PERGAMON

Omega 29 (2001) 309–317

omega
The International Journal of Management Science

www.elsevier.com/locate/dsw

Application of support vector machines in financial time series forecasting

Francis E.H. Tay *, Lijuan Cao

Department of Mechanical Engineering, National University of Singapore, 10 Kent Ridge Cresent, Singapore 119260, Singapore

Received 17 April 2000; accepted 31 March 2001

Abstract

This paper deals with the application of a novel neural network technique, support vector machine (SVM), in financial time series forecasting. The objective of this paper is to examine the feasibility of SVM in financial time series forecasting by comparing it with a multi-layer back-propagation (BP) neural network. Five real futures contracts that are collated from the Chicago Mercantile Market are used as the data sets. The experiment shows that SVM outperforms the BP neural network based on the criteria of normalized mean square error (NMSE), mean absolute error (MAE), directional symmetry (DS) and weighted directional symmetry (WDS). Since there is no structured way to choose the free parameters of SVMs, the variability in performance with respect to the free parameters is investigated in this study. Analysis of the experimental results proved that it is advantageous to apply SVMs to forecast financial time series.

Keywords: Support vector machines; Structural risk minimization principle; BP neural network; Generalization

1. Introduction

Financial time series forecasting is regarded as one of the most challenging applications of modern time series forecasting. As explained by Deboeck and Yaser [1,2], financial time series are inherently noisy, non-stationary and deterministically chaotic. The noisy characteristic refers to the unavailability of complete information from the past behavior of financial markets to fully capture the dependency between future and past prices. The information that is not included in the model is considered as noise. The non-stationary characteristic implies that the distribution of financial time series is changing over time. By deterministically chaotic, one means that financial time series are short-term random but long-term deterministic.

In recent years, neural networks have been successfully used for modeling financial time series [3–5]. Neural networks are universal function approximators that can map any non-linear function without a priori assumptions about the properties of the data [6]. Unlike traditional statistical models, neural networks are data-driven, non-parametric weak models, and they let "the data speak for themselves". Consequently, neural networks are less susceptible to the problem of model misspecification as compared to most of the parametric models. Neural networks are also more noise tolerant, having the ability to learn complex systems with incomplete and corrupted data. In addition, they are more flexible, having the capability to learn dynamic systems through a retraining process using new data patterns. So neural networks are more powerful in describing the dynamics of financial time series in comparison to traditional statistical models [7–9].

In the area of financial forecasting, the most popular neural network model is the back-propagation (BP) neural network due to its simple architecture yet powerful problem-solving ability. However, the BP neural network suffers from a number of weaknesses which include the need for a large number of controlling parameters, difficulty in obtaining a stable solution and the danger of

* Corresponding author. Tel.: +65-874-6818; fax: +65-779-1459.

E-mail addresses: mpetayeh@nus.edu.sg (F.E.H. Tay), engp8663@nus.edu.sg (L. Cao).

0305-0483/01/$ - see front matter
PII: S0305-0483(01)00026-3

over fitting. The over-fitting problem is a critical issue that usually leads to poor generalization because the neural network has too large a capacity which causes it to capture not only the useful information contained in the training data but also unwanted noises. As a result, it will end up only memorizing the training data and generalizing poorly to the out-of-sample data. This issue of generalization has long been a concern to researchers. The typical approach is to use a cross-validation data set but this method usually involves a substantial amount of computation. A variety of procedures [10–14] have been explored for enhancing the generalization ability of neural networks.

Recently, a novel neural network algorithm, called support vector machine (SVM), was developed by Vapnik and his co-workers [15]. Unlike most of the traditional neural network models which implement the *empirical risk minimization principle*, SVMs implement the *structural risk minimization principle* which seeks to minimize an upper bound of the generalization error rather than minimize the training error. This induction principle is based on the fact that the generalization error is bounded by the sum of the training error and a confidence interval term that depends on the Vapnik–Chervonenkis (VC) dimension. Based on this principle, SVMs achieve an optimum network structure by striking a right balance between the empirical error and the VC-confidence interval. This eventually results in better generalization performance than other neural network models. Another merit of SVMs is that the training of SVMs is equivalent to solving a linearly constrained quadratic programming. This means that the solution of SVMs is unique, optimal and absent from local minima, unlike other networks' training which requires non-linear optimization thus running the danger of getting stuck in a local minima. Originally, SVMs have been developed for pattern recognition problems [16,17]. However, with the introduction of Vapnik's ε-insensitive loss function, SVMs have been extended to solve non-linear regression estimation problems and they have been shown to exhibit excellent performance [18–20].

The objectives of this paper are firstly, to examine the feasibility of applying SVM in financial forecasting by comparing it with a BP neural network, and secondly, to investigate the functional characteristics of SVMs in financial forecasting. The functional characteristics are obtained through the selection of the free parameters of SVMs. Since there is a lack of a structured way to choose the free parameters of SVMs, experiments are carried out to investigate the variability in performance with respect to the free parameters.

This paper consists of five sections. Section 2 provides a brief introduction to SVMs while Section 3 contains the experimental data. The technique used for data preprocessing and statistical performance metrics are also presented in Section 3. Section 4 discusses the experimental results followed by the conclusions drawn from this study in the last section.

