Soft Comput (2015) 19:2047–2061
DOI 10.1007/s00500-014-1390-x

METHODOLOGIES AND APPLICATION

Incorporating prior knowledge and multi-kernel into linear programming support vector regression

Jinzhu Zhou · Baoyan Duan · Jin Huang · Na Li

Published online: 30 July 2014

Abstract This paper proposes a multi-kernel linear program support vector regression with prior knowledge in order to obtain an accurate data-driven model in the case of an insufficient amount of measured data. In the algorithm, multiple feature spaces have been utilized to incorporate multi-kernel functions into the framework of linear programming support vector regression (LPSVR), and then the prior knowledge which may be exact or biased from a calibrated physical simulator has also been incorporated into LPSVR by modifying optimization formulations. Moreover, a strategy of parameter selections for the proposed algorithm has been presented to facilitate practical applications. Some experiments from a synthetic example, a microstrip antenna and six-pole microwave filter have been carried out, and the experimental results show that the proposed algorithm can obtain a satisfactory data-based model in the case of the scarcity of measured data. The proposed algorithm shows great potentialities in some applications where the experimental data are insufficient for an accurate data-driven model and the prior knowledge from a calibrated physical simulator of practical applications is available.

Keywords Linear programming support vector regression · Prior knowledge · Multi-kernel · Insufficient data · Data-driven model

Communicated by V. Loia.

J. Zhou (✉) · B. Duan · J. Huang · N. Li
Key Laboratory of Electronic Equipment Structure Design of Ministry of Education, Xidian University, Xi'an, Shaanxi, People's Republic of China
e-mail: xidian_jzzhou@126.com

1 Introduction

Support vector regression (SVR), which is based on the theory of structure risk minimization in statistical learning, has been applied to many problems (Smola 2002, 2004; Cristianini and Shawe-Taylor 2000). Traditionally, support vector regression can find an estimating function by solving a quadratic program problem, which is also known as QPSVR (Smola 2004; Cristianini and Shawe-Taylor 2000; Muller et al. 2001). Subsequently, Smola has proposed linear programming support vector regression (LPSVR) (Smola 2002; Smola et al. 1999). Both LPSVR and QPSVR adopt ε-insensitive loss function and the kernel function in feature space. However, LPSVR is advantageous over QPSVR in the sparse model, ability to use more general kernel functions and fast learning ability (Lu et al. 2009; Lu and Sun 2009; Smola 2004; Zhao and Sun 2011).

Support vector regression aims at learning an unknown function based on some training data samples. However, in some practical applications, it is complex and costly to obtain sufficient experimental data. Utilizing the fewer data, one can find that it is a little difficult to obtain an accurate data-driven model. Moreover, there are many complex functions which comprise both the steep variation and the smooth variation in some engineering problems, and this will be more difficult to obtain an satisfactory model from a small amount of data samples (Clarke et al. 2005). In this paper, we will focus on the problem that how to obtain an accurate model from a limited amount of experimental data.

In order to improve the modeling accuracy, Lanckriet has proposed a multi-kernel support vector regression by using the conic combinations of kernel matrices, and formulated the algorithm as a convex quadratically constrained quadratic program (QCQP) (Lanckriet et al. 2004; Bach et al. 2004). Although the formulation yields global optimal solutions,

it is computationally inefficient and requires a commercial solver. Subsequently, the multi-kernel learning algorithm has been reformulated as a semi-infinite linear programming to obtain a general and efficient algorithm (Sonnenburg et al. 2006; Lanckriet et al. 2004). Based on the principle of kernel–target alignment and predictive accuracy, Qiu has proposed three heuristics methods to speed up the computation of QCQP formulation (Qiu and Lane 2009). In Nguyen and Tay (2008), a multi-kernel semi-parametric support vector regression has been proposed by using quadratic program solver and a semi-parametric algorithm. Instead of a single kernel, multi-scale support vector regression has been presented by using the same kernel with multiple scales (Zheng et al. 2006; Yu and Qian 2008). All of the multi-kernel support vector regression can establish an accurate model, if there is sufficient amount of training data samples (Subrahmanya and Shin 2010; Mingqing et al. 2009). However, the training data are usually so little in some practical applications that the model developed by the algorithms cannot meet some desired requirements.

In some engineering applications, a certain amount of knowledge on the problem is usually known beforehand (Sanchez 2003). This prior knowledge can take many forms such as a simulation model from a practical application, the shape of the function on a particular region and some equality and inequality constrains (Lauer and Bloch 2008a; Trnka and Havlena 2009). By utilizing the prior knowledge, one can improve the predictive accuracy of support vector regression. In Bloch et al. 2008; Lauer and Bloch (2008a), the author has reviewed three methods of incorporating the prior knowledge in support vector machine for classification, which comprise sample methods, kernel methods and the optimization methods. In Lauer and Bloch (2008b), the author explores the incorporation of different prior knowledge in support vector regression by modifying the problem formulation. In addition, the prior knowledge over arbitrary region is incorporating into a kernel approximation problem, and the region has to be discretized before including the prior knowledge to be in the learning framework as a finite set of inequalities (Olvi et al. 2004; Mangasarian and Wild 2007). Though the prior knowledge can make up a small amount of measured data, all of the algorithms incorporating prior knowledge have exploited a single kernel, and have not employed the advantages of multi-kernel functions.

The motivation of this investigation is to provide a regression algorithm which can incorporate multi-kernel functions and prior knowledge into the LPSVR. The proposed algorithm can improve the data-based modeling accuracy from an insufficient amount of measured data. The prior knowledge is from a physical simulator which is used to generate simulation data for arbitrarily chosen inputs in order to compensate for the lack of measured data in some regions of the input space. This problem, although particular, is representative of numerous situations met in engineering, where physical models, more or less accurate, exist, providing prior knowledge in the form of simulation data, and where the measured data are difficult or expensive to obtain. This paper will focus on the problem of how to utilize the prior knowledge and multi-kernel function to approximate complex functions in the case of an insufficient amount of measured data.

In this paper, multi-kernel and prior knowledge from a physical simulator were incorporated into the framework of the LPSVR to improve the modeling accuracy. The contribution of this paper is to propose a novel algorithm, termed the multi-kernel prior knowledge linear programming support vector regression (MKPLPSVR). In the algorithm, multiple feature spaces have been utilized to incorporate multi-kernel functions into the framework of the LPSVR, and then the prior knowledge which may be exact or biased from a physical simulator has also been incorporated into the LPSVR by modifying optimization objectives and inequality constraints. At the end, prior knowledge and multi-kernel functions have been simultaneously incorporated into the framework of the LPSVR, and a new formulation has been presented to solve the regression problem from a limited amount of measured data. The MKPLPSVR can be easily solved by using linear programming and is different from other multi-kernel support vector regression at the aspect of the basic principle and solution method. In addition, a strategy of parameter selections has been presented to facilitate the practical application of the proposed MKPLPSVR algorithm.

The rest of the paper is organized as follows. The LPSVR is introduced in the next section. Section 3 describes the proposed algorithm which can incorporate prior knowledge and multiple kernels into the framework of the LPSVR. In Sect. 4, we have given some experimental results from a synthetic example and two practical applications. Finally, Sect. 5 concludes the paper.

The following generic notations will be used throughout this paper. Lower case symbols such as $y, x_{ij}, y_{ij} \ldots$ refer to scalar valued objects and lower case boldface symbols such as $\boldsymbol{x}, y, \boldsymbol{\alpha}, \ldots$ refer to column vectors. The matrices are boldface and uppercase in the paper. The 1-norm $\|x\|_1$ denotes $\sum_{i=1}^{n} |x_i|$. The matrix $X \in \mathrm{R}^{N\times d}$ contains all training samples $x_i (i = 1, \ldots, N)$ as rows. The notation $A \in \mathrm{R}^{N\times n}$ will signify a real $N \times n$ matrix and the jth column of a matrix $\boldsymbol{A}$ is denoted as $\boldsymbol{A}_{\cdot j}$. The matrix $\boldsymbol{A}^{\mathrm{T}}$ will denote the transpose of $\boldsymbol{A}$. For $\boldsymbol{A} \in \mathrm{R}^{n\times N}$ and $\boldsymbol{B} \in \mathrm{R}^{n\times L}$, a kernel $\boldsymbol{K}(\boldsymbol{A}, \boldsymbol{B})$ maps $\mathrm{R}^{n\times N} \times \mathrm{R}^{n\times L}$ into $\mathrm{R}^{N\times L}$. In particular, if x and y are column vectors in R^N, then the mapping $k(\boldsymbol{x}, \boldsymbol{y})$ is a real number. The variables $\boldsymbol{o}$ and $\boldsymbol{e}$ are vectors of appropriate dimensions with all their components, respectively, equal to 0 and 1. The superscript p refers to the prior knowledge, and

the subscript k refers to the number of data samples from the prior knowledge.

2 Review of LPSVR

Let $\mathrm{M} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1, ..., N\}$ be an experimental dataset, where the input is $\boldsymbol{x}_i \in \mathrm{R}^d$ and the output is $\boldsymbol{y}_i \in \mathrm{R}$. The regression is considered as a linear function in the feature space which induced by a nonlinear mapping $\phi(\boldsymbol{x})$. The regression function is written as:

$$f(\boldsymbol{x}) = \boldsymbol{\omega} \cdot \varphi(\boldsymbol{x}) + b \tag{1}$$

where $\boldsymbol{\omega}$ is a normal vector in the feature space, and b is a bias term.

The normal vector $\boldsymbol{\omega}$ can be considered as a linear combination of the training patterns, i.e., $\boldsymbol{\omega} = \sum_{i=1}^{N} \alpha_i \phi(\boldsymbol{x}_i)$. Therefore, the regression function in the original space is expressed as:

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b \tag{2}$$

where $k(\boldsymbol{x}, \boldsymbol{x}_i) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x})$ is the kernel function which usually includes Gaussian radial basis function, polynomial kernel, and even non-Mercer kernel (Smola et al. 1999; Lu and Sun 2009; Lu et al. 2009).

