0,\ldots,0}_{N},\underbrace{0,\ldots,0}_{N},-\infty\right]^T\in\Re^{3N+1}
\end{aligned} \tag{10}$$

where

$$s=\left[\underbrace{\alpha_1^*,\ldots,\alpha_N^*}_{N},\underbrace{\alpha_1,\ldots,\alpha_N}_{N},\underbrace{\xi_1,\ldots,\xi_N}_{N},b\right]^T,\quad c=\left[\underbrace{1,\ldots,1}_{N},\underbrace{1,\ldots,1}_{N},\underbrace{C,\ldots,C}_{N},0\right]^T,\quad A=\begin{bmatrix}-K, & K, & -I, & -\vec{1}\\ K, & -K, & -I, & \vec{1}\end{bmatrix}$$

with

$$\vec{\mathbf{1}}=\left[\underbrace{1,\ldots,1}_{N}\right]^T$$

and

$$K_{ij}=k(\mathbf{x}_i,\mathbf{x}_j),\quad v=\left[\underbrace{\varepsilon-d_1,\ldots,\varepsilon-d_N}_{N},\underbrace{\varepsilon+d_1,\ldots,\varepsilon+d_N}_{N}\right]^T$$

Obviously, the linear optimization problem (10) can be solved using the interior point method or the simplex method. After (10) is solved, we can get the predictor, viz. LP-SVR, as

$$f(x)=\sum_{\mathbf{x}_i\in SV}(\alpha_i^*-\alpha_i)k(\mathbf{x}_i,x)+b \tag{11}$$

Here, *SV* represents the small set of support vectors, i.e., the training data with $\alpha_i^*-\alpha_i\neq 0$. For the QP-SVR, the training data not lying in the ε-insensitive tolerance band are the support vectors. While, in the LP-SVR case, the result is no longer true—although the solution is still sparse, any training datum could be a support vector, even if it does lie in the ε-insensitive tolerance band (Smola et al., 1999). As an extreme, when we set the width of the tolerance band zero, the sparse solution can be still obtained because of the soft constraints utilized (Drezet & Harrison, 2001). Usually, we employ the non-zero width to construct the tolerance band for the purpose of sparser LP-SVR. However, in the QP-SVR context, the sparse solution cannot be found including the zero width of the tolerance band.

3. Multikernel parametric linear programming support vector regression

For the sake of utilizing the additional knowledge about the will-learnt system, Smola et al. (1998) proposed a semiparametric predictor in the following which can be found through solving a convex optimization problem like in the conventional support vector regression:

$$f(\boldsymbol{x})=\boldsymbol{w}\cdot\varphi(\boldsymbol{x})+\sum_{j=1}^{B_r}b_j\phi_j(\boldsymbol{x}) \tag{12}$$

where $\phi_j(\boldsymbol{x})$ is a admissible function, b_j is the corresponding coefficient, namely weight. When we compare (12) with (11), it will be noted that if $\phi_j(\cdot)\equiv 1$ and $B_r=1$, (12) is equivalent to (11). In other

words, the conventional predictor is a special case of the semiparametric predictor. Here, after plugging (12) into (8), we can get the following optimization problem to generate the semiparametric predictor.

$$\begin{aligned}
\min_{\boldsymbol{\alpha}^*,\boldsymbol{\alpha},b} \quad & \left\{\sum_{i=1}^{N}(\alpha_i^*+\alpha_i)+C\sum_{i=1}^{N}(\xi_i^*+\xi_i)\right\} \\
\text{s.t.} \quad & d_i-\left(\sum_{j=1}^{N}(\alpha_j^*-\alpha_j)k(\mathbf{x}_j,\ \mathbf{x}_i)+\sum_{j=1}^{B_r}b_j\phi_j(\mathbf{x}_i)\right)\leqslant\varepsilon+\xi_i^* \\
& \sum_{j=1}^{N}\left(\alpha_j^*-\alpha_j\right)k(\mathbf{x}_j,\ \mathbf{x}_i)+\sum_{j=1}^{B_r}b_j\phi_j(\mathbf{x}_i)-d_i\leqslant\varepsilon+\xi_i \\
& \xi_i,\ \xi_i^*\geqslant 0\alpha_i^*,\ \alpha_i\geqslant 0 \quad i=1,\ldots,N
\end{aligned} \tag{13}$$

