Expert Systems with Applications 38 (2011) 2177–2186

Contents lists available at ScienceDirect

Expert Systems with Applications

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

# A multiple-kernel support vector regression approach for stock market price forecasting ☆

Chi-Yuan Yeh, Chi-Wei Huang, Shie-Jue Lee *

*Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung 804, Taiwan*

ARTICLE INFO

*Keywords:*
Stock market forecasting
Support vector regression
Multiple-kernel learning
SMO
Gradient projection

ABSTRACT

Support vector regression has been applied to stock market forecasting problems. However, it is usually needed to tune manually the hyperparameters of the kernel functions. Multiple-kernel learning was developed to deal with this problem, by which the kernel matrix weights and Lagrange multipliers can be simultaneously derived through semidefinite programming. However, the amount of time and space required is very demanding. We develop a two-stage multiple-kernel learning algorithm by incorporating sequential minimal optimization and the gradient projection method. By this algorithm, advantages from different hyperparameter settings can be combined and overall system performance can be improved. Besides, the user need not specify the hyperparameter settings in advance, and trial-and-error for determining appropriate hyperparameter settings can then be avoided. Experimental results, obtained by running on datasets taken from Taiwan Capitalization Weighted Stock Index, show that our method performs better than other methods.



## 1. Introduction

Accurate forecasting of stock prices is an appealing yet difficult activity in the modern business world. Many factors influence the behavior of the stock market, including both economic and non-economic. Therefore, stock market forecasting is regarded as one of the most challenging topics in business. In the past, methods based on statistics were proposed for tackling this problem, such as the autoregressive (AR) model (Champernowne, 1948), the autoregressive moving average (ARMA) model (Box & Jenkins, 1994), and the autoregressive integrated moving average (ARIMA) model (Box & Jenkins, 1994). These are linear models which are, more than often, inadequate for stock market forecasting, since stock time series are inherently noisy and non-stationary. Recently, nonlinear approaches have been proposed, such as autoregressive conditional heteroskedasticity (ARCH) (Engle, 1982), generalized autoregressive conditional heteroskedasticity (GARCH) (Bollerslev, 1986), artificial neural networks (ANN) (Hansen & Nelson, 1997; Kim & Han, 2008; Kwon & Moon, 2007; Qi & Zhang, 2008; Zhang & Zhou, 2004), fuzzy neural networks (FNN) (Chang & Liu, 2008; Oh, Pedrycz, & Park, 2006; Zarandi, Rezaee, Turksen, & Neshat, 2009), and support vector regression (SVR) (Cao & Tay, 2001, 2003; Fernando, Julio, & Javier, 2003; Gestel et al., 2001; Pai & Lin, 2005; Tay & Cao, 2001; Valeriy & Supriya, 2006; Yang, Chan, & King, 2002).

ANN has been widely used for modeling stock market time series due to its universal approximation property (Kecman, 2001). Previous researchers indicated that ANN, which implements the empirical risk minimization principle, outperforms traditional statistical models (Hansen & Nelson, 1997). However, ANN suffers from local minimum traps and difficulty in determining the hidden layer size and learning rate. On the contrary, SVR, proposed by Vapnik and his co-workers, has a global optimum and exhibits better prediction accuracy due to its implementation of the structural risk minimization principle which considers both the training error and the capacity of the regression model (Cristianini & Shawe-Taylor, 2000; Vapnik, 1995). However, the practitioner has to determine in advance the type of kernel function and the associated kernel hyperparameters for SVR. Unsuitably chosen kernel functions or hyperparameter settings may lead to significantly poor performance (Chapelle, Vapnik, Bousquet, & Mukherjee, 2002; Duan, Keerthi, & Poo, 2003; Kwok, 2000). Most researchers use trial-and-error to choose proper values for the hyperparameters, which obviously takes a lot of efforts. In addition, using a single kernel may not be sufficient to solve a complex problem satisfactorily, especially for stock market forecasting problems. Several researchers have adopted multiple-kernels to deal with these problems (Bach, Lanckriet, & Jordan, 2004; Bennett, Momma, & Embrechts, 2002; Crammer, Keshet, & Singer, 2003; Gönen et al., 2008; Lanckriet, Cristianini, Bartlett, Ghaoui, & Jordan, 2004; Ong, Smola, & Williamson, 2005; Rakotomamonjy, Bach, Canu, &

☆ This work was supported by the National Science Council under the grant NSC 95-2221-E-110-055-MY2.

* Corresponding author.
*E-mail address:* leesj@mail.ee.nsysu.edu.tw (S.-J. Lee).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.08.004

Grandvalet, 2007, 2008; Sonnenburg, Ratsch, Schäfer, & Schölkopf, 2006; Szafranski, Grandvalet, & Rakotomamonjy, 2008; Tsang & Kwok, 2006; Wang, Chen, & Sun, 2008).

The simplest way to combine multiple-kernels is by averaging them. But each kernel having the same weight may not be appropriate for the decision process, and therefore the main issue concerning multiple-kernel combination is to determine optimal weights for participating kernels. Lanckriet et al. (2004) used a linear combination of matrices to combine multiple-kernels. They transformed the optimization problem into a semidefinite programming (SDP) problem, which, being convex, has a global optimum. However, the amount of time and space required by this method is demanding. Other multiple-kernel learning algorithms include Bach et al. (2004), Sonnenburg et al. (2006), Rakotomamonjy et al. (2007), Rakotomamonjy, Bach, Canu, and Grandvalet (2008), Szafranski et al. (2008) and Gönen et al. (2008). These approaches deal with large-scale problems by iteratively using the sequential minimal optimization (SMO) algorithm (Platt, 1999) to update Lagrange multipliers and kernel weights in turn, i.e., Lagrange multipliers are updated with fixed kernel weights and kernel weights are updated with fixed Lagrange multipliers alternatively. Although these methods are faster than SDP, they are likely to suffer from local minimum traps. Multiple-kernel learning based on hyperkernels has also been studied (Ong et al., 2005; Tsang & Kwok, 2006). Tsang and Kwok (2006) reformulated the problem as a second-order cone programming (SOCP) form. Crammer et al. (2003) and Bennett et al. (2002) used boosting methods to combine heterogeneous kernel matrices.

We propose a regression model, which integrates multiple-kernel learning and SVR, to deal with the stock price forecasting problem. A two-stage multiple-kernel learning algorithm is developed to optimally combine multiple-kernel matrices for SVR. This learning algorithm applies SMO (Platt, 1999) and the gradient projection method (Bertsekas, 1999) iteratively to obtain Lagrange multipliers and optimal kernel weights. By this algorithm, advantages from different hyperparameter settings can be combined and overall system performance can be improved. Besides, the user need not specify the hyperparameter settings in advance, and trial-and-error for determining appropriate hyperparameter settings can then be avoided. Experimental results, obtained by running on datasets taken from Taiwan Capitalization Weighted Stock Index (TAIEX), which is a stock market index for companies traded on the Taiwan Stock Exchange, show that our method performs better than other methods.

The rest of this paper is organized as follows. Section 2 presents basic concepts about support vector regression. Section 3 describes our proposed multiple-kernel support vector regression approach for stock price forecasting. Experimental results are presented in Section 4. Finally, a conclusion is given in Section 5.

## 2. Support vector regression (SVR)

In a regression problem, we are given a set of training patterns $(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_l,y_l)$, where $\mathbf{x}_i \in \mathbb{R}^n$, $i = 1,\ldots,l$, and $y_i \in \mathbb{R}$. Each $y_i$ is the desired target, or output, value for the input vector $\mathbf{x}_i$. A regression model is learned from these patterns and used to predict the target values of unseen input vectors. SVR is a nonlinear kernel-based regression method which tries to locate a regression hyperplane with small risk in high-dimensional feature space. It possesses good function approximation and generalization capabilities.

