Applied Soft Computing 13 (2013) 947–958

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

Support vector regression with chaos-based firefly algorithm for stock market price forecasting

Ahmad Kazem[a], Ebrahim Sharifi[a], Farookh Khadeer Hussain[b,*], Morteza Saberi[c], Omar Khadeer Hussain[d]

[a] Department of Industrial Engineering, University of Tafresh, Iran
[b] Decision Support and e-Service Intelligence Lab, Quantum Computation and Intelligent Systems, School of Software, Faculty of Engineering and Information Technology, University of Technology Sydney, Ultimo, NSW, Australia
[c] Islamic Azad University, Tafresh Branch, Young Researchers Club, Tafresh, Iran
[d] School of Information Systems, Curtin University, Perth, WA, Australia

ARTICLE INFO

Article history:
Received 29 January 2012
Received in revised form 29 June 2012
Accepted 17 September 2012
Available online 8 October 2012

Keywords:
Support vector regression
Firefly algorithm
Chaotic mapping
Stock market price forecasting

ABSTRACT

Due to the inherent non-linearity and non-stationary characteristics of financial stock market price time series, conventional modeling techniques such as the Box–Jenkins autoregressive integrated moving average (ARIMA) are not adequate for stock market price forecasting. In this paper, a forecasting model based on chaotic mapping, firefly algorithm, and support vector regression (SVR) is proposed to predict stock market price. The forecasting model has three stages. In the first stage, a delay coordinate embedding method is used to reconstruct unseen phase space dynamics. In the second stage, a chaotic firefly algorithm is employed to optimize SVR hyperparameters. Finally in the third stage, the optimized SVR is used to forecast stock market price. The significance of the proposed algorithm is 3-fold. First, it integrates both chaos theory and the firefly algorithm to optimize SVR hyperparameters, whereas previous studies employ a genetic algorithm (GA) to optimize these parameters. Second, it uses a delay coordinate embedding method to reconstruct phase space dynamics. Third, it has high prediction accuracy due to its implementation of structural risk minimization (SRM). To show the applicability and superiority of the proposed algorithm, we selected the three most challenging stock market time series data from NASDAQ historical quotes, namely Intel, National Bank shares and Microsoft daily closed (last) stock price, and applied the proposed algorithm to these data. Compared with genetic algorithm-based SVR (SVR-GA), chaotic genetic algorithm-based SVR (SVR-CGA), firefly-based SVR (SVR-FA), artificial neural networks (ANNs) and adaptive neuro-fuzzy inference systems (ANFIS), the proposed model performs best based on two error measures, namely mean squared error (MSE) and mean absolute percent error (MAPE).

1. Introduction

Stock market price prediction is regarded as one of the most challenging tasks of financial time series prediction. The difficulty of forecasting arises from the inherent non-linearity and non-stationarity of the stock market and financial time series. In the past, Box–Jenkins models [1], such as the autoregressive (AR) model and the autoregressive integrated moving average (ARIMA) model, were proposed to tackle this problem. However, these models were developed based on the assumption that the time series being forecasted are linear and stationary. In recent years, nonlinear approaches have been proposed, such as autoregressive conditional heteroscedasticity (ARCH) [2], generalized autoregressive conditional heteroscedasticity (GARCH) [3], artificial neural networks (ANNs) [4–9], fuzzy neural networks (FNN) [10–13], and support vector regression (SVR) [14–22].

ANN has been widely used for modeling stock market time series due to its universal approximation property [23]. Previous researchers have indicated that ANN, which implements the empirical risk minimization principle in its learning process, outperforms traditional statistical models [4]. However, ANN suffers from local minimum traps and the difficulty of determining the hidden layer size and learning rate [24,25]. By contrast, support vector regression, originally introduced by Vapnik [24,26], 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 [25,27]. The main problem with SVR is the determination of its hyperparameters, which requires practitioner experience. Unsuitably chosen kernel functions or hyperparameter settings may lead to significantly poor performance [27–30].

* Corresponding author.
E-mail address: Farookh.Hussain@uts.edu.au (F.K. Hussain).

1568-4946/$ – see front matter.
http://dx.doi.org/10.1016/j.asoc.2012.09.024

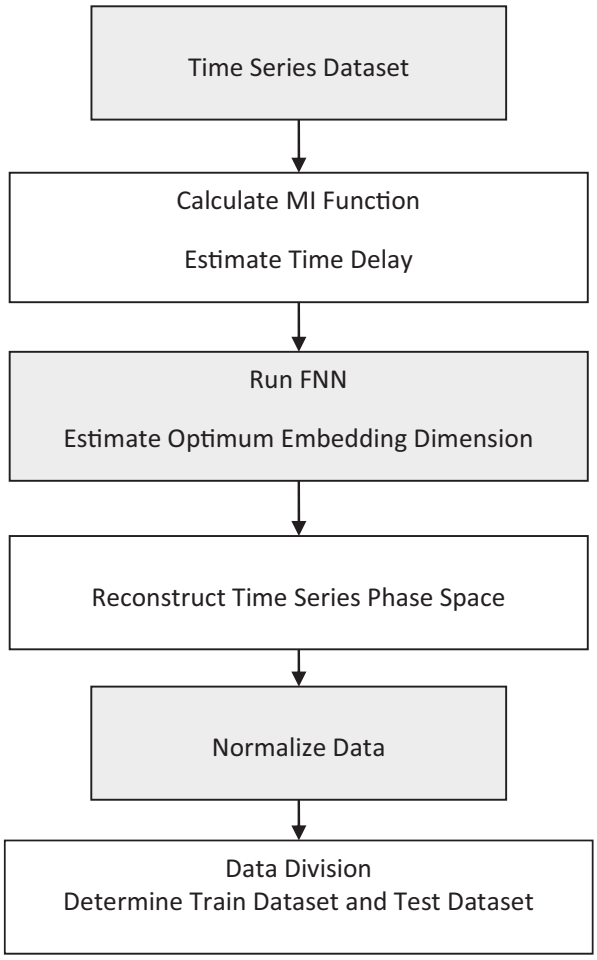


Fig. 1. Data preprocessing procedure.

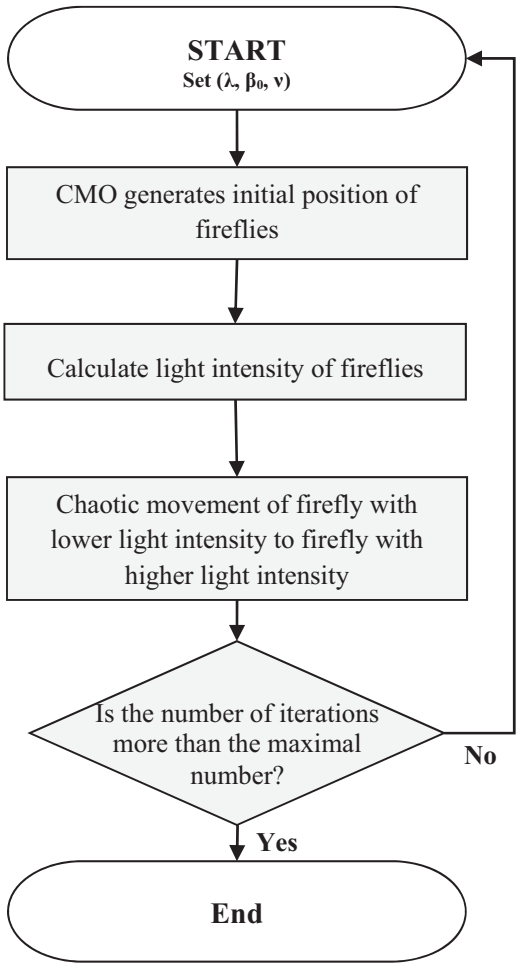


Fig. 2. Chaotic firefly algorithm.

Recently, optimization algorithms such as genetic algorithm (GA) and chaotic genetic algorithm (CGA) have been used to find the best hyperparameters for SVR [31–34].

In this paper, we propose a chaotic firefly algorithm for optimizing the SVR hyperparameters. Results show that our method performs better than SVR-FA, SVR-GA SVR-CGA, ANFIS, ANN and other previous algorithms.

