Ann Oper Res (2010) 174: 103–120
DOI 10.1007/s10479-008-0357-7

Financial market forecasting using a two-step kernel learning method for the support vector regression

Li Wang · Ji Zhu

Published online: 28 May 2008

Abstract In this paper, we propose a two-step kernel learning method based on the support vector regression (SVR) for financial time series forecasting. Given a number of candidate kernels, our method learns a sparse linear combination of these kernels so that the resulting kernel can be used to predict well on future data. The L_1-norm regularization approach is used to achieve kernel learning. Since the regularization parameter must be carefully selected, to facilitate parameter tuning, we develop an efficient solution path algorithm that solves the optimal solutions for all possible values of the regularization parameter. Our kernel learning method has been applied to forecast the S&P500 and the NASDAQ market indices and showed promising results.

Keywords Financial market forecasting · Kernel learning · LAR/LASSO · Non-negative garrote · Support vector regression

1 Introduction

Forecasting the financial market is a major challenge in both academia and business. Because of the noisy nature, financial time series are among the most difficult signals to forecast, which naturally leads to the debate on market predictability among the academics and market practitioners. The efficient market hypothesis (Fama 1970, 1991) and the random walk hypothesis (Malkiel 1973) are major theories in economics and finance. The hypotheses state that the financial market evolves randomly and no excess returns can be made by predicting and timing the market. According to these hypotheses, it is impossible to consistently outperform the market, and the simple "buy-and-hold" is the best investment strategy.

L. Wang will join Barclays Global Investors, San Francisco, CA 94105, USA.

L. Wang
Ross School of Business, University of Michigan, Ann Arbor, MI 48109, USA
e-mail: wang@umich.edu

J. Zhu (✉)
Department of Statistics, University of Michigan, Ann Arbor, MI 48109, USA
e-mail: jizhu@umich.edu

Many economists and professional investors, however, dispute these hypotheses, and they believe that the financial market is predictable to some degree. Lo et al. (2000) showed the evidence of the predictability for the financial market using technical analysis and computational algorithms. Furthermore, Lo and MacKinlay (2001) went through a series of tests and demonstrated the existence of trends and patterns in the financial market. Hirshleifer (2001) provided a survey of empirical evidence for the capital market inefficiency and reviewed the explanation for these findings from the behavioral finance perspective.

Researchers in the machine learning and data mining community have also tried to forecast the financial market, using various learning algorithms. The support vector machine (SVM) is one of the tools that have shown promising results (Cao and Tay 2001, 2003, 2001; Kim 2003; Huang et al. 2005). The SVM is a kernel based learning method. Kernel methods embed the original input space $\mathcal{X}$ into a Hilbert space $\mathcal{F}$ via kernel functions and look for linear relations in $\mathcal{F}$, which correspond to nonlinear relations in the original space $\mathcal{X}$.

Kernel functions usually contain tuning parameters. The values of the tuning parameters are crucial for the performance of the kernel methods. If a kernel function is given, the associated parameter values can be tuned by either trial-and-error or heuristic search, e.g., the gradient-based method (Chapelle et al. 2002; Keerthi et al. 2007) and the evolutionary method (Friedrichs and Igel 2005; Nguyen et al. 2007). However, in financial market forecasting, in addition to tune the parameter for a given kernel function, another relevant issue is how to combine different kernel functions in order for the traders to make better decisions. For example, traders often observe information from various sources, including technical indicators, economic indices, political news, etc. A kernel function may be derived from each information source. Then a natural question is how to combine these information (kernels) to help the trader make better forecast decisions. This problem is referred to as kernel learning or kernel selection in the literature. Traditional kernel methods rely on a single kernel function. On the other hand, kernel learning often seeks for a linear combination of candidate kernels. The candidate kernels can be obtained from different kernel functions, parameter settings and/or data sources.

In this paper, we propose a two-step kernel learning method based on the support vector regression (SVR) for financial market forecasting. In the first step, we solve a standard SVR problem using all candidate kernels simultaneously; in the second step, we use scaling parameters, one for each candidate kernel, to achieve kernel learning. The L_1-norm regularization is used to adjust the scaling parameters. Since the regularization parameter has to be selected carefully, to facilitate parameter tuning, we develop an efficient solution path algorithm, which solves the fitted model for every possible value of the regularization parameter. Then the best parameter value can be identified using a parameter selection criterion or a validation dataset.

Several other kernel learning algorithms have also been proposed recently based on different theories and frameworks. Lanckriet et al. (2004), Bach et al. (2004) and Ong et al. (2005) formulated kernel learning as semidefinite programming (SDP) problems. Fung et al. (2004) and Kim et al. (2006) proposed methods implementing kernel learning by iteratively solving quadratic programming problems. Wang et al. (2006) used an orthogonal least square forward selection procedure to achieve a sparse representation of the kernel. Cristianini et al. (2006) proposed a quantity measuring the “alignment” between a kernel and the data, and they developed algorithms to adapt the kernel matrix to the learning task. Our method, however, is based on a different framework, which is inspired by the non-negative garrote method (Breiman 1995). The non-negative garrote uses scaling parameters for variable selection in linear models; we use the similar idea for selecting kernels in our two-step kernel learning method.

The rest of the paper is organized as follows: in Sect. 2, we describe our method and the solution path algorithm; in Sect. 3, we compare the performance of different methods based on historical data of the S&P500 and the NASDAQ market indices; and we conclude the paper in Sect. 4.

2 Two-step kernel learning for the support vector regression

2.1 Support vector regression

We first briefly review the SVR and refer interested readers to Vapnik (1995) and Smola and Schoelkopf (2004) for details. Let $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ represent the n input vectors, where $\boldsymbol{x}_i \in \mathbb{R}^p$, and $\{y_1, y_2, \ldots, y_n\}$ be the corresponding output, where $y_i \in \mathbb{R}$. Let $\boldsymbol{\Phi} : \mathcal{X} \to \mathcal{F}$ be a mapping function, which transfers a vector in the original space $\mathcal{X}$ into a Hilbert space $\mathcal{F}$. The SVR solves the following problem:

$$\min_{\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}} \sum_{i=1}^{n} \ell(\xi_i) + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2 \tag{1}$$

$$\text{subject to} \quad \xi_i = y_i - \beta_0 - \boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{\Phi}(\boldsymbol{x}_i), \quad i = 1, \ldots, n. \tag{2}$$

In the above setup, $\ell(\cdot)$ is a loss function and $\lambda \geq 0$ is a regularization parameter, which controls the trade-off between the loss and the complexity of the fitted model. Figure 1 shows some of the popular loss functions that have been used for the SVR: the Laplace, the ε-insensitive, the quadratic and the Huber loss functions. The Laplace loss function is more robust to outliers than the quadratic loss function and it requires fewer parameters than the ε-insensitive and the Huber loss functions. Therefore, in this paper we focus on the Laplace loss function. However, we note that our method can be naturally applied to other loss functions as well after straightforward modifications.