Instead of choosing the flattest function, LPSVR seek the smallest combination of training patterns. According to the statistical learning theory (Vapnik 1995; Smola 2004), the coefficient α_i and the bias term b can be solved by minimizing the regularized risk function:

$$\text{Min: } Q(\boldsymbol{a}) + 2C \sum_{i=1}^{N} L(y_i - f(\boldsymbol{x}_i)) \tag{3}$$

where $Q(\boldsymbol{a})$ is a regularization term, and it is defined as $Q(\boldsymbol{a}) = \|\boldsymbol{\alpha}\|_1 = \sum_{i=1}^{n} |\alpha_i|$. The vector $\boldsymbol{\alpha} = [\alpha_1, \alpha_i, \ldots \alpha_N]^{\mathrm{T}}$ in $Q(\boldsymbol{a})$ determines the function complexity. A hyper-parameter $C > 0$ is introduced to tune the trade-off between the error minimization and the function sparsely. $L(y_i - f(\boldsymbol{x}_i))$ denotes ε-insensitive loss function:

$$L(y_i - f(\boldsymbol{x}_i)) = \begin{cases} 0, & |y_i - f(\boldsymbol{x}_i)| \le \varepsilon \\ |y_i - f(\boldsymbol{x}_i)| - \varepsilon, & \text{otherwise} \end{cases} \tag{4}$$

By introducing a slack variable ξ_i and using ε-insensitive loss function, the LPSVR is formulated as:

$$\begin{aligned} &\text{Find: } \alpha_i, \xi_i, b \\ &\text{Min: } \|\alpha\|_1 + 2C \sum_{i=1}^{N} \xi_i \\ &\text{s.t.} \begin{cases} y_i - \sum_{i=1}^{N} \alpha_i k(x_i, x_j) - b \le \varepsilon + \xi_i \\ \sum_{i=1}^{N} \alpha_i k(x_i, x_j) + b - y_i \le \varepsilon + \xi_i \\ \xi_i \ge 0 \\ \forall i = 1, 2, \ldots, N \end{cases} \end{aligned} \tag{5}$$

In order to solve the optimization above, we can decompose α_i and $|\alpha_i|$ as follows:

$$\begin{aligned} \alpha_i &= \alpha_i^+ - \alpha_i^- \\ |\alpha_i| &= \alpha_i^+ + \alpha_i^- \end{aligned} \tag{6}$$

where $\alpha_i^+, \alpha_i^- \ge 0$. Due to the nature of the constraints, typically only a subset of α_i is non-zero, and the associated training data are called support vectors (Lu and Sun 2009; Lu et al. 2009).

Substituting (6) into (5), the LPSVR can be expressed as:

$$\begin{aligned} &\text{Find: } \alpha_j^+, \alpha_j^-, \xi_i, b \\ &\text{Min: } \sum_{j=1}^{N} (\alpha_j^+ + \alpha_j^-) + 2C \sum_{i=1}^{N} \xi_i \\ &\text{s.t.} \begin{cases} y_i - \sum_{j=1}^{N} (\alpha_j^+ - \alpha_j^-) k(\boldsymbol{x}_i, \boldsymbol{x}_j) - b \le \varepsilon + \xi_i \\ \sum_{j=1}^{N} (\alpha_j^+ - \alpha_j^-) k(\boldsymbol{x}_i, \boldsymbol{x}_j) + b - y_i \le \varepsilon + \xi_i \\ \alpha_j^+ \ge 0, \ \alpha_j^- \ge 0 \\ \xi_i \ge 0. \quad (\forall i = 1, 2, \ldots, N). \end{cases} \end{aligned} \tag{7}$$

The coefficients $\alpha_j^+, \alpha_j^-, \xi_i$ and b in (7) can be solved by using *linprog* in *Matlab*. Substituting (6) into (2), Eq. (2) can be expressed as:

$$f(\boldsymbol{x}) = \sum_{j=1}^{N} (\alpha_j^+ - \alpha_j^-) k(\boldsymbol{x}, \boldsymbol{x}_j) + b \tag{8}$$

3 Proposed algorithms

In some problems of science and engineering, it is complex and costly to obtain sufficient measured data samples by some experiments. On the other hand, a simulation model built from some physical knowledge is available. In this section, we have presented an algorithm which can incorporate multi-kernel and prior knowledge into the learning framework of the LPSVR. Figure1 shows the basic idea of developing the algorithm.

From the Fig. 1, multiple feature spaces have been utilized to develop multi-kernel linear programming support vector regression (MKLPSVR). Subsequently, the optimization objectives and inequality constraints in the MKLPSVR have been modified to incorporate the prior knowledge from a simulation model or a simulator, which leads to multi-kernel prior knowledge linear programming support vector regression (MKPLPSVR). By incorporating the prior knowledge from a calibrated simulator into the MKLPSVR, we can reduce the effect of the biased data from a simulation model on the accuracy of the data-based model. The development of the MKPLPSVR is explained in the following.

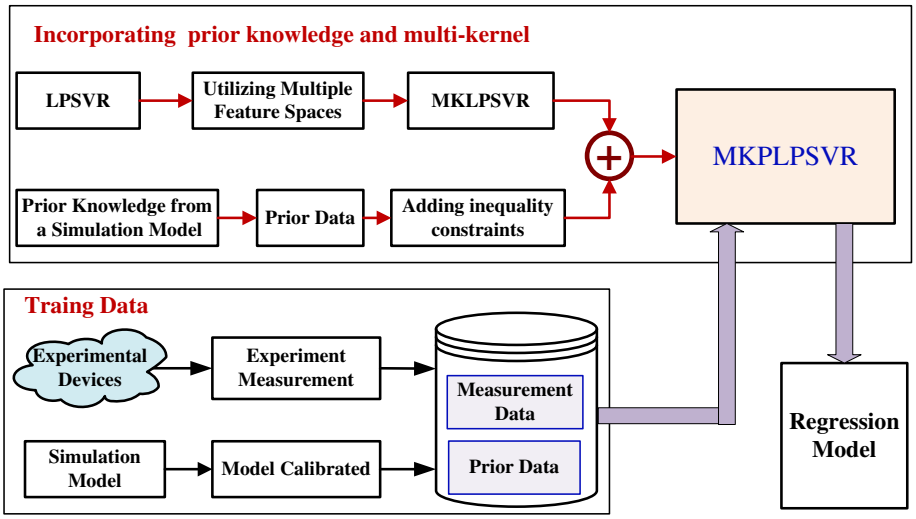

Fig. 1 Block diagram of MKPLPSVR development

3.1 Multi-kernel linear programming support vector regression

As far as the LPSVR is concerned, the function in a feature space is expressed as (1). However, the non-flat function or complicated data trend cannot be described properly in a single feature space. It is promising to seek a space which can utilize the advantage of different feature spaces (Zheng et al. 2006). Therefore, it may be better choice to consider the regression in multiple feature spaces $\boldsymbol{\omega}_1, \ldots \boldsymbol{\omega}_L$, and the function in multiple feature spaces can be written as:

$$f(\boldsymbol{x}) = \sum_{r=1}^{L} \boldsymbol{\omega}_r \cdot \varphi(\boldsymbol{x}) + b \tag{9}$$

Substituting $\boldsymbol{\omega} = \sum_{i=1}^{N} \alpha_i \varphi(\boldsymbol{x}_i)$ into (9), one can express the regression function as:

$$f(\boldsymbol{x}) = \sum_{r=1}^{L} \sum_{i=1}^{N} \alpha_{ri} k_r(\boldsymbol{x}, \boldsymbol{x}_i) + b \tag{10}$$

where L denotes the number of the kernels which are induced by a set of different feature spaces $\boldsymbol{\omega}_1, \ldots \boldsymbol{\omega}_L$. The function $k_r(\boldsymbol{x}, \boldsymbol{x}_i) = \varphi(\boldsymbol{x}_{ri}) \cdot \varphi(\boldsymbol{x})$ denotes the rth kernel, and α_{ri} is the coefficient of the corresponding kernel function.

Utilizing the method in (6), we can reformulate Eq. (10) as:

$$f(\boldsymbol{x}) = \sum_{r=1}^{L} \sum_{i=1}^{N} (\alpha_{ri}^{+} - \alpha_{ri}^{-}) k_r(\boldsymbol{x}, \boldsymbol{x}_i) + b \tag{11}$$

Equation (11) can be estimated by minimizing the risk (3) like the previous method. Since the target to be estimated is a complicated data-trend function, the minimization of the regularization term means the maximum of the function fatness, which may result in under-fitting result (Zheng et al. 2006). To avoid the problem, we have introduced a non-negative constant C_r to control the regularization term. Therefore, analogous to (3), the risk function in a multi-kernel framework is expressed as:

$$\text{Min: } \sum_{r=1}^{L} C_r \|\boldsymbol{\alpha}_r\|_1 + 2C \sum_{i=1}^{N} L(y_i - f(\boldsymbol{x}_i)) \tag{12}$$

where $Q(\boldsymbol{a}) = \sum_{r=1}^{L} C_r \|\boldsymbol{\alpha}_r\|_1$ is a regularization term, and the constant C_r penalizes non-zero coefficients $\boldsymbol{\alpha}_r$. The vector $\boldsymbol{\alpha}_r = [\alpha_{r1}, \alpha_{ri}, \ldots \alpha_{rN}]^{\mathrm{T}}$ denotes the coefficient of the rth kernel, and non-zero elements in the vector $\boldsymbol{\alpha}_r$ are also called support vectors.

Utilizing the method in (6) and (7), the MKLPSVR is expressed as:

$$\begin{aligned}
&\text{Find: } \alpha_{ri}^{+}, \alpha_{ri}^{-}, \xi_i, b \\
&\text{Min: } \sum_{r=1}^{L} C_r \sum_{i=1}^{N} (\alpha_{ri}^{+} + \alpha_{ri}^{-}) + 2C \sum_{i=1}^{N} \xi_i \\
&\text{s.t.} \begin{cases}
y_i - \sum_{r=1}^{L} \sum_{j=1}^{N} (\alpha_{rj}^{+} - \alpha_{rj}^{-}) k_r(\boldsymbol{x}_i, \boldsymbol{x}_j) - b \le \varepsilon + \xi_i \\
\sum_{r=1}^{L} \sum_{j=1}^{N} (\alpha_{rj}^{+} - \alpha_{rj}^{-}) k_r(\boldsymbol{x}_i, \boldsymbol{x}_j) + b - y_i \le \varepsilon + \xi_i \\
\alpha_{rj}^{+} \ge 0, \alpha_{rj}^{-} \ge 0 \\
\xi_i \ge 0 \quad (\forall i = 1, 2, \ldots, N)
\end{cases}
\end{aligned} \tag{13}$$

where C_r depends on the kernel parameter of the used kernel function. The coefficient $\alpha_{ri}^{+}, \alpha_{ri}^{-}$ satisfy $\alpha_{ri} = \alpha_{ri}^{+} - \alpha_{ri}^{-}$ and $|\alpha_{ri}| = \alpha_{ri}^{+} + \alpha_{ri}^{-}$.