The remainder of this paper is organized as follows. Section 2 introduces new prediction model, including delay coordinate embedding, logistic map, support vector regression and firefly algorithm. Section 3 defines the implementation steps of the proposed model. Section 4 describes the data used in this study and discusses the experimental findings. Conclusions and remarks are given in Section 5.

2. Support vector regression with chaotic firefly algorithm

In this section, we introduce delay coordinate embedding for phase space reconstruction, logistic map, support vector regression and firefly algorithm.

2.1. Delay-coordinate embedding

The analysis of time series generated by non-linear dynamic systems can be done in accordance with Taken's embedding theory [35]. Let univariate time series $\{x_i\}_{i=1}^{N}$, where N is the length of the time series, generated from a d-dimension chaotic attractor, a phase space R^d of the attractor can be reconstructed by using a delay coordinate defined as

$$X_i = (x_i, x_{i-\tau}, \ldots, x_{i-(m-1)\tau}) \quad (1)$$

where m is called the embedding dimension of reconstructed phase space and τ is the time delay constant. Choosing the correct embedding dimension is very important so that we can predict x_{t+1} [36]. Takens [35] considered that the sufficient condition for the embedding dimension is $m \geq 2d + 1$. However, too large an embedding dimension needs more observations and complex computation. Moreover, if we choose too large an embedding dimension, noise and other unwanted inputs will be highly embedded with the real source input information, which may corrupt the underlying system dynamic information. Therefore, in accordance with [37], if the dimension of the original attractor is d then an embedding dimension of $m = 2d + 1$ will be adequate for reconstructing the attractor.

An efficient method of finding the minimal sufficient embedding dimension is the false nearest neighbors (FNN) procedure, proposed by Kennel et al. [38]. Two near points in reconstructed phase space are called false neighbors if they are significantly far apart in the original phase space. Such a phenomenon occurs if we select an embedding dimension lower than the minimal sufficient value and the reconstructed attractor therefore does not preserve the topological properties of the real phase space. In this case, points are projected into the false neighborhood of other points. The idea behind the FNN procedure is as follows. Suppose X_i has a nearest

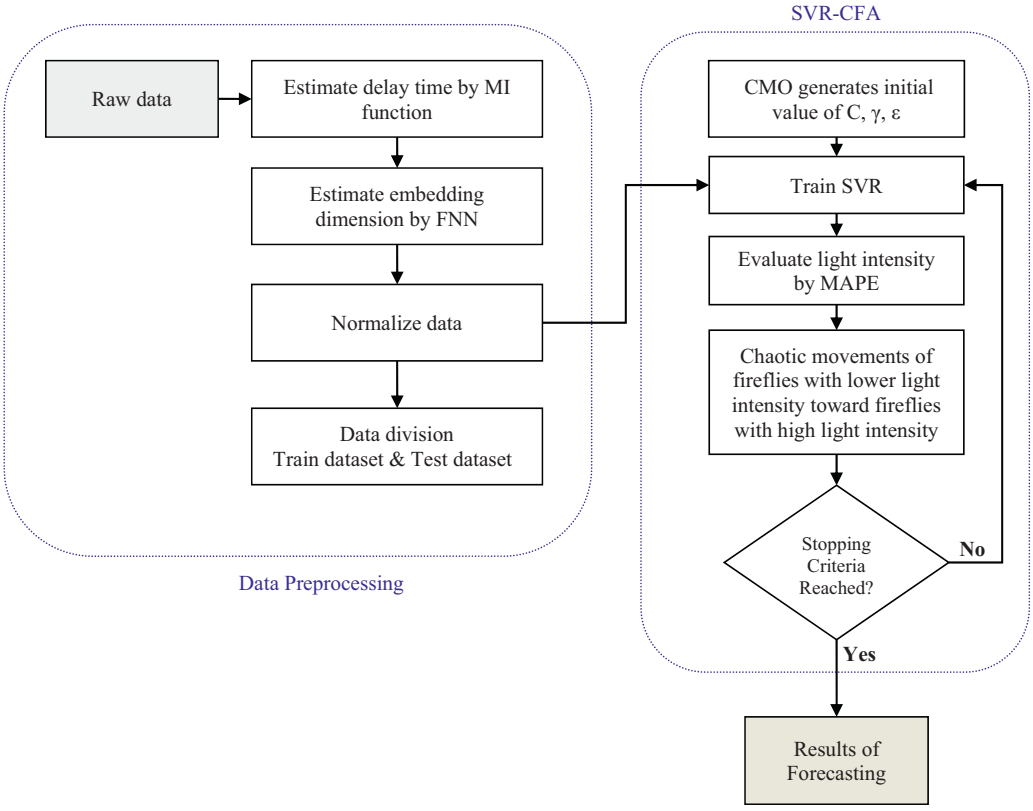


Fig. 3. SVR-CFA.

neighbor X_j in an m-dimensional space. Calculate the Euclidean distance $||X_i - X_j||$ and compute:

$$R_i = \frac{||X_{i+1} - X_{j+1}||}{||X_i - X_j||} \tag{2}$$

If R_i exceeds a given threshold R_{tol} (say, 10 or 15), the point X_j is considered as a false nearest neighbor in dimension m. We can say that the embedding dimension m is sufficiently high if the fraction of points that have false nearest neighbors is zero or considerably small.

Estimation of time delay τ is another important issue. If τ is too small, redundancy will occur and if τ is too large, it will probably lead to a complex phenomenon called irrelevance. In this study, we use the first minimum of mutual information (MI) function [39] to determine τ as follows:

$$\text{MI}\ (\tau) = \sum_{n=1}^{N-\tau} P(x_n, x_{n+\tau})\ \log_2\left(\frac{P(x_n, x_{n+\tau})}{P(x_n)P(x_{n+\tau})}\right) \tag{3}$$

where $P(x_n)$ is the probability density of x_n while $P(x_n, x_{n+\tau})$ is the joint probability density of x_n and $x_{n+\tau}$.

2.2. Logistic mapping

The simplest chaotic mapping operator (CMO), which was brought to the attention of scientists in 1976, is logistic mapping [32].

$$x_{n+1} = \mu x_n(1 - x_n) \tag{4}$$

where x_n is the nth chaotic number, n denotes the iteration number and $\mu = 4$.

Logistic mapping includes all the properties of chaotic systems such as self-similarity, ergodicity, semi-random motion, and sensitivity to initial conditions. A detailed explanation about chaotic properties can be found in [37].

As we know, diversity in the initial solution of optimization algorithms such as genetic algorithms (GA) and firefly algorithms (FA) is vital for preventing premature phenomena. Logistic mapping can provide more diversity than randomly selected initial solutions and will therefore decrease the probability of premature occurrence [31].

2.3. Support vector regressions (SVR)

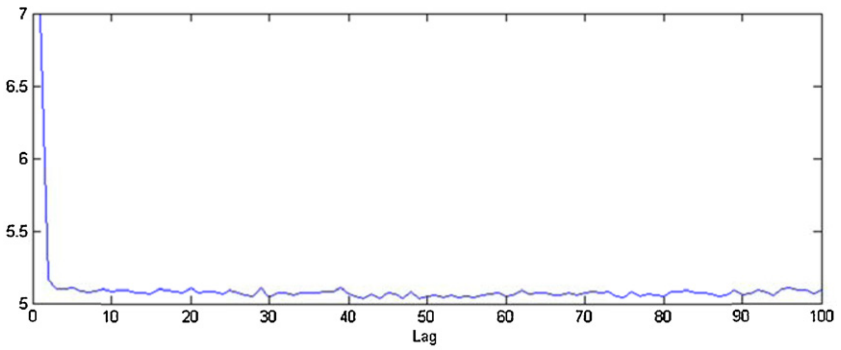
Suppose we are given a set of training patterns $(x_1, y_1), \ldots, (x_\ell, y_\ell)$, where $x_i \in R^n$, $i = 1, 2, \ldots, \ell$ and $y_i \in R$ is the target value for each input vector x_i. A regression model is trained by these patterns and used to predict the future target values. SVR is a non-linear kernel-based regression method which tries to find the best regression hyperplane with smallest structural risk in a so-called high dimensional feature space [27].