With the Laplace loss function, (1)–(2) can be written as:

$$\begin{aligned} &\min_{\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}} \sum_{i=1}^{n} (\xi_i^+ + \xi_i^-) + \frac{\lambda}{2} \|\boldsymbol{\beta}\|_2^2 \\ &\text{subject to} \quad -\xi_i^- \leq y_i - \beta_0 - \boldsymbol{\beta}^{\mathsf{T}} \boldsymbol{\Phi}(\boldsymbol{x}_i) \leq \xi_i^+, \\ &\qquad\qquad\quad \xi_i^+, \xi_i^- \geq 0, \quad i = 1, \ldots, n. \end{aligned}$$

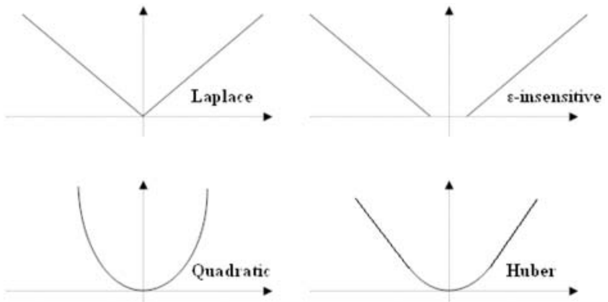

Fig. 1 Commonly used loss functions in the SVR

The corresponding Lagrangian function is:

$$\begin{aligned} L_P &= \sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) + \frac{\lambda}{2}\|\boldsymbol{\beta}\|_2^2 + \sum_{i=1}^{n}\gamma_i^+(y_i - \beta_0 - \boldsymbol{\beta}^\mathsf{T}\boldsymbol{\Phi}(\boldsymbol{x}_i) - \xi_i^+) \\ &\quad - \sum_{i=1}^{n}\gamma_i^-(y_i - \beta_0 - \boldsymbol{\beta}^\mathsf{T}\boldsymbol{\Phi}(\boldsymbol{x}_i) + \xi_i^-) - \sum_{i=1}^{n}\rho_i^+\xi_i^+ - \sum_{i=1}^{n}\rho_i^-\xi_i^-, \end{aligned}$$

where $\gamma_i^+, \gamma_i^-, \rho_i^+, \rho_i^-$ are non-negative Lagrangian multipliers. From the Lagrangian function, the original optimization problem can be transformed into its dual problem:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} -\frac{1}{2\lambda}\boldsymbol{\alpha}^\mathsf{T}\boldsymbol{K}\boldsymbol{\alpha} + \boldsymbol{y}^\mathsf{T}\boldsymbol{\alpha} \\ &\text{subject to} \quad \sum_{i=1}^{n}\alpha_i = 0, \\ &\qquad\qquad -1 \le \alpha_i \le 1, \quad i = 1, \dots, n, \end{aligned} \tag{3}$$

where $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix, with the (i,i') entry $K_{i,i'} = \boldsymbol{\Phi}(\boldsymbol{x}_i)^\mathsf{T}\boldsymbol{\Phi}(\boldsymbol{x}_{i'}) = K(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$.

Let $\widehat{\boldsymbol{\alpha}}$ represent its optimal solution, then the primal variables can be recovered as:

$$\widehat{\boldsymbol{\beta}} = \frac{1}{\lambda}\sum_{i=1}^{n}\widehat{\alpha}_i\boldsymbol{\Phi}(\boldsymbol{x}_i).$$

Notice that for any input vector $\boldsymbol{x}$, the fitted model is:

$$\widehat{f}(\boldsymbol{x}) = \widehat{\beta}_0 + \frac{1}{\lambda}\sum_{i=1}^{n}\widehat{\alpha}_i K(\boldsymbol{x}_i, \boldsymbol{x}). \tag{4}$$

Therefore, by using the kernel function $K(\cdot,\cdot)$, we can solve the SVR problem without explicitly defining the mapping function $\boldsymbol{\Phi}(\cdot)$. For example, radial basis and polynomial kernels are commonly used kernel functions:

$$\text{Radial}: \; K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/\sigma^2) \tag{5}$$

$$\text{Polynomial}: \; K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle\boldsymbol{x}, \boldsymbol{x}'\rangle)^d \tag{6}$$

where σ and d are user-specified parameters.

2.2 Model for a two-step kernel learning method

The standard SVR uses only a single mapping function $\boldsymbol{\Phi}(\cdot)$ or its corresponding kernel function $K(\cdot,\cdot)$. In our setting, we consider multiple mappings: $\boldsymbol{\Phi}_1(\cdot), \dots, \boldsymbol{\Phi}_m(\cdot)$, where m is the number of candidate mappings available. Let $K_1(\cdot,\cdot), \dots, K_m(\cdot,\cdot)$ be the corresponding kernel functions. We look for a sparse linear combination of these kernels that can be used to predict accurately on future data. For example, in the financial market forecasting setting, $K_1(\cdot,\cdot), \dots, K_m(\cdot,\cdot)$ may correspond to different information sources. Our kernel learning method results in a new kernel function $K^*(\cdot,\cdot) = s_1K_1(\cdot,\cdot) + \cdots + s_mK_m(\cdot,\cdot)$, where $s_1, \ldots, s_m \geq 0$ are the associated combination coefficients. Our goal is to learn $s_1, \ldots, s_m$, and the method we propose consists of two steps.

In the first step, we solve an SVR problem that uses all candidate mappings (or kernels) simultaneously:

$$\min_{\beta_0, \boldsymbol{\beta}_j, \xi_i^+, \xi_i^-} \sum_{i=1}^{n} (\xi_i^+ + \xi_i^-) + \frac{\lambda}{2} \sum_{j=1}^{m} \|\boldsymbol{\beta}_j\|_2^2 \tag{7}$$

$$\begin{aligned} \text{subject to} \quad & -\xi_i^- \leq y_i - \beta_0 - \sum_{j=1}^{m} \boldsymbol{\beta}_j^{\mathsf{T}} \boldsymbol{\Phi}_j(\boldsymbol{x}_i) \leq \xi_i^+, \\ & \xi_i^+, \xi_i^- \geq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{8}$$

Similar to the standard SVR, we can transform it into its dual format:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \; & -\frac{1}{2\lambda} \boldsymbol{\alpha}^{\mathsf{T}} \left(\sum_{j=1}^{m} \boldsymbol{K}_j \right) \boldsymbol{\alpha} + \boldsymbol{y}^{\mathsf{T}} \boldsymbol{\alpha} \\ \text{subject to} \quad & \sum_{i=1}^{n} \alpha_i = 0, \\ & -1 \leq \alpha_i \leq 1, \quad i = 1, \ldots, n, \end{aligned} \tag{9}$$

where the (i, i') entry of the kernel matrix $\boldsymbol{K}_j \in \mathbb{R}^{n \times n}$ is defined as $K_j(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$. Notice that this problem is exactly the same as the standard SVR, except that the kernel matrix $\boldsymbol{K}$ in (3) is replaced by the summation of candidate kernel matrices ($\sum_{j=1}^{m} \boldsymbol{K}_j$). Notice that the above dual is a quadratic programming (QP) problem, and we can solve it efficiently using the sequential minimal optimization (SMO) method (Platt 1999) or the solution path algorithm (Gunter and Zhu 2006). Let $\widehat{\boldsymbol{\alpha}}$ be its optimal solution, then each coefficient vector $\widehat{\boldsymbol{\beta}}_j$ can be written as:

$$\widehat{\boldsymbol{\beta}}_j = \frac{1}{\lambda} \sum_{i=1}^{n} \widehat{\alpha}_i \boldsymbol{\Phi}_j(\boldsymbol{x}_i), \quad j = 1, \ldots, m. \tag{10}$$

In the second step, we introduce scaling parameters s_j ($j = 1, \ldots, m$), one for each of the candidate mappings (or kernels), and consider

$$\boldsymbol{\beta}_j^{\text{new}} = s_j \widehat{\boldsymbol{\beta}}_j, \quad j = 1, \ldots, m. \tag{11}$$