2. Theory of SVMS in regression approximation

Compared to other neural network regressors, there are three distinct characteristics when SVMs are used to estimate the regression function. First of all, SVMs estimate the regression using a set of linear functions that are defined in a high dimensional space. Secondly, SVMs carry out the regression estimation by risk minimization where the risk is measured using Vapnik's ε-insensitive loss function. Thirdly, SVMs use a risk function consisting of the empirical error and a regularization term which is derived from the structural risk minimization principle.

Given a set of data points $G = \{(x_i, d_i)\}_i^n$ (x_i is the input vector, d_i is the desired value and n is the total number of data patterns), SVMs approximate the function using the following:

$$y = f(x) = w\phi(x) + b, \tag{1}$$

where $\phi(x)$ is the high dimensional feature space which is non-linearly mapped from the input space x. The coefficients w and b are estimated by minimizing

$$R_{\mathrm{SVMs}}(C) = C\frac{1}{n}\sum_{i=1}^{n} L_\varepsilon(d_i, y_i) + \frac{1}{2}||w||^2, \tag{2}$$

$$L_\varepsilon(d, y) = \begin{cases} |d - y| - \varepsilon & |d - y| \geqslant \varepsilon \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

In the regularized risk function given by Eq. (2), the first term $C(1/n)\sum_{i=1}^{n} L_\varepsilon(d_i, y_i)$ is the empirical error (risk). They are measured by the ε -insensitive loss function given by Eq. (3). This loss function provides the advantage of enabling one to use sparse data points to represent the decision function given by Eq. (1). The second term $\frac{1}{2}||w||^2$, on the other hand, is the regularization term. C is referred to as the regularized constant and it determines the trade-off between the empirical risk and the regularization term. Increasing the value of C will result in the relative importance of the empirical risk with respect to the regularization term to grow. ε is called the tube size and it is equivalent to the approximation accuracy placed on the training data points. Both C and ε are user-prescribed parameters.

To obtain the estimations of w and b, Eq. (2) is transformed to the primal function given by Eq. (4) by introducing the positive slack variables ζ_i and ζ_i^* as follows:

$$\begin{aligned} \text{Minimize} \quad & R_{\mathrm{SVMs}}(w, \zeta^{(*)}) = \tfrac{1}{2}||w||^2 + C\sum_{i=1}^{n}(\zeta_i + \zeta_i^*) \\ \text{Subjected to} \quad & d_i - w\phi(x_i) - b_i \leqslant \varepsilon + \zeta_i, \\ & w\phi(x_i) + b_i - d_i \leqslant \varepsilon + \zeta_i^*, \quad \zeta^{(*)} \geqslant 0. \end{aligned} \tag{4}$$

Finally, by introducing Lagrange multipliers and exploiting the optimality constraints, the decision function given by Eq. (1) has the following explicit form [15]:

$$f(x, a_i, a_i^*) = \sum_{i=1}^{n}(a_i - a_i^*)K(x, x_i) + b. \tag{5}$$

2.1. Lagrange multipliers

In Eq. (5), a_i and a_i^* are the so-called Lagrange multipliers. They satisfy the equalities $a_i * a_i^* = 0$, $a_i \geqslant 0$ and $a_i^* \geqslant 0$ where $i = 1, \ldots, n$, and are obtained by maximizing the dual function of Eq. (4) which has the following form:

$$R(a_i, a_i^*) = \sum_{i=1}^{n} d_i(a_i - a_i^*) - \varepsilon \sum_{i=1}^{n}(a_i + a_i^*) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(a_i - a_i^*)(a_j - a_j^*)K(x_i, x_j), \quad (6)$$

with the constraints

$$\sum_{i=1}^{n}(a_i - a_i^*) = 0,$$

$$0 \leqslant a_i \leqslant C, \qquad i = 1, 2 \ldots, n,$$

$$0 \leqslant a_i^* \leqslant C, \qquad i = 1, 2, \ldots, n.$$

Based on the Karush–Kuhn–Tucker (KKT) conditions of quadratic programming, only a certain number of coefficients $(a_i - a_i^*)$ in Eq. (5) will assume non-zero values. The data points associated with them have approximation errors equal to or larger than ε and are referred to as support vectors. These are the data points lying on or outside the ε-bound of the decision function. According to Eq. (5), it is evident that support vectors are the only elements of the data points that are used in determining the decision function as the coefficients $(a_i - a_i^*)$ of other data points are all equal to zero. Generally, the larger the ε, the fewer the number of support vectors and thus the sparser the representation of the solution. However, a larger ε can also depreciate the approximation accuracy placed on the training points. In this sense, ε is a trade-off between the sparseness of the representation and closeness to the data.

