Expert Systems with Applications 36 (2009) 7947–7951

Contents lists available at ScienceDirect

Expert Systems with Applications

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

A two-stage architecture for stock price forecasting by integrating self-organizing map and support vector regression

Sheng-Hsun Hsu [a,*], JJ Po-An Hsieh [b,1], Ting-Chih Chih [c], Kuei-Chu Hsu [a]

[a] Department of Business Administration, Chung Hua University, No. 707, Sec. 2, WuFu Road, Hsinchu, Taiwan
[b] Department of Management and Marketing, The Hong Kong Polytechnic University, Hong Kong
[c] Department of Information Management, Chung Hua University, Taiwan

ARTICLE INFO

Keywords:
Stock price prediction
Support vector machine
Self-organizing map

ABSTRACT

Stock price prediction has attracted much attention from both practitioners and researchers. However, most studies in this area ignored the non-stationary nature of stock price series. That is, stock price series do not exhibit identical statistical properties at each point of time. As a result, the relationships between stock price series and their predictors are quite dynamic. It is challenging for any single artificial technique to effectively address this problematic characteristics in stock price series. One potential solution is to hybridize different artificial techniques. Towards this end, this study employs a two-stage architecture for better stock price prediction. Specifically, the self-organizing map (SOM) is first used to decompose the whole input space into regions where data points with similar statistical distributions are grouped together, so as to contain and capture the non-stationary property of financial series. After decomposing heterogeneous data points into several homogenous regions, support vector regression (SVR) is applied to forecast financial indices. The proposed technique is empirically tested using stock price series from seven major financial markets. The results show that the performance of stock price prediction can be significantly enhanced by using the two-stage architecture in comparison with a single SVR model.

1. Introduction

Stock price prediction is an important financial subject that has attracted researchers' attention for many years. In the past, conventional statistical methods were employed to forecast stock price. However, stock price series are generally quite noisy and complex. To address this, numerous artificial techniques, such as artificial neural networks (ANN) or genetic algorithms are proposed to improve the prediction results (see Table 1). Recently, researchers are using support vector regressions (SVRs) in this area (see Table 1). SVR was developed by Vapnik and his colleagues (Vapnik, 1995). Most comparison results show that prediction performance of SVR is better than that of ANN (Huang, Nokamori, & Wang, 2005; Kim, 2003; Tay & Cao, 2001a). Reasons that are often cited to explain this superiority include the face that SVRs implement the structural risk minimization principle, while ANNs use the empirical risk minimization principle. The former seeks to minimize the misclassification error or deviation from correct solution of the training data; whereas the latter seeks to minimize the upper bound of generalization error. Solution of SVR may be global optimum while neural network techniques may offer only local optimal solutions. Besides, in choosing parameters, SVRs are less complex than ANNs.

Although researchers have shown that SVRs can be a very useful for stock price forecasting, most studies ignore that stock price series are non-stationary. That is, stock price series do not exhibit identical statistical properties at each point of time and face dynamic changes in the relationship between independent and dependent variables. Such structural changes, which are often caused by political events, economic conditions, traders' expectations and other environmental factors, are an important characteristic of equities' price series. This variability makes it difficult for any single artificial technique to capture the non-stationary property of the data. Most artificial algorithms require a constant relationship between independent and dependent variables, i.e., the data presented to artificial algorithms is generated according to a constant function. One potential solution is to hybridize several artificial techniques. For example, Tay and Cao (2001b) suggest a two-stage architecture by integrating a self-organizing map (SOM) and SVR to better capture the dynamic input–output relationships inherent in the financial data. This architecture was originally proposed by Jacobs, Jordan, Nowlan, and Hinton (1991), who were inspired by the divide-and-conquer principle that is often

[*] Corresponding author. Tel.: +886 3518 6061.
E-mail addresses: spolo@chu.edu.tw (S.-H. Hsu), JJ.Hsieh@inet.polyu.edu.hk (JJ Po-An Hsieh).
[1] Tel.: +852 2766 7359; fax: +852 2765 0611.

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.10.065

Table 1
Previous research result.