Among the various types of support vector regression, the most commonly used is $\varepsilon$-SVR which finds a regression hyperplane with an $\varepsilon$-insensitive band (Cristianini & Shawe-Taylor, 2000; Vapnik, 1995). To make the method more robust, the image of the input data does not need to lie strictly on or inside the $\varepsilon$-insensitive band. Instead, the images which lie outside the $\varepsilon$-insensitive band are penalized and slack variables are introduced to account for these images. For convenience, in the sequel, the term SVR is used to stand for $\varepsilon$-SVR. The objective function and constraints for SVR are

$$\begin{aligned}
\min_{\mathbf{w},b} \quad & \frac{1}{2}\langle \mathbf{w},\mathbf{w}\rangle + C\sum_{i=1}^{l}(\xi_i+\hat{\xi}_i),\\
\text{s.t.} \quad & (\langle \mathbf{w},\phi(\mathbf{x}_i)\rangle + b) - y_i \leqslant \varepsilon + \xi_i,\\
& y_i - (\langle \mathbf{w},\phi(\mathbf{x}_i)\rangle + b) \leqslant \varepsilon + \hat{\xi}_i,\\
& \xi_i,\hat{\xi}_i \geqslant 0, \quad i = 1,\ldots,l,
\end{aligned} \tag{1}$$

where $l$ is the number of training patterns, $C$ is a parameter which gives a tradeoff between model complexity and training error, $\xi_i$ and $\hat{\xi}_i$ are slack variables for exceeding the target value by more than $\varepsilon$ and for being below the target value by more than $\varepsilon$, respectively. Note that $\phi : X \to F$ is a possibly nonlinear mapping function from the input space to a feature space $F$. Also, $\langle\cdot,\cdot\rangle$ indicates the inner product of the involved arguments. The regression hyperplane to be derived is

$$f(\mathbf{x}) = \langle \mathbf{w},\phi(\mathbf{x})\rangle + b, \tag{2}$$

where $\mathbf{w}$ and $b$ are weight vector and offset, respectively.

To solve Eq. (1), one can introduce the Lagrangian, take partial derivatives with respect to the primal variables and set the resulting derivatives to zero, and turn the Lagrangian into the following Wolfe dual form

$$\begin{aligned}
\max_{\boldsymbol{\alpha},\hat{\boldsymbol{\alpha}}} \quad & \sum_{i=1}^{l} y_i(\hat{\alpha}_i - \alpha_i) - \varepsilon\sum_{i=1}^{l}(\hat{\alpha}_i + \alpha_i)\\
& -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)K(\mathbf{x}_i,\mathbf{x}_j),\\
\text{s.t.} \quad & \sum_{i=1}^{l}(\hat{\alpha}_i - \alpha_i) = 0,\\
& C \geqslant \alpha_i,\ \hat{\alpha}_i \geqslant 0, \quad i = 1,\ldots,l,
\end{aligned} \tag{3}$$

where $\alpha_i$ and $\hat{\alpha}_i$, $i = 1,\ldots,l$, are Lagrange multipliers, and $\boldsymbol{\alpha} = [\alpha_1,\alpha_2,\ldots,\alpha_l]$ and $\hat{\boldsymbol{\alpha}} = [\hat{\alpha}_1,\hat{\alpha}_2,\ldots,\hat{\alpha}_l]$. Note that $K(\mathbf{x}_i,\mathbf{x}_j)$ is a kernel function which represents the inner product $\langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle$. The most widely adopted kernel function is the radial basis function (RBF) which is defined as

$$\begin{aligned}
K(\mathbf{x}_i,\mathbf{x}_j) &= \langle\phi(\mathbf{x}_i),\phi(\mathbf{x}_j)\rangle,\\
&= \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2),
\end{aligned} \tag{4}$$

where $\gamma$ is the width parameter of the RBF kernel. Now, Eq. (3) can be solved by SMO (Platt, 1999). Suppose $\alpha_i^*$ and $\hat{\alpha}_i^*$, $i = 1,\ldots,,l$, are the optimal values obtained. The regression hyperplane for the underlying regression problem is then given by:

$$f(\mathbf{x}) = \sum_{i=1}^{l}(\hat{\alpha}_i^* - \alpha_i^*)K(\mathbf{x}_i,\mathbf{x}) + b^*, \tag{5}$$

where $b^* = y_k + \varepsilon - \sum_{i=1}^{l}(\hat{\alpha}_i^* - \alpha_i^*)K(\mathbf{x}_i,\mathbf{x}_k)$ is obtained from any $\alpha_k^*$ with $0 < \alpha_k^* < C$.

## 3. Proposed method

In this section, the idea of multiple-kernel support vector regression is formulated. Then a two-stage multiple-kernel learning algorithm for deriving optimal kernel weights and Lagrange multipliers is described.

### 3.1. Multiple-kernel support vector regression

The SVR method presented earlier uses a single mapping function $\phi$, and hence a single kernel function $K$. If a dataset has a locally varying distribution, using a single kernel may not catch up the varying distribution very well. Kernel fusion can help to deal with this problem. Instead of using one single mapping function, several mapping functions are combined to do aggregate mapping. A simple direct sum fusion applies a vector of $M$ mapping functions, i.e.,

$$\Phi(\mathbf{x}) = [\phi_1(\mathbf{x})\phi_2(\mathbf{x})\dots\phi_M(\mathbf{x})], \tag{6}$$

to map the input space to the feature space. We adopt the weighted sum fusion with the following mapping function:

$$\Phi(\mathbf{x}) = \left[\sqrt{\mu_1}\phi_1(\mathbf{x})\sqrt{\mu_2}\phi_2(\mathbf{x})\dots\sqrt{\mu_M}\phi_M(\mathbf{x})\right], \tag{7}$$

where $\mu_1, \mu_2, \dots, \mu_M$ are weights of component functions. Now, the regression problem includes the optimization of two parts. One part is the regression hyperplane $f(\mathbf{x})$. The other part is the weight vector $\boldsymbol{\mu} = [\mu_1, \mu_2, \dots, \mu_M]$. Note that it was shown that $\Phi$ of Eq. (7) is a valid mapping if all the weights are non-negative (Cristianini & Shawe-Taylor, 2000). Also, we require the sum of weights be unity to restrict the range of the search space to prevent the occurrence of overfitting. By referring to Eq. (1), the objective function and constraints for multiple-kernel SVR become

$$\begin{aligned}
\min_{\boldsymbol{\mu}} \min_{\mathbf{w},b} \quad & \frac{1}{2}\langle \mathbf{w}, \mathbf{w}\rangle + C\sum_{i=1}^{l}(\xi_i + \hat{\xi}_i),\\
\text{s.t.} \quad & (\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b) - y_i \leqslant \varepsilon + \xi_i,\\
& y_i - (\langle \mathbf{w}, \Phi(\mathbf{x}_i)\rangle + b) \leqslant \varepsilon + \hat{\xi}_i,\\
& \xi_i, \hat{\xi}_i \geqslant 0, \quad i = 1,\dots,l,\\
& \mu_s \geqslant 0, \quad s = 1,\dots,M,\\
& \sum_{s=1}^{M}\mu_s = 1,
\end{aligned} \tag{8}$$

where $\Phi$ is the vector of function mappings of Eq. (7).

By introducing the Lagrangian, as usual, Eq. (8) can be converted to the following Wolfe dual form:

$$\begin{aligned}
\min_{\boldsymbol{\mu}} \max_{\boldsymbol{\alpha},\hat{\boldsymbol{\alpha}}} \quad & \sum_{i=1}^{l} y_i(\hat{\alpha}_i - \alpha_i) - \varepsilon\sum_{i=1}^{l}(\hat{\alpha}_i + \alpha_i)\\
& -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)\widetilde{K}(\mathbf{x}_i, \mathbf{x}_j)\\
\text{s.t.} \quad & \sum_{i=1}^{l}(\hat{\alpha}_i - \alpha_i) = 0,\\
& C \geqslant \alpha_i,\ \hat{\alpha}_i \geqslant 0, \quad i = 1,\dots,l,\\
& \mu_s \geqslant 0, \quad s = 1,\dots,M,\\
& \sum_{s=1}^{M}\mu_s = 1
\end{aligned} \tag{9}$$

where

$$\begin{aligned}
\widetilde{K}(\mathbf{x}_i, \mathbf{x}_j) &= \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j)\rangle\\
&= \mu_1\langle \phi_1(\mathbf{x}_i), \phi_1(\mathbf{x}_j)\rangle + \mu_2\langle \phi_2(\mathbf{x}_i), \phi_2(\mathbf{x}_j)\rangle + \cdots\\
&\quad + \mu_M\langle \phi_M(\mathbf{x}_i), \phi_M(\mathbf{x}_j)\rangle\\
&= \mu_1 K_1(\mathbf{x}_i, \mathbf{x}_j) + \mu_2 K_2(\mathbf{x}_i, \mathbf{x}_j) + \cdots + \mu_M K_M(\mathbf{x}_i, \mathbf{x}_j)\\
&= \sum_{s=1}^{M}\mu_s K_s(\mathbf{x}_i, \mathbf{x}_j)
\end{aligned} \tag{10}$$

