Modern Applied Science
www.ccsenet.org/journal.html
Vol. 3, No. 12
December 2009

Prediction of Stock Market Index Movement by Ten Data Mining Techniques

Phichhang Ou (Corresponding author)
School of Business, University of Shanghai for Science and Technology
Rm 101, International Exchange Center, No. 516, Jun Gong Road, Shanghai 200093, China
Tel: 86-136-617-515-547, Fax: +86-21-55271502 E-mail: phichhang@gmail.com

Hengshan Wang
School of Business, University of Shanghai for Science and Technology
Box 461, No. 516, Jun Gong Road, Shanghai 200093, China
Tel: 86-21-5527-5971 E-mail: wanghs@usst.edu.cn

This work is supported by Shanghai Leading Academic Discipline Project, Project Number: S30504.

Abstract

Ability to predict direction of stock/index price accurately is crucial for market dealers or investors to maximize their profits. Data mining techniques have been successfully shown to generate high forecasting accuracy of stock price movement. Nowadays, in stead of a single method, traders need to use various forecasting techniques to gain multiple signals and more information about the future of the markets. In this paper, ten different techniques of data mining are discussed and applied to predict price movement of Hang Seng index of Hong Kong stock market. The approaches include Linear discriminant analysis (LDA), Quadratic discriminant analysis (QDA), K-nearest neighbor classification, Naïve Bayes based on kernel estimation, Logit model, Tree based classification, neural network, Bayesian classification with Gaussian process, Support vector machine (SVM) and Least squares support vector machine (LS-SVM). Experimental results show that the SVM and LS-SVM generate superior predictive performances among the other models. Specifically, SVM is better than LS-SVM for in-sample prediction but LS-SVM is, in turn, better than the SVM for the out-of-sample forecasts in term of hit rate and error rate criteria.

Keywords: Data mining, Stock price movement prediction, SVM, LS-SVM, NN, Bayesian classification with Gaussian processes

1. Introduction

Financial market is a complex, nonstationary, noisy, chaotic, nonlinear and dynamic system but it does not follow random walk process, (Lo & Mackinlay, 1988; Deng, 2006). There are many factors that may cause the fluctuation of financial market movement. The main factors include economic condition, political situation, traders' expectations, catastrophes and other unexpected events. Therefore, predictions of stock market price and its direction are quite difficult. In response to such difficulty, data mining (or machine learning) techniques have been introduced and applied for this financial prediction. Most of the studies have focused on the accurate forecasting of the value of stock price. However, different investors adopt different trading strategies; therefore, the forecasting model based on minimizing the error between the actual values and the forecasts may not be suitable for them. In stead, accurate prediction of movement direction of stock index is crucial for them to make effective market trading strategies. Some recent studies have suggested that trading strategies illustrated by the forecasts based on the direction of stock price change may be more effective and generate higher profit. Specifically, investors could effectively hedge against potential market risk and speculators as well as arbitrageurs could have opportunity of making profit by trading stock index whenever they could obtain the accurate prediction of stock price direction. That is why there have been a number of studies looking at direction or trend of movement of various kinds of financial instruments (such as Wu & Zhang, 1997; O'connor et al, 1997). But, these studies do not use data mining based classification techniques. Data mining techniques have been introduced for prediction of movement sign of stock market index since the results of Leung et al (2000) and Chen et al (2001), where LDA, Logit and Probit and Neural network were proposed and compared with parametric models, GMM-Kalman filter. Kim (2003) applied newly and powerful techniques of data mining, SVM and Neural network, to forecast the direction of stock index price based on economic indicators. To obtain more profits from the stock market, more and more "best" forecasting techniques are used by different traders. In stead of a single method, the traders need to use various forecasting techniques to gain multiple signals and more information about the future of the markets. Kumar & Thenmozhi (2006) collected five different approaches including SVM, Random forecast, Neural network, Logit and LDA to predict Indian stock index movement based on economic variable indicators. From the comparison, the SVM outperformed the others in forecasting S&P CNX NIFTY index direction as the model does not require any priori assumptions on data property and its algorithm results global optimal solution which is unique. Huang et al (2005) also forecasted the movement direction of Japanese stock market (NIKKEI 225 index) by various techniques such as SVM, LDA, QDA, NN and the all-in-one combined approach. The SVM approach also gives better predictive capability than other models: LDA, QDA and NN, following the out-performance of the combined model. In the study, they defined the movement of the NIKKEI 225 index based on two main factors including American stock market, S&P 500 index, which is the most influence on the world stock markets including Japanese market, and the currency exchange rate between Japanese Yen and US dollar.

In our study, ten different techniques of data mining are discussed and applied to predict price movement of Hang Seng index of Hong Kong stock market. The approaches include Linear discriminant analysis (LDA), Quadratic discriminant analysis (QDA), K-nearest neighbor classification, Naïve Bayes based on kernel estimation, Logit model, Tree based classification, neural network, Bayesian classification with Gaussian process, Support vector machine (SVM) and Least squares support vector machine (LS-SVM). The main goal is to explore the predictive ability of the ten data-mining techniques in forecasting movement direction of Hang Seng Index based on five factors, including its open price, high price, low price, S&P 500 index, and currency exchange rate between HK dollar and US dollar. The general model of stock price movement is defined as

$$D_t = f(O_{t-1}, H_{t-1}, L_{t-1}, S\,\&\,P500_{t-1}, FX_{t-1}),$$

where D_t is the direction of HSI movement at time t and is defined as a categorical value "1" if the closing price at time t is greater than the closing price at time $t-1$ and as "0", otherwise. The function $f(.)$ can be linear or nonlinear and it is estimated by the ten data-mining algorithms.

O_{t-1} denotes the open price of HSI at time $t-1$;

H_{t-1} is the high price of HSI at time $t-1$;

L_{t-1} is the low price in a day of HSI at time $t-1$;

$S\,\&\,P500_{t-1}$ is the closing price of S&P 500 index at time $t-1$.

FX_{t-1} is the currency exchange rate between HK dollar and US dollar.

All the inputs are transformed into log return to remove any trend pattern.

The remaining of the paper is organized as follow. Next sections describe the data and prediction evaluation. Section 3 briefly discusses the ten different algorithms. The final section is for conclusion.

2. Data description

We examine the daily change of closing prices of Hang Seng index based on five predictors, Open price, High price, Low price, S&P 500 index price, and Exchange rate USD against HKD. The stock prices are downloaded from the Yahoo finance and the foreign exchange rate is taken from website of Federal Reserve Bank of ST. Louis. The sample period is from Jan 03 2000 to Dec. 29 2006 so that the whole sample is of 1732 trading days. The data is divided into two sub-samples where the in-sample or training data spans from Jan 03 2000 to Dec 30 2005 with 1481 trading days. The whole year 2006 from Jan 1 2006 to Dec 29 2006 of size 250 trading days are reserved for out-of-sample or test data. Figure1 displays the actual movement of HSI closing prices for the whole sample. Figure 2 plots S&P 500 price and its log return and Figure 3 shows the plots of price and log return of exchange rate of HKD against USD. To measure the predictive performances by different models, Hit rate and Error rate are employed and defined as Hit rate = $\frac{1}{m}\sum_{i=1}^{m} I_{[A_i = P_i]}$ and Error rate = $\frac{1}{m}\sum_{i=1}^{m} I_{[A_i \neq P_i]}$ where A_i is the actual output for ith trading day and P_i is the predicted value for ith trading day, obtained from each model. Here m is the number of the out-of-sample. R software with related packages is used to conduct the whole experiment; for example Karatzoglou (2004, 2006) illustrates the R commands for SVM, LSSVM, and Bayesian classification with Gaussian Processes.