Research	Comparison algorithms	Experimental data	Results
Tay and Cao (2001a)	Back-propagation neural network (BPN) and SVR	Futures and Bonds	SVR forecasts better than the BPN
Kim (2003)	BPN and SVR	Korea composite stock price index	SVR forecasts better than the BPN in terms of movement direction
Huang et al. (2005)	SVR, Linear Discriminant analysis, Quadratic discriminant analysis, and Elman BPN	Japan NIKKEI 225 Index	SVR forecasts better than other techniques in terms of movement direction
Pai and Lin (2005)	Hybrid model of ARIMA and SVM	Ten company stocks	The hybrid model forecasts better than SVR and ARIMA
Wittkemper and Steiner (1996)	Neural networks whose topologies were optimized by genetic algorithm	Sixty-seven German stocks	The new topology can yield good forecast results

used to attack complex problems, i.e., dividing a complex problem into several smaller and simpler problems so that the original problem can be easily solved. In the two-stage architecture, the SOM serves as the "divide" function to decompose the whole financial data into regions where data points with similar statistical distribution are grouped together. After decomposing heterogeneous data into different homogenous regions, SVRs can better forecast the financial indices. Although this architecture is interesting and promising, Tay and Cao (2001b) tested the effectiveness of the architecture only on futures and bonds. Whether the architecture can be employed for stock price prediction remains to be answered.

This study aims to test the effectiveness of the architecture for stock price prediction by comparing the predictive performance of the two-stage architecture with a single SVM technique. Seven stock market indices were used for this study. This paper consists of five sections. Section 2 introduces the basic concept of SVR, SOM and the two-stage architecture. Section 3 describes research design and experiments. Section 4 presents the conclusions.

2. Methodology

2.1. Support vector machine

Support vector machine (SVM) is originated as an implementation of Vapnik's (1995) structural risk minimization (SRM) principle, which reduces empirical risk, based on bounds of generalization error. The fundamental concept in SVM is to transform the data into a higher dimensional space and to find the optimal hyperplane in the space that can maximize the margin between classes. The simplest SVM only deals with a two-class problem, in which the data is separated by a hyperplane defined by a number of *support vectors*. Support vectors are a subset of the training data used to define the boundary between two classes. As a result, support vectors contain all of the information needed to define the classifier. This property makes SVM highly insensitive to the dimensionality of the feature space.

2.2. Support vector regression

Support vector regression is closely related to SVM classifiers in terms of theory and implementation. Vapnik (1995) introduced the ε–insensitive zone in the error loss function. From a theoretical point of view, this zone represents the degree of precision at which the bounds on generalization ability apply. Training vectors that lie within this zone are deemed correct, whereas those outside this zone are deemed incorrect and contribute to the error loss function. These incorrect vectors become the support vectors (see Fig. 1). Vectors lying on and outside the dotted lines are support vectors, whereas those within the ε–insensitive zone are not important in terms of the regression function. The regression surface then can be determined only by support vectors.

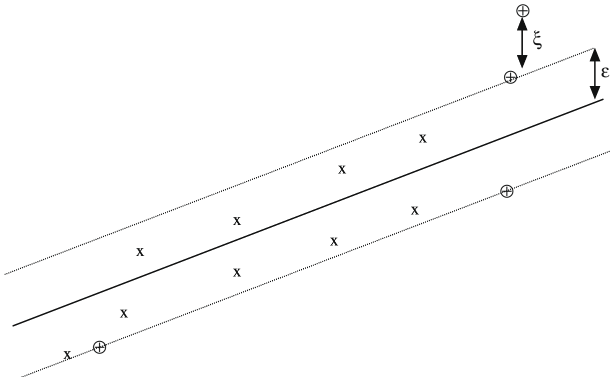

Fig. 1. Approximation function (solid line) of the SVR using an ε–insensitive zone (the area between dotted lines).

Fundamentally, SVR is linear regression in the feature space. Although it is simple and not very useful in real-world situations, it forms a building block for understanding complex SVRs. Detailed discussions of SVMs and SVRs have been given by Burges (1998), Cristianini and Shawe-Taylor (2000), and Smola and Scholkopf (1998).