is a weighted sum of $M$ kernel functions $K_1, K_2, \dots, K_M$, corresponding to mapping functions $\phi_1, \phi_2, \dots, \phi_M$, respectively. Now, if we can find $\boldsymbol{\mu}$, $\boldsymbol{\alpha}$, and $\hat{\boldsymbol{\alpha}}$ by solving Eq. (9), the regression hyperplane would be:

$$f(\mathbf{x}) = \sum_{i=1}^{l}(\hat{\alpha}_i^* - \alpha_i^*)\widetilde{K}(\mathbf{x}_i, \mathbf{x}) + b^*, \tag{11}$$

where $b^* = y_k + \varepsilon - \sum_{i=1}^{l}(\hat{\alpha}_i^* - \alpha_i^*)\widetilde{K}(\mathbf{x}_i, \mathbf{x}_k)$ is obtained from any $\alpha_k^*$ with $0 < \alpha_k^* < C$.

### 3.2. Two-stage multi-kernel learning

We develop a two-stage optimization algorithm for solving Eq. (9). The algorithm consists of two-stages in which SMO and gradient projection are applied, respectively. These stages are iteratively performed until the specified stopping criterion is met, as shown in Fig. 1. Note that the iteration number is indicated by $t$. In the first stage, the weight vector $\boldsymbol{\mu}$ is kept fixed, i.e., $\widetilde{K}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{s=1}^{M}\mu_s K_s(\mathbf{x}_i, \mathbf{x}_j)$ is known. Then Eq. (9) becomes:

$$\begin{aligned}
\max_{\boldsymbol{\alpha},\hat{\boldsymbol{\alpha}}} \quad & \sum_{i=1}^{l} y_i(\hat{\alpha}_i - \alpha_i) - \varepsilon\sum_{i=1}^{l}(\hat{\alpha}_i + \alpha_i)\\
& -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j)\widetilde{K}(\mathbf{x}_i, \mathbf{x}_j),\\
\text{s.t.} \quad & \sum_{i=1}^{l}(\hat{\alpha}_i - \alpha_i) = 0,\\
& C \geqslant \alpha_i,\ \hat{\alpha}_i \geqslant 0, \quad i = 1,\dots,l.
\end{aligned} \tag{12}$$

This equation is, obviously, identical in form to Eq. (3) and can be solved by SMO (Platt, 1999). In the second stage, the Lagrange multipliers are kept fixed, and the weight vector $\boldsymbol{\mu}$ is updated by the gradient projection method (Bertsekas, 1999). Since SMO is a standard algorithm for solving the Wolfe dual form, we won't describe it here. Detailed description about SMO can be found in Platt (1999). In the following, we describe how gradient projection is applied to obtain optimal $\boldsymbol{\mu}$ in the second stage.

Since the Lagrange multipliers are considered as known in the second stage, Eq. (9) can be rewritten as follows:

$$\begin{aligned}
\min_{\boldsymbol{\mu}} \quad & J(\boldsymbol{\mu}),\\
\text{s.t.} \quad & \mu_s \geqslant 0, \quad s = 1,\dots,M,\\
& \sum_{s=1}^{M}\mu_s = 1,
\end{aligned} \tag{13}$$

where

$$\begin{aligned}
J(\boldsymbol{\mu}) = & \sum_{i=1}^{l} y_i(\hat{\alpha}_i - \alpha_i) - \varepsilon\sum_{i=1}^{l}(\hat{\alpha}_i + \alpha_i) - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j\\
& - \alpha_j)\sum_{s=1}^{M}\mu_s K_s(\mathbf{x}_i, \mathbf{x}_j).
\end{aligned} \tag{14}$$

Note that $J(\boldsymbol{\mu})$ only depends on $\boldsymbol{\mu}$. By gradient projection (Bertsekas, 1999), we have

$$\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k + \eta^k(\hat{\boldsymbol{\mu}}^k - \boldsymbol{\mu}^k), \tag{15}$$

where $\boldsymbol{\mu}^k$ is the weight vector of the $k$th iteration, $0 < \eta^k \leqslant 1$ is the step-size, and $\hat{\boldsymbol{\mu}}^k$ is defined as

$$\hat{\boldsymbol{\mu}}^k = \begin{cases} \mathbf{z}, & \text{if } \mathbf{z} \text{ belongs to the feasible region;}\\ \mathbf{z}_\perp, & \text{otherwise,} \end{cases} \tag{16}$$

$$\mathbf{z} = \boldsymbol{\mu}^k - s^k\nabla J(\boldsymbol{\mu}^k), \tag{17}$$

where $s^k$ is a positive scalar, and $\mathbf{z}_\perp$ denotes the projection of $\mathbf{z}$ on the feasible region. The feasible region contains all the vectors

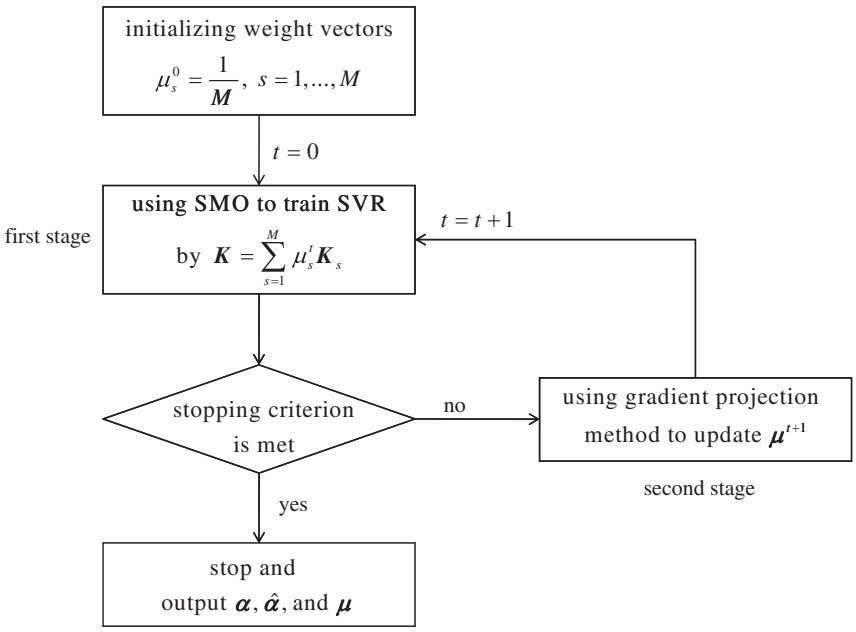


**Fig. 1.** Two-stage multiple-kernel learning algorithm.

$\mathbf{v} = [v_1, v_2, \ldots, v_M]$ such that $v_s \geqslant 0$, $1 \leqslant s \leqslant M$, and $\sum_{s=1}^{M} v_s = 1$. $\nabla J(\boldsymbol{\mu}^k)$ is the following gradient:

$$\nabla J(\mu_s^k) = \frac{\partial J}{\partial} \mu_s^k = -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} (\hat{\alpha}_i - \alpha_i)(\hat{\alpha}_j - \alpha_j) K_s(\mathbf{x}_i, \mathbf{x}_j) \tag{18}$$

for $s = 1,\ldots,M$.