One of the most popular types of SVRs is ε-SVR which locates the hyperplane with an ε-insensitive loss function [24]. The SVR function is formulated as follows:

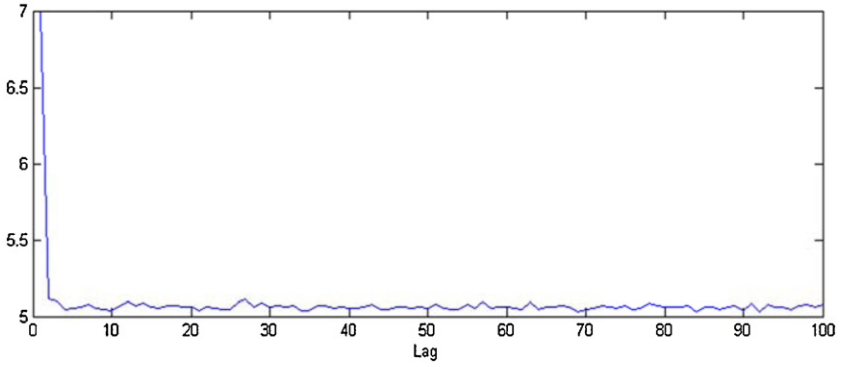
$$f(x) = w^T\varphi(x) + b \tag{5}$$

where $\varphi(x)$ is a nonlinear mapping from the input space to the feature space. w is a vector of weight coefficients and b is a bias constant. w and b are estimated by minimizing the following optimization problem:

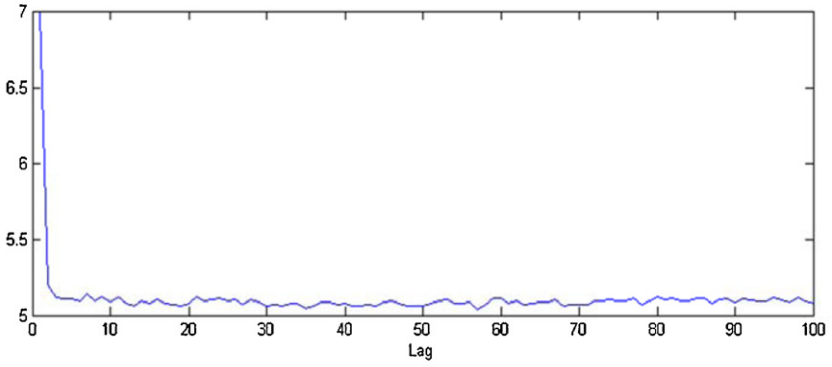
$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}||w||^2 \\ &\text{subjected to} && \begin{cases} y_i - (\langle w, \varphi(x_i)\rangle + b) \le \varepsilon \\ (\langle w, \varphi(x_i)\rangle + b) - y_i \le \varepsilon \end{cases} \end{aligned} \tag{6}$$



(a) Intel



(b) National Bankshares



(a) Microsoft

Fig. 4. The MI function of three stock market prices.

To cope with feasibility issues and to make the method more robust, points from the ε-insensitive band are not eliminated. Instead, we penalize these points by introducing slack variables ξ_i, ξ_i^* [40]:

$$\begin{aligned} &\text{minimize} && \tfrac{1}{2}||w||^2 + C\sum_{i=1}^{\ell}(\xi_i + \xi_i^*) \\ &\text{subjected to} && \begin{cases} y_i - (\langle w, \varphi(x_i)\rangle + b) \le \varepsilon + \xi_i \\ (\langle w, \varphi(x_i)\rangle + b) - y_i \le \varepsilon + \xi_i^* \\ \xi_i,\ \ \xi_i^* \ge 0 \end{cases} \end{aligned} \tag{7}$$

where the cost constant $C>0$ determines the trade-off between model complexity and training error ℓ is the number of training patterns.

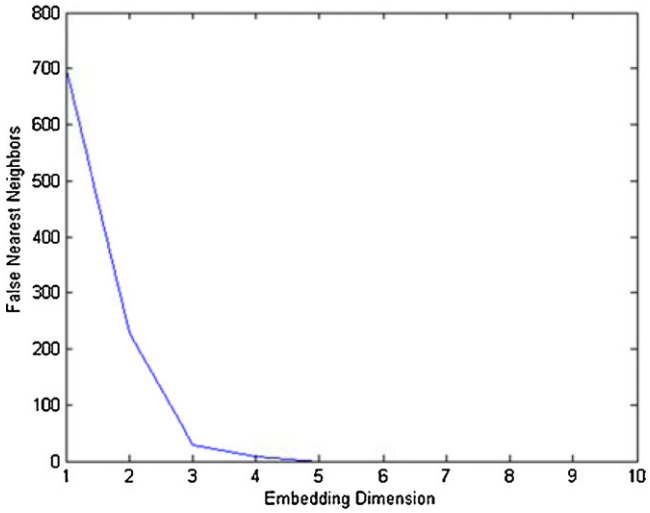
After taking the Lagrangian and conditions for optimality, we can find a model solution in dual representation [24,40].

$$f(x) = \sum_{i=1}^{\ell}(\alpha_i - \alpha_i^*)K(x_i, x) + b \tag{8}$$

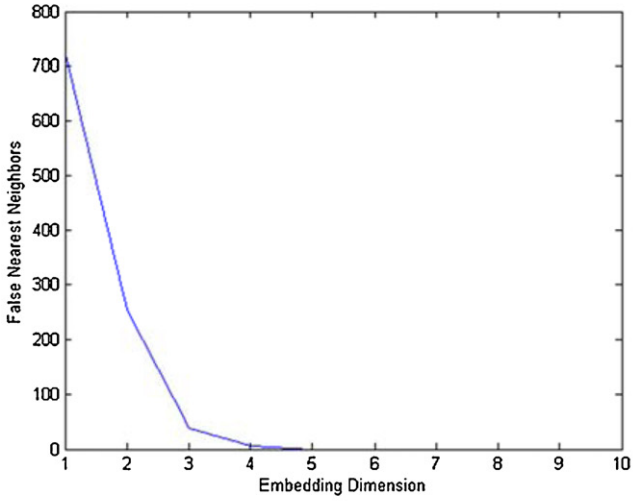
In the above formulation, α_i, α_i^* are nonzero Lagrangian multipliers and the solution for the dual problem. $K(x_i, x)$ is the kernel function which represents the inner product $\langle\varphi(x_i), \varphi(x)\rangle$. In this study, we use the radial basis function (RBF) as the kernel function because of its capabilities and simple implementation [36].

$$K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2) \tag{9}$$

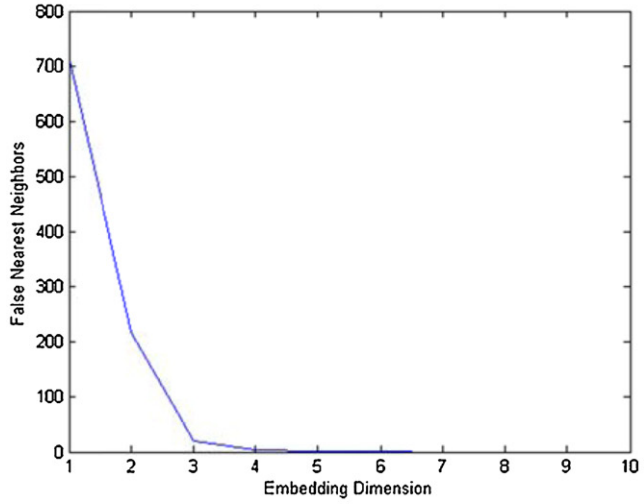
where γ is the width parameter of RBF kernel and should be selected based on heuristics.



(a) Intel



(b) National Bankshares



(c) Microsoft

Fig. 5. The embedding dimension of three stock market prices.