2.2. Kernel function

$K(x_i, x_j)$ is defined as the kernel function. The value of the kernel is equal to the inner product of two vectors X_i and X_j in the feature space $\phi(x_i)$ and $\phi(x_j)$, that is, $K(x_i, x_j) = \phi(x_i) * \phi(x_j)$. The elegance of using the kernel function is that one can deal with feature spaces of arbitrary dimensionality without having to compute the map $\phi(x)$ explicitly. Any function satisfying Mercer's condition [15] can be used as the kernel function. The typical examples of kernel function are the polynomial kernel $K(x, y) = (x * y + 1)^d$ and the Gaussian kernel $K(x, y) = \exp(-1/\delta^2(x - y)^2)$ where d is the degree of polynomial kernel and δ^2 is the bandwidth of the Gaussian kernel. The kernel parameter should be carefully chosen as it implicitly defines the structure of the high dimensional feature space $\phi(x)$ and thus controls the complexity of the final solution.

From the implementation point of view, training SVMs is equivalent to solving a linearly constrained quadratic programming (QP) with the number of variables twice as that of the training data points. The sequential minimal optimization algorithm propounded by Scholkopf and Smola [21,22] is reported to be very effective in training SVMs for solving the regression problem.

3. Experimental settings

3.1. Data sets

Five real futures contracts collated from the Chicago Mercantile Market are examined in the experiment. They are the Standard&Poor 500 stock index futures (CME-SP), United States 30-year government bond (CBOT-US), United States 10-year government bond (CBOT-BO), German 10-year government bond (EUREX-BUND) and French government stock index futures (MATIF-CAC40). The corresponding time periods used are listed in Table 1. The daily closing prices are used as the data sets.

The original closing price is transformed into a five-day relative difference in percentage of price (RDP). As mentioned by Thomason [23], there are four advantages in applying this transformation. The most prominent advantage is that the distribution of the transformed data will become more symmetrical and will follow more closely a normal distribution as illustrated in Fig. 1. This modification to the data distribution will improve the predictive power of the neural network.

The input variables are determined from four lagged RDP values based on 5-day periods (RDP-5, RDP-10, RDP-15 and RDP-20) and one transformed closing price (EMA15) which is obtained by subtracting a 15-day exponential moving average from the closing price. The subtraction is performed to eliminate the trend in price as the maximum value and the minimum value is in the ratio of about 2 : 1 in all of the five data sets. The optimal length of the moving day is not critical but it should be longer than the forecasting horizon of 5 days [24]. EMA15 is used to maintain as much information as contained in the original closing price as possible, since the application of the RDP transform to the original closing price may remove some useful information. The output variable RDP+5 is obtained by first smoothing the closing price with a 3-day exponential moving average because the application of a smoothing transform to the dependent variable generally enhances the prediction

Table 1
Five futures contracts and their corresponding time period

Futures	Time period
CME-SP	30/12/1992–30/07/1996
CBOT-US	01/01/1993–01/08/1996
CBOT-BO	01/01/1993–01/08/1996
EUREX-BUND	01/01/1993–01/08/1996
MATIF-CAC40	01/06/1995–01/02/1999

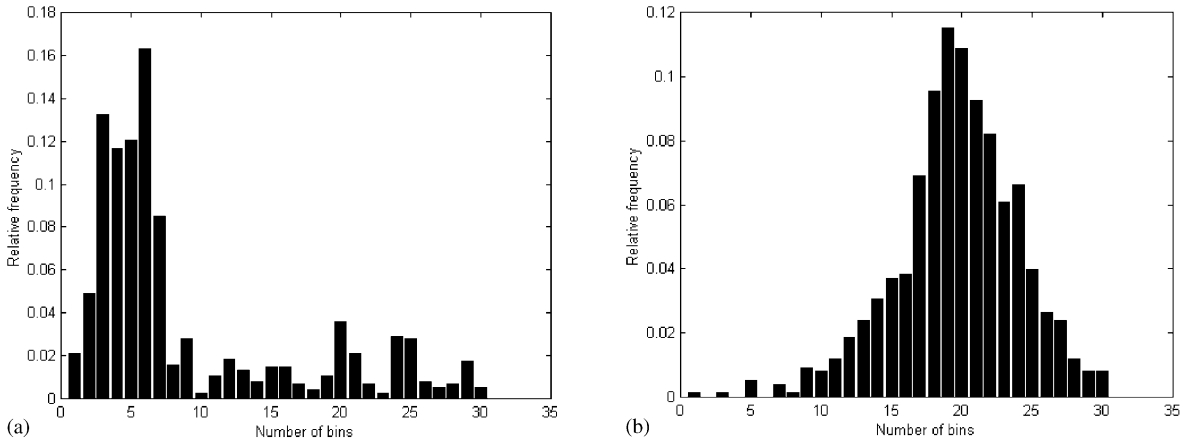

Fig. 1. Histograms of (a) CME-SP daily closing price and (b) RDP+5. RDP+5 values have a more symmetrical and normal distribution.