3. Data-mining methods

Let the training data be $\{ (x_1, y_1), \dots, (x_n, y_n) \}$ where $X = (X_1, \dots, X_p)$ denote real-valued random input vector and the response Y is categorical, i.e. $Y \in \{1, 2, \dots, K\}$. The goal is to form a predictor $G(x)$ to predict Y based on X. So $G(x)$ divides the input space (or feature vector space) into a collection of regions where each labeled by one class. For binary class problem, $Y \in \{1, 2\}$, the decision boundary between the two classes is a hyperplane in the feature space. A hyperplane in the p- dimensional input space is the set: $\{ x : \beta_0 + \sum_{i=1}^{p} \beta_i x_i = 0 \}$. (1)

The two regions separated by the hyperplane are $\{ x : \beta_0 + \sum_{i=1}^{p} \beta_i x_i > 0 \}$ and $\{ x : \beta_0 + \sum_{i=1}^{p} \beta_i x_i < 0 \}$.

Now we define Bayes classification rule. Suppose the training data $\{ (x_1, y_1), \dots, (x_n, y_n) \}$ are independent samples from the joint distribution of X and Y, $f_{X,Y}(x, y) = p_Y(y) f_{X/Y}(x / Y = y)$ and the loss function of classifying Y as $G(X) = \hat{Y}$ is $L(\hat{Y}, Y)$, where the marginal distribution of Y is specified by the pmf $p_Y(y)$ and $f_{X/Y}(x / Y = y)$ is the conditional distribution of X given $Y = y$. The goal of classification is to minimize the expected loss defined as $E_{X,Y} L(G(X), Y) = E_X [E_{Y/X} L(G(X), Y)]$. To minimize the left hand side of the expected loss, it suffices to minimize $E_{Y/X} L(G(X), Y)$ for each X. Hence the optimal classifier is $G(X) = \arg\min_y E_{Y/X=x} L(y, Y)$. For 0-1 loss function $L(y, y') = 0$ for $y = y'$ and 0 otherwise, we have $E_{Y/X=x} L(y, Y) = 1 - \Pr(Y = y / X = x)$. Therefore, the classification rule, called Bayes rule, becomes the rule of maximum a posterior probability: $G(x) = \arg\min_y E_{Y/X=x} L(y, Y) = \arg\max_y \Pr(Y = y / X = x)$.

We consider ten algorithms for classification where some of them attempt to estimate $\Pr(Y = y / X = x)$ and then apply the Bayes rule $G(x) = \arg\max \Pr(Y = y / X = x)$.The algorithms include linear discriminant analysis (LDA), Quadratic discriminant analysis (QDA), Naïve Bayes based on kernel estimation, and Logit model. Other types of data mining techniques are K-nearest neighbor classification, Tree based classification, neural network, Bayesian classification with Gaussian process, Support vector machine (SVM) and Least squares support vector machine (LS-SVM). The last three models take more advantage via kernel based methods.

3.1 Linear Discriminant Analysis

The goal here is to obtain class posteriors $\Pr(Y / X)$ for optimal classification. Suppose $f_k(x)$ is the class-conditional density of X in class $Y = K$ and let π_k be the prior probability of class K with $\sum_{k=1}^{K} \pi_k = 1$. By Bayes theorem, $\Pr(Y = k / X = x) = f_k(x)\pi_k / \sum_{i=1}^{K} f_i(x)\pi_i$. Suppose each class density is from Gaussian distribution defined as $f_k(x) = (2\pi)^{-p/2} |\Sigma_k|^{-1/2} \exp(-\frac{1}{2}(x - \mu_k)^T \Sigma_k^{-1} (x - \mu_k))$ and the classes are assumed to have a common covariance matrix $\Sigma_k = \Sigma \quad \forall k$. Considering the log ratio of the two classes k and l posteriors

$\log[\Pr(Y = k / X = x) / \Pr(Y = l / X = x)]$

$= \log[f_k(x) / f_l(x) + \log[\pi_k / \pi_l] = \log[\pi_k / \pi_l] - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1} (\mu_k - \mu_l) + x^T \Sigma^{-1} (\mu_k - \mu_l)$ which is a linear equation in x in p dimensional hyperplane defined in (1), the linear discriminant functions are obtained $\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$ and the LDA classifier is $G(x) = \arg\max_k \delta_k(x)$. From the training data, we can estimate the Gaussian distribution parameters as

$\hat{\pi}_k = N_k / N$ where N_k is the number of class k observations; $\hat{\mu}_k = \sum_{y_i = k} x_i / N_k$ and $\hat{\Sigma} = \sum_{k=1}^{K} \sum_{y_i = k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T / (N - K)$. For two classes {1, 2}, the LDA rule classifies to class 2 if $x^T \hat{\Sigma}^{-1} (\hat{\mu}_2 - \hat{\mu}_1) > \frac{1}{2} \hat{\mu}_2^T \hat{\Sigma}^{-1} \hat{\mu}_2 - \frac{1}{2} \hat{\mu}_1^T \hat{\Sigma}^{-1} \hat{\mu}_1 + \log(N_1 / N) - \log(N_2 / N)$ and class 1 otherwise. (2)

From the experimental result, we have $X = (X_1, \dots X_5)$ with dimension of p =5 so that five coefficients of the linear discriminant in (2) are obtained as the following: V1 = -0.79761825, V2 = 0.91216594, V3 = 0.74655315, V4 = -0.02241239, V5 = -1.85401191. Prior probabilities of groups: 0 and 1 are 0.5037137 and 0.4962863 respectively. For the in-sample data, the hit rate is 0.8393 and the error rate is 0.1607, while in the out-of-sample data, the hit rate and error rate are 0.8440 and 0.1560 respectively.

3.2 Quadratic discriminant analysis

For the QDA, the above Σ_k are not assumed equal for each k, then $\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k) + \log\pi_k$. The decision boundary between two classes k and l is the quadratic equation $\{ x : \delta_k(x) = \delta_l(x) \}$. The estimates for QDA are similar to those LDA except that separate covariance matrix must be estimated for each class. Here QDA needs $(K-1)\{p(p+3)/2+1\}$ parameters. See McLachlan (1992) and Duda et al (2000) for comprehensive discussion on discriminant analysis. From the prediction results, the hit rate and error rate for training data by QDA are 0.8305 and 0.1695 respectively. For the test data, the hit rate and error rate are 0.8480 and 0.1520 respectively.

3.3 K-nearest neighborhood method

K-nearest neighbor method is one of the simplest machine learning algorithms used for classifying objects based on closest training examples in the feature space. An object is classified by a majority being assigned to the class most common amongst its k nearest neighbors. Formally, the k-nearest neighbor approach uses the training data set $\{ (x_1, y_1), \ldots, (x_n, y_n) \}$ closest in input space to x to form $\hat{Y}$. Specifically, the k-nearest neighbor fit for $\hat{Y}$ is defined as $\hat{y}(x) = \frac{1}{k}\sum_{x_i \in N_k(x)} y_i$ where $N_k(x)$ is the neighborhood of x defined by k closest points x_i in the training example. That is, we find the k observations with x_i closest to x in input space and average their responses. k is estimated by cross-validation technique. The algorithm starts with the determination of the optimal k based on RMSE done by cross validation technique, then calculate the distance between the query distance and all the training samples. After sorting the distance and determination of the nearest neighbors based on the kth minimum distance, gather the Y of the nearest neighbors. Finally, use simple majority of the category Y of nearest neighbors as the prediction value of the query distance. Noticeably, the k-nearest neighbor approach does not rely on prior probabilities like LDA and QDA.