To solve for s_j, we consider to regularize the L_1-norm of s_j (Tibshirani 1996; Bradley and Mangasarian 1998), i.e.:

$$\min_{s_j, \beta_0, \xi_i^+, \xi_i^-} \sum_{i=1}^{n} (\xi_i^+ + \xi_i^-) \tag{12}$$

$$\text{subject to} \quad -\xi_i^- \leq y_i - \beta_0 - \sum_{j=1}^{m} s_j \widehat{\boldsymbol{\beta}}_j^{\mathsf{T}} \boldsymbol{\Phi}_j(\boldsymbol{x}_i) \leq \xi_i^+,$$

$$\sum_{j=1}^{m} s_j \le C, \tag{13}$$

$$s_j \ge 0, \quad j = 1, \ldots, m, \tag{14}$$

$$\xi_i^+, \xi_i^- \ge 0, \quad i = 1, \ldots, n, \tag{15}$$

where $C \ge 0$ is a regularization parameter. Notice that in the second step, $\widehat{\boldsymbol{\beta}}_j$ are known and fixed, and we are interested in solving for β_0 and s_j. Denote the solution as $\widehat{\beta}_0^*$ and $\widehat{s}_j$ ($j = 1, \ldots, m$); using (10) and (11), we then have the final fitted model:

$$\begin{aligned} \widehat{f}(\boldsymbol{x}) &= \widehat{\beta}_0^* + \sum_{j=1}^{m} \widehat{s}_j \widehat{\boldsymbol{\beta}}_j^{\mathsf{T}} \boldsymbol{\Phi}_j(\boldsymbol{x}) \\ &= \widehat{\beta}_0^* + \sum_{j=1}^{m} \widehat{s}_j \left(\frac{1}{\lambda} \sum_{i=1}^{n} \widehat{\alpha}_i K_j(\boldsymbol{x}_i, \boldsymbol{x}) \right). \end{aligned}$$

The final fitted model can be further written as:

$$\widehat{f}(\boldsymbol{x}) = \widehat{\beta}_0^* + \frac{1}{\lambda} \sum_{i=1}^{n} \widehat{\alpha}_i K^*(\boldsymbol{x}_i, \boldsymbol{x}), \tag{16}$$

with the corresponding learned new kernel:

$$K^*(\cdot, \cdot) = \sum_{j=1}^{m} \widehat{s}_j K_j(\cdot, \cdot). \tag{17}$$

In (13), we apply the L_1-norm regularization idea (Breiman 1995; Tibshirani 1996; Bradley and Mangasarian 1998), which has been used for variable selection in the setting of linear regression: the singularities of the constraints $\sum_{j=1}^{m} s_j \le C$ and $s_j \ge 0$ tend to generate a sparse solution for $\boldsymbol{s}$, i.e., some of the s_j's are estimated to be exact zero. The idea is similar to the non-negative garrote method (Breiman 1995), which uses scaling parameters for variable selection in the setting of linear regression. In our method, we use scaling parameters to select kernels. If an $\widehat{s}_j$ is equal to zero, the corresponding kernel is removed from the estimated combined kernel (17), hence the method implements automatic kernel selection or kernel learning. The intuition is that if the jth mapping (or kernel) is not important for prediction, the corresponding $\widehat{\boldsymbol{\beta}}_j^{\mathsf{T}} \boldsymbol{\Phi}_j(\boldsymbol{x})$ tends to have a small "magnitude," hence the s_j gets heavily penalized and tends to be estimated as zero, while if the jth mapping (or kernel) is important for the prediction, $\widehat{\boldsymbol{\beta}}_j^{\mathsf{T}} \boldsymbol{\Phi}_j(\boldsymbol{x})$ tends to have a big "magnitude," hence s_j gets lightly penalized and tends to be estimated as non-zero.

2.3 The solution path algorithm

The regularization parameter C controls the bias-variance trade-off. Increasing the value of C tends to select more kernels and generates a more complicated model, which reduces the bias but increases the variance of the fitted model; and vice versa when decreasing the value of C. Hence the value of C is critical for the performance of the fitted model. In practice, people can pre-specify a number of values for C, solve the optimization problem for each of them, and use a validation set or some model selection criterion to choose the best model. Instead of using such a trial-and-error approach, we develop an efficient solution path algorithm that computes the solutions for all possible values of C, which facilitates the selection for an optimal value of C. Starting from $C = 0$, the algorithm continuously increases C and calculates $\widehat{s}_j$ and $\widehat{\beta}_0^*$ along the path, until the solution does not further change with C. The algorithm takes advantage of the piecewise linearity property of $\widehat{s}_j$ and $\widehat{\beta}_0^*$ with respect to C. We acknowledge that our algorithm is inspired by the LAR/LASSO method (Efron et al. 2004) and the general piecewise linear solution path strategy of Rosset and Zhu 2007. Similar ideas have also been used in the SVM to generate the optimal solution path of the regularization parameter for over-fitting control (Hastie et al. 2004; Gunter and Zhu 2006) or of "hyperparameters" for the kernel function (Wang et al. 2007). However, we make a note that the algorithm we develop here is significantly different from the earlier algorithms, because we are now dealing with a non-differentiable loss function *and* a non-differentiable penalty, while all the earlier algorithms deal with cases where either the loss function is differentiable or the penalty is differentiable.

The overall computational cost of our solution path algorithm is approximately $O(\max(n, m)m^2)$. For expositional clarity, we delay the details of the algorithm in the appendix.

3 Financial market forecasting

In this section, we apply our two-step kernel learning method to forecast the financial market and compare its performance to three other methods, specifically, the buy-and-hold strategy, the standard SVR and the kernel learning method proposed by Lanckriet et al. (2004). We chose to compare with Lanckriet et al. (2004) simply because none of the current kernel selection methods have their code publicly available, and the method in Lanckriet et al. (2004) seems to be the most convenient for implementation (and it also works well in practical problems). The experiments were based on historical prices of the S&P500 and the NASDAQ composite indices. All these methods were tested on 2,500 trading days from 11/1997 to 10/2007, which covers for around 10 years.

From the daily closing prices of the S&P500 and the NASDAQ indices, we first calculated the 5 (weekly), 10 (biweekly), 20 (monthly), and 50-day (quarterly) moving averages, which are widely used technical indicators by traders. To illustrate our preprocessing procedure, let P_t denote the closing price on day t and $A_{t,T}$ be the T-day moving average on day t, where $A_{t,T}$ is calculated as:

$$A_{t,T} = \frac{1}{T} \sum_{k=t-T+1}^{t} P_k, \quad \text{for } T = 5, 10, 20, 50.$$

Notice that P_t can also be considered as $A_{t,1}$. Then we calculated the moving average log-return $R_{t,T}$ (including the daily log-return) for each day t:

$$R_{t,T} = \ln(A_{t,T}) - \ln(A_{t-T,T}), \quad \text{for } T = 1, 5, 10, 20, 50.$$

We chose to use the log-return, which is a more widely used measure in finance than the raw return, because the return tends to have a log-normal distribution, while the log-return has an approximate normal distribution. Since the log-return was used in our experiments, the cumulative log-return over a certain period was simply the summation of the daily log-returns over this period. Overall we had 5 time series: (1) $R_{t,1}$, (2) $R_{t,5}$, (3) $R_{t,10}$, (4) $R_{t,20}$, and (5) $R_{t,50}$. The output variable was $R_{t+1,1}$, the next day log-return.

Then we constructed the input features. For the jth time series, we extracted p_j data points to construct the input features. Specifically, let

$$\boldsymbol{x}_t^j = [R_{t-(p_j-1)T_j,T_j}, R_{t-(p_j-2)T_j,T_j}, \ldots, R_{t,T_j}], \quad \text{for } j = 1, \ldots, 5,$$

where $T_1 = 1$, $T_2 = 5$, $T_3 = 10$, $T_4 = 20$ and $T_5 = 50$. Notice that as T_j increases, the corresponding time series R_{t,T_j} becomes smoother, hence fewer data points were needed for $\boldsymbol{x}_t^j$. Specifically, we let $p_1 = 10$, $p_2 = 8$, $p_3 = 6$, $p_4 = 4$ and $p_5 = 4$. The overall input features for day t were then:

$$\boldsymbol{x}_t = [\boldsymbol{x}_t^1, \boldsymbol{x}_t^2, \boldsymbol{x}_t^3, \boldsymbol{x}_t^4, \boldsymbol{x}_t^5].$$

The intuition is that different $\boldsymbol{x}_t^j$ represent different characteristics underlying the financial market. For example, $\boldsymbol{x}_t^1$ and $\boldsymbol{x}_t^2$ capture the short-term (daily and weekly) behaviors of the market, while $\boldsymbol{x}_t^4$ and $\boldsymbol{x}_t^5$ capture the long-term (monthly and quarterly) trends.