Table 2
Input and output variables[a]

Indicator	Calculation
Input variables	
EMA15	$P(i) - \overline{\mathrm{EMA}_{15}(i)}$
RDP-5	$(p(i) - p(i-5))/p(i-5) * 100$
RDP-10	$(p(i) - p(i-10))/p(i-10) * 100$
RDP-15	$(p(i) - p(i-15))/p(i-15) * 100$
RDP-20	$(p(i) - p(i-20))/p(i-20) * 100$
Output variable	
RDP+5	$(\overline{p(i+5)} - \overline{p(i)})/\overline{p(i)} * 100$
	$\overline{p(i)} = \overline{\mathrm{EMA}_3(i)}$

[a] $\mathrm{EMA}_n(i)$ is the n-day exponential moving average of the ith day; and $p(i)$ is the closing price of the ith day.

Table 3
Performance metrics and their calculations[a]

Metrics	Calculation
NMSE	$\mathrm{NMSE} = 1/(\delta^2 n) * \sum_{i=1}^{n}(a_i - p_i)^2$
	$\delta^2 = 1/(n-1) * \sum_{i=1}^{n}(a_i - \bar{a})^2$
MAE	$\mathrm{MAE} = 1/n * \sum_{i=1}^{n} \lvert a_i - p_i \rvert$
DS	$\mathrm{DS} = 100/n * \sum_{i=1}^{n} d_i$
	$d_i = \begin{cases} 1 & (a_i - a_{i-1})(p_i - p_{i-1}) \geqslant 0 \\ 0 & \text{otherwise} \end{cases}$
WDS	$\mathrm{WDS} = \sum_{i=1}^{n} d_i \lvert a_i - p_i \rvert / \sum_{i=1}^{n} d_i' \lvert a_i - p_i \rvert$
	$d_i = \begin{cases} 0 & (a_i - a_{i-1})(p_i - p_{i-1}) \geqslant 0 \\ 1 & \text{otherwise} \end{cases}$
	$d_i' = \begin{cases} 1 & (a_i - a_{i-1})(p_i - p_{i-1}) \geqslant 0 \\ 0 & \text{otherwise} \end{cases}$

[a] a_i and p_i are the actual values and predicted values.

performance of the neural network [24]. The calculations for all the indicators are given in Table 2.

The long left tail in Fig. 1(b) indicates that there are outliers in the data set. Since outliers may make it difficult or time-consuming to arrive at an effective solution for the SVMs, RDP values beyond the limits of ± 2 standard deviations are selected as outliers. They are replaced with the closest marginal values. Another pre-processing technique used in this study is data scaling. All the data points are scaled into the range of $[-0.9, 0.9]$ as the data points include both positive values and negative values. All of the five data sets are partitioned into three parts according to the time sequence. The first part is used for training, the second part used for validation is to select optimal parameters for the SVMs and to prevent the over-fitting problem found in the BP neural network. The last part is used for the purpose of testing. There are a total of 907 data patterns in the training set, 200 data patterns in both the validation set and the test set in all the data sets.

3.2. Performance criteria

The prediction performance is evaluated using the following statistical metrics, namely, the normalized mean squared error (NMSE), mean absolute error (MAE), directional symmetry (DS) and weighted directional symmetry (WDS). The definitions of these criteria can be found in Table 3. NMSE and MAE are the measures of the deviation between the actual and predicted values. The smaller the values of NMSE and MAE, the closer are the predicted time series values to the actual values. DS provides an indication of the correctness of the predicted direction of RDP+5 given in the form of percentages (a large value suggests a better predictor). The weighted directional symmetry measures both the magnitude of the prediction error and the direction. It penalizes the errors related to incorrectly predicted direction and rewards those associated with correctly predicted direction. The smaller the value of WDS, the better is the

forecasting performance in terms of both the magnitude and direction.

4. Experimental results

4.1. Results of SVMs and BP

In this investigation, the Gaussian function is used as the kernel function of the SVMs because Gaussian kernels tend to give good performance under general smoothness assumptions. Consequently, they are especially useful if no additional knowledge of the data is available [22]. This is also demonstrated in the experiment by comparing the results obtained using the Gaussian kernel with results obtained using the polynomial kernel. The polynomial kernel gives inferior results and takes a longer time in the training of SVMs. The choice of $\delta^2 = 10$, $C = 100$ and $\varepsilon = 0.001$ is because these values produced the best possible results according to the validation set. The sequential minimal optimization algorithm for solving the regression problem is implemented in this experiment and the program is developed using VC^{++} language [25].

A standard three-layer BP network is used as a benchmark. There are five nodes in the input layer which is equal to the number of indicators. The output node is equal to 1, whereas the number of hidden nodes is determined by using the formula, $W < = M/15$, where W is the number of interconnection weights that satisfies the following equality:

$$W = (I + O) * H, \tag{7}$$

where M is the number of training examples, I the number of input nodes, O the number of output nodes, and H the number of hidden nodes.

The size of the network is controlled by ensuring that the ratio of the number of training samples to the number of weights is equal to or larger than 15. Based on Eq. (7), the maximal number of hidden nodes is 10. Varying the number of hidden nodes from 4 to 10, it was determined that using eight hidden nodes in the BP network gives the best performance using the same four criteria. The selection of the learning rate as 0.01 and the momentum term as 0.9 is because a BP network with these settings as the learning parameters has the best prediction performance with the least number of epochs [6]. The hidden nodes use the sigmoid transfer function and the output node uses the linear transfer function.