The projection $\mathbf{z}_\perp$ of $\mathbf{z}$ onto the feasible region can be obtained by solving the following constraint problem:

$$\begin{aligned} \min_{\mathbf{z}_\perp} \quad & \|\mathbf{z} - \mathbf{z}_\perp\|^2, \\ \text{s.t.} \quad & \text{all components of } \mathbf{z}_\perp \text{ are non-negative and} \\ & \text{their sum is unity}, \end{aligned} \tag{19}$$

which can be reformulated as the following form of quadratic programming:

$$\begin{aligned} \min_{\mathbf{z}_\perp} \quad & \frac{1}{2}(\mathbf{z}_\perp)^T H \mathbf{z}_\perp - \mathbf{z}^T \mathbf{z}_\perp, \\ \text{s.t.} \quad & \mathbf{k}_s^T \mathbf{z}_\perp \geqslant 0, \quad 1 \leqslant s \leqslant M, \\ & \mathbf{e}^T \mathbf{z}_\perp = 1, \end{aligned} \tag{20}$$

where $H$ is an identity matrix of rank $M$, $\mathbf{k}_s$ is an $M$-vector with the $s$th component being 1 and the other components being 0, and $\mathbf{e}$ is an $M$-vector with all components being 1. The step-size $\eta^k$ is determined by using the Armijo rule along the feasible direction. Here, by choosing $\beta$ and $\sigma$ with $0 < \beta < 1$ and $0 < \sigma < 1$, we can set $\eta^k = \beta^{m_k}$, where $m_k$ is the first non-negative integer $m$ for which

$$\begin{aligned} & J(\boldsymbol{\mu}^{k+1}) - J(\boldsymbol{\mu}^{k+1} + \beta^m(\hat{\boldsymbol{\mu}}^{k+1} - \boldsymbol{\mu}^{k+1})) \\ & \geqslant -\sigma\beta^m \nabla J(\boldsymbol{\mu}^{k+1})^T (\hat{\boldsymbol{\mu}}^{k+1} - \boldsymbol{\mu}^{k+1}). \end{aligned} \tag{21}$$

The detailed procedure of the gradient projection algorithm is depicted in Fig. 2. Note that the iteration number is indicated by $k$. In each application of gradient projection, $k$ starts with 0 and the initial weights are set to $\boldsymbol{\mu}^t$. Then Eq. (15) is iteratively applied until the stopping criterion is met. When the algorithm terminates, the final weights obtained are set to $\boldsymbol{\mu}^{t+1}$.

## 4. Experimental results

To test the forecasting performance of our proposed method, we have conducted three experiments on the datasets taken from Taiwan Capitalization Weighted Stock Index (TAIEX). We also compare the performance of our proposed method with that of other methods, i.e., single kernel support vector regression (SKSVR) (Tay & Cao, 2001), autoregressive integrated moving average (ARIMA) model (Box & Jenkins, 1994), and TSK type fuzzy neural network (FNN) (Chang & Liu, 2008). For convenience, we abbreviate our multiple-kernel support vector regression method as MKSVR.

### 4.1. Experiment I

First of all, we compare the performance of MKSVR with that of SKSVR. In this experiment, the daily stock closing prices of TAIEX for the period of October 2002 to December 2005 are used, and a one-season moving-window testing approach is used for generating the training/validating/testing data. Four datasets, DS-I to DS-IV, are formed, following the way done in Tay and Cao (2001). For instance, DS-I contains the daily stock closing prices from October 2002 to September 2004 selected as training dataset, the daily stock closing prices from October 2004 to December 2004 selected as validating dataset, and the daily stock closing prices from January 2005 to March 2005 selected as testing dataset. The corresponding time periods for DS-I to DS-IV are listed in Table 1.

Given the original daily stock closing prices $\mathbf{p} = \{p_1, p_2, \ldots, p_t, \ldots\}$, we follow (Tay & Cao, 2001) to derive training patterns $(\mathbf{x}_t, y_t)$ for SKSVR and MKSVR. Firstly, the $n$-day exponential moving average of the $t$th day, $\mathrm{EMA}_n(t)$, is defined as

$$\mathrm{EMA}_n(t) = \mathrm{EMA}_n(t-1) + \alpha \times (p_t - \mathrm{EMA}_n(t-1)), \tag{22}$$

where $p_t$ is the $t$th day daily stock closing prices and $\alpha = \frac{2}{1+n}$. The output variable $y_t$ is defined by:

$$y_t = \mathrm{RDP}_{+5}(t) = \frac{\mathrm{EMA}_3(t) - \mathrm{EMA}_3(t-5)}{\mathrm{EMA}_3(t-5)} \times 100. \tag{23}$$

The input vector $\mathbf{x}_t$ consists of five components, i.e., $\mathbf{x}_t = [x_{t,1}\ x_{t,2}\ x_{t,3}\ x_{t,4}\ x_{t,5}]$. A transformed closing price is obtained by subtracting a $n$-day EMA from the closing price, defined by:

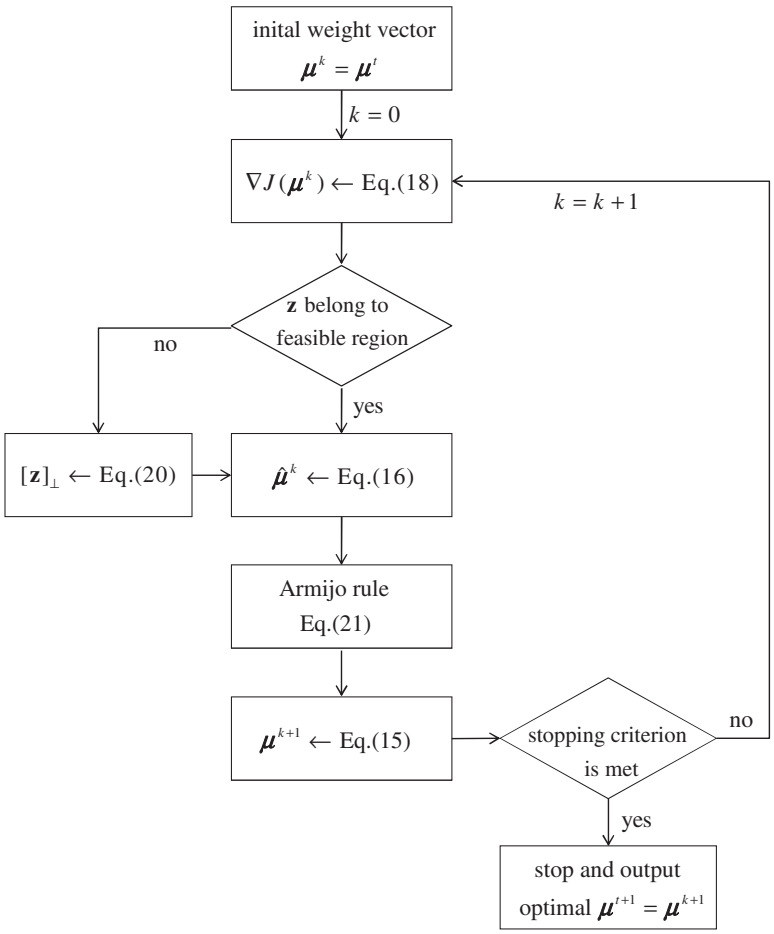


**Fig. 2.** Gradient projection for multiple-kernel learning.

**Table 1**
The datasets for Experiment I.

| Datasets | Training | Validating | Testing |
|---|---|---|---|
| DS-I | 2002/10 – 2004/09 | 2004/10 – 2004/12 | 2005/01 – 2005/03 |
| DS-II | 2003/01 – 2004/12 | 2005/01 – 2005/03 | 2005/04 – 2005/06 |
| DS-III | 2003/04 – 2005/03 | 2005/04 – 2005/06 | 2005/07 – 2005/09 |
| DS-IV | 2003/07 – 2005/06 | 2005/07 – 2005/09 | 2005/10 – 2005/12 |

$$\widehat{\mathrm{EMA}}_n(t) = p_t - \mathrm{EMA}_n(t) \tag{24}$$

and a lagged relative difference in percentage of price (RDP) is defined as:

$$\mathrm{RDP}_{-n}(t) = \frac{p_t - p_{t-n}}{p_{t-n}} \times 100. \tag{25}$$

Then the input variables are defined as $x_{t,1} = \widehat{\mathrm{EMA}}_{15}(t-5)$, $x_{t,2} = \mathrm{RDP}_{-5}(t-5)$, $x_{t,3} = \mathrm{RDP}_{-10}(t-5)$, $x_{t,4} = \mathrm{RDP}_{-15}(t-5)$, and $x_{t,5} = \mathrm{RDP}_{-20}(t-5)$. The root mean squared error (RMSE) is adopted for performance comparison, and is defined as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y_t - \hat{y}_t)^2}, \tag{26}$$

where $y_t$ and $\hat{y}_t$ are desired output and predicted output, respectively.