According to Proposition 1, we reformulate (13) in the economy size as

$$\begin{aligned}
\min \quad & \boldsymbol{c}^T\boldsymbol{s} \\
\text{s.t.} \quad & A\boldsymbol{s}\leqslant\boldsymbol{v} \\
& \boldsymbol{s}\geqslant\left[\underbrace{0,\ldots,0}_{N},\underbrace{0,\ldots,0}_{N},\underbrace{0,\ldots,0}_{N},\underbrace{-\infty,\ldots,-\infty}_{B_r}\right]^T\in\Re^{3N+B_r}
\end{aligned} \tag{14}$$

where

$$\boldsymbol{s}=\left[\underbrace{\alpha_1^*,\ldots,\alpha_N^*}_{N},\underbrace{\alpha_1,\ldots,\alpha_N}_{N},\underbrace{\xi_1,\ldots,\xi_N}_{N},\underbrace{b_1,\ldots,b_{B_r}}_{B_r}\right]^T,$$

$$c=\left[\underbrace{1,\ldots,1}_{N},\underbrace{1,\ldots,1}_{N},\underbrace{C,\ldots,C}_{N},\underbrace{0,\ldots,0}_{B_r}\right]^T,$$

$$A=\begin{bmatrix}-K, & K, & -I, & -\Phi\\ K, & -K, & -I, & \Phi\end{bmatrix}$$

with (10) and (14) is still a linear programming problem. Although the coefficients of those admissible functions are not regularized in the cost function, the generated semiparametric predictor cannot cause the overfitting phenomena because B_r is kept sufficiently smaller than N (Nguyen & Tay, 2008). In the following section, the simulation results also show that the overfitting cannot arise. So far, there is another very significant problem not to be solved, that is, how to select the admissible functions $\phi(\cdot)$s. In Smola et al. (1998) and Li et al. (2007), the admissible functions are determined using the explicit functions according to the additional knowledge about the identification system. This is very limited because we usually do not own the prior knowledge about the problem, where what admissible functions to be selected will baffle us. As we know, usually, a kernel function can induce an implicit and nonlinear mapping which is associated with the selected training data, and the selected data, viz. so-called support vectors, chosen by the support vector learning algorithms commonly play an important role in the process of the modeling. Naturally, a good idea comes to us, that is, the nonlinear mappings induced from the kernel functions are chosen as the admissible functions, and these admissible functions may be infinite-dimensional and implied. Besides, if the admissible functions are constructed using different kind kernel functions or different parameter kernel functions, then we can get a multikernel semiparametric learning avenue. To be more important, the will-identified system usually holds different data trends in different regions. If the single fixed kernel does not easily fit the system globally, it is necessary to consider that different data trends are fitted using different kernel functions. Actually, we are capable of finishing this learning algorithm through multiple stages. Firstly, we can roughly utilize the conventional LP-SVR to select support vectors, and then these support vectors and their corresponding kernel functions are used to construct the admissible functions for the next training stage. This procedure can be repeated many times until our requirements are satisfied, and during each stage, the chosen kernel function may be different. Hence, the final semiparametric predictor is the multiple kernels. Finally, we summarize the procedure of MSLP-SVR below

Algorithm 1. MSLP-SVR

1. Define a positive integer M, a array $T=[T_1,\ldots,T_M]$, where T_j is a positive integer, $j=1,\ldots,M$, and a kernel function set $\{k_1(\cdot,\cdot),\ldots,k_M(\cdot,\cdot)\}$; and let $n=1$.
2. If $n==1$, use the conventional LP-SVR with the kernel function $k_n(\cdot,\cdot)$ to select T_n training data, namely so-called support vectors; else utilize the MSLP-SVR with $k_n(\cdot,\cdot)$, viz. (14), to choose T_n training data as support vectors.
3. If $n==M$, stop; else utilize the $(T_1+\cdots+T_n)$ training data and their corresponding kernel functions to construct the admissible functions for the next MSLP-SVR training.
4. Let $n=n+1$ and go to step 2.