It is not clear a priori which features are important for predicting the next day return, neither how they should be combined to predict. We treated these features separately by applying separate kernels to them and used our method to automatically select and combine them to help us decide whether we should "long" or "short" for the next day. Since the radial basis kernel is the most popular kernel that gets used in practical problems and it has also been proved to be useful in financial market forecasting, for example, see Cao and Tay (2003), Kim (2003) and Huang et al. (2005), we chose to use the radial basis kernel (5). The radial basis kernel contains a scaling parameter σ^2, which controls the "effective support" of the kernel function. To illustrate our method, we considered two possible values for σ^2. Specifically, for each feature $\boldsymbol{x}_t^j$ (which is a p_j-vector, $j = 1, \ldots, 5$), we considered two candidate radial basis kernel functions:

$$K_j(\boldsymbol{x}_t^j, \boldsymbol{x}_{t'}^j) = \exp(-\|\boldsymbol{x}_t^j - \boldsymbol{x}_{t'}^j\|^2/m_j^2) \quad \text{and} \quad K_j(\boldsymbol{x}_t^j, \boldsymbol{x}_{t'}^j) = \exp(-\|\boldsymbol{x}_t^j - \boldsymbol{x}_{t'}^j\|^2/(10m_j^2)),$$

where m_j^2 is the median value of $\|\boldsymbol{x}_t^j - \bar{\boldsymbol{x}}^j\|^2$. Therefore, there were a total of $5 \times 2 = 10$ candidate kernel functions.

In evaluating the fitted model, we used a simple strategy to invest: let $\widehat{f}_t$ be the forecasted next day log-return on day t, if $\widehat{f}_t \geq 0$, we buy at the closing price on day t; otherwise, we short it. Then the log-return, $\widehat{R}_t$, associated with day t based on our strategy, can be computed as:

$$\widehat{R}_t = \begin{cases} |R_{t+1,1}|, & \text{if } R_{t+1,1} \cdot \widehat{f}_t \geq 0; \\ -|R_{t+1,1}|, & \text{otherwise.} \end{cases}$$

Figure 2 illustrates how we selected the tuning parameters λ in (9) and C in (12) for our two-step procedure and how we evaluated the fitted model $\widehat{f}$ from our two-step procedure. Starting from 11/1997, we firstly trained models (via the solution path algorithm) on 150 consecutive data points (the training set); secondly, we applied the obtained models on the next 10 data points (the validation set) and selected the values of λ and C that had the best performance; thirdly, we combined 140 data points of the training set and all 10 data points of the validation set into a new set, which we call the "true training set," and trained the final model using the selected values of λ and C; lastly, we applied the final model on another 10 data points (the test set) and recorded its performance. We then shifted forward for 10 data points (or 10 trading days) and repeated the same procedure. We repeated this for 250 times, which correspond to 2,500 trading days until 10/2007. Notice that there was no overlap between any two test sets.

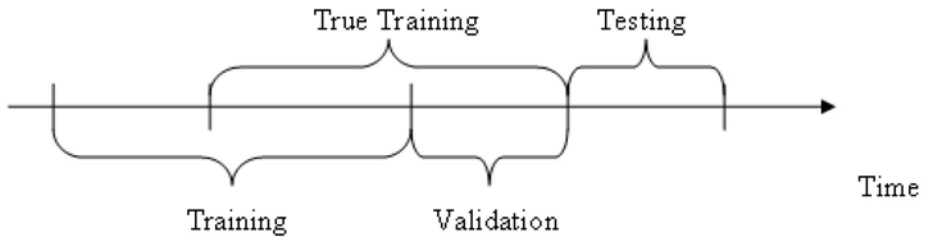

Fig. 2 Training, validation and test sets

Table 1 Performance on the S&P500 and NASDAQ indices. "Average biweekly return" is the average of log-returns over 250 different test sets, and the numbers in the parentheses are the corresponding standard errors. "p-values" were computed from the paired t-test against the buy-and-hold strategy

Method	Average biweekly log-return (%)	p-value
S&P500:		
Buy-and-hold	0.198 (0.217)	–
SVR	0.278 (0.216)	0.814
Kernel learning	0.529 (0.231)	0.338
Two-step procedure	0.855 (0.231)	0.0296
NASDAQ:		
Buy-and-hold	0.202 (0.351)	–
SVR	0.608 (0.374)	0.448
Kernel learning	0.382 (0.339)	0.740
Two-step procedure	1.126 (0.361)	0.0504

For the standard SVR, we used $\boldsymbol{x}_t$ as the input feature, which contains information from different time series without differentiating them, and considered one single radial basis kernel $K(\boldsymbol{x}_t, \boldsymbol{x}_{t'}) = \exp(-\|\boldsymbol{x}_t - \boldsymbol{x}_{t'}\|^2/\sigma^2)$. There were two tuning parameters, the λ as in (3) and the σ. For λ, we used the solution path algorithm that was developed in Gunter and Zhu (2006), which solves solutions for all possible values of λ, and the optimal λ was selected using a validation set. For σ, we first computed the median of $\|\boldsymbol{x}_t - \bar{\boldsymbol{x}}\|$, where $\boldsymbol{x}_t$ is the input feature vector of the tth training observation and $\bar{\boldsymbol{x}}$ is the corresponding mean. Denote it as $m_{\boldsymbol{x}}$. Then we searched the optimal σ over $(0.2m_{\boldsymbol{x}}, 0.4m_{\boldsymbol{x}}, \ldots, 2m_{\boldsymbol{x}})$. For the kernel learning method by Lanckriet et al. (2004), we selected the tuning parameters in a similar fashion. All selected final models were evaluated on the same test sets.

Since there are 5 trading days per week, the summation of log-returns on each test set represents the biweekly log-return. Table 1 shows the average biweekly log-returns among 250 test sets on the two markets from four different methods. The average biweekly log-returns of our method are 4.3 and 5.6 times of the buy-and-hold strategy on the S&P500 and the NASDAQ, respectively. We also used the paired t-test to compare the log-returns of our method against the buy-and-hold strategy. The p-value for the S&P500 is 0.03, and the p-value for the NASDAQ is 0.05. The standard SVR method and the kernel learning method by Lanckriet et al. (2004) also performed better than the buy-and-hold strategy, but the improvements are not statistically significant based on the paired t-test. Figures 3 and 4 show the cumulative log-returns for all these methods on the two markets. We can see that the log-returns of our method have a consistently increasing trend over the 2,500 trading day period.