Results: Table 1 displays the process of choosing k by cross-validation technique in the experiment. We consider k = 30 as an initial range and then select optimal k in the range. The best k = 10 is obtained corresponding to smallest error (0.1708). [Insert Table 1 around here]. The performance results are given as the following. For the training data, the hit rate is 0.8312 and the error rate is 0.1688 and for the test data, the hit rate is 0.7960 and the error rate is 0.2040.

3.4 Naïve Bayes classification method

This is a well established Bayesian method primarily formulated for performing classification tasks. Given its simplicity, i.e., the assumption that the independent variables are statistically independent, Naive Bayes models are effective classification tools that are easy to use and interpret. Naive Bayes is particularly appropriate when the dimensionality of the feature space (i.e., number of input variables) is high (a problem known as the curse of dimensionality). Mathemcally, Naïve Bayes model requires an assumption that given a class $Y = j$, the features X_k are independent so that $f_j(X) = \prod_{k=1}^{p} f_{jk}(X_k)$. The estimates of $f_{jk}(.)$ is from the training data via kernel smoothing. Naïve Bayes classification is $G(x) = \arg\max_j f_j(X)\pi_j$ and π_j is estimated by the sample proportions. See Mitchell (2005) for precise explanation. From the result obtained in the experiment, the hit rate and error rate for in-sample data are 0.8386 and 0.1614 respectively. For the out-of-sample, the hit rate is 0.8280 and the error rate is 0.1720.

3.5 Logit model

Logistic regression refers to methods for describing the relationship between a categorical response variable and a set of predictor variables. It can be used to predict a dependent variable on the basis of independents and to determine the percent of variance in the dependent variable explained by the independents. The logistic regression applies maximum likelihood estimation after transforming the dependent into a logit variable. In this way, logistic regression estimates the probability of a certain event occurring. Note that logistic regression calculates changes in the log odds of the dependent, not changes in the dependent itself. From the Friedman et al (2008), the model for logistic regression is given as: $\pi(x) = \Pr(Y=1/X=x) = \dfrac{\exp(\beta_0 + \sum_{i=1}^{p}\beta_i X_i)}{1+\exp(\beta_0 + \sum_{i=1}^{p}\beta_i X_i)}$ for two classes of output Y. We obtain β's using the maximum likelihood approach.

Logit is given by: $G(x) = \log\left(\dfrac{\pi(x)}{1+\pi(x)}\right) = \log\dfrac{\Pr(Y=1/X=x)}{\Pr(Y=0/X=x)} = \beta_0 + \sum_{i=1}^{p}\beta_j X_j$. The curve of $\pi(x)$ are called sigmoid because they are S-shape and therefore nonlinear. Statisticians have chosen the logistic distribution to model binary data because of its flexibility and interpretability. The minimum for $\pi(x)$ is attained at $\lim_{a\to-\infty} e^a/(1+e^a) = 0$, and the maximum for $\pi(x)$ is obtained at $\lim_{a\to\infty} e^a/(1+e^a) = 1$.

From the experiment, the coefficients of β's estimated by maximum likelihood are obtained as $\beta_0 = -0.0031$, $\beta_1 = -2.3285$, $\beta_2 = 2.6697$, $\beta_3 = 2.4129$, $\beta_4 = 0.0736$ and $\beta_5 = -3.1932$. The performance results show that the hit rate and error rate for in-sample data is 0.8474 and 0.1526 respectively while the hit rate and error rate for out-of-sample data are 0.8560 and 0.1440 respectively.

3.6 Tree based classification

Classification tree is one of the main techniques used in Data Mining. Classification trees are used to predict membership of objects in the classes of a categorical dependent variable from their measurements on one or more predictor variables. The goal of classification trees is to predict or explain responses on a categorical dependent variable, and as such, the available techniques have much in common with the techniques used in the more traditional methods of Discriminant Analysis, Cluster Analysis, Nonparametric Statistics, and Nonlinear Estimation. The flexibility of classification trees makes them a very attractive analysis option as it does not require any assumption on the distribution like traditional statistical methods. Technically, the Tree-based methods partition the feature space into a set of rectangles, and then fit a simple model in each one. Starting from the root of a tree, the feature space X containing all examples is split recursively into subsets usually two at a time. Each split depends on the value of only a unique variable of input x. If x is categorical, the split is of the form $x \in A$ or $x \notin A$ where A is subset of X. The goodness of split is measured by an impurity function defined for each node. The basic idea is to choose a split such that the child nodes are purer than their parent node. The split continues till the end subsets (leaf nodes) are 'pure'; that is till one class dominates. For an impurity function ϕ, define the impurity measure by $i(t) = \phi(\Pr(1/t), \Pr(2/t), \ldots, \Pr(K/t))$ where $\Pr(j/t)$ is the estimated probability of class j within node t. The goodness of a split s for node t is

$\phi(s,t) = \Delta I(s,t) = i(t) - p_R i(t_R) - p_L i(t_L)$ where p_R and p_L are the proportions of the samples in node t that go to the right node and left node respectively. Possible impurity functions:

a. Entropy: $i(t) = \sum_{j=1}^{K} \Pr(j/t) \log(1/\Pr(j/t))$

b. Gini index: $i(t) = \sum_{j=1}^{K} \Pr(j/t)(1-\Pr(j/t))$ where $\Pr(j/t) = \frac{1}{N(t)} \sum_{x \in t} I_{[j_x = j]}$, $I_{[j_x = j]} = 1$ if $j_x = j$ and 0 otherwise,

and $N(t)$ is the total number of samples in node t. The criteria is to stop splitting a node t when $\max_{s \in S} \Delta I(s,t) < \beta$, β is the chosen threshold. It is not trivial to choose β as it leads to overfitting or underfitting problems for a new data prediction. To solve this problem, we go for pruning approach. The idea is to obtain the subtree from the initial large tree. One of the most popular techniques is the Cost-complexity pruning. Let $T_{\max}$ be initial large tree and let the pruned subtree by $T \le T_{\max}$. Then the cost complexity measure $C_\alpha(T)$ is defined as $C_\alpha(T) = R(T) + \alpha \left|\tilde{T}\right|$, $\left|\tilde{T}\right|$ denotes the number of leaf nodes. $R(T) = \sum_{t \in \tilde{T}} \Pr(t)\, r(t)$ is the error measure of T, $r(t) = 1 - \max_j \Pr(j/t)$ and $\alpha \ge 0$ is the complexity parameter. Here $C_\alpha(T)$ represents the tradeoff between the cost of a tree and its complexity. The goal of cost-complexity pruning is for each α choose a tree $T(\alpha) \le T_{\max}$ such that $C_\alpha(T)$ is minimized. The estimation of α is achieved by crossvalidation. We choose $\hat{\alpha}$ that minimizes the cross-validation sum of squares. Thus the final tree is $T(\hat{\alpha})$. Tree based method is considered one of top ten algorithm of data mining technique (Wu et al, 2008). We refer to Quinlan (1986) and Friedman et al (2008) for detailed discussion on the tree based classification. For illustrative application of Tree method in predicting stock price behavior is referred to Pearson (2004).