After finishing Algorithm 1, we obtain T_M support vectors and $(T_1+\cdots+T_{M-1})$ admissible functions. In summary, those training data associated with their corresponding admissible functions may be called as support vectors except that their corresponding support weights are not regularized in the cost function. Roughly speaking, we get $\sum_{j=1}^{M}T_j$ support vectors and M kinds of kernel functions to model the final semiparametric predictor. Generally, $(T_1+\cdots+T_{M-1})$ is sufficiently smaller than N to avoid the overfitting phenomena. How to select T_j and $k_j(\cdot,\cdot)$ $(j=1,\ldots,M)$ is still an open problem. Of course, the traditional techniques such as cross validation are candidate choices. As for M, according to our experience, M is usually small. In our following experiments, $M=2$ is satisfying. If the computational complexity of the conventional LP-SVR is $O(\cdot)$, then the computational burden of MSLP-SVR is $M\cdot O(\cdot)$, which is comparable with the standard LP-SVR due to M being very small like $M=2$.

4. Experiments

For the sake of showing the efficacy of our proposed MSLP-SVR, we will do experiments on the synthetic and real-world data sets and compare it with other multikernel linear programming method. In this letter, the Gaussian $k(\mathbf{x}_i,\mathbf{x}_j)=\exp\left(-\frac{\|\mathbf{x}_i-\mathbf{x}_j\|^2}{2\gamma^2}\right)$ is selected as the kernel functions. As for the multikernel, we use different width kernel parameters to construct different Gaussian kernel functions. In addition, all the experiments are performed using the interior point method on a personal computer with AMD 3200+ (2.01 GHz) processor, 512 MB memory, and Windows XP operation system in a MATLAB 7.1 environment. To compare conveniently, one performance index, viz. root mean squared error (RMSE), is defined as a derivation measurement between the target and the predictive values.

4.1. Simulations on the synthetic and real-world data sets

In this experiment, from the following univariate curve (Nguyen & Tay, 2008),

$$f(x)=\begin{cases}1.25x+15, & x<-4\\ -9.246x-26.984, & -4\leqslant x<-2\\ 4.246x, & -2\leqslant x<0\\ 10e^{-0.05x-0.5}\sin((0.03x+0.7)x) & x\geqslant 0\end{cases}$$

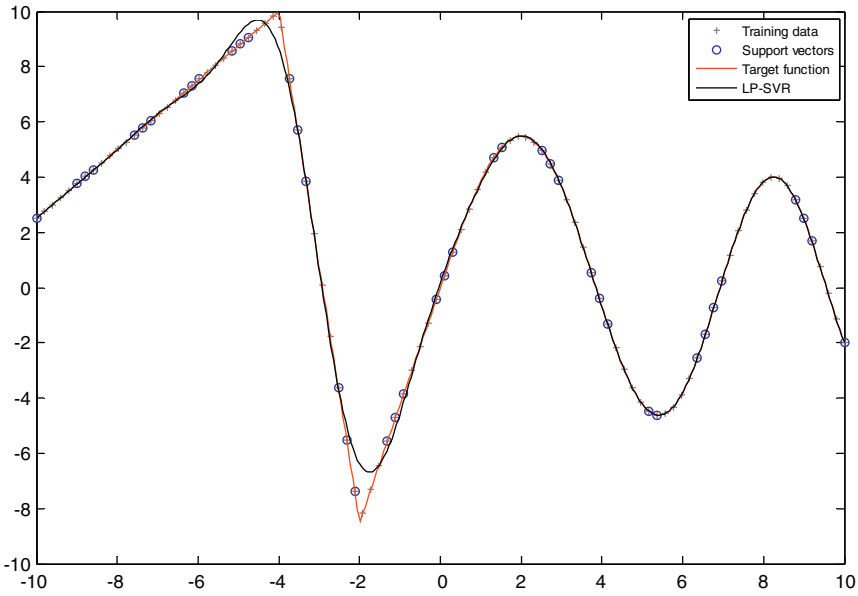

Fig. 1. Simulation results of the conventional LP-SVR ($\gamma_1 = 1.5$).

we generate a training data set *zigzag* of size 100, which are uniformly sampled from the range $[-10,10]$. Moreover, in this range, 333 testing data are uniformly taken as the testing data set. Firstly, we utilize the conventional LP-SVR, namely single kernel LP-SVR, with different kernel widths to fit this curve. These two kernel widths are $\gamma_2 = 1.5$ and $\gamma_1 = 0.3$ which are derived from Nguyen and Tay (2008). Meantime, we manually tune the tolerance parameter ε to guarantee that the number of support vectors (#SV) is 43 which is in accordance with Nguyen and Tay (2008), and we set $C = 2^6$. The simulation results are listed in Figs. 1 and 2. From Fig. 1, we know that the conventional LP-SVR with wide width kernel parameter fits the smooth parts very well, but it fails to track the two sharp edges. Conversely, although the LP-SVR of narrow width kernel parameter somewhat forms the two sharp edges, yet the smooth parts are matched deterioratedly. These phenomena accord with our experience: usually, the wider width kernels fitting the smooth variation can achieve good results, while the narrower ones match the steep variation well.