Given set of training data $\{(x_1,y_1),\ldots,(x_l, y_l)\} \subset X \times R$, where X denotes the space of input patterns. The goal of SVR is to find a function $f(x)$ that deviates not more than ε from the targets y_i for all the training data, and at the same time, is as flat as possible. Let linear function $f(x)$ takes the form:

$$f(x) = w^T x + b \text{ with } w \in X, \quad b \in R \tag{1}$$

Flatness in (1) means smaller $\|w\|$. The problem can then be formulated as

$$\begin{aligned} &\min \quad \tfrac{1}{2}\|w\|^2 \\ &\text{s.t.} \quad \begin{cases} y_i - w^T x_i - b \leqslant \varepsilon \\ w^T x_i + b - y_i \leqslant \varepsilon \end{cases} \end{aligned} \tag{2}$$

However, not all problems are linearly separable. To cope with this issue, non-negative slack variables, ξ_i, ξ_i^*, are introduced to deal with the otherwise infeasible constraints of optimization problem (2). The new formation is then stated as

$$\begin{aligned} &\min \quad \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ &\text{s.t.} \quad \begin{cases} y_i - w^T x_i - b \leqslant \varepsilon + \xi_i \\ w^T x_i + b - y_i \leqslant \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geqslant 0 \end{cases} \end{aligned} \tag{3}$$

The constant C determines the trade-off of error margin between the flatness of $f(x)$ and the amount of deviation in excess of ε that is tolerated. To enable the SVR to predict a nonlinear situation, SVR maps the input data into a feature space. The mapping of X into the feature space F is denoted by

$$\Phi : \Re^n \rightarrow F$$
$$x \mapsto \Phi(x)$$

The decision function can be computed by the inner products of $\Phi(x)^{\mathrm{T}}\Phi(x_i)$ without explicitly mapping x into a higher dimension, which saves considerable computation efforts. $\Phi(x)^{\mathrm{T}}\Phi(x_i)$ is then called kernel function $K(x,z) \equiv \Phi(x)^{\mathrm{T}}\Phi(z)$.

2.3. Self-organizing map

SOM was first introduced by Kohonen (1995) and has attracted substantial research interest in a wide range of applications. For example, SOM has been shown to be quite effective in organizing large amounts of text data (Kohonen et al., 2000; Yang, Chen, & Hong, 2003). In essence, SOM is an unsupervised learning method that clusters objects having multi-dimensional attributes into a lower-dimensional space. The objective of SOM is to maximize the degree of similarity of patterns within a cluster, minimize the similarity of patterns belonging to different clusters, and then present the results in a lower-dimensional space.

The SOM network is a fully connected network, containing two layers of nodes – an input layer and an output layer. The output layer is usually in the shape of a two-dimensional grid, acting as a distribution layer. The number of nodes in the input layer is equal to the number of features associated with the input. Each output node has the same number of features as the input nodes. The input layer, as well as each output node, can be represented as a vector that contains the number of features of the input. The topology of the Kohonen SOM network is shown in Fig. 2.

The SOM technique is based on the associative neural properties of the brain in that regions of neurons are operating in a centralized and localized manner to achieve tasks (Smith & Gupta, 2000). To replicate the whole process of human brain in SOM, the learning process of SOM is as follows: when an input pattern is presented to SOM, the winning node, defined as one whose weights are most similar to the input vector, receives the most learning by strengthening its weights. Weights of the surrounding neurons are also strengthened a little so that this area is more likely to fire up when a similar input pattern is presented next time. The localized manner in SOM is implemented by adjusting two parameters: the neighborhood size and the learning rate. Let us denote $R(t)$ as the neighborhood size and $\eta(t)$ as the learning rate for weight update. The amount of learning of each neuron is determined by