For SKSVR, there are three parameters that have to be determined in advance while using RBF kernel, i.e., $C$, $\varepsilon$, and $\gamma$. We examine the forecasting performance of SKSVR with $C = 1$ and $\varepsilon = 0.001$.

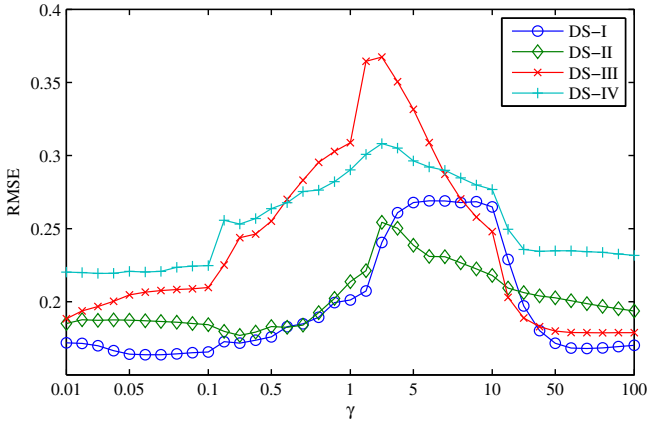


**Fig. 3.** Forecasting performance of SKSVR with different hyperparameters in Experiment I.

**Table 2**
Performance comparison between best SKSVR and MKSVR in Experiment I.

| Methods | Datasets | | | |
|---|---|---|---|---|
| | DS-I | DS-II | DS-III | DS-IV |
| SKSVR | 0.170 | 0.179 | 0.188 | 0.234 |
| MKSVR | 0.161 | 0.174 | 0.179 | 0.219 |

Besides, we try with 37 different settings of hyperparameter $\gamma$, from 0.01 to 0.09 with a stepping factor of 0.01, from 0.1 to 0.9

with a stepping factor of 0.1, from 1 to 10 with a stepping factor of 1, and from 10 to 100 with a stepping factor of 10. The forecasting performance obtained by SKSVR on the four datasets is shown in Fig. 3. From this figure, we can see that SKSVR requires different $\gamma$ settings for different datasets to obtain good performance. For DS-I, the best performance occurs when $0.05 \leqslant \gamma \leqslant 0.1$. For DS-II, the best performance occurs when $0.1 \leqslant \gamma \leqslant 0.5$. For DS-III, the best performance occurs when $50 \leqslant \gamma \leqslant 100$. For DS-IV, the best performance occurs when $0.01 \leqslant \gamma \leqslant 0.05$. The best RMSE values obtained by SKSVR are listed in Table 2.

For multiple-kernel learning, a kernel combining all the 37 different RBF kernels is considered, i.e., $\gamma \in \{0.01, 0.02, \ldots, 0.09, 0.1, 0.2, \ldots, 0.9, 1, 2, \ldots, 9, 10, 20, \ldots, 100\}$. Therefore, the combined kernel matrix is a weighted sum of 37 kernel matrices, i.e., $\widetilde{\mathbf{K}} = \mu_1\mathbf{K}_1 + \mu_2\mathbf{K}_2 + \cdots + \mu_{37}\mathbf{K}_{37}$ where $\mu_1$ denotes the kernel weight for the first kernel matrix with $\gamma = 0.01$ and $\mu_2$ denotes the kernel weight for the second kernel matrix with $\gamma = 0.02$, etc. The RMSE values obtained by MKSVR for the four datasets are also listed in Table 2. Obviously, MKSVR performs better than the best SKSVR for each dataset. Note that we don't need to specify the hyperparameter settings in advance, and trial-and-error for determining appropriate hyperparameter settings is avoided.

### 4.2. Experiment II

In this experiment, we compare the performance of MKSVR with that of ARIMA (Box & Jenkins, 1994). The daily stock closing prices of TAIEX for the period of January 2004 to December 2005

**Table 3**
The datasets for Experiment II and Experiment III.

| Datasets | Training | Validating | Testing |
|---|---|---|---|
| DS-V | 2004/01 – 2004/09 | 2004/10 – 2004/12 | 2005/01 – 2005/03 |
| DS-VI | 2004/04 – 2004/12 | 2005/01 – 2005/03 | 2005/04 – 2005/06 |
| DS-VII | 2004/07 – 2005/03 | 2005/04 – 2005/06 | 2005/07 – 2005/09 |
| DS-VIII | 2004/10 – 2005/06 | 2005/07 – 2005/09 | 2005/10 – 2005/12 |

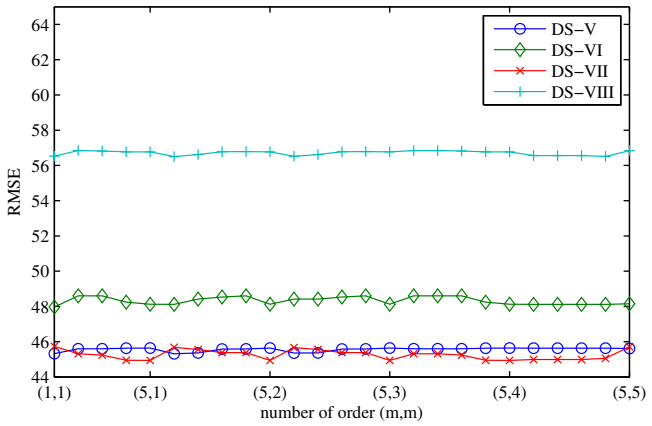


**Fig. 5.** Forecasting performance of ARIMA with different parameters in Experiment II.

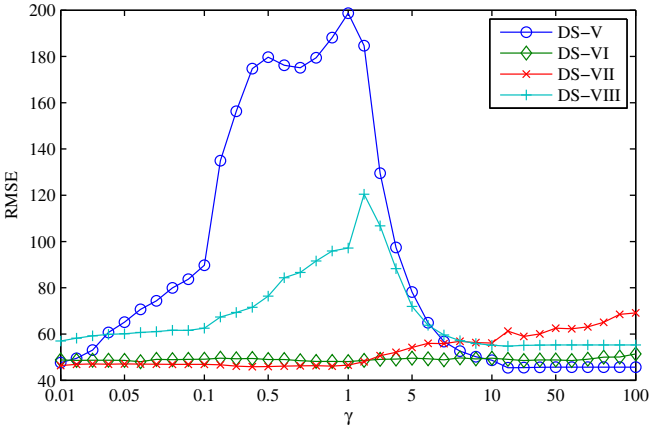


**Fig. 6.** Forecasting performance of SKSVR with different hyperparameters in Experiment II.

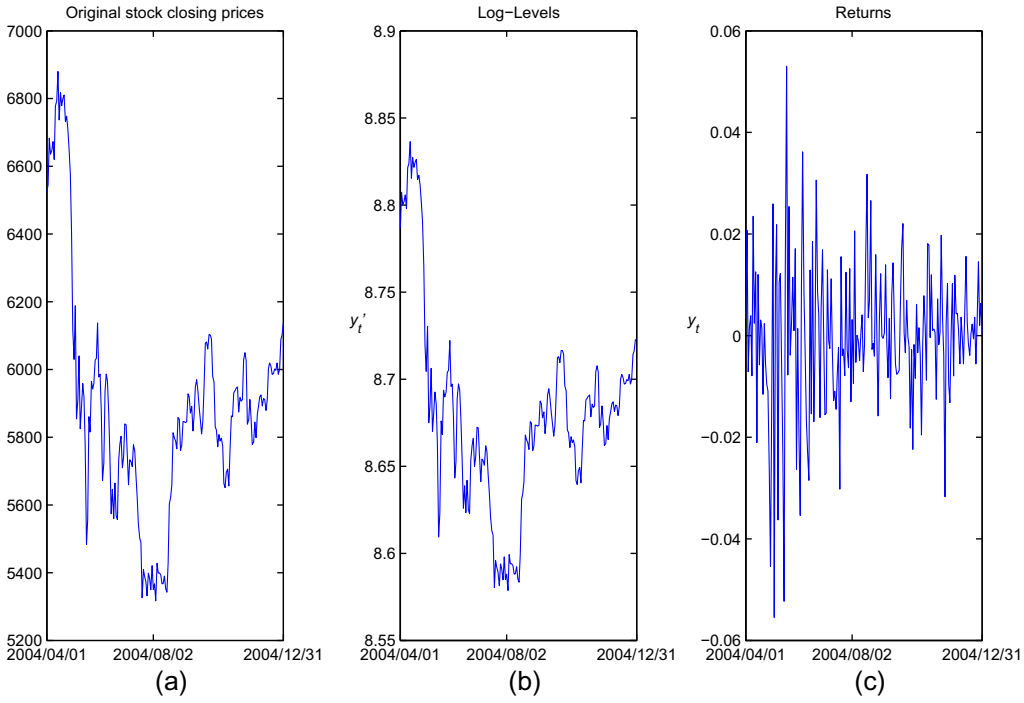


**Fig. 4.** An example of (a) **p**, (b) **y**′, and (c) **y**, TAIEX (2004/04/01 – 2004/12/31).

are used. The one-season moving-window testing approach used in Pai and Lin (2005) for generating the training/validating/testing data is adopted and four datasets, DS-V to DS-VIII, are obtained. For instance, DS-V contains the daily stock closing prices from January 2004 to September 2004 selected as training dataset, the daily stock closing prices from October 2004 to December 2004 selected as validating dataset, and the daily stock closing prices from January 2005 to March 2005 selected as testing dataset. The corresponding time periods for DS-V to DS-VIII are listed in Table 3.