Experimental results:

Denote variables actually used in tree construction by V1 = Open price, V2 = Low price, V3=High price, V4= S&P500, V5 = FX, V6 = Close price. The class "1" is when the next price is larger than the previous price, while the class "0" is refers to when the next price is smaller than the previous price. Table 2 show the process of pruning tree.

From the Table 2, we choose CP = 0.0036281 corresponding to 17 splits based on the smallest value of the x_error: 0.40272. The initial large tree is not displayed to reduce the space but pruned tree is illustrated in the appendix section. The performance result shows that the hit rate is 0.8717 and error rate is 0.1283 in the training data; while the test data, the hit rate is 0.8 and the error rate is 0.2.

3.7 Neural network for classification

Haykin (1994) defines neural network as a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects: (1) Knowledge is acquired by the network through a learning process, and (2) Interneuron connection strengths known as synaptic weights are used to store the knowledge. The literature on neural network is enormous, and its application spreads over many scientific areas see Bishop (1995) and Ripley (1996) for detailed. Recently, the neural network has been well known for good capability in forecasting stock market movement. Let's go directly to its formulation.

Let X be input vector and Y be output taking value as categorical. Following notations in Hastie (1996), the neural network model can be represented as

$$z_j = \sigma(\alpha_{0j} + \alpha_j^T x)\,,\quad j = 1,\dots,m$$

$$\hat{y}_k = f_k(\beta_{0k} + \beta_k^T z)\,,\quad k = 1,\dots,q$$

where $\sigma(z) = 1/(1+e^{-z})$ activation function called sigmoid. The parameters α_{jl} and β_{kj} are known as weights and α_{0j} and β_{0k} are bias. Here we use $f_k(v) = 1/(1+e^{-v})$ the inverse logit for binary classification.

To learn the neural network, back propagation is used.

Suppose we use least squares on a sample of training data to learn the weights:

$$R(\alpha,\beta) = \sum_{i=1}^{N} \sum_{j=1}^{k} (y_k^i - \hat{y}_k^i)^2\,,$$

$$\frac{\partial R^i}{\partial \beta_{kj}} = -2(y_k^i - \hat{y}_k^i) f_k'(\beta_k^T z^i) z_j^i\,,$$

$$\frac{\partial R^i}{\partial \alpha_{jl}} = -\sum_{k=1}^{K} 2(y_k^i - \hat{y}_k^i) f_k'(\beta_k^T z^i)\beta_{kj}\sigma'(\alpha_j^T x^i) x_l^i\,.$$

The ith component is denoted by superscript i.

Gradient update at the (r+1)st iteration,

$$\beta_{kj}^{(k+1)} \quad \leftarrow \quad \beta_{kj}^{(r)} - \gamma_r \sum_i \frac{\partial R^i}{\partial \beta_{kj}^{(r)}}$$

$$\alpha_{jl}^{(r+1)} \quad \leftarrow \quad \alpha_{jl}^{(r)} - \gamma_r \sum_i \frac{\partial R^i}{\partial \alpha_{jl}^{(r)}}\,.$$

Here γ_r is the learning rate.

In the experimental analysis, $l = 1,\dots,5$; $j = 1,2$; $k = 1,2$ so that a 5-2-2 network with 18 weights are obtained with the following estimated weights:

$\alpha_{01} = -9.33$ $\alpha_{11} = 2.57$ $\alpha_{12} = -6.14$ $\alpha_{13} = -3.06$ $\alpha_{14} = 2.40$ $\alpha_{15} = 2.70$

$\alpha_{02} = -0.19$ $\alpha_{21} = -0.84$ $\alpha_{22} = 0.86$ $\alpha_{23} = 0.87$ $\alpha_{24} = 0.09$ $\alpha_{25} = -1.15$

$\beta_{01} = 2.59$ $\beta_{11} = 3.99$ $\beta_{21} = -5.82$

$\beta_{02} = -2.57$ $\beta_{12} = -3.99$ $\beta_{22} = 5.80$

From the prediction performances, the hit rate is 0.8481 and error rate is 0.1519 in the in-sample data. For the out-of-sample, 0.8520 and 0.1480 are hit rate and error rate respectively.

3.8 Bayesian classification for Gaussian process

Gaussian processes are based on the prior assumption that adjacent observations should convey information about each other. Particularly, it is assumed that the observed variables follow normal distribution and that the coupling between them takes place by covariance matrix of a normal distribution. Using the kernel matrix as the covariance matrix is a convenient way of extending Bayesian modeling of linear estimators to nonlinear situations.

In regression problem, the goal is to predict a real valued output based on a set of input variables. It is possible to carry out nonparametric regression using Gaussian process. With Gaussian prior and Gaussian noise model, the solution of the regression problem can be obtained via Kernel function placed on each training data; the coefficients are determined by solving a linear system. If the parameter θ indexed in the Gaussian process are unknown, Bayesian inference can be carried out for them. Gaussian process can be extended to classification problems by defining a Gaussian process over y , the input to the sigmoid function. The goal is to predict $\Pr(Y = j / X = x)$, $j = 1,\dots,K$. For binary case, $j = 0,1$, $\Pr(Y = 1/ X = x)$ is estimated by $\sigma(y(x))$ where $\sigma(y) = 1/(1+e^{-y})$. The idea is to place the Gaussian prior on $y(x)$ and combine it with the training data $D = (x_i, t_i)$, $i = 1,\dots,n$ to obtain predictions for new x points. Bayesian treatment is imposed by integrating over uncertainty in y and in the parameters that control the Gaussian prior. Then the Laplace's approximation is employed to obtain the results of the integration over y . Specifically, let $\Pr(y)$ be the prior of $y = (y(x_1), y(x_2), \dots, y(x_n))$ so that $\Pr(y_*, y)$ is the joint distribution including y_*. Given new input x_*, we want to predict $y_* = y(x_*)$ based on the $D = (x_i, t_i)$, $i = 1, \dots, n$. Let $\Pr(t / y)$ be the probability of observing the particular values $t = (t_1, t_2, \dots, t_n)^T$ given actual values y (i.e., noise model). Then we have

$\Pr(y_* / t) = \int \Pr(y_*, y / t) dy = \frac{1}{\Pr(t)} \int \Pr(y_*, y) \Pr(y) \Pr(t / y) dy = \int \Pr(y_*, y) \Pr(y / t) dy$. Hence the predictive distribution for y_* is found from the marginalization of the product of the prior and the noise model.

The integral terms are estimated by Laplace's approximation. William et al (1999) and Rasmussen et al (2006) provided a comprehensive and detailed discussion on the Bayesian classification with Gaussian processes.

The following shortly describes the experimental results obtained from training and forecasting the movement of HSI by Bayesian classification with Gaussian processes.

Problem type: classification

Gaussian Radial Basis kernel function hyperparameter: sigma = 0.414967184016261

Number of training instances learned: 1481

Train error: 0.238758065

Cross validation error: 0.1647560

In-sample data: Hit rate is 0.8595 and error rate is 0.1405.

Out-of-sample data: Hit rate is 0.8520 and error rate is 0.1480.