$$\eta(t)e^{\frac{-d}{R(t)}} \tag{4}$$

When we let these two parameters – $R(t)$ and $\eta(t)$ – reduce over time, we observe that Eq. (4) will slowly decrease and the weight-updating process will gradually stabilize. Eq. (4) also shows that the amount of learning is the highest at the winning neuron, and decreases as the distance between a neuron and the winning neuron increases. This process of weight-updating will be performed for a specified number of iterations. The detailed steps of the algorithm are presented in Fig. 3.

```
Begin
    Set neighborhood parameters $R$
    Set learning rate parameters $\eta$
    Initialize weights randomly
    While
        For each input pattern x, compute the distance:
            $d_j \leftarrow \|x - w_j\|$
            Find index $j$ such that $d_j$ is a minimum (winning node)
            For unit $j$ and its neighborhoods, updates according to rules
                $\varsigma(t) \leftarrow \eta(t)\exp(-d/R(t))$
                $w_{ji} \leftarrow w_{ji} + \varsigma(t) \times (x_i - w_{ji})$
            Reduce learning rate $\eta(t)$ and radius of $R(t)$ of neighborhood
    Until (weight convergence or maximum no. of iterations is exceeded)
End
```

Fig. 3. SOM algorithm.

2.4. A two-stage architecture

A time series is a sequence of data points recorded sequentially in time. Time series forecasting is to predict future values based on past values and other variables. One problem in financial time series forecasting is that time series are non-stationary. The non-stationary property implies that the statistical distributions of a time series can change over time. This change may be caused by economic recession or growth, or political or environmental events. The non-stationary property will lead to a gradual change in the relationship between independent and dependent variables, i.e., the time series may have different predictor functions in different time periods. However, most learning algorithms require a constant relationship between independent and dependent variables (Cao & Gu, 2002). As a result, it is challenging to predict such structural changes of financial time series.

To address this issue, this study employs a two-stage architecture to better predict financial indices (see Fig. 4). In the first stage, the SOM is used to decompose the whole input space into regions where data points with similar statistical distributions are grouped together, so as to capture the non-stationary property of financial series. After decomposing heterogeneous data points into different homogenous regions, SVMs can then better forecast the financial indices. As demonstrated by Tay and Cao (2001b), this two-stage architecture can capture the dynamic input–output relationship inherent in futures and bonds prediction. However, whether the architecture can be used for stock price prediction remains to be answered.

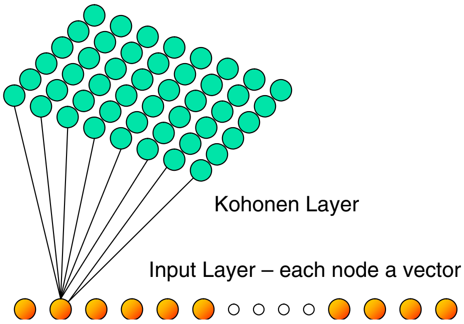

Fig. 2. Kohonen SOM topology.

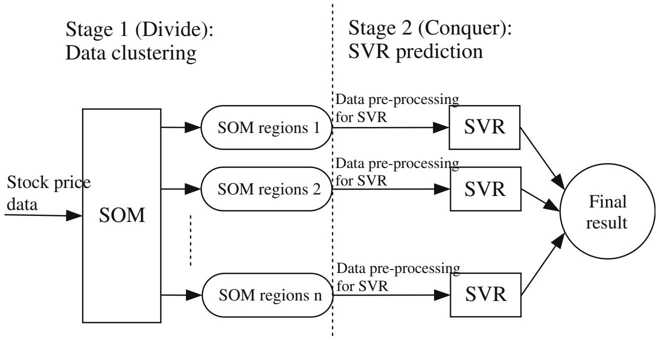

Fig. 4. The two-staged architecture.

3. Results

3.1. Data sets

We examined seven major stock market indices in this study, including the Nikkei 225 (NK), the All Ordinaries (AU), the Hang Seng (HS), the Straits Times (ST), the Taiwan Weighted (TW), the KOSPI (KO), and Dow Jones (DJ). Data were collected mostly from Yahoo Finance. Daily closing prices were used. The whole data set covers the period from July 1, 1997 to May 31, 2002. We believe that the time periods cover many important economic events, which are sufficient for testing the issue of non-stationary properties inherent in financial data.

3.2. Data processing

Original closing price was transformed into four-lagged relative difference in percentage of price (RDP), including RDP−5, RDP−10, RDP−15 and RDP−20 and one transformed closing index (EMA15). EMA15 was obtained by subtracting a fifteen-day exponential moving average from the closing indices. Input variables include RDP−5, RDP−10, RDP−15, RDP−20, and EMA15. According to Tay and Cao (2001a), this transformation can make the distribution of the data more symmetrical, thus improving the predictive power of artificial methods. The output variable RDP+5 is obtained by first smoothing the closing index with a three-day exponential moving average, because the application of a smoothing transformation to the dependent variable generally enhances the prediction performance of artificial methods. The calculations for all variables can be found in Table 2.