In the training of the BP network, the number of epochs is first chosen as 6000 as there is no prior knowledge of this value before the experiment. The behavior of the NMSE is given in Fig. 2(a). In BP, it is evident that the NMSE on the training set decreases monotonically during the entire training period. In contrast, the NMSE on the validation set decreases for the first few hundreds of epochs but increases for the remaining epochs. This indicates that over-fitting has occurred in the BP network. Hence, in the later training, the BP network is first trained using 500 epochs since the NMSE of the validation set decreases in that period. The validation set is presented to the network at every 10 epochs thereafter. If the NMSE on the validation set shows any tendency for increasing, the training of the BP network will be stopped. This reduces the possibility of over-fitting. In comparison, for the SVMs, the NMSE on both the training and the validation set fluctuate during the initial training period but gradually converge to a constant value, as is illustrated in Fig. 2(b).

The results of SVMs and BP on the test set are collated and the averages of the best five records obtained in 30 trials are given in Table 4. It can be observed that in the CME-SP, CBOT-US, CBOT-BO and MATIF-CAC40 futures, SVMs have much smaller NMSE, MAE and WDS but larger DS than BP. This indicates that SVMs significantly outperform BP in these futures, and they can forecast more accurately and capture the turning points better than BP. In the EUREX-BUND contract, the difference in performance is smaller than that of the other four but nevertheless, SVMs still outperform BP. Therefore, it can be concluded that SVMs provide a promising technique in financial times series forecasting.

4.2. Sensitivity of SVMs to parameters

In the above experiments, the kernel parameter δ^2, C and ε are selected based on the validation set. Making use of a validation set is still not a structured way to select the free parameters as this iterative process involves expensive computation. In this section, the NMSE and the number of support vectors with respect to the three free parameters are investigated. Only the results for CME-SP are illustrated as the same can be applied to the other data sets.

Fig. 3(a) gives the NMSE of SVMs at various δ^2, in which C and ε are, respectively, fixed at 10 and 0.001. The figure shows that the NMSE on the training set increases with δ^2. On the other hand, the NMSE on the test set decreases initially but subsequently increases as δ^2 increases. This indicates that too small a value of δ^2 (0.1–1) causes SVMs to over-fit the training data while too large a value of δ^2 (100–100,000) causes SVMs to under-fit the training data. An appropriate value for δ^2 would be between 1 and 100. In this aspect, it can be said that δ^2 plays an important role on the generalization performance of SVMs. Fig. 3(b) shows that the number of support vector decreases initially and then increases with δ^2 as most of the training data points are converged to the support vectors in the over-fitting and under-fitting cases.

Fig. 4 gives the results of various C where δ^2 is chosen as 10 based on the last experiment and ε is still fixed at 0.001. It can be observed that the NMSE on the training set decreases monotonically as C increases. In contrast, the NMSE on the test set decreases when C increases from 0.1 to 10 and maintains an almost constant value as C increases from 10 to 100. However, it starts to increase again when C increases