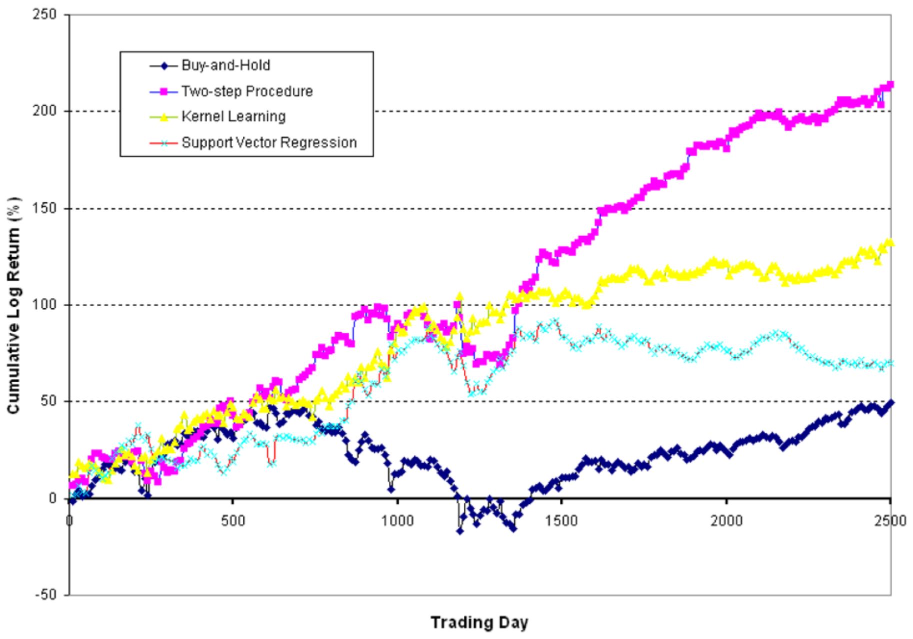

Fig. 3 Cumulative log-returns of the four methods on the S&P500 index

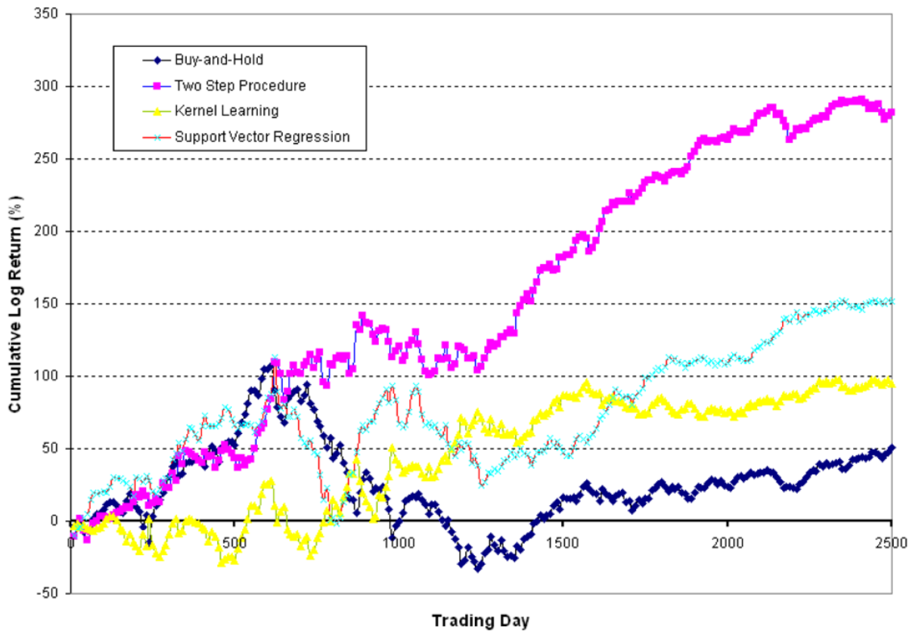

Fig. 4 Cumulative log-returns of the four methods on the NASDAQ index

Table 2 Kernel selection on the S&P500 index. *Each row* corresponds to a candidate kernel; the subscript "1" corresponds to $\sigma^2 = m_j^2$ and the subscript "2" corresponds to $\sigma^2 = 10m_j^2$; the superscript indicates the part of the input features that the kernel is based on. The *second column* contains the average estimated combining coefficient s_j over the 250 different trials, and the numbers in the parentheses are the corresponding standard errors. The *third column* records the selection frequency for each kernel out of 250 trials

Kernel	Two-step procedure		Kernel learning	
	s_j	Frequency	s_j	Frequency
$K_1^{\boldsymbol{x}_1}$	1.2500 (0.2373)	250/250	0.9837 (0.0269)	250/250
$K_2^{\boldsymbol{x}_1}$	0.3710 (0.4015)	154/250	0.0001 (0.0013)	2/250
$K_1^{\boldsymbol{x}_2}$	0.7710 (0.4361)	218/250	0.0144 (0.0267)	82/250
$K_2^{\boldsymbol{x}_2}$	0.2386 (0.3884)	86/250	0.0000 (0.0000)	0/250
$K_1^{\boldsymbol{x}_3}$	0.5749 (0.5506)	157/250	0.0015 (0.0053)	24/250
$K_2^{\boldsymbol{x}_3}$	0.1582 (0.3474)	49/250	0.0000 (0.0000)	0/250
$K_1^{\boldsymbol{x}_4}$	0.3917 (0.5591)	97/250	0.0001 (0.0005)	5/250
$K_2^{\boldsymbol{x}_4}$	0.1682 (0.3722)	47/250	0.0000 (0.0000)	0/250
$K_1^{\boldsymbol{x}_5}$	0.2935 (0.4950)	74/250	0.0002 (0.0018)	4/250
$K_2^{\boldsymbol{x}_5}$	0.1745 (0.3722)	51/250	0.0000 (0.0000)	0/250

To further investigate the effects of different characteristics of the financial time series, we also recorded how each kernel was selected. Table 2 summarizes the results for our method and the kernel learning method in Lanckriet et al. (2004) on the S&P 500 index. As we can see, the two methods behaved very differently on this particular dataset. For our two-step kernel learning method, overall, the short-term (daily and weekly) trends had a bigger impact than the long-term (monthly and quarterly) trends in predicting the next day return, however, the long-term trends also played a significant role. On the other hand, the kernel learning method by Lanckriet et al. (2004) tended to generate a more sparse model: Most of the time, it only selected the kernel that contained the daily information. Regarding the scaling parameter of the radial basis kernel, the value m_j^2 seemed to be preferred over $10m_j^2$. The results on the NASDAQ index are similar.

4 Discussion

In this paper, we have proposed a kernel learning method based on the support vector regression for financial market forecasting. Our method consists of two steps, where a similar idea was used by non-negative garrote (Breiman 1995) for variable selection in the setting of linear regression. In the first step, we fit a standard SVR using all candidate kernels; in the second step, we use scaling parameters, one for each candidate kernel, and search for a sparse linear combination of the kernels via the L_1-norm regularization. For the second step, we have also developed an efficient solution path algorithm that solves the optimal solutions for all possible values of the regularization parameter.

Our two-step kernel learning method shows promising results in forecasting the financial market. The trading strategy based on our method consistently outperforms the market, and the excess returns are statistically significant. Readers might be curious why we are publishing this work instead of making money for ourselves from the financial market. The main reason is that in the current experiment, transaction costs have not been considered. Suppose that the transaction cost takes off 0.1% of the capital for each trade, then overall it will eliminate 250% of the return over the 2,500-day trading period, which makes our method unprofitable. One may also argue that the transaction cost might become ignorable if our trading volume is large enough. However, large orders can also have impact on the market price and we do not have related data to discover the range of order sizes, within which we do not change the market behavior at its closing time. Although our method and the current trading strategy can not bring up economic benefits, they did demonstrate that the financial market is not completely efficient and is predictable to some degree, which is aligned with the conclusions made by many previous work.

Appendix

In the appendix, we describe the details of the solution path algorithm that computes the solutions of (12)–(15) for all possible values of C.