Utilizing linear programming to solve Eq. (13), one can obtain the function in Eq. (11). In the paper, the MKLPSVR is a generalized version of the LPSVR. Compared with other

multi-kernel support vector regression (Bach et al. 2004; Sonnenburg et al. 2006; Nguyen and Tay 2008; Pozdnoukhov and Kanevski 2008; Mingqing et al. 2009; Qiu and Lane 2009; Varma and Babu 2009), the MKLPSVR has the following differences. Firstly, the basic principle and solution of the MKLSVR are different from the existing multi-kernel support vector regression, which have exploited convex quadratically constrained quadratic program. Secondly, the MKLPSVR can exploit non-Mercer kernels, which provides more flexibility in designing the kernel function.

3.2 Incorporating prior knowledge into MKLPSVR

In practice, it is complex and costly to obtain sufficient measured data. On the other hand, a simulation model built from some physical knowledge is available. Using a calibrated simulator, one can obtain enough prior data, but which may be biased from the measured results. In order to reduce the effect of the biased prior data, this subsection has presented an approach to incorporate the prior data from a calibrated simulator into the MKLPSVR.

Let the prior dataset $\mathrm{P} = \left\{(z_k^p, y_k^p), z_k^p \in \mathrm{R}^d, y_k^p \in \mathrm{R}, k = 1, 2, \ldots N_k\right\}$ from a calibrated simulator. Obviously, the prior data will satisfy the equation in the simulator:

$$f(z_k^p) = y_k^p \ (k = 1, 2, \ldots, N_k). \tag{14}$$

The equality constraints can be added to the formulation (13) without changing the linear programming nature. However, this will lead to an exact fit to the data points, which may not be advised if the prior data are biased from the measured results. Moreover, all the equality constraints may lead to an unfeasible problem if they cannot be satisfied simultaneously (Lauer and Bloch 2008b; Bloch et al. 2008; Zhou et al. 2011). Therefore, soft constraints have been utilized in Eq. (14) by introducing a positive slack variable $\boldsymbol{u} = [u_1, u_1, \ldots, u_k]^{\mathrm{T}}$. The slack variable can bound the errors between the prior data (z_k^p, y_k^p) and the regression function $f(z_k^p)$ in the following inequality:

$$\left|y_k^p - f(z_k^p)\right| \le u_k \ (\forall k = 1,2, \ldots, N_k). \tag{15}$$

In order to include almost exact or biased knowledge from a prior simulator, it is possible to authorize violations of the constraints (15) that are less than a threshold ε^p. Therefore, by applying ε-insensitive loss function to the error u_k, one can obtain the following inequality:

$$\left|y_k^p - f(z_k^p)\right| \le u_k + \varepsilon^p \ (\forall k = 1, 2, \ldots, N_k). \tag{16}$$

In order to minimize the error $\boldsymbol{u} = [u_1, u_1, \ldots, u_k]^{\mathrm{T}}$, the l_1 norm of $\boldsymbol{u}$ is added to (12) by introducing a trade-off parameter λ which can tune the influence of the prior data on the regression function. Therefore, by adding inequality constraints (16) and the l_1 norm of the slack vector, the MKLPSVR in (13) has been modified to reduce the influence of biased prior data from a simulator on the modeling accuracy. The modified algorithm called as MKPLPSVR in this paper is expressed as:

$$\begin{aligned}
&\text{Find: } \alpha_{ri}^+, \alpha_{ri}^-, \xi_i, u_k, b\\
&\text{Min: } \sum_{r=1}^{L} C_r \sum_{i=1}^{N} (\alpha_{ri}^+ + \alpha_{ri}^-) + 2C \sum_{i=1}^{N} \xi_i + \lambda \sum_{k=1}^{N_k} u_k\\
&\text{s.t.} \begin{cases}
y_i - \sum_{r=1}^{L} \sum_{j=1}^{N} (\alpha_{rj}^+ - \alpha_{rj}^-) k_r(\boldsymbol{x}_i, \boldsymbol{x}_j) - b \le \varepsilon + \xi_i\\
\sum_{r=1}^{L} \sum_{j=1}^{N} (\alpha_{rj}^+ - \alpha_{rj}^-) k_r(\boldsymbol{x}_i, \boldsymbol{x}_j) + b - y_i \le \varepsilon + \xi_i\\
y_k^p - \sum_{r=1}^{L} \sum_{j=1}^{N} (\alpha_{rj}^+ - \alpha_{rj}^-) k_r(z_k^p, \boldsymbol{x}_j) - b \le \varepsilon^p + u_k\\
\sum_{r=1}^{L} \sum_{j=1}^{N} (\alpha_{rj}^+ - \alpha_{rj}^-) k_r(z_k^p, \boldsymbol{x}_j) + b - y_k^p \le \varepsilon^p + u_k\\
\alpha_{rj}^+ \ge 0, \ \alpha_{rj}^- \ge 0 \quad (\forall i = 1, 2, \ldots, N)\\
\xi_i \ge 0, \ u_k \ge 0 \quad (\forall k = 1, 2, \ldots N_k).
\end{cases}
\end{aligned} \tag{17}$$

In order to facilitate the solution, we have reformulated the Eq. (17) in the following vector form:

$$\begin{aligned}
&\text{Find: } \boldsymbol{s}\\
&\text{Min: } \boldsymbol{h}^{\mathrm{T}}\boldsymbol{s}\\
&\text{s.t.} \begin{cases} \boldsymbol{A}\boldsymbol{s} \le \boldsymbol{B}\\ \boldsymbol{s} \ge \boldsymbol{l} \end{cases}
\end{aligned} \tag{18}$$

where

$$\begin{aligned}
&\boldsymbol{s} = \left[\boldsymbol{\alpha}_1^+, \ldots, \boldsymbol{\alpha}_L^+, \boldsymbol{\alpha}_1^-, \ldots, \boldsymbol{\alpha}_L^-, \boldsymbol{\xi}, \boldsymbol{u}, b\right]^{\mathrm{T}}\\
&\boldsymbol{h} = \left[C_1\boldsymbol{e}, \ldots C_L\boldsymbol{e}, C_1\boldsymbol{e}, \ldots C_L\boldsymbol{e}, 2C\boldsymbol{e}, \lambda\boldsymbol{e}^p, 0\right]^{\mathrm{T}}\\
&\boldsymbol{B} = \left[\varepsilon\boldsymbol{e} + \boldsymbol{y}, \varepsilon\boldsymbol{e} - \boldsymbol{y}, \varepsilon^p\boldsymbol{e} + \boldsymbol{y}^p, \varepsilon^p\boldsymbol{e} - \boldsymbol{y}^p\right]^{\mathrm{T}}\\
&\boldsymbol{l} = \left[\boldsymbol{o}_1, \ldots, \boldsymbol{o}_L, \boldsymbol{o}_1, \ldots, \boldsymbol{o}_L, \boldsymbol{o}, \boldsymbol{o}^p, -\infty\right]^{\mathrm{T}}\\
&\boldsymbol{A} = \begin{bmatrix}
\boldsymbol{K}_1, \ldots, \boldsymbol{K}_L, -\boldsymbol{K}_1, \ldots, -\boldsymbol{K}_L, -\boldsymbol{E}, \boldsymbol{Z}^k, \boldsymbol{e}\\
-\boldsymbol{K}_1, \ldots, -\boldsymbol{K}_L, \boldsymbol{K}_1, \ldots, \boldsymbol{K}_L, -\boldsymbol{E}, \boldsymbol{Z}^k, -\boldsymbol{e}\\
\boldsymbol{K}_1^p, \ldots, \boldsymbol{K}_L^p, -\boldsymbol{K}_1^p, \ldots, -\boldsymbol{K}_L^p, \boldsymbol{Z}^p, -\boldsymbol{E}^p, \boldsymbol{e}^p\\
-\boldsymbol{K}_1^p, \ldots, -\boldsymbol{K}_L^p, \boldsymbol{K}_1^p, \ldots, \boldsymbol{K}_L^p, \boldsymbol{Z}^p, -\boldsymbol{E}^p, -\boldsymbol{e}^p
\end{bmatrix}.
\end{aligned}$$

In the optimization, the slack vector $\boldsymbol{\xi}$ and $\boldsymbol{u}$ represent $\boldsymbol{\xi} = [\xi_1, \xi_2, \ldots, \xi_N]^{\mathrm{T}}$ and $\boldsymbol{u} = [u_1, u_1, \ldots, u_k]^{\mathrm{T}}$, respectively. The vector $\boldsymbol{\alpha}_r^+$ and $\boldsymbol{\alpha}_r^-$ denote $\boldsymbol{\alpha}_r^+ = \left[\alpha_{r1}^+, \alpha_{r2}^+, \ldots, \alpha_{rN}^+\right]^{\mathrm{T}}$ and $\boldsymbol{\alpha}_r^- = \left[\alpha_{r1}^-, \alpha_{r2}^-, \ldots, \alpha_{rN}^-\right]^{\mathrm{T}}$, respectively. The vector $\boldsymbol{e} = [1, 1, \ldots, 1]^{\mathrm{T}}$ and $\boldsymbol{o} = [0, 0, \ldots, 0]^{\mathrm{T}}$ denote $N \times 1$ column vector. The vector $\boldsymbol{e}^p = [1, 1, \ldots, 1]^{\mathrm{T}}$ and $\boldsymbol{o}^p = [0, 0, \ldots, 0]^{\mathrm{T}}$ denote $N_k \times 1$ column vector. The matrix $\boldsymbol{K}_r (r = 1, 2, \ldots, L)$ denotes a $N \times N$ kernel matrix calculated by the rth kernel function, and every element in the matrix is calculated by the rth kernel function $k_r(\boldsymbol{x}_i, \boldsymbol{x}_j)$. $\boldsymbol{E}$ is a $N \times N$ identity matrix. $\boldsymbol{E}^p$ denotes a $N_k \times N_k$

identity matrix, $\boldsymbol{Z}^k$ denotes a $N \times N_k$ zero matrix, and $\boldsymbol{Z}^p$ denotes a $N_k \times N$ zero matrix. $\boldsymbol{K}_r^p (r = 1, 2, \ldots, L)$ denotes a $N_k \times N$ kernel matrix calculated by the rth kernel function, and every element in the matrix is calculated by the predefined kernel function $k_r(\boldsymbol{z}_p^k, \boldsymbol{x}_j)$. The vector $\boldsymbol{y} \in \mathrm{R}^{N\times 1}$contains all training samples $y_i (i = 1, \ldots, N)$ as rows, and the vector $\boldsymbol{y}^p \in \mathrm{R}^{N_k\times 1}$contains all prior data samples $y_i^p (i = 1, \ldots, N_k)$ as rows.