Why do these phenomena arise for this curve? The main reason is that this curve holds different data trends in different regions. Hence, it is necessary to utilize multiple kernels to fit this plot so that different data trends are matched using different kernels. Here, we employ the Algorithm 1 to improve the regression accuracy, where $M = 2$, $k_1(\cdot,\cdot)$ and $k_2(\cdot,\cdot)$ are parameterized by $\gamma_1 = 1.5$ and $\gamma_2 = 0.3$. In addition, ε_1 and ε_2 are tuned manually so as to keep the same number of support vectors as the above for fair comparison. Fig. 3 depicts the simulation results of Algorithm 1. According to this figure, compared with Fig. 1, the effectiveness of tracking the two sharp edges is improved greatly and the fitting results of the smooth parts do not deteriorate. In contrast to Fig. 2, the entire fitting results are enhanced tremendously. As for the specific simulation results, they are listed in Table 1. In addition, from the fitting result, viz. RMSE = 0.0574, given by Nguyen and Tay (2008), our experimental results, namely RMSE = 0.0435, is comparable, and the MSLP-SVR holds the computational advantage over the multikernel method proposed by Nguyen and Tay (2008) due to the linear programming versus the quadratic programming. From the synthetic example above, we can understand the effectiveness of the multikernel learning algorithms in a way. In the following, we will give the experimental results on the real world data sets.

In this paragraph, two multivariate data sets from the internet webpage,[1] *auto_price* and *triazines*, are used to test the effectiveness of the MSLP-SVR. Before simulations, we normalize the input data into the closed interval $[0,1]$ without normalization on the outputs. As for the model selection, the optimal pair (γ^*, C^*) is determined by the commonly-used cross validation technique (CV) from the candidate pairs $\{2^{-4}, 2^{-3}, \ldots, 2^3, 2^4\} \times \{2^{-4}, 2^{-3}, \ldots, 2^9, 2^{10}\}$ for the learning algorithms, and the tolerance parameter, viz. ε, is also chosen according to the CV. The validation set is randomly selected about 2/5 of the training set. In addition, the number of the training data (trNum) and the number of the testing data (teNum) about each data set are specified in the items of Table 2. As for detailed experimental results, they are tabulated, i.e., Table 2. From Table 2, it is easily comprehended that the MSLP-SVR can improve the simulation results of the conventional LP-SVR more or less, which is also favor for our proposed MSLP-SVR.

4.2. Comparison with other multikernel linear programming method

In this section, we will compare MSLP-SVR with the linear programming multikernel learning algorithm proposed by Zheng, Wang, and Zhao (2006). Two simulation examples are both from their paper, one of which is the mixture function; another is the oscillation function.

4.2.1. Example 1: mixture function

The mixture function is: $f(x) = \frac{1}{\sqrt{2\pi}\times 0.3}\exp\left(-\frac{(x-2)^2}{2\times 0.3^2}\right) + \frac{1}{\sqrt{2\pi}\times 1.2}\exp\left(-\frac{(x-7)^2}{2\times 1.2^2}\right)$. We also uniformly generate 30 training sets of size 100 with the normal distribution noise $(0, 0.05^2)$. $M = 2$ is chosen for MSLP-SVR, and the kernel parameters $\gamma_1 = 0.3$ and $\gamma_2 = 1.2$ are selected from Zheng et al. (2006) . In addition, we let $C = 2^4$, which is approximately equal to the value used by Zheng et al.

4.2.2. Example 2: oscillating function

This example is a highly oscillating function: $r(x) = \sin\left(\frac{2\pi(0.35\times 10+1)}{0.35x+1}\right)$. In this case, we uniformly generate 30 training sets of size 100 by adding an independent normal distribution noise

[1] Available from the URL:http://www.liaad.up.pt/~ltorgo/Regression/.