Problem setup

To simplify the notation, we define:

$$\widehat{f}_j(\boldsymbol{x}) = \widehat{\boldsymbol{\beta}}_j^{\mathsf{T}} \boldsymbol{\Phi}_j(\boldsymbol{x}) = \frac{1}{\lambda}\sum_{i=1}^{n} \widehat{\alpha}_i K_j(\boldsymbol{x}_i, \boldsymbol{x}). \tag{18}$$

Then the Lagrangian function of (12)–(15) is:

$$\begin{aligned} L_P = {} & \sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) + \eta\left(\sum_{j=1}^{m} s_j - C\right) - \sum_{j=1}^{m}\varepsilon_j s_j \\ & + \sum_{i=1}^{n}\gamma_i^+\left(y_i - \beta_0 - \sum_{j=1}^{m} s_j \widehat{f}_j(\boldsymbol{x}_i) - \xi_i^+\right) \\ & - \sum_{i=1}^{n}\gamma_i^-\left(y_i - \beta_0 - \sum_{j=1}^{m} s_j \widehat{f}_j(\boldsymbol{x}_i) + \xi_i^-\right) - \sum_{i=1}^{n}\rho_i^+\xi_i^+ - \sum_{i=1}^{n}\rho_i^-\xi_i^-, \end{aligned}$$

where $\gamma_i^+, \gamma_i^-, \eta, \varepsilon_j, \rho_i^+, \rho_i^- \geq 0$ are Lagrangian multipliers. Let $\gamma_i = \gamma_i^+ - \gamma_i^-$ $(i = 1, \ldots, n)$, then the Karush-Kuhn-Tucker (KKT) conditions are:

$$\frac{\partial}{\partial \beta_0}: \quad \sum_{i=1}^{n}\gamma_i = 0, \tag{19}$$

$$\frac{\partial}{\partial s_j}: \quad -\sum_{i=1}^{n}\gamma_i \widehat{f}_j(\boldsymbol{x}_i) + \eta - \varepsilon_j = 0, \tag{20}$$

$$\frac{\partial}{\partial \xi_i^+}: \quad 1 - \gamma_i^+ - \rho_i^+ = 0, \tag{21}$$

$$\frac{\partial}{\partial \xi_i^-}: \quad 1 - \gamma_i^- - \rho_i^- = 0. \tag{22}$$

Let $f(\boldsymbol{x}_i)=\beta_0+\sum_{j=1}^m s_j\widehat{f}_j(\boldsymbol{x}_i)$, then the complementary slackness conditions are:

$$\gamma_i^+(y_i-f(\boldsymbol{x}_i)-\xi_i^+)=0, \tag{23}$$

$$\gamma_i^-(y_i-f(\boldsymbol{x}_i)-\xi_i^-)=0, \tag{24}$$

$$\eta\left(\sum_{j=1}^m s_j-C\right)=0, \tag{25}$$

$$\rho_i^+\xi_i^+=0, \tag{26}$$

$$\rho_i^-\xi_i^-=0, \tag{27}$$

$$\varepsilon_j s_j=0. \tag{28}$$

Note that all the Lagrangian multipliers are non-negative. Let $\xi_i=\xi_i^+-\xi_i^-$ and $\rho_i=\rho_i^+-\rho_i^-$, conditions (21) to (28) lead to the following relationships:

$$\begin{array}{lll} y_i-f(\boldsymbol{x}_i)>0: & \gamma_i=1, & \xi_i>0;\\ y_i-f(\boldsymbol{x}_i)<0: & \gamma_i=-1, & \xi_i<0;\\ y_i-f(\boldsymbol{x}_i)=0: & -1\le\gamma_i\le 1, & \xi_i=0. \end{array}$$

Using these relationships, we can define the following sets:

- $\mathcal{R}=\{i:y_i-f(\boldsymbol{x}_i)>0,\ \gamma_i=1\}$ (the Right portion of the Laplace loss function)
- $\mathcal{E}=\{i:y_i-f(\boldsymbol{x}_i)=0,\ 0\le\gamma_i\le 1\}$ (the Elbow of the Laplace loss function)
- $\mathcal{L}=\{i:y_i-f(\boldsymbol{x}_i)<0,\ \gamma_i=-1\}$ (the Left portion of the Laplace loss function)
- $\mathcal{A}=\{j:s_j\neq 0\}$ (the Active set of the kernels)

Note that if $i\in\mathcal{L}$ or $\mathcal{R}$, then γ_i is known; only when $i\in\mathcal{E}$, the γ_i becomes unknown. From the KKT conditions and the derived relationships, we can get two linear systems for the primal and dual variables. We call the following equations as the primal system:

$$y_i-\beta_0-\sum_{j\in\mathcal{A}}s_j\widehat{f}_j(\boldsymbol{x}_i)=0,\quad i\in\mathcal{E}, \tag{29}$$

$$\sum_{j\in\mathcal{A}}s_j=C. \tag{30}$$

According to (28), if $j\in\mathcal{A}$ ($s_j\neq 0$), then $\varepsilon_j=0$. So we also have the following dual system:

$$\sum_{i\in\mathcal{E}}\gamma_i+|\mathcal{R}|-|\mathcal{L}|=0, \tag{31}$$

$$\sum_{i=1}^n\gamma_i\widehat{f}_j(\boldsymbol{x}_i)-\eta=0,\quad j\in\mathcal{A}. \tag{32}$$

For any optimal solution at any C, the primal and the dual systems must hold; and vice versa, if we solve these systems, we obtain the optimal solution. Note that in the primal system there are $|\mathcal{A}|+1$ unknowns and $|\mathcal{E}|+1$ equations; in the dual system, there are

$|\mathcal{E}|+1$ unknowns and $|\mathcal{A}|+1$ equations. Therefore, to satisfy both systems, we must have $|\mathcal{A}|=|\mathcal{E}|$.

Initial conditions

Initially, the algorithm starts from $C=0$, where $s_j=0$ and $f(\boldsymbol{x}_i)=\beta_0$. Therefore, (12)–(15) is reduced to an optimization problem with only one parameter β_0. Since $y_1,\dots,y_n$ are real numbers, we assume that their values are distinct. Let $y_{(1)},y_{(2)},\dots,y_{(n)}$ be the output values in ascending order, then the solution for the initial β_0 is:

$$\widehat{\beta_0}=y_{(\lceil n/2\rceil)},$$

where $\lceil\cdot\rceil$ is the ceiling function. According to the definitions of $\mathcal{L},\mathcal{R}$ and $\mathcal{E}$, we have $|\mathcal{L}|=\lceil n/2\rceil-1$, $|\mathcal{R}|=n-\lceil n/2\rceil$ and $|\mathcal{E}|=1$. Based on the derived relationships, if $y_i>\widehat{\beta_0}$ then $\gamma_i=1$; if $y_i<\widehat{\beta_0}$ then $\gamma_i=-1$. Let i^* denote the point in $\mathcal{E}$, i.e., $\mathcal{E}=\{i^*\}$. From (19), we get: if n is odd, then $\gamma_{i^*}=0$; otherwise, $\gamma_{i^*}=1$.

When C increases from 0, some s_j will become non-zero and join $\mathcal{A}$. If C is increased by an infinitesimal amount: $C=\Delta C\to 0^+$, exactly one s_j will be added into $\mathcal{A}$, because we currently have $|\mathcal{E}|=1$, and $|\mathcal{A}|=|\mathcal{E}|$ has to hold. To determine which s_j will join $\mathcal{A}$, we consider the following problem:

$$\min_{\beta_0,j\in\{1,\dots,m\}}\sum_{i=1}^{n}|y_i-\beta_0-\Delta C\cdot\widehat{f_j}(\boldsymbol{x}_i)|. \tag{33}$$

According to the derived relationships in “Problem setup”, problem (33) is equivalent to:

$$\min_{\beta_0,j\in\{1,\dots,m\}}\sum_{i=1}^{n}\gamma_i(y_i-\beta_0-\Delta C\cdot\widehat{f_j}(\boldsymbol{x}_i)). \tag{34}$$

Let $\mathcal{A}=\{j^*\}$, and j^* is determined by:

$$j^*=\operatorname{argmax}_{j\in\{1,\dots,m\}}\sum_{i=1}^{n}\gamma_i\widehat{f_j}(\boldsymbol{x}_i).$$

Now, we have initial $\widehat{\beta_0}$, $\mathcal{A},\mathcal{L},\mathcal{R}$ and $\mathcal{E}$, and we also need the initial value for the non-negative dual variable η as $\Delta C\to 0^+$. According to (32), it is determined by:

$$\eta=\sum_{i=1}^{n}\gamma_i\widehat{f_{j^*}}(\boldsymbol{x}_i).$$

In the following, we use the superscript ℓ to indicate the iteration number. After the initial setup, $\ell=0$ and $\mathcal{L}^\ell,\mathcal{R}^\ell,\mathcal{E}^\ell,\mathcal{A}^\ell,s_j^\ell,\gamma_i^\ell,\beta_0^\ell$ and η^ℓ are all available.