Given the original daily stock closing prices $\mathbf{p} = \{p_1, p_2, \ldots p_t, \ldots\}$, we follow (Box & Jenkins, 1994) to derive training patterns $(\mathbf{x}_t, y_t)$ for this experiment. Firstly, the natural logarithmic transformation is applied to the original daily stock closing prices $\mathbf{p} = \{p_1, p_2, \ldots p_t, \ldots\}$, resulting in another time series $\mathbf{y}' = \{y'_1, y'_2, \ldots y'_t, \ldots\}$ where $y'_t = \ln(p_t)$. The output sequence is $\mathbf{y} = \{y_1, y_2, \ldots y_t, \ldots\}$ where $y_t$ is defined by:

$$y_t = y'_t - y'_{t-1}. \tag{27}$$

An example of these three different sequences is shown in Fig. 4. The input vector $\mathbf{x}_t$ consists of three parts, an autoregressive part, an integrated part, and a moving average part, characterized by three parameters $m$, $o$, $n$ indicating the order of the autoregressive part, the order of the differencing part, and the order of the moving average part, respectively. To distinguish different models, the notation ARIMA $(m, o, n)$ is used. Each input vector consists of $(m + n)$ components, i.e., $\mathbf{x}_t = [x_{t,1}\ x_{t,2} \ldots x_{t,m+n}]$. The values of the components depend on the model used. For instance, for ARIMA $(2, 1, 3)$ we have $x_{t,1} = y_{t-1}$, $x_{t,2} = y_{t-2}$, $x_{t,3} = \epsilon_{t-1}$, $x_{t,4} = \epsilon_{t-2}$, and $x_{t,5} = \epsilon_{t-3}$ where $\epsilon_{t-1}$, $\epsilon_{t-2}$, and $\epsilon_{t-3}$ are forecast errors. The RMSE is defined as follows:

$$\begin{aligned}\text{RMSE} &= \sqrt{\frac{1}{T}\sum_{t=1}^{T}(p_t - \hat{p}_t)^2} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\exp(y'_t) - \exp(\hat{y}'_t))^2} \\ &= \sqrt{\frac{1}{T}\sum_{t=1}^{T}(\exp(y'_t) - \exp(\hat{y}_t + y'_{t-1}))^2},\end{aligned} \tag{28}$$

where $\hat{y}_t$ is the predicted output obtained from the predictor.

**Table 4**
Performance comparison among best ARIMA, best SKSVR, and MKSVR in Experiment II.

| Methods | Datasets | | | |
|---|---|---|---|---|
| | DS-V | DS-VI | DS-VII | DS-VIII |
| ARIMA | 45.421 | 48.400 | 45.674 | 56.957 |
| SKSVR | 45.686 | 48.667 | 46.401 | 55.294 |
| MKSVR | 45.634 | 47.297 | 44.142 | 54.882 |

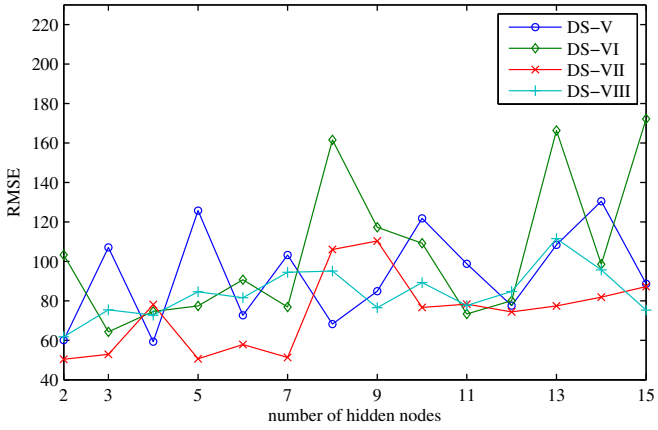


**Fig. 8.** Forecasting performance of FNN with different numbers of hidden nodes in Experiment III.

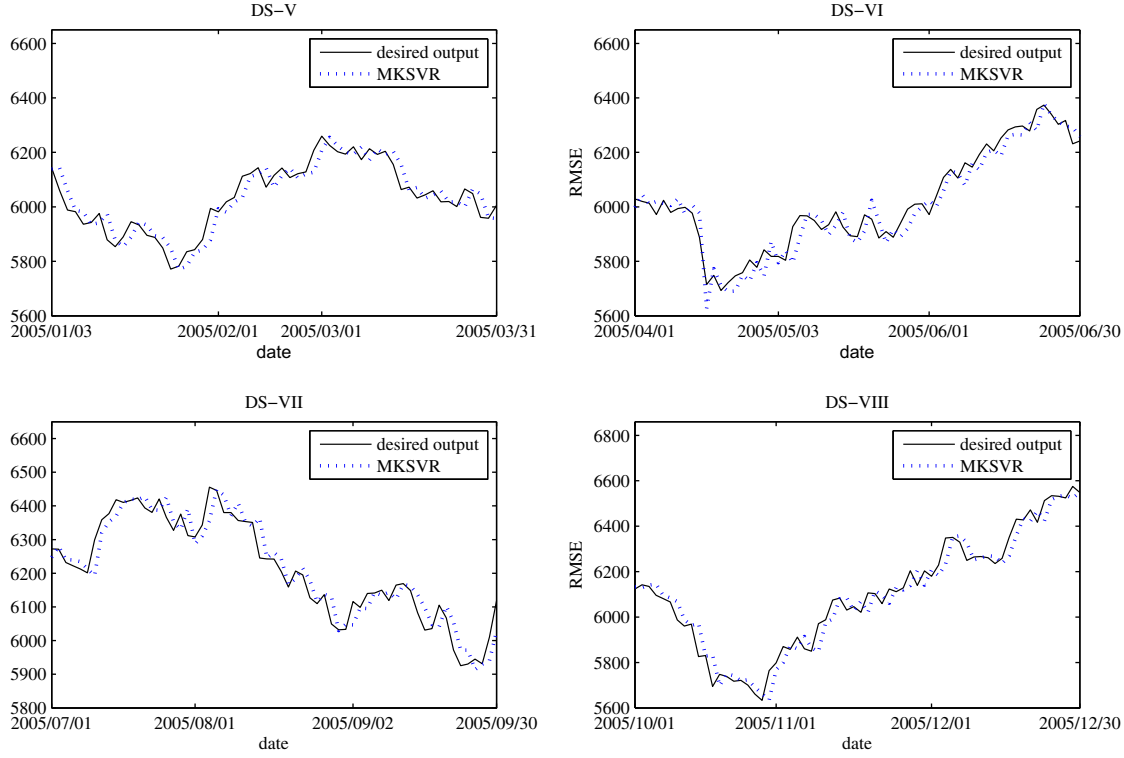


**Fig. 7.** Forecasting results by MKSVR in Experiment II.

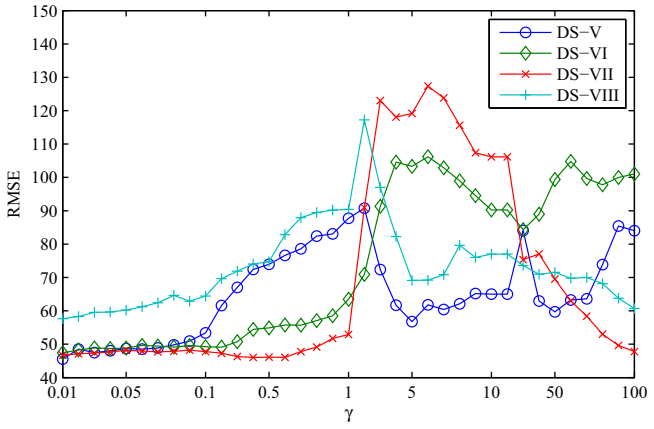


**Fig. 9.** Forecasting performance of SKSVR with different hyperparameters in Experiment III.

**Table 5**
Performance comparison among best FNN, best SKSVR, and MKSVR in Experiment III.

| Methods | Datasets | | | |
|---|---|---|---|---|
| | DS-V | DS-VI | DS-VII | DS-VIII |
| FNN | 59.260 | 64.232 | 50.395 | 61.774 |
| SKSVR | 45.543 | 47.434 | 46.669 | 57.625 |
| MKSVR | 45.531 | 47.398 | 45.907 | 57.301 |