3.9 Support vector machine for classification

A popular machine learning algorithm with neural network type is SVM, support vector machine, developed by Vapnik (1995). The SVM is a kernel based learning approach like the above Gaussian processes for classification. However, the SVM does not require any assumptions on the data property like in Gaussian processes. The SVM has been successfully applied for various areas of predictions; for instance, in financial time series forecasting (Mukherjee et al, 1997; Tay and Cao, 2001), marketing (Bend-David and Lindenbaum, 1997), estimating manufacturing yields (Stoneking, 1999), text categorization (Joachims, 2002), face detection using image (Osuna et al, 1997), handwritten digit recognition (Burges and Schokopf, 1997); Cortes and Vapnik, 1995), medical diagnosis (Tarassenko et al, 1995).

The SVM formulation can be started as the following.

Given a training set $(x_i, y_i), i = 1, 2, \dots, N$ with input data $x_i \in R^n$ and corresponding binary class label $y_i \in \{-1, 1\}$, SVM algorithm seeks the separating hyperplane with largest margin. The problem can be formulated as follow:

$$\min_{w,b,\xi} \frac{1}{2} w^T w \qquad (3)$$

subject to $y_i(w^T x_i + b) \ge 1 \quad i = 1, \dots, N$. (4)

Standard method to solve the problem (3)-(4) is convex programming, where Lagrange method is applied to transfer primal to dual problems of optimization. Specifically, we first construct Lagrangian

$$L_P = \frac{1}{2} w^T w - \sum_{i=1}^{N} \lambda_i [y_i (w^T x_i + b) - 1] \qquad (5)$$

λ_i are nonnegative Lagrange multipliers corresponding to (4). The solution is achieved by a saddle point of the Lagrangian which has to be minimized with respect to w and b and maximized with respect to λ_i.

Differentiating (5) and set the results equal to zero,

$$\frac{\partial L_P}{\partial w} = w - \sum_{i=1}^{N} \lambda_i y_i x_i = 0 \qquad (6)$$

$$\frac{\partial L_P}{\partial b} = \sum_{i=1}^{N} \lambda_i y_i = 0 \qquad (7)$$

The optimal solution is obtained from (6) as $w^* = \sum_{i=1}^{N} \lambda_i^* y_i x_i$ (8)

where * denotes optimal values. Now substituting (8) and (7) into (5),

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} w^T w = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j \quad (9)$$

The dual problem is posed as quadratic programming:

maximize L_D

subject to $\sum_{i=1}^{N} \lambda_i y_i = 0$

$0 \le \lambda_i \le C$, $i = 1, \dots, N$.

The conditions $\lambda_i^* [y_i (w^{*T} x_i + b^*) - 1] = 0$, $i = 1, \dots, N$ (10)

implies that $\lambda_i > 0$ only when constraint (4) active. The vectors for which $\lambda_i > 0$ are called support vectors.

By (10), we obtain $b^* = y_i - w^{*T} x_i$ for any support vector x_i. By linearity of the inner product and (8), the decision function for the linear separable case is

$$f(x) = sign\left(w^T x + b\right) = sign\left(\sum_{i=1}^{N} y_i \lambda_i^* (x^T x_i) + b^* \right).$$

For linearly non-separable case, we introduce a new set of variables $\{ \xi_i \}, i = 1, \dots, N$ that measure the amount of violation of the constraints.

Thus (3) and (4) are modified as

$$\min_{w,b,\xi_i} \frac{1}{2} w^T w + C \left(\sum_{i=1}^{N} \xi_i \right)^k \quad (11)$$

subject to $y_i \left(w^T x_i + b\right) \ge 1 - \xi_i$, $i = 1, \dots, N$ (12)

$\xi_i \ge 0$, $i = 1, \cdots, N$ (13)

where C and k are predetermined parameters defines the cost of constraints.

The Lagrangian is constructed as

$$L_P = \frac{1}{2} w^T w - \sum_{i=1}^{N} \lambda_i [y_i (w^T x_i + b) - 1 + \xi_i] - \sum_{i=1}^{N} \gamma_i \xi_i + C \left(\sum_{i=1}^{N} \xi_i \right)^k \quad (14)$$

where λ_i and γ_i are Lagrange multipliers which are associated with constraints (12) and (13) respectively. The solution to this problem is determined by minimizing L_P with respect to w, k and b and maximizing with respect to λ_i and γ_i.

Differentiating (14) and setting equal to zero,

$$\frac{\partial L_P}{\partial w} = w - \sum_{i=1}^{N} \lambda_i y_i x_i = 0 \quad (15)$$

$$\frac{\partial L_P}{\partial b} = \sum_{i=1}^{N} \lambda_i y_i = 0 \quad (16)$$

$$\frac{\partial L_P}{\partial k} = \begin{cases} kC \left(\sum_{i=1}^{N} \xi_i \right)^{k-1} - \lambda_i - \gamma_i = 0 \ , & k > 1 \\ C - \lambda_i - \gamma_i = 0 \ , & k = 1 \end{cases}$$

or $\delta - \lambda_i - \gamma_i = 0$ by denoting $\sum_{i=1}^{N} \xi_i = \left(\delta / Ck\right)^{1/k-1}$ for $k > 1$. (17)

From (15), $w^* = \sum_{i=1}^{N} \lambda_i^* y_i x_i$. (18)

Substituting (18),(17),(16) into (14), we obtain

$$L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j - \frac{\delta^{k/k-1}}{(kC)^{1/k-1}} (1 - \frac{1}{k}) .$$

This leads to the dual problem as

maximize $L_D = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j x_i^T x_j - \frac{\delta^{k/k-1}}{(kC)^{1/k-1}} (1 - \frac{1}{k})$

subject to $\sum_{i=1}^{N} \lambda_i y_i = 0$

$0 \le \lambda_i \le \delta$, $i = 1,\ldots,N$.

When $k = 1$,

$$\max L_D = \sum_{i=1}^{N}\lambda_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j x_i^T x_j$$

subject to $\sum_{i=1}^{N} \lambda_i y_i = 0$

$0 \le \lambda_i \le C$, $i = 1,\ldots,N$.

Hence the classifier is

$$f(x) = sign\left(\sum_{i=1}^{N} y_i \lambda_i^* (x^T x_i) + b^*\right)$$

where $b^* = y_i - w^{*T} x_i$ for any support vector x_i such that $0 < \lambda_i < C$ following from $\lambda_i^*[y_i(w^{*T} x_i + b^*) - 1 + \xi_i] = 0$, $i = 1,\ldots,N$.

For nonlinear classifier, we need to map the input variable x into a higher dimensional feature space and work with linear classification in that space, i.e., $x \to \phi(x) = (\alpha_1\phi_1(x),\ldots,\alpha_n\phi_n(x),\ldots)$ where $\{\alpha_n\}_{n=1}^{\infty}$ are some real numbers and $\{\phi_n\}_{n=1}^{\infty}$ are some real functions.

The solution of the SVM has the form

$$f(x) = sign\left(\phi(x)^T w^* + b^*\right) = sign\left(\sum_{i=1}^{N} y_i \lambda_i^* \phi(x)^T \phi(x_i) + b^*\right)$$

To avoid complex calculation of scalar product $\phi(x)^T \phi(x_i)$, we introduce kernel function:

$K(x, y) \equiv \phi(x)^T \phi(x) = \sum_{n=1}^{\infty} \alpha_n^2 \phi_n(x)\phi_n(y)$ which satisfies Mercer’s conditions.

Hence, $f(x) = sign\left(\sum_{i=1}^{N} y_i \lambda_i^* K(x, x_i) + b^*\right)$. In this work, Gaussian kernel or RBF(radial basis function) is used as it tends to give good performance under general smoothing assumptions. The kernel is defined as

$K(x_1, x_2) = \exp(-\gamma \|x_1 - x_2\|^2)$. The kernel and regularized parameters (C, γ) with range $[2^{-5}, 2^5]$ are tuned by gridsearch technique to avoid overfiting problem.