Using linear programming to solve the formulation, one will obtain a regression function as shown in (11). The solution procedure is summarized below.

Algorithm: MKPLPSVR

1. Prepare a training dataset which includes a measured dataset $\{(\boldsymbol{x}_i, \boldsymbol{y}_i), i = 1,..., N_m\}$ from a fine model and a prior dataset $\left\{\left(\boldsymbol{z}_k^p, y_k^p\right), k = 1, 2, \cdots N_k\right\}$ from a simulator.
2. Determine the total number $N = N_m + N_k$ of the training dataset.
3. Define the number L and kernel parameters of the used kernel functions.
4. Select some hyper-parameters such as the parameters C, λ, ε and ε^p.
5. Calculate the $N \times N$ kernel matrix $\boldsymbol{K}_r (r = 1, 2, \cdots, L)$ from the training dataset.
6. Calculate the $N_k \times N$ kernel matrix $\boldsymbol{K}_r^p (r = 1, 2, \cdots, L)$ from the prior dataset
7. Construct the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{h}$, $\boldsymbol{B}$ and $\boldsymbol{l}$ in **(18)**.
8. Solve the optimization in **(18)** by using *linprog* algorithm.
9. Computer $\alpha_{ri} = \alpha_{ri}^+ - \alpha_{ri}^-$ by using $\boldsymbol{s}$ in **(18)** and obtain the function **(11)**.

3.3 Strategy for parameter selections in MKPLPSVR

In this section, a strategy has been presented to select some hyper-parameters in the MKPLPSVR. Figure 2 shows the flowchart of parameter selections for the MKPLPSVR.

In the Fig. 2, we suppose that the kernel function is firstly given beforehand according to some experiences. Generally, the number $L = 2, 3$ of the kernel function may be sufficient for dealing with most of practical problems (Pozdnoukhov and Kanevski 2008). The error threshold ε and ε^p are proportional to the noise level of training dataset, and the empirical tuning from Cherkassky and Yunqian (2004) has been applied to determine the ε and ε^p which are expressed as:

$$\begin{aligned} \varepsilon &= 3\sigma\sqrt{\ln N/N} \\ \varepsilon^p &= 3\sigma^p\sqrt{\ln N_k/N_k} \end{aligned} \tag{19}$$

where the standard deviation σ and σ^p are estimated from N training dataset and N_k prior dataset, respectively.

The constant C_r penalizes the non-zero variables $\alpha_{ri}^+, \alpha_{ri}^-$ in the MKPLPSVR. To avoid over-fitting, this paper has chosen $C_r = 1/\delta_r$ where δ_r denotes the kernel parameter of the rth kernel function. The remaining parameters including the rth kernel parameter δ_r and the hyper-parameters C and λ will be searched by using fivefold cross-validation (Phienthrakul and Kijsirikul 2010; Pasolli et al. 2012; Huang 2012). The searching procedure is described as below:

1. Specify the search range of C, λ and δ_r.
2. Initialize $C = \hat{C}$, $\lambda = \hat{\lambda}$ and $\delta_r = \hat{\delta}_r$.
3. Divide the training dataset into five subsets.
4. Choose four subsets to train the MKPLPSVR, and apply the remaining subset to evaluate the model accuracy (cross-validation error, test error, etc.).
5. Search C, λ and δ_r by using optimization algorithm such as PSO or GA.
6. Evaluate the model accuracy whether it satisfies a predefined terminal condition or not.
7. Update the search direction and go to step 3.
8. Obtain the optimal hyper-parameters C, λ and δ_r.

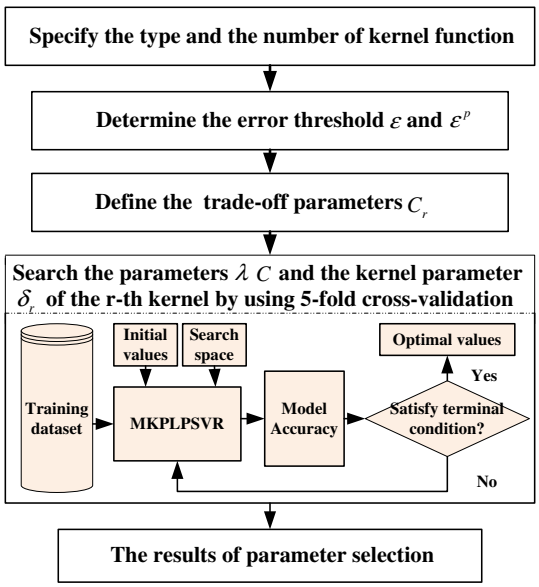

Fig. 2 Flowchart of parameter selections procedure

4 Experimental results

In this section, we will validate the proposed algorithm by using a synthetic example, a microstrip antenna and six-pole microwave filter. Moreover, the following two criteria are used to evaluate the generalization performance.

$$\mathrm{RMSE} = \sqrt{N^{-1}\sum_{i=1}^{N}(y_i - f(\boldsymbol{x}_i))^2} \tag{20}$$

$$\mathrm{MAE} = \max(|y_i - f(\boldsymbol{x}_i)|) \tag{21}$$

where $f(\boldsymbol{x}_i)$ is the predicted value, y_i is the corresponding measured value, N is the number of testing samples. RMSE denotes the root mean squared error, and MAE denotes the maximum absolute error.

4.1 Complex function approximation

In the subsection, we will validate the proposed algorithm by a synthetic example, and five different algorithms have been employed to approximate the following function.

$$y = \begin{cases} -4x - 8, & -3 \le x < -1 \\ -3x^3 - 5x^2 + 5x + 3, & -1 \le x < 1 \\ 2\sin(\exp(1.2x)) + 0.3552, & 1 \le x \le 3. \end{cases} \tag{22}$$

From the range $[-3, 3]$ of the function above, we have generated 13 training data by adding a Gaussian noise $N(0, 0.1^2)$. Then, in order to simulate the prior knowledge, we have applied the same function to generate 35 data samples with a Gaussian noise $N(0, 0.2^2)$ and taken them as the prior data. Finally, 201 data points are also taken uniformly from the same function as a testing data. Figure 3 shows the testing data, the training data and the prior data.

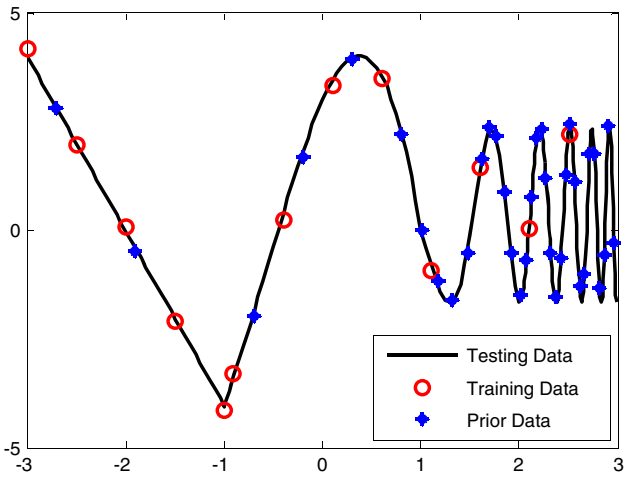

Fig. 3 Data samples

Utilizing the data, we will approximate the function separately by using the LPSVR, the MKLPSVR and the MKPLPSVR. In addition, the SimpleMKL in (Rakotomamonjy et al. 2008) and the PLPSVR in Zhou and Huang (2010) have been used to compare their performance with the MKPLPSVR. The strategy in Sect. 3.3 has been applied to determine the hyper-parameters in the MKPLPSVR.

In order to validate the proposed algorithm, we have designed two groups of experiments. In the first group of experiment, 13 training data will used to develop a regression function, and 35 data samples from the prior knowledge are only used during the course of calculating the constraints of the optimization formulation. In the second group of experiment, we will utilize the 35 prior data samples to extend the 13 training data samples, and then apply the extended data samples to develop a regression function. After obtaining a regression model, we will verify the model by using the 201 testing data.

In the first group of experiment, we have chosen $C = 100$, $\varepsilon = 0.01$, $\varepsilon^p = 0.04$ for all the algorithms. Both the LPSVR and the PLPSVR exploited only a Gaussian kernel with the kernel parameter $\sigma = 0.0803$. However, the MKLPSVR, the SimpleMKL and the MKPLPSVR employed a Gaussian kernel, a polynomial kernel and a wavelet kernel (Lu et al. 2009) with the kernel parameters 2, 0.058 and 0.0125, respectively. In the second group, we have chosen $C = 150$, $\varepsilon = 0.01$, $\varepsilon^p = 0.04$ for all the algorithms. Similarly, both the LPSVR and the PLPSVR only exploited a Gaussian kernel with the kernel parameter $\sigma = 0.013$. The MKLPSVR, the SimpleMKL and the MKPLPSVR utilized a Gaussian kernel, a polynomial kernel and a wavelet kernel with the kernel parameters 2, 0.055 and 0.0122, respectively.

Utilizing the data samples and parameters above, we will establish the models separately by using five algorithms. Figure 4 shows the approximating results in the first group of experiment. Figure 4 shows that all of the algorithms cannot accurately approximate the steep variation of the actual function, due to only the 13 training samples. However, compared with Fig. 4a and b, we can find that the multi-kernel algorithms such as the MKLPSVR, the MKPLPSVR and the SimpleMKL are able to more accurately approximate the flat variation than other algorithms.

In order to clearly show the performance, we have presented some statistical results verified by 201 testing data in Table 1. From Table 1, we can also find that the number of support vector (NSV) is almost the same among the functions. However, the model calculated by the MKPLPSVR has the smallest RMSE and MAE among the five models.