RDP values that lie beyond ±2 standard deviations were first identified as outliers and then replaced with the closet marginal values. About 80% of the data was used for training, and 20% for testing. Data were scaled into the range of $[-0.9, 0.9]$ to normalize each feature component so that larger input attributes do not overwhelm smaller inputs.

3.3. Performance criteria

The prediction performance is evaluated using the following statistical methods: 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 actual values and predicted values. The smaller the values of NMSE and MAE, the closer are the predicted time series values in relation to the actual values. Although predicting the actual levels of price changes is desirable, in many cases, the direction of the change is equally important. DS provides the correctness of the predicted direction of RDP+5 in terms of percentage; and the large values of DS suggest a better predictor. WDS measures the magnitude of the prediction error as well as the direction. It penalizes errors related to incorrectly predicted directions and rewards those associated with correctly predicted directions. The smaller the value of WDS, the better is the forecasting performance in terms of both magnitude and direction.

Table 2
Input and output variables.

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

Table 3
Performance metrics and their calculation.

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

3.4. SOM implementation

The determination of the size of SOM is not an easy task, because the statistical properties of the data are not always available. To avoid the trial-and-error process of determining the size of the SOM map, researchers have proposed several methods for auto-determination (Dittenbach, Rauber, & Merkl, 2002; Fritzke, 1995). Those networks can automatically determine the map size that is suitable for the specific data distribution at hand. Therefore, this study employs the growing hierarchical self-organizing map (GHSOM), developed by Dittenbach et al. (2002). GHSOM is a SOM technique which automatically grows the map size both in a hierarchical and horizontal way. Thus, besides basic parameters (e.g., learning rate and neighborhood range), GHSOM needs extra parameters, including the initial map size, the horizontal growing parameter, and the hierarchical growing parameter. We set the initial map size at 2×2 units and let the GHSOM determine the best map size. Horizontal and hierarchical growing parameters can suggest GHSOM when to stop growing horizontally and/or hierarchically. We set horizontal growing parameter at 0.05 and hierarchical growing parameter at 1. Some partitions of GHSOM may involve very few data. One characteristic of the SOM is that similar types of input data are mirrored to a large extent by their geographical vicinity within the representation space. Thus, when some partitions have very few data ($n < 30$), we merge the data into their neighborhood partitions.

3.5. SVM implementation

The typical kernel functions are the polynomial kernel $k(x,y) = (x \times y + 1)^d$ and the Gaussian kernel $k(x,y) = \exp(-(x-y)^2/\delta^2)$, where d is the degree of the polynomial kernel and δ^2 is the bandwidth of the Gaussian kernel. In our experiment, we chose the Gaussian kernel as our kernel function because it tends to achieve better performance. Tay and Cao (2001a) showed that SVRs are insensitive to ε, as long as it is a reasonable value. Thus, we choose 0.001 for ε. In determining the kernel bandwidth δ^2 and the margin C, tenfold cross validation technique was used to choose parameters that yield the best results. Subsequently, this set of parameters was applied to the test data set. The parameters tried in the tenfold cross validation process were $\delta^2 \in \{2, 1, 0.5, 0.1, 0.01, 0.001, 0.0001\}$ and $C \in \{1000, 750, 500, 100, 50, 2\}$. A SVM implementation called LIBSVM was used in this work. We used the LIBSVM because it uses the state-of-the-art optimization method.[2]

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

Table 4
Prediction performance of the two-stage architecture and the single SVM.