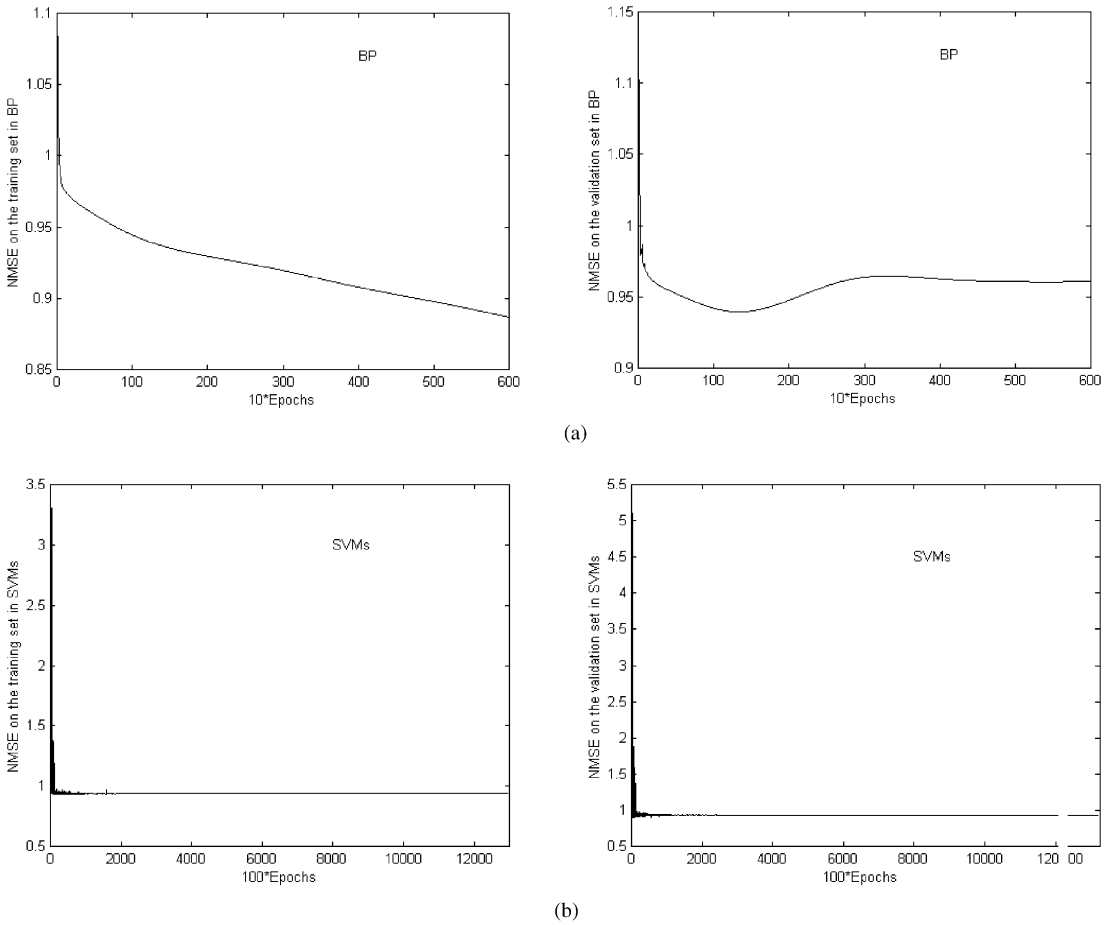

Fig. 2. (a) The behavior of NMSE in BP. The NMSE on the training set decreases for the entire period (left) while the NMSE on the validation set (right) decreases in the first hundreds of epochs but subsequently increases in the remaining epochs. This suggests that over-fitting has occurred in BP. (b) The behavior of NMSE in SVMs. The NMSE on both the training set (left) and validation set (right) fluctuate during the initial period and then gradually converge to a constant value.

Table 4
Results of SVMs and BP on the test set

Futures	SVMs				BP			
	NMSE	MAE	DS	WDS	NMSE	MAE	DS	WDS
CME-SP	0.9365	0.2361	58.29	0.7380	1.0485	0.2556	55.27	0.8273
CBOT-US	1.2970	0.3425	45.22	0.9039	1.3424	0.3609	44.72	0.9882
CBOT-BO	1.1349	0.2989	46.73	0.9563	1.1877	0.3149	41.70	1.1093
EUREX-BUND	1.3337	0.3502	43.21	1.0542	1.3478	0.3520	41.20	1.0977
MATIF-CAC40	1.2083	0.4105	45.22	0.9827	1.2379	0.4224	42.22	1.0960

beyond 100. The reason lies in that a small value for C will under-fit the training data because the weight placed on the training data is too small thus resulting in large values of NMSE on both the training and test sets. On the contrary, when C is too large, SVMs will over-fit the training set, leading to a deterioration in the generalization performance. In this case, an appropriate choice for C would be between 10 and 100. The number of support vectors decreases slightly as C increases (Fig. 4(b)) because the number of the support vectors with Lagrange coefficients $|a_i - a_i^*| = c$ decreases as C increases.

Fig. 5 gives the results of SVMs with various ε where both δ^2 and C are fixed at 10. Fig. 5(a) shows that the NMSE on both the training set and the test set is very stable

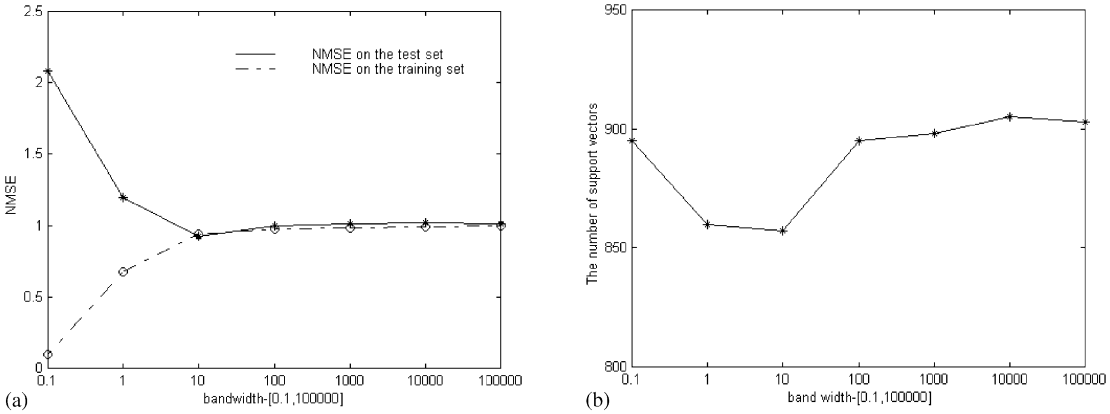

Fig. 3. The results of various δ^2 in which $C = 10$ and $\varepsilon = 0.001$. (a) The NMSE. A small value for δ^2 over-fits the training data while a large value for δ^2 under-fits the training data. (b) The number of support vectors. There are more support vectors in the over-fitting and under-fitting cases.