Figure 5 shows the approximating results in the second group of experiment. From Fig. 5a, we can find that all of the algorithms can approximate the steep variation of the function. However, the results from Fig. 5b show that other algorithms besides the MKPLPSVR and the MKLPSVR cannot

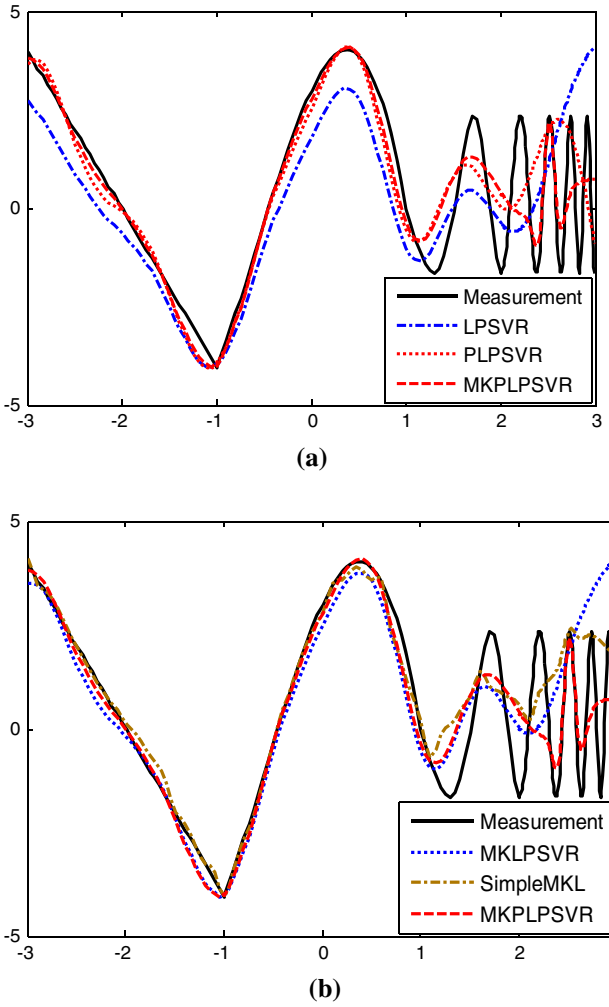

Fig. 4 Comparison of predicted results in the first group of experiment. **a** Results of LPSVR, PLPSVR, MKPLPSVR and measurement. **b** Results of MKLPSVR, MKPLPSVR, SimpleMKL and measurement

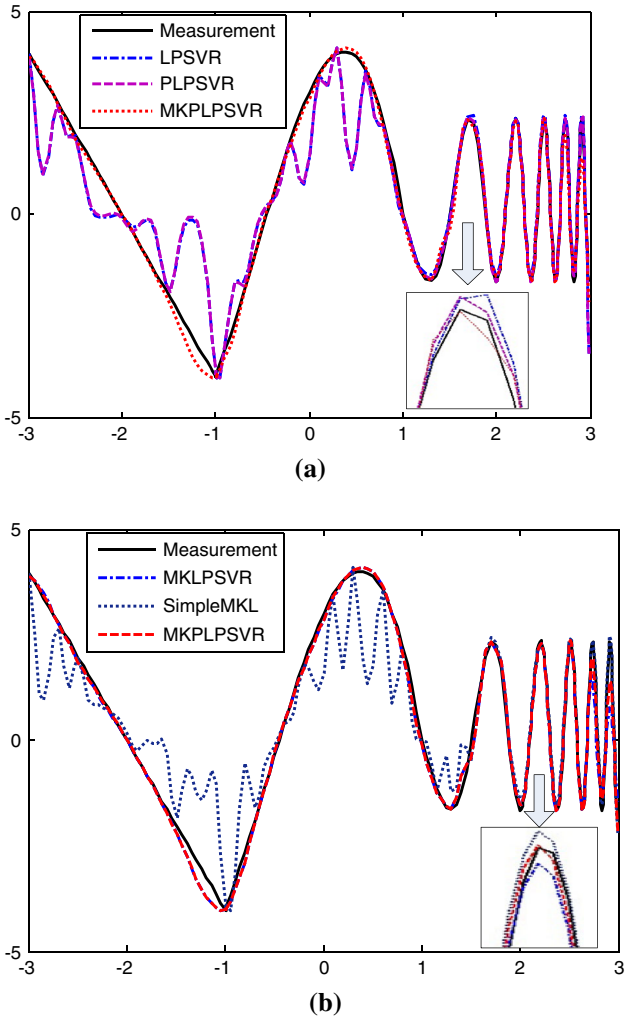

Fig. 5 Comparison of predicted results in the second group of experiment. **a** Results of LPSVR, PLPSVR, MKPLPSVR and measurement. **b** Results of MKLPSVR, MKPLPSVR, SimpleMKL and measurement

Table 1 Errors and number of support vector

Algorithm	NSV	RMSE	MAE
LPSVR	**11**	1.848857	5.735477
PLPSVR	12	1.328779	3.880107
MKLPSVR	12	1.845337	5.735477
SimpleMKL	12	1.4378	3.8105
MKPLPSVR	12	**0.972362**	**2.486516**

Bold values indicate the best ones

Table 2 Errors and number of support vector

Algorithm	NSV	RMSE	MAE
LPSVR	**37**	0.737101	2.988316
PLPSVR	**37**	0.736426	2.053115
MKLPSVR	**37**	0.325079	1.573869
SimpleMKL	42	0.7522	2.7198
MKPLPSVR	**37**	**0.212162**	**0.955358**

Bold values indicate the best ones

accurately approximate the flat variation. A possible explanation is that incorporating multi-kernel into the LPSVR has improved the accuracy of approximating a function with both the steep variation and smooth variation.

Table 2 shows some statistical results of the five regression functions separately calculated by 201 testing data. From Table 2, we can find that the function developed by the MKPLPSVR is the most accurate among all of the functions. Compared with the SimpleMKL, the MKPLPSVR is also advantageous over the SimpleMKL in the aspects of model sparsity and generation performance. Compared with Tables 1 and 2, we find that the function developed in the second group of experiment is more accurate than the one developed in the first group of experiment. The reason is that the prior data have been utilized to extend the few training data samples.

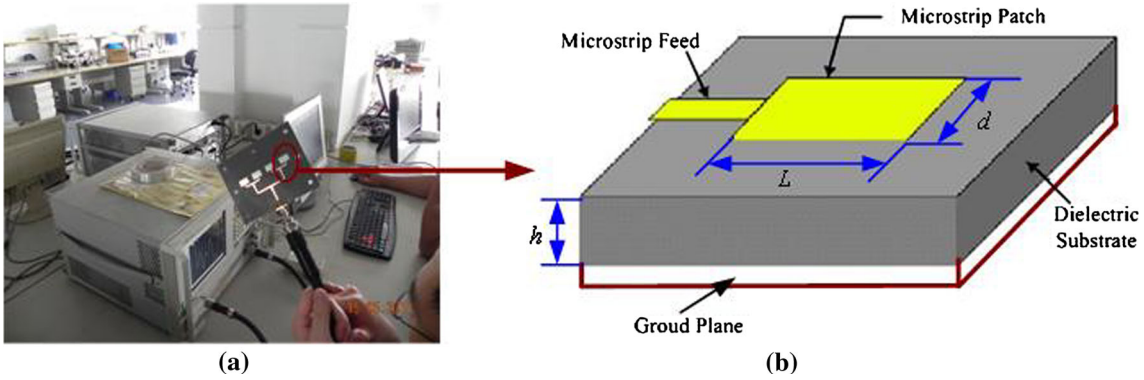

Fig. 6 Experimental devices and structural sketch for microstrip antennas. **a** Experimental devices. **b** Structural sketch of microstrip antennas

From these comparing results, we can find that the function approximated by the MKPLPSVR is the most accurate among all of the functions when few measurements are available. The MKPLPSVR is more effective to approximate a complex function with both steep and smooth variations than the LPSVR or PLPSVR with a simple kernel. The reason is that incorporating a multi-kernel into the LPSVR has improved the accuracy of approximating a complex function. In terms of the generation performance, the MKPLPSVR is also superior to the multi-kernel algorithms such as the SimpleMKL and the MKLPSVR. A possible explanation is that the MKPLPSVR has utilized the prior data to extend the few training data samples and improve the modeling accuracy. It follows that the introduction of prior knowledge and multi-kernel functions into the framework of the LPSVR can improve the modeling accuracy for a small dataset.

4.2 Bandwidth calculation in microstrip antennas

During the course of designing a microstrip antenna, the antenna bandwidth can be calculated by empirical formulas or numerical techniques. Although empirical formulas can facilitate the design, its accuracy is limited. The numerical techniques based on the electromagnetic theory can obtain an accurate result. However, its solution is relatively time-consuming. In order to accurately predict the bandwidth and reduce the computational effort, the subsection will apply the proposed algorithm to build a hybrid model of bandwidth calculation which can be integrated into a microwave CAD tool.

Consider a rectangular patch of width d and length L over a ground plane with a substrate thickness h and a substrate dielectric permittivity ε_r, as shown in Fig. 6. The rectangular antenna bandwidth $\mathrm{BW}_{\mathrm{exp}}$ can be evaluated by using the empirical formula from Sagiroglu et al. (1999):

$$\mathrm{BW}_{\mathrm{exp}} = \left[89\left(\frac{hd}{\varepsilon_r \lambda_0^2}\right)^{0.45} + 91\left(\frac{h}{\lambda_0}\right) \right] \% \tag{23}$$

where λ_0 is the free space wavelength at the resonant frequency f_r. The dielectric permittivity $\varepsilon_r = \left(\frac{c}{f_r \lambda_d}\right)^2$ is related to the dielectric loss tangent $\tan\delta$, and c is the velocity of electromagnetic waves in free space, and λ_d is the wavelength in the dielectric substrate.

The empirical formula can be used to calculate the bandwidth quickly, but the results are not in agreement with the experimental results. This paper has presented a hybrid model for antenna bandwidth, and the hybrid model is expressed as:

$$\Delta\mathrm{BW} = h_s(\boldsymbol{x}) \tag{24}$$

$$\mathrm{BW} = \mathrm{BW}_{\mathrm{exp}} + \Delta\mathrm{BW} \tag{25}$$

Equation (24) is a support-vector model which corrects the difference between the empirical formula and the experimental results. The structural parameters $h, d, \varepsilon_r, \lambda_0, f_r$ and $\tan\delta$ will influence the bandwidth BW. However, the research in the literature (Sagiroglu et al. 1999) shows that the bandwidth depends on three independent variables $\boldsymbol{x} = [h/\lambda_d, d, \tan\delta]^{\mathrm{T}}$. From the literature in Sagiroglu et al. (1999), 27 measured data have been obtained and taken as the measured dataset $\mathrm{S} = \left\{(\boldsymbol{x}, \Delta BW), \boldsymbol{x} \in \mathrm{R}^{27\times 3}, \Delta BW \in \mathrm{R}^{27\times 1}\right\}$. In order to improve the modeling accuracy, a calibrated electromagnetic simulator has been utilized to generate 8 prior data shown in Table 3.