		NMSE	MAE	DS	WDS
NK	SVR + SOM	1.165	0.216	55.80	0.847
	SVR	1.283	0.242	50.83	0.893
AU	SVR + SOM	1.078	0.132	53.36	0.850
	SVR	1.091	0.142	50.85	0.900
HS	SVR + SOM	0.913	0.101	59.07	0.805
	SVR	0.963	0.117	37.50	1.653
ST	SVR + SOM	0.949	0.082	53.94	1.002
	SVR	1.053	0.096	34.98	1.621
TW	SVR + SOM	1.006	0.156	51.35	0.957
	SVR	1.434	0.239	48.31	1.223
KO	SVR + SOM	1.013	0.143	53.53	1.095
	SVR	1.032	0.157	40.60	1.507
DJ	SVR + SOM	1.055	0.138	50.63	1.006
	SVR	1.186	0.160	47.35	1.088

3.6. Results

The results of the two-stage architecture and the single SVM model are shown in Table 4. In terms of NMSE, MAE and WDS, we observe that the two-stage architecture achieves smaller values than the single SVM model does, on the test data set. This suggests that the two-stage architecture can have smaller deviations between predicted and actual values than the single SVM model. The values of DS are larger in the two-staged architecture than in the single SVM model. This suggests that in terms of correctness of the predicted direction of RDP+5, the two-stage architecture offers better prediction. The results are consistent in all seven data sets. A paired *t*-test is also performed to check whether there is significant difference in the four performance criterion between the two methods. The calculated *t*-value for NMSE, MAE, DS, and WDS are 2.28 ($p < 0.1$), 2.73 ($p < 0.05$), 3.10 ($p < 0.05$), and 2.81 ($p < 0.05$), respectively. This shows that the two-stage architecture outperforms the single SVM model. The findings are compatible with the conclusions by Tay and Cao (2001b).

4. Conclusion

The study shows that the performance of stock price prediction can be significantly enhanced by using the two-stage architecture in comparison with a single SVM model. The results may be attributable to the fact that financial time series are non-stationary and, therefore, the two-stage architecture can better capture the characteristics by decomposing the whole financial series into smaller homogenous regions. After decomposing the data, SVRs can better predict financial indices. The results suggest that the two-stage architecture provides a promising alternative for financial time series forecasting. Future research can further testing the idea of the two-stage architecture on other non-stationary data to evaluate the generalizability of the architecture.

Acknowledgement

The authors are grateful for the financial support from the Hong Kong Polytechnic University (Grant # G-SAA3).

References

Burges, C. J. C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2*(2), 1–47.

Cao, L., & Gu, Q. (2002). Dynamic support vector machines for non-stationary time series forecasting. *Intelligent Data Analysis, 6*, 67–83.

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

Dittenbach, M., Rauber, A., & Merkl, D. (2002). Uncovering hierarchical structure in data using the growing hierarchical self-organizing map. *Neurocomputing, 48*, 199–216.

Fritzke, B. (1995). Growing grid – A self-organizing network with constant neighborhood range and adaptation strength. *Neural Processing Letters, 2*(5), 9–13.

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

Jacobs, R. A., Jordan, M. A., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*, 79–87.

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

Kohonen, T. (1995). *Self-organizing maps*. Berlin: Springer-Verlag.

Kohonen, T., Kaski, S., Lagus, K., Salojvi, J., Paatero, V., & Sarela, A. (2000). Self organization of a massive document collection. *IEEE Transactions on Neural Networks, 11*(3), 574–585.

Pai, P.-F., & Lin, C.-S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. *Omega, 33*, 497–505.

Smith, K. A., & Gupta, J. N. D. (2000). Neural networks in business: Techniques and applications for the operations research. *Computers and Operations Research, 27*, 1023–1044.

Smola, A. J., & Scholkopf, B. (1998). A tutorial on support vector regression. NeuroCOLT Technical Report, TR Royal Holloway College, London, UK.

Tay, F. E. H., & Cao, L. J. (2001a). Application of support vector machines in financial time series forecasting. *Omega, 29*, 309–317.

Tay, F. E. H., & Cao, L. J. (2001b). Improved financial time series forecasting by combining support vector machines with self-organizing feature map. *Intelligent Data Analysis, 5*, 339–354.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. New York: Springer-Verlag.

Wittkemper, H.-G., & Steiner, M. (1996). Using neural networks to forecast the systematic risk of stocks. *European Journal of Operational Research, 90*, 577–588.

Yang, C. C., Chen, H., & Hong, K. (2003). Visualization of large category map for Internet browsing. *Decision Support Systems, 35*, 89–102.