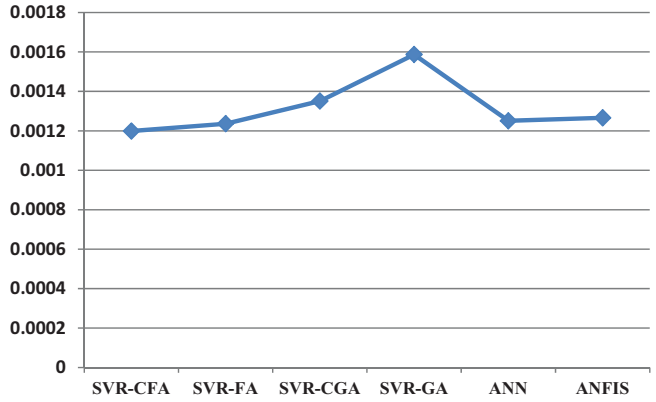


Fig. 6. The average of MSE for all models.

2.4. Firefly algorithm (FA)

The firefly algorithm is a metaheuristic optimization algorithm inspired by the flashing behavior of fireflies [41]. Fireflies use their natural glowing mechanism to attract other fireflies. In this algorithm, each firefly represents a possible solution and its light intensity is proportional to its objective function value. Fireflies with lower light intensity (fitness) move toward fireflies with higher light intensity by the following formulation:

$$x_i = x_i + \beta(x_j - x_i) + \alpha(u - 0.5) \tag{10}$$

$$\beta = \beta_0[\exp(-\lambda \cdot r_{ij}^2)] \tag{11}$$

where x_i is a firefly with higher light intensity, x_j is a firefly with lower light intensity; λ is the absorption coefficient, r_{ij} is the Euclidean distance between x_i and x_j; β_0 is the maximum attractiveness value and α is a trade-off constant which determines the random behavior of movement. u is a random number in the interval (0,1).

After such movements, all fireflies move toward the neighborhood of the best firefly, improving their personal fitness. The firefly with the highest light intensity moves randomly in the search space to improve global fitness. After reaching the defined maximum iterations, the firefly with the highest light intensity is considered as the best solution. The whole procedure of FA can be briefly explained as follows. First, initial positions of fireflies are generated randomly. Second, each firefly is evaluated by a given fitness function. Third, the fireflies with lower fitness values move toward the fireflies with higher fitness values by Eq. (10). For the firefly with the highest fitness value, the second part of Eq. (10) becomes zero.

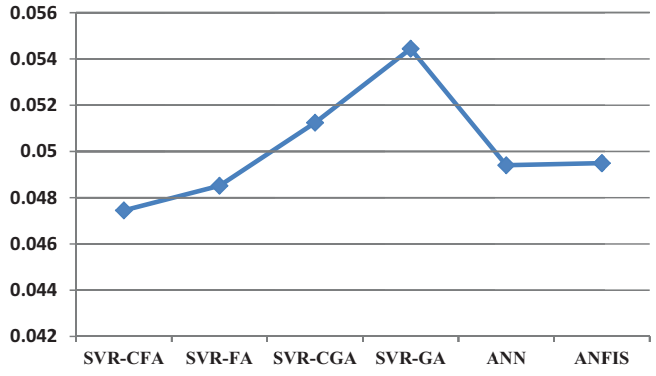


Fig. 7. The average of MAPE for all models.

Therefore it moves randomly proportional to the coefficient α. The above procedure continues until a given number of iterations are reached.

3. Proposed integrated algorithm

In this section, we introduce our proposed algorithm, including data pre-processing and the chaotic firefly algorithm.

3.1. Data preprocessing

First, the MI function (Eq. (3)) is calculated for the financial time series dataset. Second, the first delay time in which MI function minimum value occurs is considered as the optimum time delay. Third, the false nearest neighbors (FNN) method is employed to find the minimum sufficient embedding dimension. Fourth, according to the optimum time delay and embedding dimension, the time series phase space is reconstructed to reveal its unseen dynamics. Then, we use Eq. (12) to normalize the data in the interval (0,1) and fit them for the RBF kernel function.

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}} \tag{12}$$

Finally, the time series dataset is divided into two datasets, namely a training dataset and a testing dataset. Fig. 1 presents the data preprocessing procedure.

3.2. Proposed chaotic firefly algorithm

Firefly algorithms (FAs), like other nature-inspired optimization algorithms, use a random approach to generate an initial solution. However, this approach has two major shortcomings, namely slow convergence and becoming trapped in local optima, caused by reduced population diversity. In this approach, the initial positions of fireflies are not necessarily fully diversified in the search space [32]. To improve initial solution diversity and the quality of the initial population, a CMO (Eq. (1)) is used instead of a random approach to generate an initial solution.

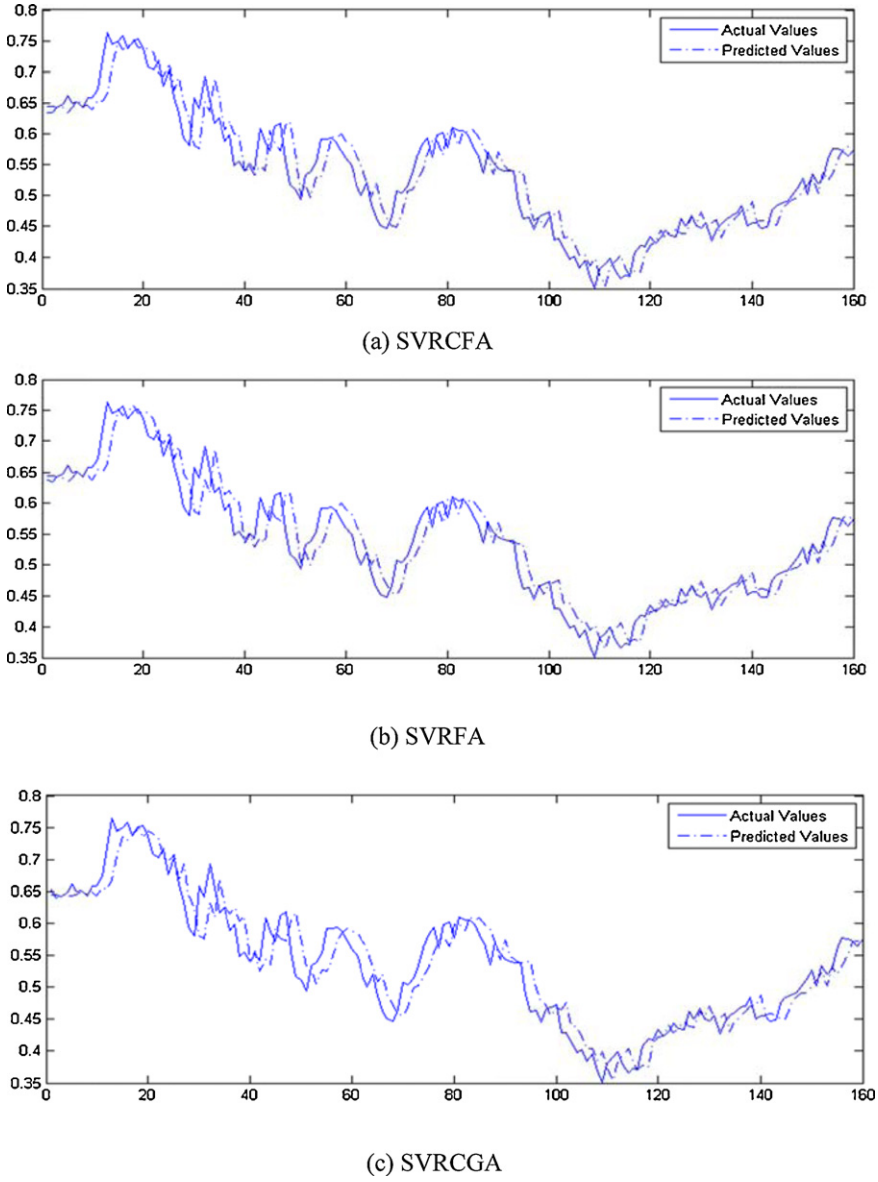
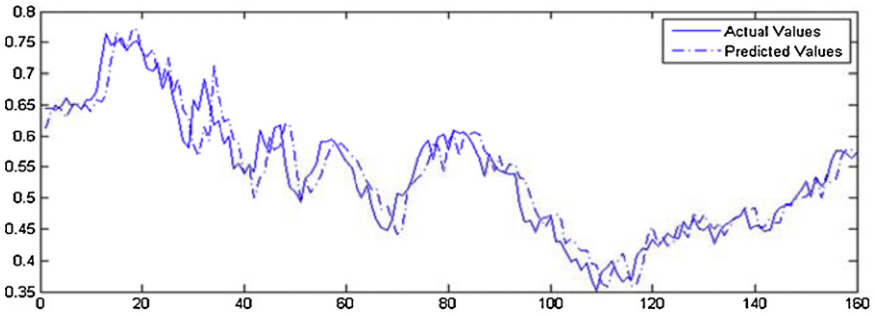
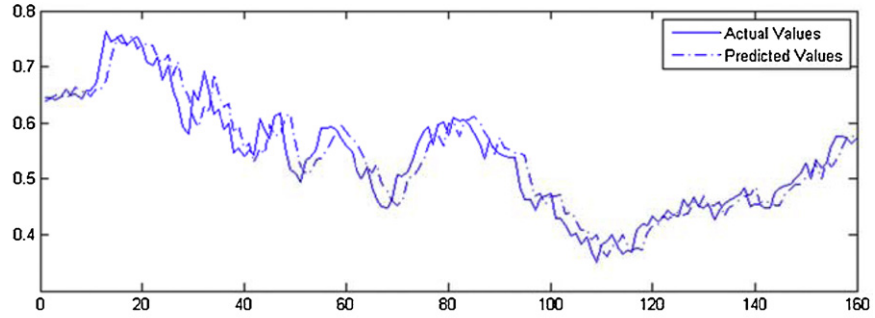