By applying the above algorithm of SVM to train the data under study, training results are obtained. Table 3 illustrates the cross-validation error corresponding to the tuning parameters (C , γ). From the Table 3, we obtain the best hyperparameters C = 2^4 and $\gamma = 2^4$ (the Gaussian kernel function parameter) with the smallest error of ten fold cross-validation = 0.000006. Considering the in-sample data, the hit rate is 1.000 and error rate is 0.000 but for the out-of-sample, the hit rate is 0.860 and the error rate is 0.140.

3.10 Least square support vector machine for classification

LSSVM is a new version of SVM modified by Suykens et al (1999). LSSVM uses least square loss function to obtain a set of linear equations in dual space so that learning rate is faster and the complexity of calculation in convex programming (in SVM) is relaxed. In addition, the LSSVM avoids the drawback faced by SVM such as trade-off parameters (C, σ^2, ε) selection; instead the LSSVM requires only two hyper-parameters (γ, σ^2) to train the model. According to Suykens et al (2001), the equality constraints of LSSVM can act as recurrent neural network and nonlinear optimal control. Due to these nice properties, LSSVM has been successfully applied for classification and regression problems, including time series forecasting. Further application can be found in Van Gestel et al (2004) for detailed discussion on classification performance of LSSVM and Van Gestel et al (2001) and Ye et al (2004) for predictive capability of LSSVM in chaotic time series prediction.

Formally, given a training set $(x_i, y_i), i = 1,2,\ldots,N$ with input data $x_i \in R^n$ and corresponding binary class label $y_i \in \{-1, 1\}$,

LSSVM formulation is represented as follow:

$\min_{w,b,e} J(w,e) = \frac{1}{2} w^T w + \gamma \frac{1}{2} \sum_{i=1}^{N} e_i^2$ subject to the equality constraints $y_i[w^T \phi(x_i) + b] = 1 - e_i$, $i = 1, \ldots, N$. This formulation consists of equality instead of inequality constraints and takes into account a squared error with regularization term similar to ridge regression.

The solution is obtained after constructing the Lagrangian:

$$L(w,b,e\,;\alpha) = J(w,b,e) - \sum_{i=1}^{N} \alpha_i \{ y_i[w^T \phi(x_i) + b] - 1 + e_i \} ,$$

where α_i are Lagrange multipliers that can be positive or negative in the LSSVM formulation. From the conditions for optimality, one obtains the Karush-Kuhn-Tucker (KKT)

$$\begin{cases} \frac{\partial L}{\partial w} = 0 \to w = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i) \\ \frac{\partial L}{\partial b} = 0 \to \sum_{i=1}^{N} \alpha_i y_i = 0 \\ \frac{\partial L}{\partial e_i} = 0 \to \alpha_i = \gamma_i e_i \quad i = 1, \ldots, N \\ \frac{\partial L}{\partial \alpha_i} = 0 \to y_i[w^T \phi(x_i) + b] - 1 + e_i = 0\ , \ i = 1, \ldots, N. \end{cases}$$

By eliminating w and e ,

$$\left[\begin{array}{c|c} 0 & y^T \\ \hline y & \Omega + \gamma^{-1} I \end{array} \right] \left[\begin{array}{c} b \\ \hline \alpha \end{array} \right] = \left[\begin{array}{c} 0 \\ \hline 1_v \end{array} \right]$$

with $y = [y_1, \ldots, y_N]$, $1_v = [1, \ldots, 1]$ $e = [e_1, \ldots, e_N]$, $\alpha = [\alpha_1, \ldots, \alpha_N]$.

Matrix $\Omega_{ij} = y_i y_j \phi(x_i)^T \phi(x_j) = y_i y_j K(x_i, x_j)$ for $i, j = 1, \ldots, N$ satisfies Mercer's condition and the LS-SVM model for estimating classifier is obtained as

$$y(x) = sign\Big(\sum_{i=1}^{N} \alpha_i y_i K(x, x_i) + b \Big).$$

In this work, Gaussian kernel or RBF(radial basis function) is used as it tends to give good performance under general smoothing assumptions. The kernel is defined as $K(x_1, x_2) = \exp(-\frac{1}{\sigma^2} \|x_1 - x_2\|^2)$. The kernel and regularized parameters (γ, σ^2) are tuned by gridsearch technique to avoid overfiting problem.

The experimental results show that the obtained hyper-parameters: gamma is 0.50691 chosen from the range [0.04978707, 148.4132] and sigma square is 8.7544 selected from the range [0.082085, 12.1825]. The cost of ten fold cross-validation is 0.033291. from the forecasting performance, the in-sample data hit rate is 0.8528 and error rate is 0.1472. For the out-of-sample data, the hit rate is 0.8640 and error rate is 0.1360.

To sum it up, Table 4 below gives a summary result of prediction performances by the ten different approaches. From the Table 4, we can see that almost all the algorithms generate high hit rate (more than 80%) and low error rate (less than 20%). By comparison, LSSVM ranks first one as it outperforms the other model though it is not better than SVM for in-sample prediction. The superior performances of the SVM models in this study also support the results in the literature. Bayesian classification for Gaussian process produces good prediction as neural network, following the SVM and LS-SVM. K-nearest neighbor approach gives the worst predictive ability and then Tree classification is the next one.

4. Conclusion

In this paper, we apply ten different techniques of data mining to forecast movement direction of Hang Seng index from Hong Kong stock market. All algorithms produce good prediction with hit rate more than 80%. The LS-SVM and SVM outperform the other models since theoretically they don't require any priori assumption on data property and their algorithms guarantee to efficiently obtain global optimal solution which is unique. The other models may be reliable for other markets, especially when the data fall into each of their properties. As can be seen in the figure 1-3 in appendix section, different stock prices behave differently. Therefore, all the approaches are recommended for forecasters of stock index movement and the better models, SVM and LS-SVM, are more preferred.

References

Ben-David, S., & Lindenbaum, M. (1997). Learning distributions by their density levels: A paradigm for learning without a teacher, *Journal of Computer and System Sciences* 55, 171-182.

Bishop, C. (1995). *Neural Networks for Pattern Recognition,* Clarendon Press, Oxford.

Burges, C.J.C., & Schokopf, B. (1997). Improving the Accuracy and Speed of Support Vector Machines, Advances in Neural Information Processing Systems, *MIT Press*, Cambridge, MA. pp. 475-481.

Cao, L.J., & Tay, F. (2001). Application of support vector machines in financial time series forecasting, *International Journal of Management Science*, pp. 309-317.

Chen, An-Sing, Daouk, Hazem & Leung, Mark T. Application of Neural Networks to an Emerging Financial Market: Forecasting and Trading the Taiwan Stock Index. [Online] Available: http://ssrn.com/abstract=237038 or DOI: 10.2139/ssrn.237038 (July 2001)

Cortes, C., & Vapnik, V.N. (1995). Support Vector Networks, *Machine Learning* 20, 273-297.

Deng, Min. Pattern and Forecast of Movement in Stock Price. [Online] Available: http://ssrn.com/abstract =217048 or DOI: 10.2139/ssrn.217048 (November 8, 2006)

Duda, R., Hart, P., & Stork, D. (2000). *Pattern Classification* (Second Edition), Wiley, New York.