To compare ARIMA and MKSVR, we consider 25 models which are ARIMA $(m,1,n)$ with $m \in \{1,2,3,4,5\}$ and $n \in \{1,2,3,4,5\}$. The forecasting performance obtained by ARIMA on the four datasets is shown in Fig. 5. Interestingly, little variation occurs among different parameter settings with ARIMA. We run SKSVR and MKSVR on these four datasets, with the same settings as in Experiment I. The forecasting performance obtained by SKSVR is shown in Fig. 6. From this figure, we can see that SKSVR requires different $\gamma$ settings for different datasets to obtain good performance. The best RMSE values obtained by ARIMA and SKSVR are listed in Table 4. The RMSE values obtained by MKSVR for these datasets are also listed in Table 4. Obviously, MKSVR can do equally well as, or even better than, the best ARIMA and SKSVR for each dataset. However, we don't need to worry about the hyperparameter settings in MKSVR. Fig. 7 shows the forecasting results for datasets DS-V to DS-VIII by MKSVR.

### 4.3. Experiment III

In this experiment, we compare the performance of MKSVR with that of FNN (Chang & Liu, 2008). We use the same datasets used in Experiment II, as listed in Table 1. Given the original daily stock closing prices $\mathbf{p} = \{p_1, p_2, \ldots p_t, \ldots\}$, we follow (Chang & Liu, 2008) to derive training patterns $(\mathbf{x}_t, y_t)$ for this experiment. Let $y'_t$ be $p_t$, i.e., $y'_t = p_t$. Two technical indices, SMA and BIAS, are used in generating the input vector $\mathbf{x}_t$. SMA, abbreviated for simple moving average, is used to emphasize the direction of a trend and to smooth out price and volume fluctuations. The $n$-day SMA of the $t$th day is defined as follows:

$$\mathrm{SMA}_n(t) = \frac{\sum_{i=t}^{t-5} p_i}{n}. \tag{29}$$

BIAS is used to observe the difference between the closing price and the moving average line. The $n$-day BIAS of the $t$th day is defined as follows:

$$\mathrm{BIAS}_n(t) = \frac{p_t - \mathrm{SMA}_n(t)}{\mathrm{SMA}_n(t)} \times 100. \tag{30}$$

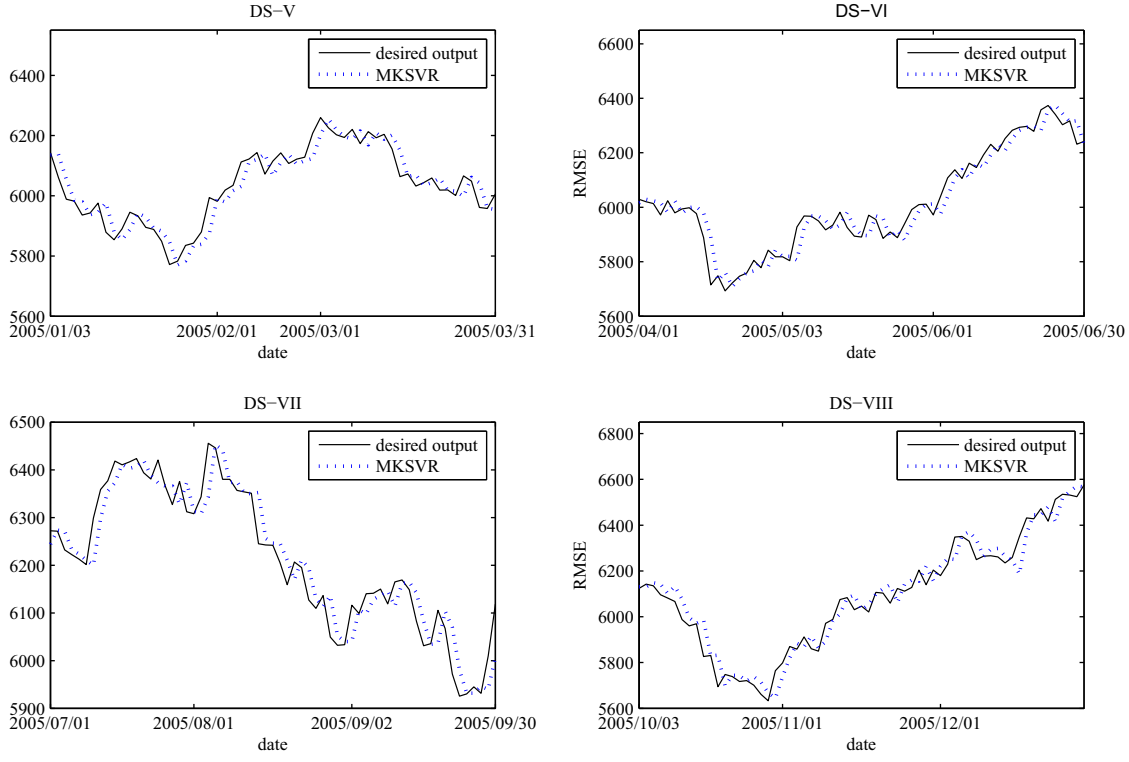


**Fig. 10.** Forecasting results by MKSVR in Experiment III.

Let $x'_{t,1} = \mathrm{SMA}_6(t-1)$ and $x'_{t,2} = \mathrm{BIAS}_6(t-1)$. Now the underlying dataset is partitioned into $K$ clusters by $k$-means (Hartigan & Wong, 1979), a popular clustering algorithm. Then the output variable $y_t$ is

$$y_t = \frac{y'_t - \bar{y}'_j}{\sigma_{y'_j}}, \tag{31}$$

where $y'_t$ belongs to the $j$th cluster, and $\bar{y}'_j$ and $\sigma_{y'_j}$ are the mean and standard deviation in the $y'$ direction of the $j$th cluster. The input vector $\mathbf{x}_t = [x_{t,1}\ x_{t,2}]$ is obtained by:

$$x_{t,i} = \frac{x'_{t,i} - \bar{x}'_{j,i}}{\sigma_{x'_{j,i}}} \tag{32}$$

for $i = 1, 2$, where $[x'_{t,1}\ x'_{t,2}]$ belongs to the $j$th cluster, and $\bar{x}'_{j,i}$ and $\sigma_{x'_{j,i}}$ are the mean and standard deviation, respectively, in the $i$th direction of the $j$th cluster. The RMSE is defined as follows:

$$\begin{aligned}\mathrm{RMSE} &= \sqrt{\frac{1}{T}\sum_{t=1}^{T}(p_t - \hat{p}_t)^2} = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(y'_t - \hat{y}'_t)^2} \\ &= \sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(y'_t - \left(\hat{y}_t \times \sigma_{y'_j} + \bar{y}'_j\right)\right)^2},\end{aligned} \tag{33}$$

where $\hat{y}_t$ is the predicted output and $j$ is the index of its corresponding cluster.