Solution path

When C increases by an infinitesimal amount, the sets $\mathcal{L},\mathcal{R},\mathcal{E}$ and $\mathcal{A}$ do not change due to their continuity with respect to C. Therefore the structure of the primal system

(29)–(30) does not change. The right derivatives of the primal variables with respect to C can be obtained by:

$$\frac{\Delta\beta_0}{\Delta C} + \sum_{j\in\mathcal{A}} \frac{\Delta s_j}{\Delta C} \widehat{f}_j(\boldsymbol{x}_i) = 0, \quad i \in \mathcal{E} \tag{35}$$

$$\sum_{j\in\mathcal{A}} \frac{\Delta s_j}{\Delta C} = 1. \tag{36}$$

Since $|\mathcal{E}| = |\mathcal{A}|$, $\frac{\Delta\beta_0}{\Delta C}$ and $\frac{\Delta s_j}{\Delta C}$ are constant and can be uniquely determined, assuming the linear system is non-singular. Therefore, if C increases by a small enough amount, the primal variables and fitted values are linear functions of C:

$$\beta_0 = \beta_0^\ell + \frac{\Delta\beta_0}{\Delta C}(C - C^\ell),$$

$$s_j = s_j^\ell + \frac{\Delta s_j}{\Delta C}(C - C^\ell),$$

$$f(\boldsymbol{x}_i) = f^\ell(\boldsymbol{x}_i) + \frac{\Delta\beta_0}{\Delta C}(C - C^\ell) + \sum_{j\in\mathcal{A}} \frac{\Delta s_j}{\Delta C} \widehat{f}_j(\boldsymbol{x}_i)(C - C^\ell).$$

If the increase in C is large enough, some of the sets $\mathcal{L}$, $\mathcal{R}$, $\mathcal{E}$ and $\mathcal{A}$ may change. Specifically, one of the following two *events* may occur in the primal system:

- a point leaves $\mathcal{L}$ or $\mathcal{R}$ and joins $\mathcal{E}$, i.e., its residual $y_i - f(\boldsymbol{x}_i)$ changes from non-zero to zero; or
- an active s_j becomes inactive, i.e., s_j reduces from a positive value to zero.

To determine ΔC for the first *event*, we calculate for every $i \notin \mathcal{E}$:

$$\Delta\widehat{C}_i = \max\left\{-\frac{f^\ell(\boldsymbol{x}_i)}{\frac{\Delta\beta_0}{\Delta C} + \sum_{j\in\mathcal{A}} \frac{\Delta s_j}{\Delta C} \widehat{f}_j(\boldsymbol{x}_i)}, 0\right\}.$$

For the second *event*, we calculate

$$\Delta\widetilde{C}_j = \max\left\{-s_j^\ell / \frac{\Delta s_j}{\Delta C}, 0\right\}, \quad \text{for } j \in \mathcal{A}.$$

Therefore, the overall ΔC is

$$\Delta C = \min\{\Delta\widehat{C}_i\ (i \notin \mathcal{E}, \Delta\widehat{C}_i > 0),\ \Delta\widetilde{C}_j\ (j \in \mathcal{A}, \Delta\widetilde{C}_j > 0)\},$$

and we get the updated variable $s_j^{\ell+1}$, $\beta_0^{\ell+1}$ and $C^{\ell+1}$. If the first *event* occurs, $|\mathcal{E}|$ will increase by 1; if the second *event* occurs, $|\mathcal{A}|$ will decrease by 1. Therefore, it always leads to $|\mathcal{E}| = |\mathcal{A}| + 1$ after an *event* occurs. Since the relation $|\mathcal{E}| = |\mathcal{A}|$ does not hold, we have to restore it by taking one of the two *actions*:

- removing a point from $\mathcal{E}$; or
- adding a new s_j into $\mathcal{A}$.

Solving the dual system helps us determine which *action* to take.

Since we now have $|\mathcal{E}| = |\mathcal{A}| + 1$, in the dual system (31)–(32), there is one more unknown than the number of equations. In other words, there is one degree of freedom in the system, and we can express γ_i $(i \in \mathcal{E})$ as a linear function of η. The derivatives of γ_i with respect to η can be solved by the following equations:

$$\sum_{i \in \mathcal{E}} \frac{\Delta \gamma_i}{\Delta \eta} = 0, \tag{37}$$

$$\sum_{i \in \mathcal{E}} \frac{\Delta \gamma_i}{\Delta \eta} \widehat{f}_j(\boldsymbol{x}_i) = 1, \quad j \in \mathcal{A}. \tag{38}$$

In (37)–(38), there are $|\mathcal{E}|$ unknowns and $|\mathcal{A}|+1$ equations, so the values of $\frac{\Delta\gamma_i}{\Delta\eta}$ $(i \in \mathcal{E})$ can be uniquely determined. To determine the $\Delta\eta$ for taking the first *action* (removing a point from $\mathcal{E}$), we calculate for every $i \in \mathcal{E}$:

$$\Delta\widehat{\eta}_i = \begin{cases} -\gamma_i^\ell / \frac{\Delta\gamma_i}{\Delta\eta}, & \text{if } \frac{\Delta\gamma_i}{\Delta\eta} > 0, \\ (1-\gamma_i^\ell)/\frac{\Delta\gamma_i}{\Delta\eta}, & \text{if } \frac{\Delta\gamma_i}{\Delta\eta} < 0, \\ -\infty, & \text{otherwise.} \end{cases}$$

On the other hand, since (32) must hold for $j \in \mathcal{A}$, we can determine the $\Delta\eta$ for taking the second *action* (adding a new s_j into $\mathcal{A}$) by calculating:

$$\Delta\widetilde{\eta}_j = \begin{cases} \frac{\eta^\ell - \sum_{i=1}^n \gamma_i^\ell \widehat{f}_j(\boldsymbol{x}_i)}{\sum_{i\in\mathcal{E}} \frac{\Delta\gamma_i}{\Delta\eta}\widehat{f}_j(\boldsymbol{x}_i) - 1}, & \text{if } \sum_{i\in\mathcal{E}} \frac{\Delta\gamma_i}{\Delta\eta}\widehat{f}_j(\boldsymbol{x}_i) - 1 < 0, \\ -\infty, & \text{otherwise,} \end{cases}$$

for every $j \notin \mathcal{A}$. When η reduces to $\eta^\ell + \Delta\widetilde{\eta}_j$, s_j will join $\mathcal{A}$. Since η can only decrease, the overall $\Delta\eta$ is

$$\Delta\eta = \max\left\{\Delta\widehat{\eta}_i (i \in \mathcal{E}), \Delta\widetilde{\eta}_j (j \notin \mathcal{A})\right\}, \tag{39}$$

and we get the updated dual variables $\gamma_i^{\ell+1}$, $\eta^{\ell+1}$ and the sets $\mathcal{L}^{\ell+1}$, $\mathcal{R}^{\ell+1}$, $\mathcal{E}^{\ell+1}$ and $\mathcal{A}^{\ell+1}$.

Once an *action* is taken, we restore $|\mathcal{E}| = |\mathcal{A}|$. The algorithm keeps increasing C and alternates between reaching the next *event* and taking an *action*, until η reduces to 0.