Based on the dataset, this section will establish the model separately by using the LPSVR, the MKLPSVR and the MKPLPSVR. Moreover, the SimpleMKL in Rakotomamonjy et al. (2008) and the PLPSVR in Zhou et al. (2010) have also been utilized to establish the model. Two groups of experiments were designed to validate the model. In the first group, 27 training samples will be used to establish the model; meanwhile, 8 data samples from the prior knowledge will be only used for the calculation of the constraints in MKPLPSVR. In the second group, we have extended the 27 training data with 8 prior data, and then applied the extended data to establish the model.

Table 3 Data samples from prior knowledge

No.	h/λ_d	d (mm)	$\tan\delta$	BW (%)
1	0.0085	8.5	0.001	1.16
2	0.2148	10.8	0.002	21.13
3	0.1519	7.9	0.002	18.2
4	0.066	13.37	0.002	7.81
5	0.0384	18.1	0.001	4.87
6	0.1976	10.2	0.002	20.45
7	0.1263	9.05	0.002	15.64
8	0.0843	15.32	0.002	10.16

In the first group, we have chosen the hyper-parameters $C = 100$, $\varepsilon = 0.01$, $\varepsilon^p = 0.001$ for all the algorithms. Both the LPSVR and the PLPSVR exploited a Gaussian kernel with the kernel parameter $\sigma = 0.0803$, and the MKLPSVR, the SimpleMKL and the MKPLPSVR exploited a Gaussian kernel and a polynomial kernel with the kernel parameters 0.0803 and 2, respectively. In the second group, we have chosen the hyper-parameters $C = 150$, $\varepsilon = 0.01$, $\varepsilon^p = 0.001$. Both the LPSVR and the PLPSVR used a Gaussian kernel with the kernel parameter $\sigma = 0.09$, and the MKLPSVR, the SimpleMKL and the MKPLPSVR exploited a Gaussian kernel and a polynomial kernel with the kernel parameters 0.09 and 0.155, respectively.

Using the data samples and the parameters above, different hybrid models were developed by using five algorithms. Finally, six testing data samples have been applied to verify the models. Tables 4 and 5 give some comparisons between the measured results and the predicted ones. Table 6 presents the number of support vectors and the statistic errors calculated by using the same testing data in two groups of experiments. From Tables 4 and 5, we can find that the results predicted by the MKPLPSVR are in very good agreement with the measurements among the five algorithms.

As seen in Table 6, the model developed by the MKPLPSVR is more accurate than the ones developed by other algorithms in the first group. Similarly, the model developed by the MKPLPSVR in the second group is also more accurate than the ones developed by other algorithms. Moreover, the MKPLPSVR uses the fewest number of support vector among the algorithms. The results indicate that the MKPLPSVR can improve the modeling accuracy in the case of the scarcity of measurement data available.

Compared with two groups of experiments, we find that the results calculated in the second group are more accurate than the ones in the first group, due to the incorporation of the prior knowledge in the second group. Moreover, the model developed by the MKPLPSVR in the second group is more accurate than the result BW_{EDBD} from the literature (Sagiroglu et al. 1999). From the comparisons, we can conclude that it is effective to improve the modeling accuracy from an insufficient amount of measurement data by simultaneously incorporating prior knowledge and multiple kernels into the framework of the LPSVR.

4.3 Application in a microwave filter tuning device

In the subsection, the proposed algorithm has been applied to develop a model which is particularly suited to an automatic tuning device for microwave filters (Zhou et al. 2010; Zhou and Huang 2013). A six-pole microwave filter with center frequency of 1,810 MHz, bandwidth of 10 MHz is tuned

Table 4 Comparisons of predicted results in the first group of experiment

No.	Measured BW (%)	LPSVR	PLPSVR	MKLPSVR	SimpleMKL	MKPLPSVR
1	4.9	6.616262	6.535426	4.309119	4.186733	4.786704
2	7.7	7.311341	7.374298	7.626643	7.652719	7.763086
3	16	14.81919	15.43764	15.4274	15.6498	15.50704
4	18.2	18.20299	18.02852	18.22053	18.4185	18.04538
5	20.3	19.29236	19.81466	20.21762	20.09769	20.10524
6	21.2	20.81712	20.90465	21.46231	21.17879	21.25276

Table 5 Comparisons of predicted results in the second group of experiment

No.	Measured BW (%)	LPSVR	PLPSVR	MKLPSVR	SimpleMKL	MKPLPSVR
1	4.9	4.796841	4.988748	4.53827	4.500532	4.873588
2	7.7	7.548884	7.493423	7.535338	7.540162	7.640877
3	16	15.49525	15.68391	15.94323	15.73599	15.93453
4	18.2	18.2219	18.21511	18.32455	18.49765	18.28414
5	20.3	20.23177	20.0456	20.45567	20.41423	20.54020
6	21.2	20.99816	21.13577	21.32475	21.32602	21.26145

Table 6 Statistic errors and number of support vector

	Algorithm	NSV	RMSE	MAE
Group 1	LPSVR	25	0.970641	1.716262
	PLPSVR	26	0.758193	1.635426
	MKLPSVR	**22**	0.355529	0.590881
	SimpleMKL	**22**	0.347100	0.713300
	MKPLPSVR	23	**0.232541**	**0.492964**
Group 2	LPSVR	28	0.235982	0.50475
	PLPSVR	27	0.191284	0.316086
	MKLPSVR	**21**	0.189954	0.36173
	SimpleMKL	23	0.24910	0.39950
	MKPLPSVR	**21**	**0.113308**	**0.210204**
Sagiroglu et al. (1999)	BW_{EDBD}	–	0.18940	0.30600

Bold values indicate the best ones

by an automatic tuning device, as shown in Fig. 7. In this example, we will develop a filter-tuning model which reveals the influence of the inserting depth of the six tunable screws on the filter response.

In order to formulate the problem, we assume that the inserting depth of the six tunable screws at the benchmark, is $\boldsymbol{L}_0 = [t_1, t_2, ..., t_6]^{\mathrm{T}}$, where the ideal coupling matrix $\boldsymbol{M_0}$ can be obtained from the initial design stage of the microwave filter. When the filter is adjusted, the tunable screws will be rotated $\Delta \boldsymbol{D} = [\Delta d_1, \Delta d_2, ..., \Delta d_6]^{\mathrm{T}}$ degree which makes the inserting depth of tunable screws alter $\Delta \boldsymbol{L} = \Delta \boldsymbol{D}^{\mathrm{T}} \boldsymbol{R}$ with the given thread pitch $\boldsymbol{R}$ of tunable screws. As a result, the actual coupling matrix $\boldsymbol{M}$ of the filter has a changing amount $\Delta \boldsymbol{M}$, and the filter response is also affected. In order to formulate the influence, we assume that there is a mapping between the inserting depth (or the rotating degrees) of the six tunable screws and the changing amount $\Delta \boldsymbol{M}$ of the coupling matrix (Zhou et al. 2010; Zhou and Huang 2013), and it can be formulated as:

$$\Delta \boldsymbol{M} = \boldsymbol{f}(\Delta \boldsymbol{D}). \tag{26}$$

If the formulation in (26) has been obtained, the actual coupling matrix $\boldsymbol{M}$ can be expressed as:

$$\boldsymbol{M} = \boldsymbol{M_0} + \Delta \boldsymbol{M}. \tag{27}$$

According to the topology of the filter in Fig. 7a, the coupling matrix $\boldsymbol{M}$ is expressed as:

$$\boldsymbol{M} = \begin{bmatrix} m_{11} & m_{12} & 0 & 0 & 0 & 0 \\ m_{12} & m_{22} & m_{23} & 0 & m_{25} & 0 \\ 0 & m_{23} & m_{33} & m_{34} & 0 & 0 \\ 0 & 0 & m_{34} & m_{44} & m_{45} & 0 \\ 0 & m_{25} & 0 & m_{45} & m_{55} & m_{56} \\ 0 & 0 & 0 & 0 & m_{56} & m_{66} \end{bmatrix}. \tag{28}$$

In this example, the ideal coupling matrix $\boldsymbol{M}_0$ in the benchmark is determined at the initial design stage by using the synthesis of microwave filters according to some predefined specification (Zhou et al. 2010; Zhou and Huang 2013), and it is given in the followings.

$$\boldsymbol{M}_0 = \begin{bmatrix} 0.0001 & 0.8720 & 0 & 0 & 0 & 0 \\ 0.8720 & 0.0001 & 0.6048 & 0 & 0.109 & 0 \\ 0 & 0.6048 & 0.0008 & -0.6781 & 0 & 0 \\ 0 & 0 & -0.6781 & -0.0032 & 0.6048 & 0 \\ 0 & 0.109 & 0 & 0.6408 & 0.001 & 0.872 \\ 0 & 0 & 0 & 0 & 0.872 & 0.001 \end{bmatrix}. \tag{29}$$

Following the analysis in Zhou and Huang (2013); Zhou et al. (2010), the filter response is a function of the coupling matrix M, and it is expressed as:

$$\begin{aligned} S_{21}(f) &= -2\mathrm{j}\sqrt{R_1 R_2}\left[\left(\frac{f_0}{BW}\left(\frac{f}{f_0} - \frac{f_0}{f}\right)\boldsymbol{I} - \mathrm{j}\boldsymbol{R} + \boldsymbol{M}\right)^{-1}\right]_{21} \\ S_{11}(f) &= 1 + 2\mathrm{j}\sqrt{R_1}\left[\left(\frac{f_0}{BW}\left(\frac{f}{f_0} - \frac{f_0}{f}\right)\boldsymbol{I} - \mathrm{j}\boldsymbol{R} + \boldsymbol{M}\right)^{-1}\right]_{11} \end{aligned} \tag{30}$$

where the scattering parameters $S_{21}(f)$ and $S_{11}(f)$ denote the transfer and reflection characteristics of the filter response, respectively. $\boldsymbol{I}$ is a unity matrix. $\boldsymbol{R}$ is a diagonal matrix with all elements equal to zero except $R_{11} = R_1$, $R_{nn} = R_2$ and $R_{ii} = \frac{f_0}{BW \cdot Q}$ $(1 < i < n)$. The operational frequency is denoted by the variable f. As for the filter, the unload $Q = 3804$, the desired central frequency $f_0 = 1810$, and the desired bandwidth $BW = 10$ MHz are given at the design stage of the filter. Both input coupling R_1 and output coupling R_2 are equal to 1.043 in this example.