McLachlan, G. J. (1992), *Discriminant Analysis and Statistical Pattern Recognition*, Wiley, New York.

Friedman, J., Hastie, T., & Tibshirani, R. (2008). *The Elements of Statistical Learning:Data Mining, Inference and Prediction*, Second Edition Springer.

Hastie, T. (1996). Neural Network, *Encyclopedia of Biostatistics*, John Wiley.

Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*, Macmillan, New York.

Huang, W., Nakamori, Y. & Wang, S.Y. (2005). Forecasting stock market movement direction with support vector machine, *Journal of Computers & Operational Research,* pp. 2513-2522.

Joachims, T. (2002). Learning to Classify Text using Support Vector Machines, *Kluwer Academic Publshers*, London.

Karatzoglou, A., Mayer, D., & Hornik, A. (2006). Support Vector Machines in R, *Journal of Statistical Software.*

Karatzoglou, A., Smola, A., Hornik, A., & Zeileis, A.. (2004). "An S4 Package for Kernel Methods in R", *Journal of Statistical Software*.

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

Kumar, Manish and Thenmozhi, M. Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest, Indian Institute of Capital Markets 9th Capital Markets Conference Paper. [Online] Available: http://ssrn.com/abstract=876544(February 06, 2006)

Leung, Mark T., Daouk, Hazem, & Chen, An-Sing. Forecasting Stock Indices: A Comparison of Classification and Level Estimation Models. [Online] Available: http://ssrn.com/abstract=200429 (March 1999)

Lo, A.W., & MacKinlay, A.C. (1988). Stock market prices do not follow random walks: Evidence from a simple specification test, *Review of Financial Studies* 1, 41-66.

Mitchell, T. M. (2007). *Machine learning*, McGraw Hill, USA.

Mukherjee, S., Osuna, E., & Girosi, F. (1997). Nonlinear prediction of chaotic time series using support vector machine. *In: Proceedings of the IEEE workshop on Neural Networks for Signal Processing*, Amelia Island, FL., pp. 511-520.

O'Connor, M., Remus, W., & Griggs, K. (1997). Going up-going down: How good are people at forecasting trends and changes in trends? *Journal of Forecasting* 16, 165-176.

Osuna, E.E, Freund, R., & Girosi, F. (1997). Support Vector Machines: Training and Applications, *MIT press*, USA.

Pearson, R.A.(2004). Tree structures for predicting stock price behaviour, *ANZIAM J.*, Austral. Mathematical Soc. 45, pp C950-C963

Quinlan, J.R. (1986). Induction of Decision Trees, *Journal of Machine Learning*, 1:81-106, Kluwer Academic Publishers.

Rasmussen, C. E. & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*, the MIT Press, ISBN 026218253X.

Ripley, B.D. (1996). *Pattern recognition and neural networks,* Cambridge University Press.

Stoneking, D. (1999). Improving the manufacturability of electronic designs, *IEEE Spectrum* 36, 70-76.

Suykens, J.A.K, & Vandewalle, J., (1999). Least squares support vector machine classifiers. *Neural Processing Letters* (9) 293-300.

Suykens, J.A.K., Vandewalle, J., & De Moor, B. (2001). Optimal control by least squares support vector machines. *Neural Networks* (14) 23-35.

Tarassenko, L., Hayton, P., Cerneaz, N., & Brady, M. (1995). Novelty detection for the identification of masses in mammograms. *In: Proceedings fourth IEE International Conference on Artificial Neural Networks*, Cambridge, pp. 442-447.

Van Gestel, T., Suykens, J.A.K, Baesens, B., Viaene, S., Vanthienen, J., Dedene, G., Moor B.D., & Vandewalle, G. (2004). Benchmarking Least Squares Support Vector Machine Classifiers, *Machine Learning*, (54) 5-32.

Van Gestel, T.V., Suykens, J.A.K, Baestaens, D.E., Lambrechts, A., Lanckriet G., Vandaele B., De Moor B., & Vandewalle, J.(2001). Financial Time Series Prediction using Least Squares Support Vector Machines within the Evidence Framework, *IEEE Transactions on Neural Networks,* (12) 809-821.

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

Williams, C. K. I. & Barber, D. (1999). Bayesian classification with Gaussian Processes, *Pattern Analysis and Machine Intelligence, 20(12) 1342-1351*, IEEE.

Wu, Y., & Zhang, H. (1997). Forward premiums as unbiased predictors of future currency depreciation: A non-parametric analysis", *Journal of International Money and Finance* 16, 609-623.

Wu, X.D., Kumar V., Quinlan J.R., Ghosh J., Yang Q., Motoda, H., McLachlan G.J., Ng A., Liu B., Yu P.S., Zhou Z.H., Steinbach, M., Hand, D.J., & Steinberg, D. (2008). Top 10 algorithms in data mining, *Knowledge Information System*, Springer-Verlag London.

Ye, M.Y., & Wang, X.D. (2004). Chaotic time series prediction using least squares support vector machine, *J. Chinese Physics*, IOP Publishing Ltd. IP address: 130.120.82.1000.

Appendices:

Table 1. Selection of k by cross-validation technique

k = 1	2	3	4	5	6	7	8	9	10
0.2302	0.2275	0.1964	0.2032	0.1897	0.1917	0.1789	0.1796	0.1735	**0.1708**
k =11	12	13	14	15	16	17	18	19	20
0.1729	0.1729	0.1742	0.1735	0.1769	0.1816	0.1762	0.1823	0.1769	0.1789
k = 21	22	23	24	25	26	27	28	29	30
0.1749	0.1776	0.1729	0.1769	0.1729	0.1769	0.1776	0.1762	0.1749	0.1749

The table displays the mean square error and corresponding value of k. The optimal k is 10 with smallest MSE 0.1708.

Table 2. Process of pruning tree

	CP	n.split	rel error	x_error	x_std
1	0.5387755	0	1.00000	1.01769	0.026178
2	0.0285714	1	0.46122	0.47211	0.022177
3	0.0149660	3	0.40408	0.43673	0.021573
4	0.0117914	5	0.37415	0.41905	0.021250
5	0.0088435	9	0.32653	0.42041	0.021275
6	0.0081633	11	0.30884	0.41905	0.021250
7	0.0068027	13	0.29252	0.41361	0.021148
8	0.0047619	15	0.27891	0.40816	0.021044
9	***0.0036281***	***17***	***0.26939***	***0.40272***	***0.020938***
10	0.0027211	20	0.25850	0.40408	0.020965
11	0.0020408	21	0.25578	0.40680	0.021017
12	0.0017007	25	0.24762	0.41497	0.021173
13	0.0013605	29	0.24082	0.41497	0.021173
14	0.0010000	36	0.23129	0.42857	0.021426

Denote variables actually used in tree construction by V1 = Open price, V2 = Low price, V3=High price, V4= S&p500, V5 = FX, V6 = Close price. The class “1” is when the next price is larger than the previous price, while the class “0” is refers to when the next price is smaller than the previous price.