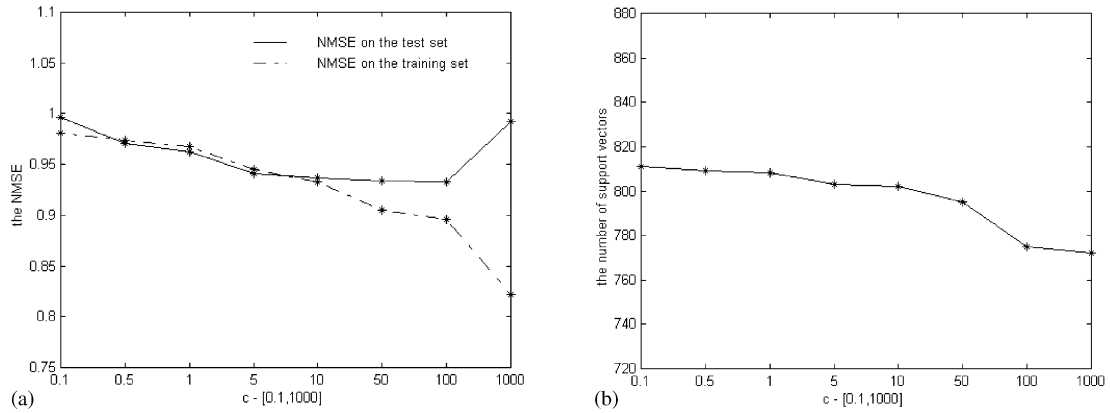

Fig. 4. The results of various C in which $\delta^2 = 10$ and $\varepsilon = 0.001$. (a) The NMSE. A small value for C under-fits the training data while a large value for C over-fits the training data. (b) The number of support vectors. The number of support vectors decreases slightly as C increases.

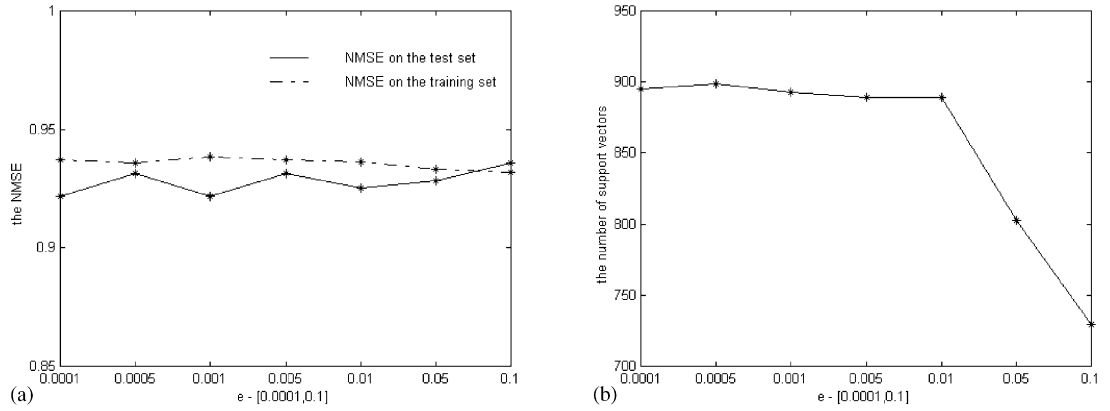

Fig. 5. The results of various ε in which $\delta^2 = 10$ and $C = 10$. (a) The NMSE. NSME is not affected much by ε. (b) The number of support vectors. The number of support dramatically decreases as ε increases.

and relatively unaffected by changes in ε. This indicates that the performance of SVMs is insensitive to ε. However, the number of support vector decreases as ε increases, especially when ε is larger than 0.01 as illustrated in Fig. 5(b). This is consistent with the result obtained in [15] that the number of support vector is found to be a decreasing function of ε.

5. Conclusions

The use of SVMs in financial time series forecasting is studied in this paper. The study concluded that SVMs provide a promising alternative to the BP neural network for financial time series forecasting. As demonstrated in the experiment, SVMs forecast significantly better than the BP network in the CME-SP, CBOT-US, CBOT-BO and MATIF-CAC40 futures and slightly better in the EUREX-BUND. The superior performance of SVMs over BP is due to the following reasons:

(1) SVMs implement the structural risk minimization principle which minimizes an upper bound of the generalization error rather than minimizes the training error. This eventually leads to better generalization than the BP network which implements the empirical risk minimization principle.
(2) There are fewer free parameters compared to the BP network. In SVMs, there are only three free parameters, namely, δ^2, C and ε. As illustrated in the experiment, the performance of SVMs is insensitive to ε when a reasonable value is selected for ε. However, for the BP network, there are a large number of controlling parameters which include the number of hidden layers, the number of hidden nodes, the learning rate, the momentum term, epochs, transfer functions and weights initialization methods. All of them are selected empirically. It is a difficult task to obtain an optimal combination of parameters which produces the best prediction performance.
(3) The BP network may not converge to global solutions. The gradient descent BP algorithm optimizes the weights in a way that the summed square error is minimized along the steepest slope of the error surface. Global solution is not guaranteed because the algorithm can get stuck in a local minima of the error surface. In the case of SVMs, training SVMs is equivalent to solving a linearly constrained quadratic programming, and the solution of SVMs is unique, optimal and global.
(4) The use of validation set to stop the training of the BP network requires much experience and care. Although we have the luxury of using the validation set, it is still difficult to guarantee that there is no over-fitting in the BP network. This is often considered as a weakness of the method.