For FNN, standard three-layer networks are adopted. There are 2 nodes in the input layer and 1 node in the output layer. To examine the effect of different architectures on the performance, we set the number of hidden nodes from 2 to 15 with a stepping factor of 1 in the hidden layer. A hybrid learning algorithm incorporating particle swarm optimization (PSO) and recursive least square (RLS) is used for refining the antecedent parameters and the consequent parameters, respectively. The forecasting performance obtained by FNN with different numbers of hidden nodes is depicted in Fig. 8. From this figure, we can see that FNN requires different numbers of hidden nodes for different datasets to obtain good performance. We run SKSVR and MKSVR on these four datasets, with the same settings as in Experiment I. The forecasting performance obtained by SKSVR is shown in Fig. 9. Again, we can see that SKSVR requires different $\gamma$ settings for different datasets to obtain good performance. The best RMSE values obtained by FNN and SKSVR are listed in Table 5. The RMSE values obtained by MKSVR for the four datasets are also listed in Table 5. Obviously, MKSVR works better than the best FNN and SKSVR for each dataset, and we don't need to do trail-and-error with MKSVR. Fig. 10 shows the forecasting results for datasets DS-V to DS-VIII by MKSVR.

## 5. Conclusion

We have proposed a multiple-kernel support vector regression approach for stock market price forecasting. A two-stage multiple-kernel learning algorithm is developed to optimally combine multiple-kernel matrices for support vector regression. The learning algorithm applies sequential minimal optimization and gradient projection iteratively to obtain Lagrange multipliers and optimal kernel weights. By this algorithm, advantages from different hyperparameter settings can be combined and overall system performance can be improved. Besides, the user need not specify the hyperparameter settings in advance, and trial-and-error for determining appropriate hyperparameter settings can then be avoided. Experimental results, obtained by running on datasets taken from Taiwan Capitalization Weighted Stock Index, have shown that our method performs better than other methods.

## References


Bach, F. R., Lanckriet, G. R. G., & Jordan, M. I. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. In: *Proceedings of the 21th international conference on machine learning* (pp. 6–13).

Bennett, K. P., Momma, M., & Embrechts, M. J. (2002). MARK: A boosting algorithm for heterogeneous kernel models. In *Proceedings of the eigth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 24–31).

Bertsekas, D. P. (1999). *Nonlinear programming* (2nd ed.). Massachusetts, USA: Athena Scientific.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroscedasticity. *Journal of Econometrics, 31*(3), 307–327.

Box, G. E. P., & Jenkins, G. M. (1994). *Time series analysis: Forecasting and control* (3rd ed.). Englewood Cliffs: Prentice Hall.

Cao, L., & Tay, F. E. H. (2001). Financial forecasting using support vector machines. *Neural Computing & Applications, 10*(2), 184–192.

Cao, L., & Tay, F. E. H. (2003). Support vector machine with adaptive parameters in financial time series forecasting. *IEEE Transactions on Neural Networks, 14*(6), 1506–1518.

Champernowne, D. G. (1948). Sampling theory applied to autoregressive schemes. *Journal of the Royal Statistical Society: Series B, 10*, 204–231.

Chang, P.-C., & Liu, C.-H. (2008). A TSK type fuzzy rule based system for stock price prediction. *Expert Systems with Application, 34*(1), 135–144.

Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning, 46*(1–3), 131–159.

Crammer, K., Keshet, J., & Singer, Y. (2003). Kernel design using boosting. In S. Becker, S. Thrun, & K. Obermayer (Eds.). *Advances in neural information processing systems* (Vol. 15, pp. 537–544). Cambridge, MA, USA: MIT Press.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge, UK: Cambridge University Press.

Duan, K., Keerthi, S., & Poo, A. N. (2003). Evaluation of simple performance measures for tuning SVM hyperparameters. *Neurocomputing, 51*, 41–59.

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica, 50*(4), 987–1008.

Fernando, P.-C., Julio, A. A.-R., & Javier, G. (2003). Estimating GARCH models using support vector machines. *Quantitative Finance, 3*(3), 163–172.

Gestel, T. V., Suykens, J. A. K., Baestaens, D. E., Lambrechts, A., Lanckriet, G., Vandaele, B., et al. (2001). Financial time series prediction using least squares support vector machines within the evidence framework. *IEEE Transactions on Neural Networks, 12*(4), 809–821.

Gönen, M., & Alpaydin, E. (2008). Localized multiple kernel learning. In *Proceedings of the 25th international conference on machine learning* (pp. 352–359).

Hansen, J. V., & Nelson, R. D. (1997). Neural networks and traditional time series methods: A synergistic combination in state economic forecasts. *IEEE Transactions on Neural Networks, 8*(4), 863–873.

Hartigan, J. A., & Wong, M. A. (1979). A K-means clustering algorithm. *Applied Statistics, 28*, 100–108.

Kecman, V. (2001). *Learning and soft computing: Support vector machines. Neural networks, and fuzzy logic models*. Cambridge, MA, USA: MIT Press.

Kim, K. J., & Han, I. (2008). Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert Systems with Application, 19*(2), 125–132.

Kwok, J. T.-Y. (2000). The evidence framework applied to support vector machines. *IEEE Transactions on Neural Networks, 11*(5), 1162–1173.

Kwon, Y.-K., & Moon, B.-R. (2007). A hybrid neurogenetic approach for stock forecasting. *IEEE Transactions on Neural Networks, 18*(3), 851–864.

Lanckriet, G. R. G., Cristianini, N., Bartlett, P., Ghaoui, L. E., & Jordan, M. I. (2004). Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research, 5*, 27–72.

Oh, S.-K., Pedrycz, W., & Park, H.-S. (2006). Genetically optimized fuzzy polynomial neural networks. *IEEE Transactions on Fuzzy Systems, 14*(1), 125–144.

Ong, C. S., Smola, A. J., & Williamson, R. C. (2005). Learning the kernel with hyperkernels. *Journal of Machine Learning Research, 6*, 1043–1071.

Pai, P.-F., & Lin, C.-S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega: The International Journal of Management Science, 33*(6), 497–505.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. J. C. Burges, & A. J. Smola (Eds.). *Advances in kernel methods: Support vector learning* (Vol. 11, pp. 185–208). Cambridge, MA, USA: MIT Press.

Qi, M., & Zhang, G. P. (2008). Trend timevseries modeling and forecasting with neural networks. *IEEE Transactions on Neural Networks, 19*(5), 808–816.

Rakotomamonjy, A., Bach, F. R., Canu, S., & Grandvalet, Y. (2008). Simplemkl. *Journal of Machine Learning Research, 9*, 2491–2521.

Rakotomamonjy, A., Bach, F. R., Canu, S., & Grandvalet, Y. (2007) More efficiency in multiple kernel learning. In *Proceedings of the 24th international conference on machine learning* (pp. 775–782).

Sonnenburg, S., Ratsch, G., Schäfer, C., & Schölkopf, B. (2006). Large scale multiple kernel learning. *Journal of Machine Learning Research, 7*, 1531–1565.

Szafranski, M., Grandvalet, Y., & Rakotomamonjy, A. (2008). Composite kernel learning. In *Proceedings of the 25th international conference on machine learning* (pp. 1040–1047).

Taiwan Stock Exchange Corporation. <http://www.twse.com.tw/>.
Tay, F. E. H., & Cao, L. (2001). Application of support vector machines in financial time series forecasting. *Omega: The International Journal of Management Science, 29*(4), 309–317.
Tsang, I. W.-H., & Kwok, J. T.-Y. (2006). Efficient hyperkernel learning using second-order cone programming. *IEEE Transactions on Neural Networks, 17*(1), 48–58.
Valeriy, G., & Supriya, B. (2006). Support vector machine as an efficient framework for stock market volatility forecasting. *Computational Management Science, 3*(2), 147–160.
Vapnik, V. (1995). *The nature of statistical learning theory*. New York, USA: Springer-Verlag.
Wang, Z., Chen, S., & Sun, T. (2008). MultiK-MHKS: A novel multiple kernel learning algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 30*(2), 348–353.
Yang, H., Chan, L., & King, I. (2002). Support vector machine regression for volatile stock market prediction. In *Proceedings of the third international conference on intelligent data engineering and automated learning* (pp. 391–396).
Zarandi, M. H. F., Rezaee, B., Turksen, I. B., & Neshat, E. (2009). A type-2 fuzzy rule-based expert system model for stock price analysis. *Expert Systems with Application, 36*(1), 139–154.
Zhang, D., & Zhou, L. (2004). Discovering golden nuggets: Data mining in financial application. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 34*(4), 513–522.