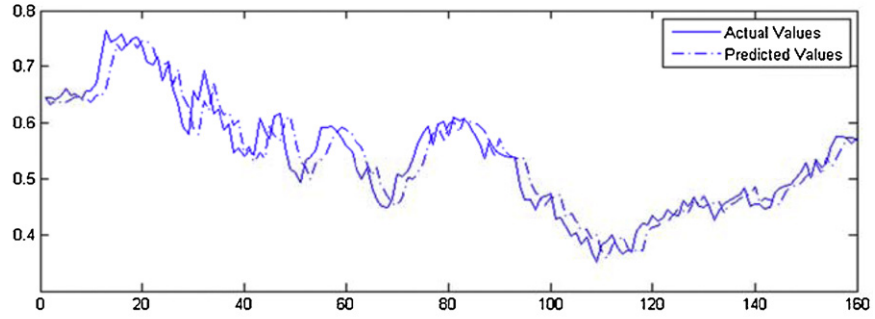
Fig. 8. Model forecasts on Intel stock price.



(d) SVRGA



(e) ANN



(f) ANFIS

Fig. 8. (*Continued*)

A random component is used by FAs to estimate the new positions of fireflies. However, this kind of movement does not necessarily search the whole solution space. Therefore, to perform ergodic searching of the solution space, we use a chaotic component instead of a random component. Fig. 2 shows the flowchart of the proposed procedure of the chaotic firefly algorithm.

The stepwise workings of the chaotic firefly algorithm are as follows:

Step 1: Generate initial positions of fireflies by CMO. The values of the three hyperparameters in a SVR model in ith iteration can be represented as $X_p^{(i)}$, $p = C, \gamma, \varepsilon$. Set $i = 0$, and employ Eq. (12) to map the three parameters among the intervals (Min_p, Max_p) into chaotic variable $x_p^{(i)}$ in the interval (0,1).

$$x_p^{(i)} = \frac{X_p^{(i)} - Min_p}{Max_p - Min_p}, \quad p = C,\ \gamma,\ \varepsilon \tag{13}$$

Then, adopt Eq. (4) to compute the next iteration chaotic variable $x_p^{(i+1)}$. Transform $x_p^{(i+1)}$ to obtain three parameters for the next iteration, $X_p^{(i+1)}$ by the following Eq. (14)

$$X_p^{(i+1)} = Min_p + x_p^{(i+1)}(Max_p - Min_p) \tag{14}$$

Step 2: Evaluate light intensity. Evaluate the light intensity (forecasting errors) of each firefly. In this study, we use a mean absolute percentage error (MAPE) as the fitness function. The MAPE is calculated as Eq. (15):

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - f_i}{y_i}\right| \tag{15}$$

where y_i and f_i represent the actual and forecast values, and N is the number of forecasting periods.

Step 3: Chaotic movement of fireflies. Fireflies with lower light intensity (fitness) move toward fireflies with higher light intensity and the positions of fireflies are updated. The firefly with the highest light intensity moves chaotically in the solution space using Eq. (9). Instead of random component, $\alpha(u - 0.5)$, we use a chaotic component, $\delta(x^{(n)})$, where

$x^{(n)}$ is a chaotic variable generated by Eq. (3), and δ is the annealing operation resulting from Eq. (14) [31]:

$$\delta = 1 - \left| \frac{n-1}{n} \right|^{v} \tag{16}$$

where n is the iteration number and v is an integer.

Step 4: Stopping condition. If the number of iterations is equal to a given scale, then the best fireflies (with highest light intensity) are presented as a solution; otherwise go back to step 2.

Fig. 3 shows the complete procedure of the proposed SVR-CFA.

4. Case study

In this section, we test the proposed algorithm with three different daily stock market prices, namely Intel, National Bank shares and Microsoft. These datasets are selected from the numerous stocks available in the NASDAQ stock market due to their challenging behavior and many direction changes in the selected time span. The proposed algorithm is compared with SVR-GA, SVR-CGA, SVR-FA, SVR-CFA, ANN and ANFIS.

Table 1
The optimal τ and m.

	Intel	National Bank shares	Microsoft
τ	2	3	3
m	5	5	7

4.1. Data collection and performance evaluation

Daily closing (last) stock market prices for Microsoft (from 9/12/2007 to 11/11/2011), Intel (from 9/12/2007 to 11/11/2010) and National Bank shares (from 6/27/2008 to 8/29/2011) were extracted from NASDAQ historical quotes. The dataset was divided into two sets, a training dataset and a testing dataset; 80% of the daily data (a total of 640 observations) were used for the training dataset and the remainder of the daily data (a total of 160 observations) were used for the testing dataset.