The investigations of the parameters of SVMs show that δ^2 and C play an important role in the performance of SVMs. Improper selection of the two parameters can cause either over-fitting or under-fitting of the training data. Although the performance of SVMs is insensitive to ε, the number of support vectors can be greatly reduced by using a larger ε thus resulting in a sparse representation of the solution. Based on the results obtained, it is important to develop a structured way of selecting optimum parameters of SVMs considering their significant impact on the performance.

Since SVMs offer so many advantages in comparison with the BP network, future research will explore the possibility of refining the SVMs in order to achieve higher generalization performance.

Acknowledgements

We would like to express our special thanks to Mr. Manzoor U.L.S. and Ms. Hong X.B. for their assistance in the writing of this paper. In addition, we would like to thank Man-Drapeau Research Company for providing the data.

References

[1] Hall JW. Adaptive selection of U.S. stocks with neural nets. In: GJ Deboeck (Ed.), Trading on the edge: neural, genetic, and fuzzy systems for chaotic financial markets. New York: Wiley, 1994.
[2] Yaser SAM, Atiya AF. Introduction to financial forecasting. Applied Intelligence 1996;6:205–13.
[3] Cheng W, Wanger L, Lin CH. Forecasting the 30-year US treasury bond with a system of neural networks. Journal of Computational Intelligence in Finance 1996;4:10–6.
[4] Sharda R, Patil RB. A connectionist approach to time series prediction: an empirical test. In: Trippi, RR, Turban, E, (Eds.), Neural Networks in Finance and Investing, Chicago: Probus Publishing Co., 1994, pp. 451–64.
[5] Van E, Robert J. The application of neural networks in the forecasting of share prices. Haymarket, VA, USA: Finance & Technology Publishing, 1997.
[6] Haykin S. Neural networks: a comprehensive foundation. Englewood Cliffs, NJ: Prentice Hall, 1999.
[7] Kaastra I, Milton SB. Forecasting futures trading volume using neural networks. The Journal of Futures Markets 1995;15(8):853–970.
[8] Zhang GQ, Michael YH. Neural network forecasting of the British Pound/US Dollar exchange rate. Omega 1998;26(4):495–506.
[9] Chiang WC, Urban TL, Baildridge G. A neural network approach to mutual fund net asset value forecasting. Omega 1996;24(2):205–15.
[10] Pan ZH, Wang XD. Wavelet-based density estimator model for forecasting. Journal of Computational Intelligence in Finance 1998;6:6–13.
[11] Kimoto T, Asakawa K, Yoda M, Takeoka M. Stock market prediction system with modular neural networks. In: Trippi, RR, Turban, E, (Eds.), Neural Networks in Finance and Investing, Chicago: Probus Publishing Co., 1994, pp. 343–57.

[12] Dorsey R, Sexton R. The use of parsimonious neural networks for forecasting financial time series. Journal of Computational Intelligence in Finance 1998;6:24–30.
[13] Abecasis SM, Lapenta SL. Modeling the MERVAL index with neural networks and the discrete wavelet transform. Journal of Computational Intelligence in Finance 1997;5: 15–9.
[14] VanAussem A, Campbell J, Murtagh F. Wavelet-based feature extraction and decomposition strategies for financial forecasting. Journal of Computational Intelligence in Finance 1998;6:5–12.
[15] Vapnik VN. The nature of statistical learning theory. New York: Springer, 1995.
[16] Schmidt M. Identifying speaker with support vector networks. Interface '96 Proceedings, Sydney, 1996.
[17] Joachimes T. Text categorization with support vector machines. Technical Report, ftp://ftp-ai.informatik.uni-dortmund.de/pub/Reports/report23.ps.z.
[18] Muller KR, Smola A, Scholkopf B. Prediction time series with support vector machines. Proceedings of International Conference on Artificial Neural Networks, Lausanne, Switzerland, 1997, p. 999.
[19] Mukherjee S, Osuna E, Girosi F. Nonlinear prediction of chaotic time series using support vector machines. Proceedings of IEEE NNSP'97, Amelia Island, FL, 1997.
[20] Vapnik VN, Golowich SE, Smola AJ. Support vector method for function approximation, regression estimation, and signal processing. Advances in Neural Information Processing Systems 1996;9:281–7.
[21] Smola AJ, Scholkopf B. A tutorial on support vector regression, NeuroCOLT Technical Report TR Royal Holloway College, London, UK, 1998.
[22] Smola AJ. Learning with Kernels. PhD Thesis, GMD, Birlinghoven, Germany, 1998.
[23] Thomason M. The practitioner methods and tool. Journal of Computational Intelligence in Finance 1999;7(3):36–45.
[24] Thomason M. The practitioner methods and tool. Journal of Computational Intelligence in Finance 1999;7(4):35–45.
[25] Murray WH. Microsoft C/C++7: The complete reference. Berkeley, CA: Osborne McGraw-Hill, 1992.