From the formulation above, the key of developing a filter-tuning model is to establish the relationship in (26).

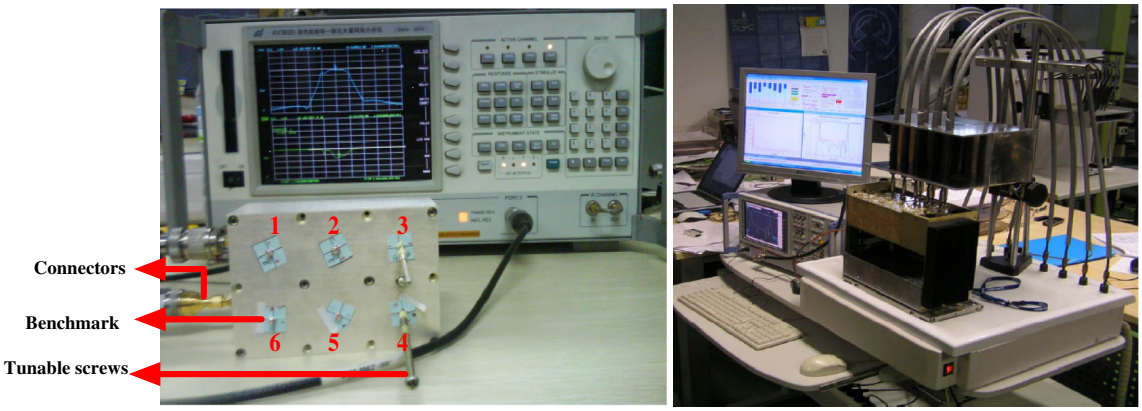

Fig. 7 Experimental system for filter-tuning devices. **a** Six-pole microwave filter. **b** Automated tuning device

In this example, we will build the relationship by using different algorithms. Firstly, some training data were required. According to the data acquisition method from the literature (Zhou et al. 2010), 45 measured data have been collected by a skilled operator. The number of the measured data are too small to obtain a satisfactory model. Moreover, the measurement is costly. Therefore, 30 prior data from a calibrated simulator have been used to make up a small amount of measured data. Finally, we have obtained a training data set $S = \{(\Delta \boldsymbol{D}, \Delta \boldsymbol{M}), \Delta \boldsymbol{D} \in R^{75\times 6}, \Delta \boldsymbol{M} \in R^{75\times 12}\}$ which consists of a measured data set $M = \{(\Delta \boldsymbol{D}, \Delta \boldsymbol{M}), \Delta \boldsymbol{D} \in R^{45\times 6}, \Delta \boldsymbol{M} \in R^{45\times 12}\}$ and a prior data set $P = \{(\Delta \boldsymbol{D}^{p}, \Delta \boldsymbol{M}^{p}), \Delta \boldsymbol{D}^{p} \in R^{30\times 6}, \Delta M^{p} \in R^{30\times 12}\}$.

Based on the dataset, this section established the Eq. (26) separately by using the LPSVR, the MKLPSVR and the MKPLPSVR, and the SimpleMKL in Rakotomamonjy et al. (2008) and the PLPSVR in Zhou and Huang (2010). Because the algorithms can only obtain a multi-input and one-output function, this example has applied the meta-model method in Zhou et al. (2010); Zhou and Huang (2013) to build the multi-input and multi-output model shown in Eq. (26). Every meta-model was established independently by the algorithms, and then all of the meta-models were combined into the coupling matrix in (28).

Two groups of experiments, which are the same as the previous two examples, were designed to validate the proposed algorithm. In the first group of experiments, both the LPSVR and the PLPSVR exploited a Gaussian kernel with the kernel parameter $\sigma = 0.0803$. The MKLPSVR, the SimpleMKL, and the MKPLPSVR used a Gaussian kernel and a polynomial kernel with the kernel parameters 0.1 and 1.2, respectively. Other hyper-parameters such as $C = 150$, $\lambda = 100$, $\varepsilon = 0.08$, and $\varepsilon^{p} = 0.05$ were used. In the second group, both the LPSVR and the PLPSVR used a Gaussian kernel with the kernel parameter $\sigma = 0.22$. The MKLPSVR, the SimpleMKL and the MKPLPSVR exploited a Gaussian kernel and a polynomial kernel with the kernel parameters 0.22 and 1.2, respectively. Other hyper-parameters such as $C = 135$, $\lambda = 100$, $\varepsilon = 0.05$, and $\varepsilon^{p} = 0.03$ were used. Once the relationship in (26) was developed separately by using the five algorithms, 10 testing data samples were applied to validate them. Tables 7 and 8 present the maximum absolute errors predicted by the five algorithms in the two groups of experiments. Figure 8 shows their statistical average root mean square error predicted by the five algorithms.

As seen in Tables 7 and 8, the maximum absolute error predicted by the MKPLPSVR is smaller than the ones predicted by other algorithms. Moreover, the results in the second group of experiment are more accurate than the ones in the first group of experiment, and the reason is that the prior data are utilized to extend the few training data. Figure 8 also clearly shows the results predicted by the MKPLPSVR are the most accurate among all of the algorithms. Compared with the MKLPSVR and the PLPSVR, the MKPLPSVR shows a better performance. The reason is that the prior knowledge and multi-kernel have been simultaneously incorporated into the LPSVR. On the contrary, the MKLPSVR and the PLPSVR have separately used multiple kernel or prior knowledge in the LPSVR. Compared with the SimpleMKL and the MKLPSVR, the MKPLPSVR is superior to the SimpleMKL in terms of the generation performance. A possible explanation is that the MKPLPSVR has utilized the prior data from a calibrated simulator to extend the few training data samples and improve the data-based modeling accuracy. It follows that the introduction of prior knowledge from a calibrated simulator and multi-kernel into the LPSVR can improve the data-based modeling accuracy when the amount of measured data are scarce.

The electrical performance was also evaluated by combining the Eqs. (27) and (30). Figure 9 presents a group of

Table 7 Maximum absolute error in the first group of experiment

	LPSVR	PLPSVR	MKLPSVR	SimpleMKL	MKPLPSVR
Δm_{12}	0.077143	0.063357	0.102968	0.073364	**0.040525**
Δm_{23}	0.099318	0.090955	0.092435	0.087449	**0.054519**
Δm_{34}	0.127449	0.116274	0.097622	0.108876	**0. 088329**
Δm_{45}	0.099116	**0.071106**	0.110378	0.083556	0.078589
Δm_{56}	0.037533	0.029308	0.032982	0.033988	**0.028326**
Δm_{25}	0.031512	0.028793	0.041770	0.026224	**0.014317**
Δm_{11}	0.020497	0.030887	0.033800	0.018054	**0.012462**
Δm_{22}	0.008779	0.009981	0.016872	0.004182	**0.004123**
Δm_{33}	0.007089	0.005091	**0.003963**	0.008225	0.004944
Δm_{44}	0.025286	0.036175	0.031635	0.021927	**0.017730**
Δm_{55}	0.002369	0.003478	0.003575	0.004758	**0.001594**
Δm_{66}	0.008937	0.013951	0.011308	0.012313	**0.008125**

Bold values indicate the best ones

Table 8 Maximum absolute error in the second group of experiment

	LPSVR	PLPSVR	MKLPSVR	SimpleMKL	MKPLPSVR
Δm_{12}	0.047693	0.035111	0.041286	0.042148	**0.033853**
Δm_{23}	0.049001	0.039814	0.022129	0.041325	**0.017476**
Δm_{34}	0.017806	0.011423	0.01433	0.015428	**0.006774**
Δm_{45}	0.050443	0.04122	0.033502	0.041335	**0.022978**
Δm_{56}	0.017830	0.013941	0.018179	0.014061	**0.012134**
Δm_{25}	0.020355	0.018554	0.017881	0.018356	**0.010986**
Δm_{11}	0.010713	0.015503	0.012193	0.010539	**0.009933**
Δm_{22}	0.001937	0.007174	0.001857	0.002437	**0.001647**
Δm_{33}	0.002445	0.001999	0.004728	0.002423	**0.001779**
Δm_{44}	0.009433	0.008468	0.006295	0.009263	**0.004362**
Δm_{55}	0.002329	0.010767	0.00393	0.002689	**0.001586**
Δm_{66}	0.003854	0.002508	0.007496	0.002267	**0.001741**

Bold values indicate the best ones

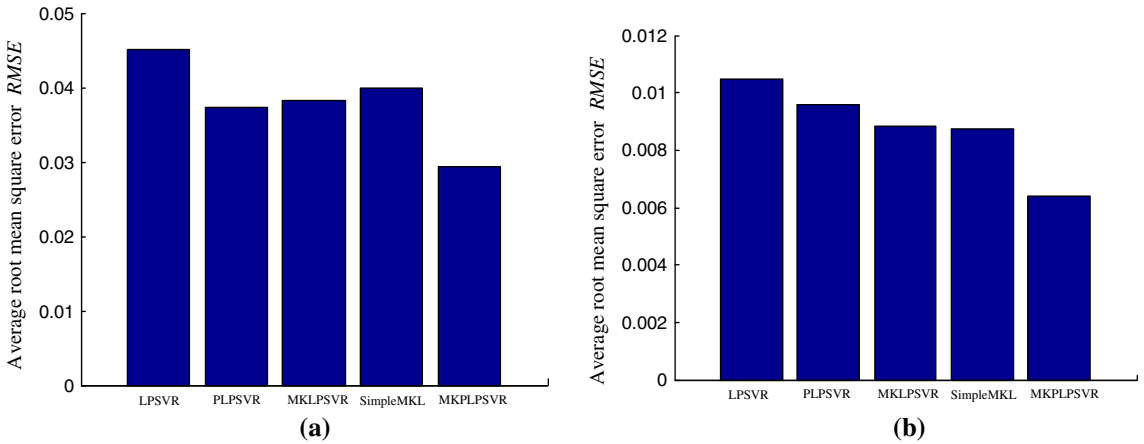

Fig. 8 Average root mean square error. **a** The first group of experiment, **b** the second group of experiment

comparing result between the five predicted responses and the measurement result.