Between any two consecutive events the optimal solutions are linear in C; after an *event* occurs, the derivatives with respect to C are changed. Therefore, the solution path is piecewise linear in C, and each *event* corresponds to a kink on the path. Figure 5 shows the solution path for a randomly generated data set with 50 samples and 10 candidate radial basis kernels. For this particular data set, it takes 48 iterations to compute the entire solution path. We can see that the 10 candidate kernels are selected one by one as the algorithm proceeds (or the value of C increases). The algorithm ends up with $\widehat{s}_j = 1$ for all j, which corresponds to the scenario where the constraint $\sum_{j=1}^m s_j \leq C$ is fully released.

For clarity, we summarize our path algorithm in Table 3.

Computational cost

Our kernel learning method consists of two steps: solving a QP problem and running our solution path algorithm. The first problem has been extensively addressed in the literature, e.g., the SMO algorithm (Platt 1999; Fan et al. 2005) and the solution path algorithm (Gunter and Zhu 2006). In the second step, our solution path algorithm solves a series of LP problems. The major computation at each iteration is to solve the primal system (35)–(36) and the dual system (37)–(38). Since $|\mathcal{A}|$ is upper bounded by m, the computational complexity

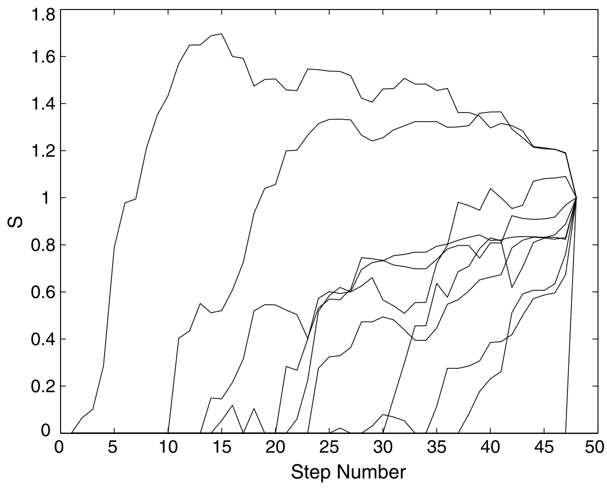

Fig. 5 Piecewise linear solution path for a randomly generated data set

Table 3 Outline of the algorithm

```
Initialization: Calculate $\beta_0^0$ and $\eta^0$; determine $\mathcal{A}^0, \mathcal{L}^0, \mathcal{R}^0, \mathcal{E}^0$ and set $\ell = 0$;
Step 1: Solve the primal system (35)–(36);
Step 2: Calculate $\Delta C$ ; identify the next event for the primal system
        and update the primal variables;
Step 3: Solve the dual system (37)–(38);
Step 4: Calculate $\Delta\eta$ ; identify the next event for the dual system
        and update the dual variables;
Step 5: If the stopping criterion is met, then stop the algorithm;
Step 6: Otherwise, let $\ell = \ell + 1$ and goto Step 1.
```

seems to be $O(m^3)$ for each iteration. However, between any two consecutive iterations, only one element in the sets $\mathcal{L}$, $\mathcal{E}$, $\mathcal{R}$ and $\mathcal{A}$ tends to get changed, so using inverse updating and downdating the computational complexity can be reduced to $O(m^2)$. It is difficult to predict the total number of iterations for completing the algorithm, but our experience suggests that $O(\max(n,m))$ is a reasonable estimate. The heuristic is that the algorithm needs m iterations to include all the s_j's into $\mathcal{A}$ and n iterations to move all the points to $\mathcal{E}$. Therefore, the overall computational cost of our solution path algorithm is approximately $O(\max(n,m)m^2)$.

References

Bach, F., Lanckriet, G., & Jordan, M. (2004). Multiple kernel learning, conic duality, and the SMO algorithm. In *Proceedings of the 21st international conference on machine learning* (p. 6).

Bradley, P., & Mangasarian, O. (1998). Feature selection via concave minimization and support vector machines. In *Machine learning proceedings of the fifteenth international conference* (pp. 82–90).

Breiman, L. (1995). Better subset regression using the nonnegative garrote. *Technometrics*, *37*, 373–384.

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

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

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

Cristianini, N., Kandola, J., Elisseeff, A., & Taylor, J. (2006). *On kernel target alignment. Innovations in machine learning: theory and applications*, Berlin: Springer.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *The Annals of Statistics*, *32*, 407–499.

Fama, E. (1970). Efficient capital markets: a review of theory and empirical work. *Journal of Finance*, *25*, 383–417.

Fama, E. (1991). Efficient capital markets: II. *Journal of Finance*, *46*, 1575–1617.

Fan, R., Chen, P., & Lin, C. (2005). Working set selection using second order information for training support vector machines. *Journal of Machine Learning Research*, *6*, 1889–1918.

Friedrichs, F., & Igel, C. (2005). Evolutionary tuning of multiple SVM parameters. *Neurocomputing*, *64*, 107–117.

Fung, G., Dundar, M., Bi, J., & Rao, B. (2004). A fast iterative algorithm for fisher discriminant using heterogeneous kernels. In *Proceedings of the 21st international conference on machine learning* (p. 40).

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

Gunter, L., & Zhu, J. (2006). Computing the solution path for the regularized support vector regression. In *Advances in neural information processing systems* (pp. 483–490).

Hastie, T., Rosset, S., Tibshirani, R., & Zhu, J. (2004). The entire regularization path for the support vector machine. *Journal of Machine Learning Research*, *5*, 1391–1415.

Hirshleifer, D. (2001). Investor psychology and asset pricing. *The Journal of Finance*, *4*, 1533.

Huang, W., Nakamori, Y., & Wang, S. (2005). Forecasting stock market movement direction with using support vector machine. *Computers and Operations Research*, *32*, 2513–2522.

Keerthi, S., Sindhwani, V., & Chapelle, O. (2007). An efficient method for gradient-based adaptation of hyperparameters in SVM models. In *Advances in neural information processing systems* (pp. 217–224).

Kim, K. (2003). Financial time series forecasting using support vector machines. *Neurocomputing*, *55*, 307–319.

Kim, S., Magnani, A., & Boyd, S. (2006). Optimal kernel selection in kernel fisher discriminant analysis. In *Proceedings of the 23rd international conference on machine learning* (pp. 465–472).

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

Lo, A., & MacKinlay, C. (2001). *A non-random walk down wall street*. Princeton: Princeton University Press.

Lo, A., Mamaysky, H., & Wang, J. (2000). Foundations of technical analysis: computational algorithms, statistical inference, and empirical implementation. *Journal of Finance*, *55*, 1705–1765.

Malkiel, J. (1973). *A random walk down wall street*. New York: W.W. Norton & Company.

Nguyen, H., Ohn, S., & Choi, W. (2007). Combined kernel function for support vector machine and learning method based on evolutionary algorithm. In *Proceedings of the 11th international conference on neural information processing* (pp. 1273–1278).

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

Platt, J. (1999). Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods—support vector learning*. Cambridge: MIT Press.

Rosset, S., & Zhu, J. (2007). Piecewise linear regularized solution paths. *The Annals of Statistics*, *35*, 1012–1030.

Smola, A., & Schoelkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*, 199–222.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society, Series B*, *58*, 267–288.

Vapnik, V. (1995). *The nature of statistical learning theory*. Berlin: Springer.

Wang, G., Yeung, D., & Lochovsky, F. (2007). A kernel path algorithm for support vector machines. In *Proceedings of the 24th international conference on machine learning* (pp. 951–958).

Wang, X., Chen, S., Lowe, D., & Harris, C. (2006). Sparse support vector regression based on orthogonal forward selection for the generalised kernel model. *Neurocomputing*, *70*, 462–474.