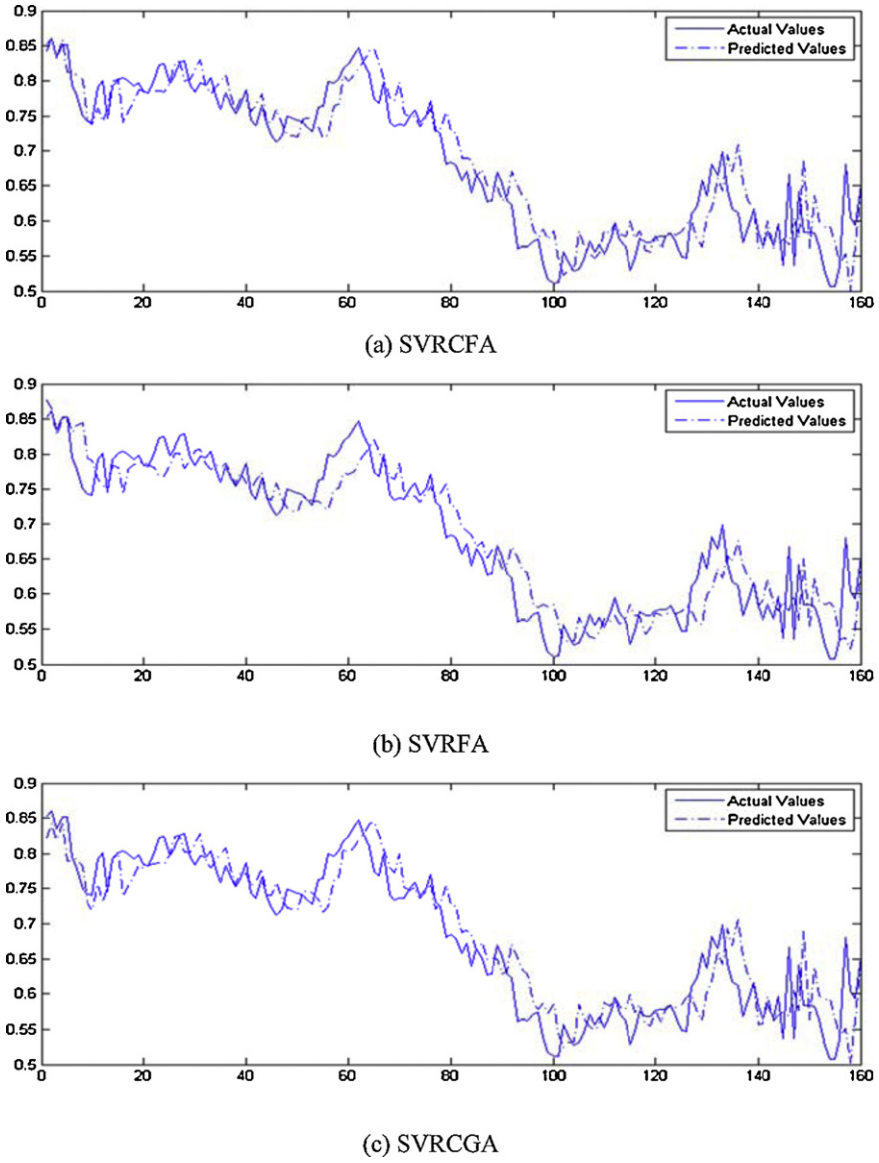
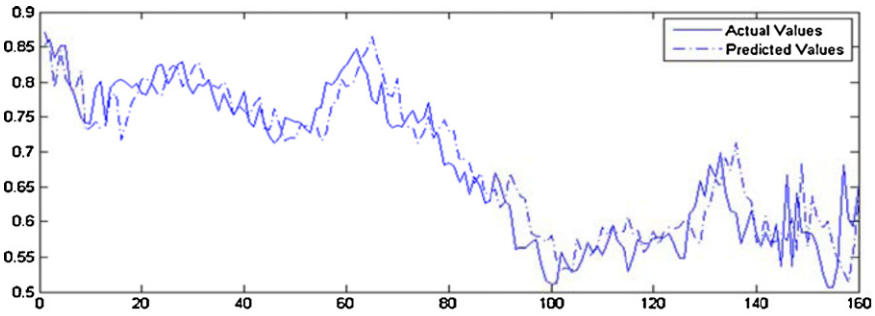
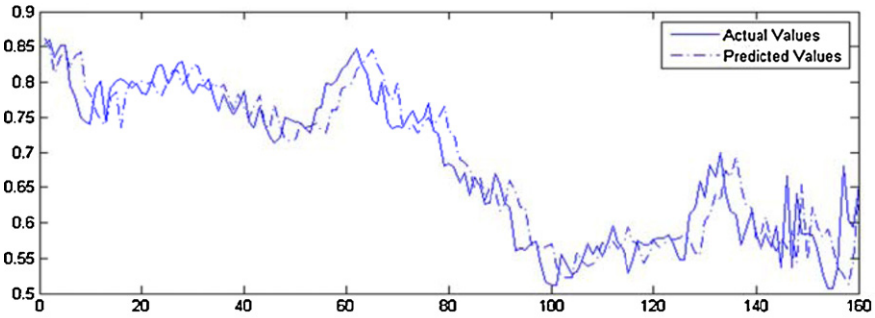


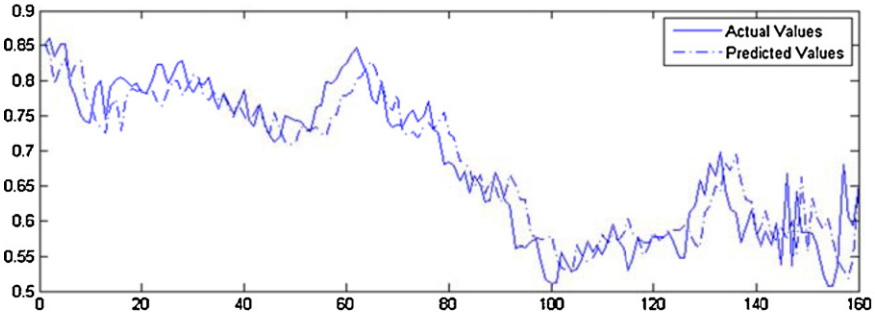
Fig. 9. Model forecasts on National Bank shares stock price.



(d) SVRGA



(e) ANN



(f) ANFIS

Fig. 9. (*Continued*)

For the analysis of predicting models, MSE (mean square error) and MAPE (Eq. (15)) were employed as forecasting indices to measure their performance. The formulation of MSE is as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - f_i)^2 \tag{17}$$

Table 2
The optimal γ, C and ε.

	γ	C	ε
(a) Intel			
SVR-GA	0.8291	5844.6	0.0719
SVR-CGA	5.6984	0.3109	0.0182
SVR-FA	0.1666	10,000	0
SVR-CFA	0.1011	8997.6	0
(b) National Bank shares			
SVR-GA	0.2198	2528.7	0.0781
SVR-CGA	0.2325	9112.0	0.1033
SVR-FA	0.0001	8586.5	0.1037
SVR-CFA	0.2458	5830.9	0.1042
(c) Microsoft			
SVR-GA	8.3900	934.2623	0.1321
SVR-CGA	5.6984	0.3109	0.0182
SVR-FA	0.0001	8058.0	0.0326
SVR-CFA	0.0047	7930.6	0.0415

4.2. Parameter setting in CFA algorithm

The parameters of the CFA algorithm in the proposed model for three numerical examples are experimentally set. The number of fireflies is 20, the maximum number of iterations is 200, β_0 is 4, the constant of the annealing operator is 0.25 and the absorption coefficient is 1.

4.3. Phase space reconstruction

In the phase space reconstruction, we used Hao Cheng's Fractal MATLAB toolbox to select the optimal delay time and embedding dimension. Fig. 4(a)–(c) shows the mutual function of each dataset, and Fig. 5(a)–(c) shows FNN results for each dataset. Table 1 shows the optimal m and τ.

These optimal embedding dimensions and delay times are used to construct the input matrix. The data were fed to SVR, and the hyperparameters of SVR were optimized by GA, CGA, FA and CFA.

Table 2 shows the optimized values of the hyperparameters for each algorithm.

We also fed the reconstructed phase space matrix into the ANN and ANFIS models. The best possible ANN and ANFIS structures were used for comparison with the proposed model.

4.4. Performance comparison

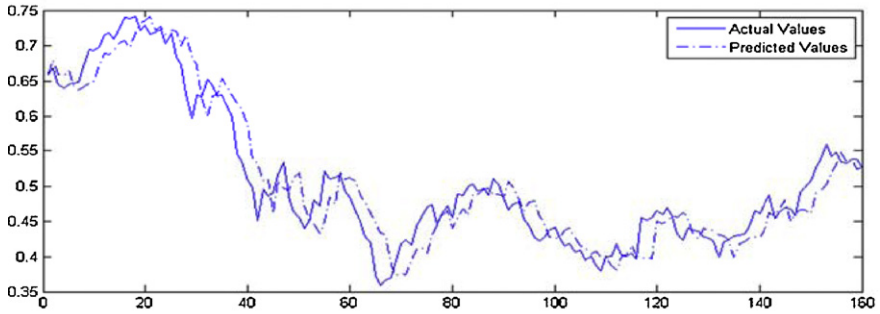
The performance comparison of six models on two indices, MSE and MAPE are reported in Tables 3 and 4. The average errors of SVR-CFA for MSE and MAPE are 0.001199 and 0.047449 respectively, and SVR-CFA is ranked first. The average errors of SVR-FA for MSE and MAPE are 0.001236 and 0.048515 respectively. SVR-FA is worse than SVR-CFA in all measures and is ranked second. The average errors of ANN for MSE and MAPE are 0.001251 and 0.049405 respectively; it is ranked third. The average errors of ANFIS for MSE and MAPE are 0.001266 and 0.049494 respectively; it is ranked fourth. The average errors of SVR-CGA for MSE and MAPE are 0.001351 and 0.051246 respectively. This model is ranked fifth. Finally, the average errors of SVR-GA for MSE and MAPE are 0.001587 and 0.054442

Table 3
Models' performance under MSE.