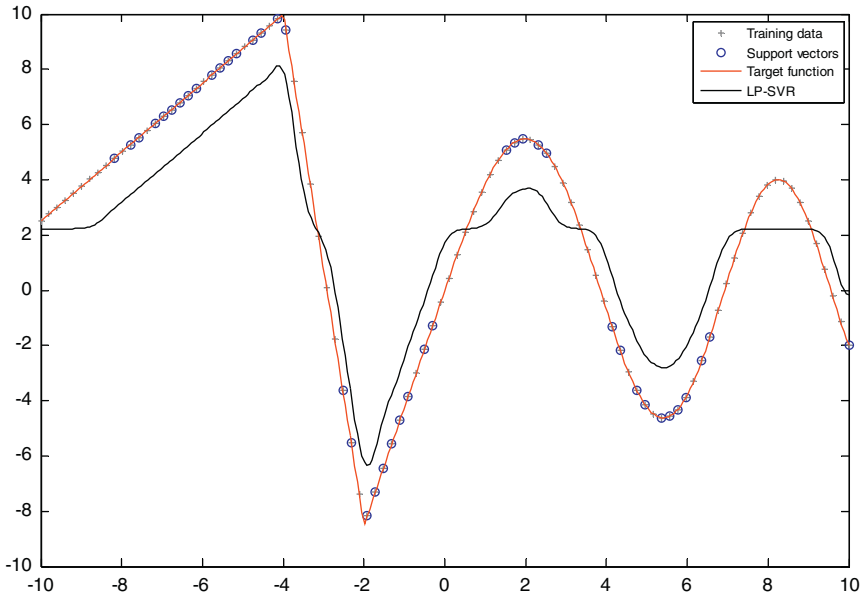

Fig. 2. Simulation results of the conventional LP-SVR ($\gamma_1 = 0.3$).

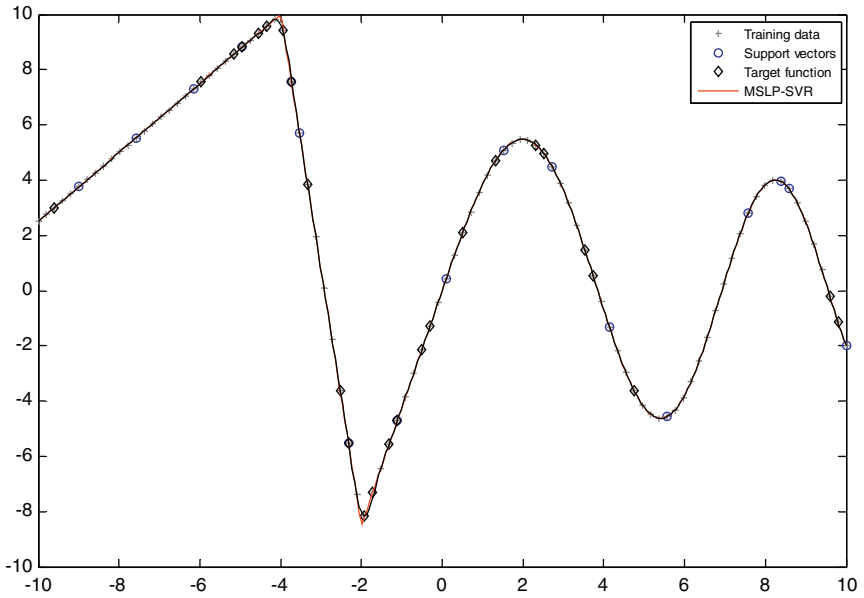

Fig. 3. Simulation results of the MSLP-SVR.

Table 1
Simulation results on the *zigzag* data set.

Algorithms	RMSE	T_1	T_2	#SV	γ	ε
Conventional LP-SVR	1.6041	N/A	N/A	43	0.3	1.8201
	2.95E−01	N/A	N/A	43	1.5	4.13E−06
MSLP-SVR	**4.35E−02**	17	26	43	{1.5,0.3}	{0.04,0.003}

Table 2
Simulation results on the real-world data sets.