The results in the Fig. 9 show that the model developed by the MKPLPSVR is much closer to the measurements than other ones. Comparing the transfer characteristics with the reflection characteristics, we find that the transfer characteristics calculated by MKPLPSVR is much closer to the measurement than the reflection characteristics calculated by MKPLPSVR. From the comparing results, we can find that the proposed MKPLPSVR is effective in solv-

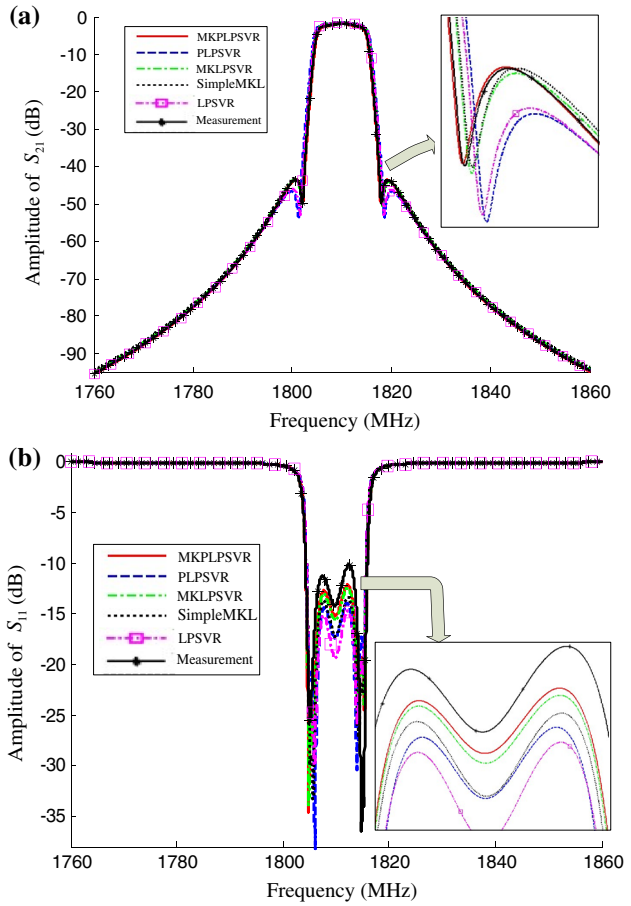

Fig. 9 Comparison of the electrical performance. **a** Transfer characteristics, **b** reflection characteristics

ing the modeling problem from a small amount of measured data.

5 Conclusions

In order to obtain an accurate model from an insufficient amount of measurement data, this paper has presented multi-kernel linear program support vector regression with prior knowledge. In the algorithm, multiple feature spaces have been utilized to incorporate multi-kernel functions into the framework of the LPSVR, and then the prior knowledge from a physical simulator has also been incorporated into the framework of the LPSVR by modifying optimization objectives and inequality constraints. At the end, prior knowledge and multi-kernel functions have been simultaneously incorporated into the framework of LPSVR. In addition, a strategy of parameter selections has been presented to facilitate the practical application of the proposed MKPLPSVR algorithm. Some experiments from a synthetic example, a microstrip antenna and a six-pole microwave filter have been carried out, and the experimental results show that the model developed by MKPLPSVR is more accurate than the ones developed by the other algorithms.

The proposed MKPLPSVR provides an approach to solve the scarcity of measured data in practice, and the MKPLPSVR shows great potential in some problems where a sufficient measurement data is difficult and costly to obtain, but the prior knowledge data from a physical simulator is available. The proposed MKPLPSVR algorithm can apply to the field of computer-aided modeling and system identification. By incorporating prior knowledge into the MKLPSVR, one can reduce the effect of the biased data from a calibrated physical simulator on the modeling accuracy. Although this paper focuses on the regression from a limited amount of measured data, the same technique can be applied to the classification problem in the case of the scarcity of measurement data, if the prior data from a physical simulator are available. Possible future extension is to solve the problem of model selection for the proposed algorithm, which is very significant to find an efficient method to automatically determine the type, the number of kernel functions and the hyper-parameters of support vector regression.

Acknowledgments This work was supported by National Natural Science Foundation of China (Grant No. 51305323, 51305322 and 51035006) and the Fundamental Research Funds for the Central Universities (Grant No JB140404).

References

Bach FR, Lanckriet GRG, Jordan MI Multiple kernel learning, conic duality, and the SMO algorithm. In: Banff, Alta, Canada, 2004. Proceedings, Twenty-First International conference on machine learning, ICML 2004. Association for Computing Machinery, pp 41–48

Bloch G, Lauer F, Colin G, Chamaillard Y (2008) Support vector regression from simulation data and few experimental samples. Inf Sci 178(20):3813–3827

Cherkassky V, Yunqian M (2004) Practical selection of SVM parameters and noise estimation for SVM regression. Neural Netw 17:113–126

Clarke SM, Griebsch JH, Simpson TW (2005) Analysis of support vector regression for approximation of complex engineering analyses. Trans ASME J Mech Design 127(6):1077–1087

Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines. Cambridge University Press, Cambridge

Huang CF (2012) A hybrid stock selection model using genetic algorithms and support vector regression. Appl Soft Comput 12(2):807–818

Lanckriet GRG, Cristianini N, Bartlett P, El Ghaoui L, Jordan MI (2004) Learning the kernel matrix with semidefinite programming. J Mach Learn Res 5:27–72

Lauer F, Bloch G (2008a) Incorporating prior knowledge in support vector machines for classification: a review. Neurocomputing 71(7–9):1578–1594

Lauer F, Bloch G (2008b) Incorporating prior knowledge in support vector regression. Mach Learn 70(1):89–118

Lu Z, Sun J (2009) Non-Mercer hybrid kernel for linear programming support vector regression in nonlinear systems identification. Appl Soft Comput 9(1):94–99

Lu Z, Sun J, Butts KR (2009) Linear programming support vector regression with wavelet kernel: a new approach to nonlinear dynamical systems identification. Math Comput Simul 79(7):2051–2063

Mangasarian OL, Wild EW (2007) Nonlinear knowledge in kernel approximation. IEEE Trans Neural Netw 18(1):300–306

Mingqing Hu, Chen Yiqiang (2009) Building sparse multiple-kernel SVM classifiers. IEEE Trans Neural Netw 20(5):827–839

Muller KR, Mika S, Ratsch G, Tsuda K, Scholkopf B (2001) An introduction to kernel-based learning algorithms. IEEE Trans Neural Netw 12(2):181–201

Nguyen C-V, Tay DBH (2008) Regression using multikernel and semiparametric support vector algorithms. IEEE Signal Process Lett 15:481–484

Mangasarian Olvi L, Shavlik Jude W (2004) Knowledge-based kernel approximation. J Mach Learn Res 5:1127–1141

Pasolli L, Notarnicola C, Bruzzone L (2012) Multi-objective parameter optimization in support vector regression: general formulation and application to the retrieval of soil moisture from remote sensing data. IEEE J Select Topics Appl Earth Observ Remote Sens 5(5):1495–1508

Phienthrakul T, Kijsirikul B (2010) Evolutionary strategies for hyperparameters of support vector machines based on multi-scale radial basis function kernels. Soft Comput 14(7):681–699

Pozdnoukhov A, Kanevski M (2008) Multi-scale support vector algorithms for hot spot detection and modelling. Stoch Environ Res Risk Assess 22(5):647–660

Qiu S, Lane T (2009) A framework for multiple kernel support vector regression and its applications to siRNA efficacy prediction. IEEE/ACM Trans Comput Biol Bioinform 6(2):190–199

Rakotomamonjy A, Bach FR, Canu S, Grandvalet Y (2008) SimpleMKL. J Mach Learn Res 9:2491–2521

Sagiroglu S, Guney K, Erler M (1999) Calculation of bandwidth for electrically thin and thick rectangular microstrip antennas with the use of multilayered perceptrons. Int J RF Microwave Comput Aided Eng 9(3):277–286

Sanchez AVD (2003) Advanced support vector machines and kernel methods. Neurocomputing 55(1–2):5–20

Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Stat Comput 14:199–220

Smola A, Schoelkopf B, Raetsch G (1999) Linear programs for automatic accuracy control in regression. In: Proceedings of the Ninth International Conference on Artificial Neural Networks, Edinburgh, UK. IEE Conference Publication. IEE, pp 575–580

Smola A, Scholkopf B (2002) Learning with Kernels. MIT Press, Cambridge

Sonnenburg S, Ratsch G, Schafer C, Scholkopf B (2006) Large scale multiple kernel learning. J Mach Learn Res 7:1531–1565

Subrahmanya N, Shin YC (2010) Sparse multiple kernel learning for signal processing applications. IEEE Trans Pattern Anal Mach Intell 32(5):788–798

Trnka P, Havlena V (2009) Subspace like identification incorporating prior information. Automatica 45(4):1086–1091

Vapnik VN (1995) The nature of statistical learning theory. Springer, Berlin

Varma M, Babu BR (2009) More generality in efficient multiple kernel learning. In, Montreal, QC, Canada, 2009. Proceedings of the 26th International Conference on machine learning, ICML. Omnipress, pp 1065–1072

Zhao YP, Sun JG (2011) Multikernel semiparametric linear programming support vector regression. Expert Systems Appl 38:1611–1618

Yu Y, Qian F (2008) Multi-scale linear programming support vector regression for ethylene distillation modeling. In: Chongqing, China. Proceedings of the World Congress on Intelligent Control and Automation (WCICA). Institute of Electrical and Electronics Engineers Inc., pp 1548–1552

Zheng D, Wang J, Zhao Y (2006) Non-flat function estimation with a multi-scale support vector regression. Neurocomputing 70(1–3):420–429

Zhou J, Huang J, Xue X (2011) Modeling and optimization for electromechanical coupling of cavity filters based on support vector regression. (Dianzi Yu Xinxi Xuebao) J Electron Inf Technol 33(11):2780–2784

Zhou J, Duan B, Huang J (2010) Influence and tuning of tunable screws for microwave filters using least squares support vector regression. Int J RF Microwave Comput Aided Eng 20:422–429

Zhou J, Huang J (2010) Incorporating priori knowledge into linear programming support vector regression. In: 2010 IEEE International Conference on Intelligent Computing and Integrated Systems, ICISS2010, October 22, 2010–October 24, 2010, Guilin, China. IEEE Computer Society, pp 591–595

Zhou J, Huang J (2013) Intelligent tuning for microwave filters based on multi-kernel machine learning model. In: 5th IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications, MAPE 2013, October 29, 2013–October 31, 2013. Chengdu, China, pp 259–266