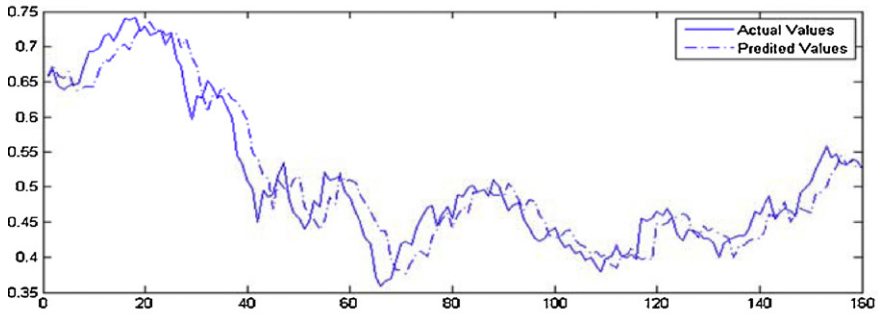
Model	Intel	National Bank shares	Microsoft
SVR-GA	0.00107575	0.00176314	0.00192163
SVR-CGA	0.00103765	0.00158184	0.00143307
SVR-FA	0.000988423	0.00161896	0.00110032
SVR-CFA	0.000959743	0.00157299	0.00106339
ANN	0.0010064	0.001607	0.00114
ANFIS	0.000998	0.001672	0.001212

Table 4
Models' performance under MAPE.

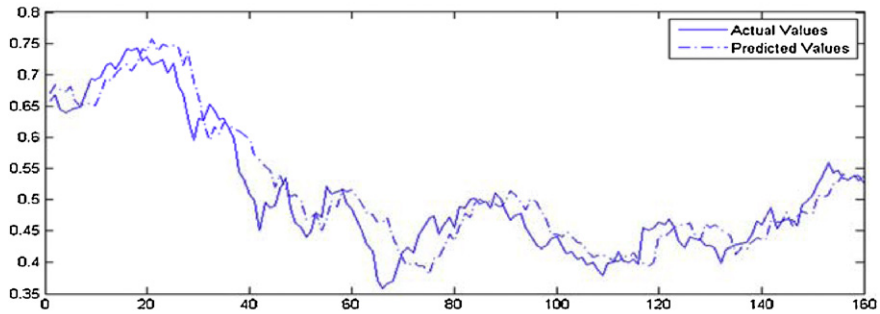
Model	Intel	National Bank shares	Microsoft
SVR-GA	0.047385	0.049147	0.066795
SVR-CGA	0.046709	0.045997	0.061031
SVR-FA	0.045626	0.047267	0.052653
SVR-CFA	0.044594	0.045847	0.051907
ANN	0.047088	0.046742	0.054386
ANFIS	0.04635	0.047982	0.056344



(a) SVRCFA

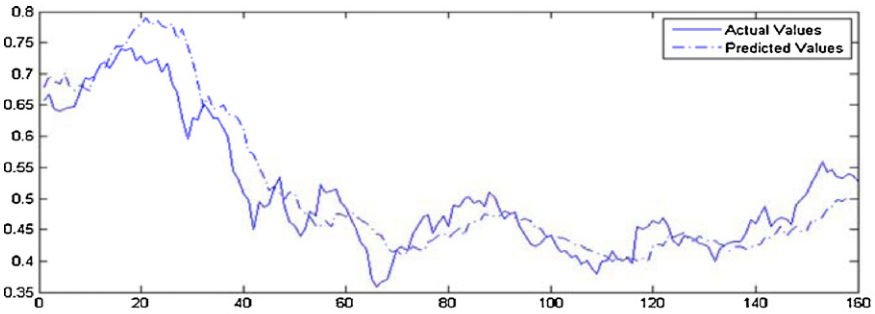


(b) SVRFA

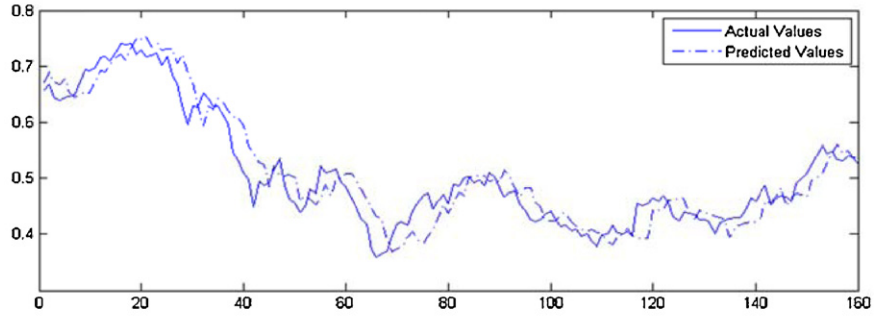


(c) SVRCGA

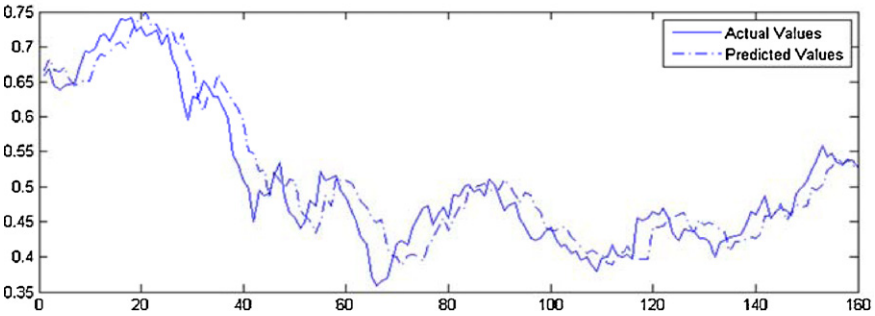
Fig. 10. SVR model forecasts on Microsoft stock price.



(d) SVRGA



(e) ANN



(f) ANFIS

Fig. 10. (*Continued*)

respectively. This is the worst of all the models. Figs. 6 and 7 show the average MSE and MAPE values for all six models.

To present a detailed view of the fitted values of the models against stock market values, the actual values and predicted values for all SVR models are shown in Figs. 8–10.

To discuss the results of the models in more detail, paired *t*-tests are performed to examine which model significantly outperforms the other models. The paired *t*-test is a parametric statistical test for two related numerical samples with a null hypothesis of equality in mean. A detailed explanation for the paired *t*-test is provided in [42]. In this study, the residuals of each forecasting model are used to construct the *t*-test statistic and the *p*-value for each *t*-test on our three stock market datasets is then calculated and presented in Table 5. The tests are performed at a significance level of 95%, therefore *p*-values <0.05 indicate models that vary significantly.

According to Table 5, based on the significance level of 95%, there is an almost-significant difference between the SVR-CFA model and other models. No statistically significant difference can be seen for the other three SVR-based models. The ANN and ANFIS models are significantly different from the SVR-based models. However, in most cases, there is no significant difference between the SVR-GA and ANN models.

Table 5
p-Values for paired *t*-tests.

Models	SVR-FA	SVR-CGA	SVR-GA	ANN	ANFIS
Intel					
SVR-CFA	0.039	0.240	0.058	0.026	0.043
SVR-FA		0.009	0.557	0.171	0.00
SVR-CGA			0.199	0.00	0.031
SVR-GA				0.678	0.026
ANN					0.00
National Bank shares					
SVR-CFA	0.00	0.00	0.006	0.001	0.00
SVR-FA		0.047	0.377	0.190	0.109
SVR-CGA			0.205	0.154	0.00
SVR-GA				0.012	0.00
ANN					0.002
Microsoft					
SVR-CFA	0.029	0.00	0.005	0.00	0.00
SVR-FA		0.00	0.011	0.004	0.00
SVR-CGA			0.211	0.00	0.00
SVR-GA				0.115	0.495
ANN					0.009

Overall, based on MSE and MAPE measures, we conclude that SVR-CFA performs best in terms of prediction accuracy. In addition,

its implementation is much easier than that of traditional models such as ANFIS and ANN.