Data sets	Algorithms	trNum	teNum	RMSE	T_1	T_2	#SV	γ^*	ε
Auto_price	Conventional LP-SVR	120	39	2.70E+03	N/A	N/A	65	2^0	{550}
$C^* = 2^7$	MSLP-SVR	120	39	**2.22E+03**	15	50	65	$\{2^2, 2^0\}$	{10,10}
Triazines	Conventional LP-SVR	140	46	1.72E−01	N/A	N/A	51	$\{2^0\}$	{0.1}
$C^* = 2^5$	MSLP-SVR	140	46	**1.55E−01**	8	43	51	$\{2^{-2}, 2^0\}$	{0.3,0.1}

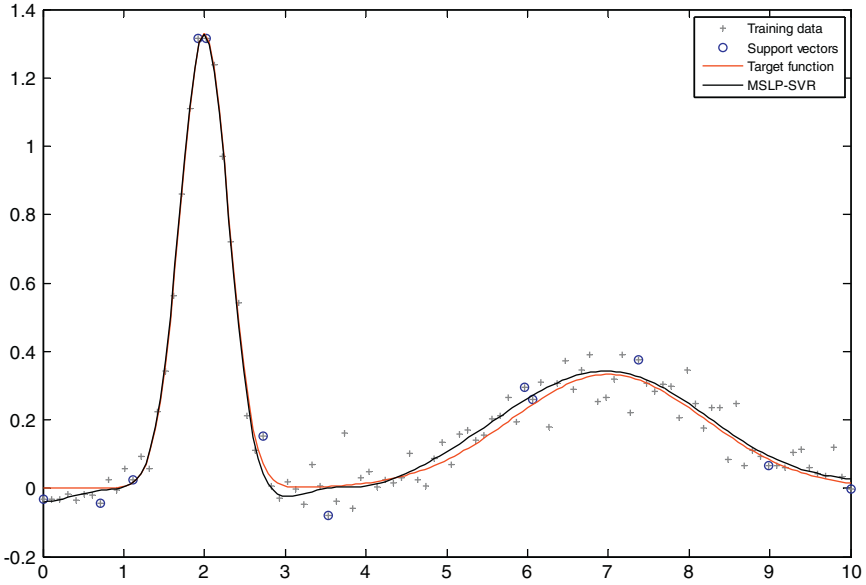

Fig. 4. Simulation results of mixture function using MSLP-SVR.

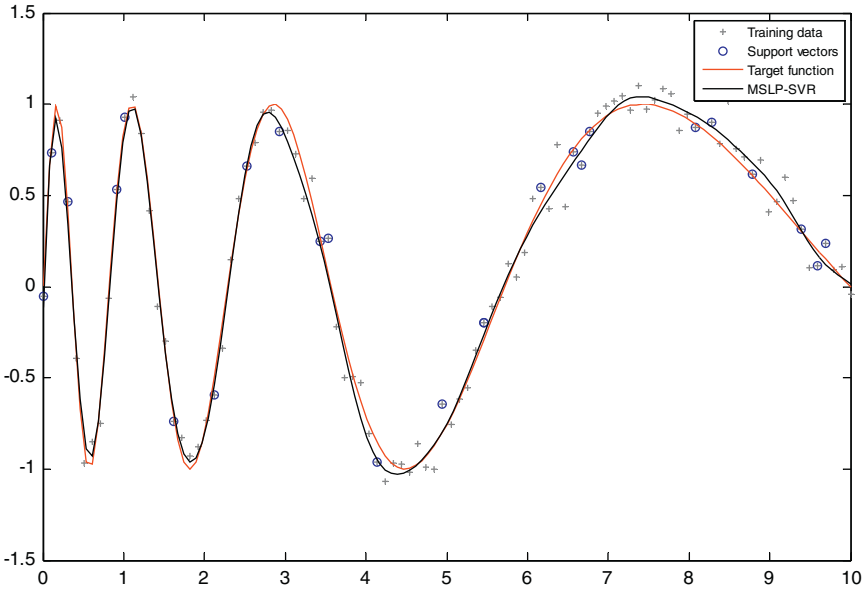

Fig. 5. Simulation results of oscillating function using MSLP-SVR.

$(0, 0.1^2)$, which is the same as Zheng et al. (2006). We also choose $M = 2$ for MSLP-SVR, and the kernel parameters $\gamma_1 = 1.4$ and $\gamma_2 = 0.4$ are decided by the cross validation technique. The validation set is additionally sampled from the oscillating function. Here, the approximately equal regularization parameter is utilized, viz. $C = 2^5$.