5. Conclusion

This study developed a novel hybrid model based on a chaotic firefly algorithm and support vector regression for stock market price forecasting. The contribution of the proposed algorithm is mainly the integration of chaotic motion with a firefly algorithm as a simple and novel optimization method. Second, this new integrated chaotic algorithm is incorporated to find the best hyperparameters of SVR namely, cost, RBF kernel function width parameter and radius of the epsilon tube. Third, the implementation of phase space reconstruction in the data preprocess procedure makes the behavior of the financial time series more recognizable for learning machines such as ANN and SVR. The proposed model has three stages. In the first stage, unobserved time series features are extracted by using phase space reconstruction according to Takens' theorem. In the second stage, a chaotic firefly algorithm is applied to optimize SVR hyperparameters based on MAPE. Finally in the third stage, the optimized SVR is used to forecast stock market prices.

Chaos, one of the most important natural motions in the world, can be used as a powerful driver of nature-inspired search algorithms such as GA and FA. A chaotic mapping operator (CMO) improves the search ability of the firefly algorithm in the solution space by increasing the quality of the initially generated population and prevents it from becoming trapped in local optima.

In addition, the use of structural risk minimization (SRM) in the SVR training process makes it more robust than ANN- or ANFIS-based models which use empirical risk minimization (ERM).

In summary, the SVR-CFA model is more suitable than the other three methods for describing the data relationship between input and output. This highly effective forecasting framework can be applied to other problems involving financial forecasting.

References

[1] G.E.P. Box, G.M. Jenkins, Time Series Analysis: Forecasting and Control, third ed., Prentice Hall, Englewood Cliffs, CA, 1994.
[2] R.F. Engle, Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation, Econometrica 50 (1982) 987–1008.
[3] T. Bollerslev, Generalized autoregressive conditional heteroscedasticity, Journal of Econometrics 31 (1986) 307–327.
[4] J.V. Hansen, R.D. Nelson, Neural networks and traditional time series methods: a synergic combination in state economic forecasts, IEEE Transactions on Neural Networks 8 (1997) 863–873.
[5] K.J. Kim, I. Han, Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index, Expert Systems with Application 19 (2008) 125–132.
[6] Y.K. Kwon, B.R. Moon, A hybrid neurogenetic approach for stock forecasting, IEEE Transactions on Neural Networks 18 (2007) 851–864.
[7] M. Qui, G.P. Zhang, Trend time series modeling and forecasting with neural networks, IEEE Transactions on Neural Networks 19 (2008) 808–816.
[8] D. Zhang, L. Zhou, Discovering golden nuggets: data mining in financial applications, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 34 (2004) 513–522.
[9] L. Yu, S. Wang, K.K. Lai, A neural-network-based nonlinear meta modeling approach to financial time series forecasting, Applied Soft Computing 9 (2009) 563–574.
[10] P.C. Chang, C.H. Liu, A TSK type fuzzy rule based system for stock price prediction, Expert Systems with Applications 34 (2008) 135–144.
[11] S.K. Oh, W. Pedrycz, H.S. Park, Genetically optimized fuzzy polynomial neural networks, IEEE Transactions on Fuzzy Systems 14 (2006) 125–144.
[12] M.H.F. Zarandi, B. Rezaee, I.B. Turksen, E. Neshat, A type-2 fuzzy rule based expert system model for stock price analysis, Expert Systems with Applications 36 (2009) 139–154.
[13] C.F. Liu, C.Y. Yeh, S.J. Lee, Application of type-2 neuro-fuzzy modeling in stock price prediction, Applied Soft Computing 12 (2012) 1348–1358.
[14] L. Cao, F.E.H. Tay, Financial forecasting using support vector machines, Neural Computing & Applications 10 (2001) 184–192.
[15] L. Cao, F.E.H. Tay, Support vector machine with adaptive parameters in financial time series forecasting, IEEE Transactions on Neural Networks 14 (2003) 1506–1518.
[16] P.C. Fernando, A.A.R. Julio, G. Javier, Estimating GARCH models using support vector machines, Quantitative Finance 3 (2003) 163–172.
[17] T.V. Gestel, J.A.K. Suykens, D.E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, Financial time series prediction using least squares support vector machines within the evidence framework, IEEE Transactions on Neural Networks 12 (2001) 809–821.
[18] P.F. Pai, C.S. Lin, A hybrid ARIMA and support vector machines model in stock price forecasting, Omega: The International Journal of Management Science 33 (2005) 497–505.
[19] F.E.H. Tay, L. Cao, Application of support vector machines in financial time series forecasting, Omega: The International Journal of Management Science 29 (2001) 309–317.
[20] G. Valeriy, B. Supriya, Support vector machine as an efficient framework for stock market volatility forecasting, Computational Management Science 3 (2006) 147–160.
[21] H. Yang, L. Chan, I. King, Support vector machine regression for volatile stock market prediction, in: Proceedings of The Third International Conference on Intelligent Data Engineering and Automated Learning, 2002, pp. 391–396.
[22] K.J. Kim, Financial time series forecasting using support vector machines, Neurocomputing 55 (2003) 307–319.
[23] V. Kecman, Learning and Soft Computing: Support Vector Machines, Neural Networks and Fuzzy Logic Models, MIT Press, Cambridge, MA, 2001.
[24] V. Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[25] C.Y. Yeh, C.W. Huang, S.J. Lee, A multiple-kernel support vector regression approach for stock market price forecasting, Expert Systems with Applications 38 (2010) 2177–2186.
[26] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, Cambridge, UK, 2000.
[27] C.Y. Yeh, C.W. Huang, S.J. Lee, A multiple-kernel support vector regression approach for stock market price forecasting, Expert Systems with Applications 38 (2011) 2177–2186.
[28] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Machine Learning 46 (2002) 131–159.
[29] K. Duan, S. Keerthi, A.N. Poo, Evaluation of simple performance measures for tuning SVM hyperparameters, Neurocomputing 51 (2003) 41–59.
[30] J.T.Y. Kwok, The evidence framework applied to support vector machines, IEEE Transactions on Neural Networks 11 (2000) 1162–1173.
[31] C.T. Cheng, Optimizing hydropower reservoir operation using hybrid genetic algorithm and chaos, Water Resource Management 22 (2008) 895–909.
[32] W.C. Hong, Y. Dong, L.Y. Chen, S.Y. Wei, SVR with hybrid chaotic genetic algorithms for tourism demand forecasting, Applied Soft Computing 11 (2010) 1881–1890.
[33] G. Jirong, Z. Mingcang, J. Liuguangyan, Housing price forecasting based on genetic algorithm and support vector regression, Expert Systems with Applications 38 (2010) 3383–3386.
[34] C.F. Huang, A hybrid stock selection model using genetic algorithms and support vector regression, Applied Soft Computing 12 (2012) 807–818.
[35] F. Takens, Detecting strange attractors in turbulence, Lecture Notes in Mathematics 898 (1981) 366–381.
[36] S.H. Huang, P.J. Chuang, C.F. Wu, H.J. Lai, Chaos-based support vector regressions for exchange rate forecasting, Expert Systems with Applications 37 (2010) 8590–8598.
[37] T. Sauer, J.A. Yorke, M. Casdagli, Embedology, Journal of Statistical Physics 65 (1991) 579.
[38] M. Kennel, R. Brown, H.D.I. Abarbanel, Determining embedding dimension for phase space reconstruction using geometrical construction, Physical Reviews A 45 (1992) 3403–3411.
[39] H.D.I. Abarbanel, Analysis of Observed Chaotic Data, Springer, New York, 1996.
[40] A.J. Smola, B. Scholkopf, A Tutorial on Support Vector Regression, Neuro COLT Technical Report, 2003.
[41] X.S. Yang, Nature Inspired Metaheuristic Algorithms, Luniver Press, Frome, UK, 2008.
[42] S.M. Ross, Probability and Statistics for Engineers and Scientist, fourth ed., Elsevier Academic Press, Burlington, MA, 2009.