Figs. 4 and 5 give the simulation results of the mixture and the oscillating functions using the MSLP-SVR algorithm, respectively. Meanwhile, the detailed simulation results are listed in Table 3. Except the training time (trTime), the other items involving Zhang et al.'s proposed method are obtained from their paper. Due to no training time in their paper, we have to realize their algorithm

Table 3
Comparison results.

Data sets	Algorithms	trTime (s)	#SV	RMSE
Mixture	Zheng et al.'s algorithm	5.13	15.8	0.0202
	MSLP-SVR	**3.08**	**13.0**	**0.0181**
Oscillating	Zheng et al.'s algorithm	6.03	27.0	0.0581
	MSLP-SVR	**3.20**	**25.0**	**0.0444**

to compare conveniently. The experimental results about the MSLP-SVR are the average of 30 training data sets.

According to Table 3, we know that MSLP-SVR takes advantages of training computational complexity, the number of support vector and the regression accuracy over the algorithm proposed by Zheng et al. As for the computational complexity, the linear programming problem proposed by Zheng et al. concerns $(2m + 2)N + 1$ variables and $(2m + 4)N$ constraints, where m is the number of the selected kernel functions, while in the case of MSLP-SVR, it involves $(3N + B_r)$ variables and $(5N)$ constraints. As we know, the computational complexity of the linear programming is related to the number of the variables and the number of the constraints. Generally, $m > 1$, such as $m = 3$ for the mixture function and $m = 4$ for the oscillating function, so, despite requiring M calculation circles, MSLP-SVR owns the advantage of the computational complexity due to M being very small like $M = 2$. Moreover, MSLP-SVR holds the superiority of the memory space to the Zheng et al.'s method, i.e., their algorithm needs $O(4mN^2)$, while ours only requires $O(6N^2)$.

5. Conclusions

In many real life fields, we usually encounter some unknown systems to identify, and these systems sometimes hold different data trends in different regions, i.e. some parts are steep variations while others are smooth variations. In this context, if we use the traditional kernel learning algorithms like the single kernel LP-SVR to identify these systems, the fitting effectiveness is commonly not satisfactory. For this case, there are two candidate methods to mitigate this embarrassment. One is multikernel trick which has been becoming one hot topic in the kernel learning domain. It is capable of utilizing different kind kernel functions or different parameter kernel functions to fit different data trends in different regions for the unknown systems. On the other hand, if we have the additional knowledge about the unknown systems in advance, of course, it is a very good choice to identify these systems by virtue of the priori knowledge. For example, Smola et al. (1998) utilized the additional knowledge to construct the admissible functions, viz. the semiparametric technique, so as to improve the learning effectiveness. However, generally, we do not have the additional knowledge about the will-identified system. In this case, it is very hard to construct the explicit admissible functions which are usually beneficial to model the systems. Hence, in this letter, we exploit the nonlinear mappings induced from the kernel functions as the admissible functions to improve the regression accuracy of the conventional LP-SVR. This novel predictor, named as MSLP-SVR, combines the semiparametric technique with the multikernel trick, which can improve the fitting effectiveness of the conventional LP-SVR more or less with the comparable computational complexity.

The experimental results on the synthetic and real-world data sets show the effectiveness of the MSLP-SVR. Meantime, compared with other multikernel linear programming learning algorithm, our proposed MSLP-SVR own the superiorities of the number of support vectors, the computational complexity, and regression accuracy. Although we propose MSLP-SVR for the regression problem, similarly, it can be extended to classification realm. In addition, in the linear programming learning algorithms, we are able to obtain a solvable linear programming problem even using non-Mercer kernels (Lu & Sun, 2009), so MSLP-SVR may be expanded with non-Mercer hybrid kernels, which is our future work.

Acknowledgment

This research was supported by the National Natural Science Foundation of China under Grant Nos. 50576033 and 51006052.

References

Bi, J., Zhang, T., Bennett, K. (2004). Column-generation boosting methods for mixture of kernels. In *KDD* (pp. 521–526).
Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*(2), 121–167.
Chang, C.C., Lin, C.J. (2001). *LIBSVM: A library for support vector machines*. <http://www.csie.ntu.edu.tw/~cjlin>.
Chen, S., & Wang, M. (2005). Seeking multi-thresholds directly from support vectors for image segmentation. *Neurocomputing, 67*(1–4), 335–344.
Collobert, R., & Bengio, S. (2001). SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research, 1*(2), 143–160.
Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines*. Cambridge University Press.
Drezet, P.M.L., Harrison, R.F. (1998). Support vector machines for system identification. In *Proceedings of the 1998 international conference on control. Part 1 (of 2)*.
Drezet, P. M. L., & Harrison, R. R. (2001). A new method for sparsity control in support vector classification and regression. *Pattern Recognition, 34*, 111–125.
Guo, G., Li, S. Z., & Chan, K. L. (2001). Support vector machines for face recognition. *Image and Vision Computing, 19*(9–10), 631–638.
Hadzic, I., Kecman, V. (2000). Support vector machines trained by linear programming theory and application in image compression and data classification. In *Proceedings of the 5th seminar on neural network applications in electrical engineering*.
Joachims, T. (1999). Making large-scale SVM learning practical. In *Advances in kernel methods-support vector machine, Cambridge, MA, USA*.
Kecman, V. (2001). *Learning and soft computing: Support vector machines, neural networks, and fuzzy logic models*. MIT Press.
Kimeldorf, G. S., & Wahba, G. (1970). A correspondence between Bayesian estimation on stochastic processes and smoothing by splines. *Annals of Mathematical Statistics, 41*, 495–502.
Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Joran, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research, 5*, 27–72.
Lau, K. W., & Wu, Q. H. (2008). Local prediction of non-linear time series using support vector regression. *Pattern Recognition, 41*(5), 1539–1547.
Lee, K. L., & Billings, S. A. (2002). Time series prediction using support vector machines, the orthogonal and the regularized orthogonal least-squares algorithms. *International Journal of Systems Science, 33*, 811–821.
Li, Q., Jiao, L., & Hao, Y. (2007). Adaptive simplification of solution for support vector machine. *Pattern Recognition, 40*(3), 972–980.
Li, W., Lee, K.-H., & Leung, K.-S. (2007). Generalized regularized least-squares learning with predefined features in a Hilbert space. *Advances in Neural Information Processing Systems* (Vol. 19). Cambridge, MA: MIT Press.
Lu, Z., & Sun, J. (2009). Non-Mercer hybrid kernel for linear programming support vector regression in nonlinear systems identification. *Applied Soft Computing, 9*, 94–99.
Nguyen, D.-D., & Ho, T.-B. (2006). A bottom-up method for simplifying support vector solutions. *IEEE Transactions on Neural Networks, 17*(3), 792–796.
Nguyen, C.-V., & Tay, D. B. H. (2008). Regression using multikernel and semiparametric support vector algorithms. *IEEE Signal Processing, 15*, 481–484.
Ong, C. S., Smola, A. J., & Williamson, R. C. (2005). Learning the kernel with hyperkernels. *Journal of Machine Learning Research, 6*, 1043–1071.
Osuna, E., Freund, R., Girosi, F. (1997). An improved training algorithm for support vector machines. In *Proceedings of neural networks for signal processing VII, New York, USA*.
Platt, J.C. (1998). Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods-support vector machines, Cambridge, MA, USA*.
Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. Cambridge: MIT Press.
Shevade, S. K., Keerthi, S. S., Bhattacharyya, C., & Murthy, K. R. K. (2000). Improvements to the SMO algorithm for SVM regression. *IEEE Transactions on Neural Network, 11*(5), 1188–1193.
Smola, A.J., Frie, T.T., Schölkopf, B. (1998). Semiparametric support vector and linear programming machines. In *Proceedings of the 1998 conference on advances in neural information processing systems II, Cambridge, MA, USA* (pp. 585–591).
Smola, A., Schölkopf, B., Rätsch, G. (1999). Linear programs for automatic accuracy control in regression. In *Proceedings of the 1999 the 9th international conference on artificial neural networks*.
Vapnik, V. N. (1995). *The nature of statistical theory*. New York: Springer-Verlag.
Weston, J., Gammerman, A., Stitson, M. O., Vapnik, V., Vovk, V., & Watkins, C. (1999). *Support vector density estimation, advances in kernel methods-support vector learning*. Cambridge, MA: MIT.
Zheng, D., Wang, J., & Zhao, Y. (2006). Non-flat function estimation with a multi-scale support vector regression. *Neurocomputing, 70*, 420–429.