We choose CP = 0.0036281 corresponding to 17 splits based on the smallest value of the x_error : 0.40272 to obtain the following pruned tree:

Pruned Tree

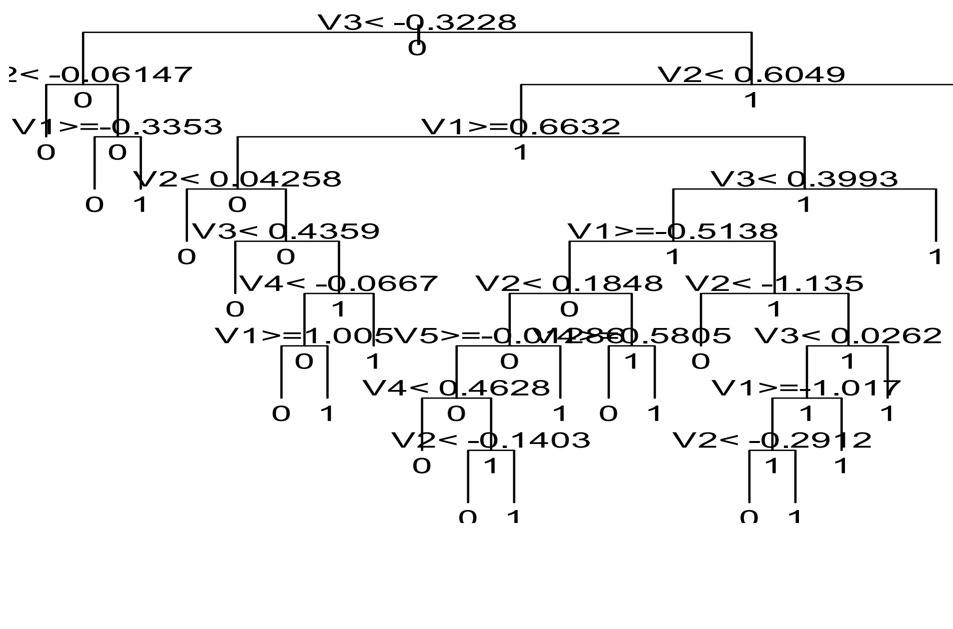


Table 3. Hyperparamters selection based on 10 fold cross-validation for SVM

C/γ	2^{-5}	2^{-4}	2^{-3}	2^{-2}	2^{-1}	2^{0}	2^{1}	2^{2}	2^{3}	2^{4}	2^{5}
2^{-5}	0.1679	0.1596	0.1532	0.1489	0.1455	0.1487	0.1619	0.2479	0.3967	0.4647	0.484945
2^{-4}	0.1576	0.1511	0.1449	0.1424	0.1369	0.1358	0.1426	0.1643	0.2962	0.4112	0.449012
2^{-3}	0.1498	0.1441	0.1400	0.1353	0.1304	0.1246	0.1247	0.1345	0.1699	0.3159	0.381449
2^{-2}	0.1439	0.1391	0.1347	0.1305	0.1233	0.1150	0.1084	0.1073	0.1134	0.1723	0.263409
2^{-1}	0.1405	0.1363	0.1329	0.1269	0.1183	0.1069	0.0930	0.0806	0.0676	0.0618	0.095686
2^{0}	0.1384	0.1341	0.1306	0.1241	0.1141	0.1006	0.0830	0.0626	0.0373	0.0152	0.004014
2^{1}	0.1364	0.1332	0.1285	0.1200	0.1110	0.0926	0.0746	0.0519	0.0238	0.0061	0.000574
2^{2}	0.1351	0.1314	0.1271	0.1170	0.1066	0.0874	0.0680	0.0409	0.0143	0.0018	0.000006
2^{3}	0.1339	0.1296	0.1252	0.1152	0.1007	0.0823	0.0623	0.0306	0.0082	0.00009	0.000006
2^{4}	0.1330	0.1283	0.1226	0.1136	0.0951	0.0776	0.0536	0.0210	0.0030	**0.000006**	0.000006
2^{5}	0.1311	0.1279	0.1206	0.1105	0.0896	0.0719	0.0443	0.0162	0.0004	0.000006	0.000006

The table illustrates the cross-validation error corresponding to the tuning parameters (C , γ). The training result shows that (C , γ)=(2^4, 2^4) which corresponds to the smallest training error = 0.000 006.

Table 4. Summary result of prediction performances by ten different approaches

	In-sample		Out-of-sample		
Predictors	Hit Rate	Error Rate	Hit Rate	Error Rate	Rank
LDA	0.8393	0.1607	0.8440	0.1560	6
QDA	0.8305	0.1695	0.8480	0.1520	5
K-NN	0.8312	0.1688	0.7960	0.2040	9
Naïve Bayes	0.8386	0.1614	0.8280	0.1720	7
Logit model	0.8474	0.1526	0.8560	0.1440	3
Tree classification	0.8717	0.1283	0.8000	0.2000	8
Neural network	0.8481	0.1519	0.8520	0.1480	4
Gaussian process	0.8595	0.1405	0.8520	0.1480	4
SVM	1.0000	0.0000	0.8600	0.1400	2
LS-SVM	0.8528	0.1472	0.8640	0.1360	1

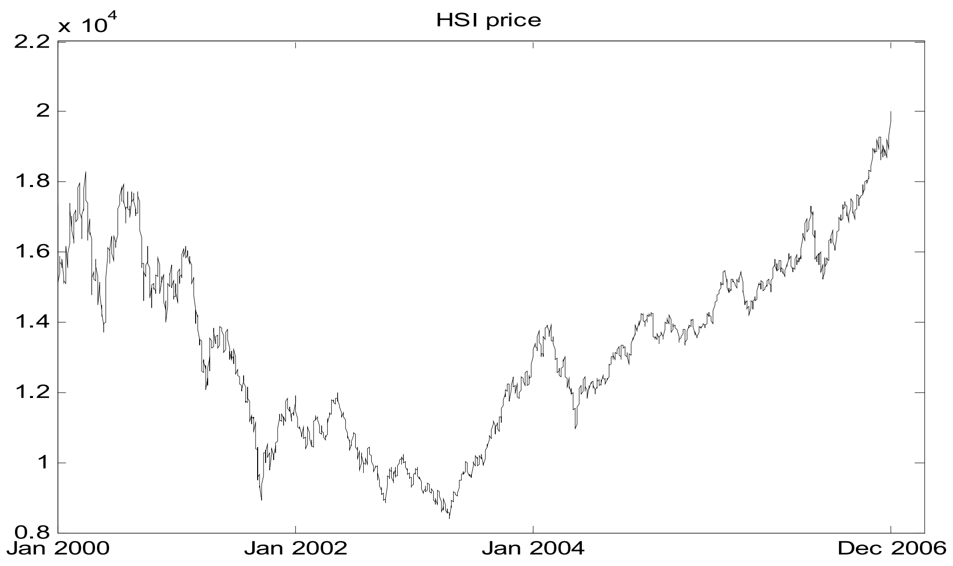
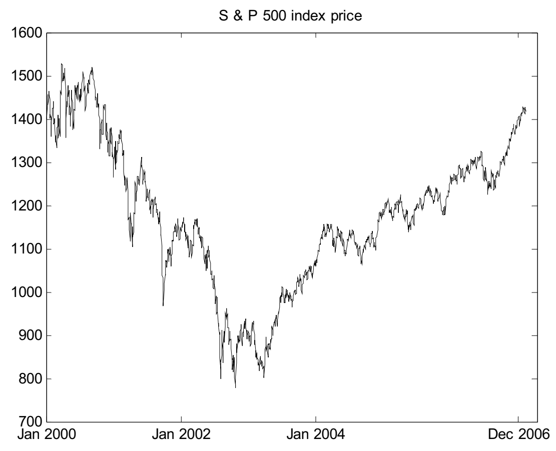


Figure 1. Daily closing prices of Hang Seng Index



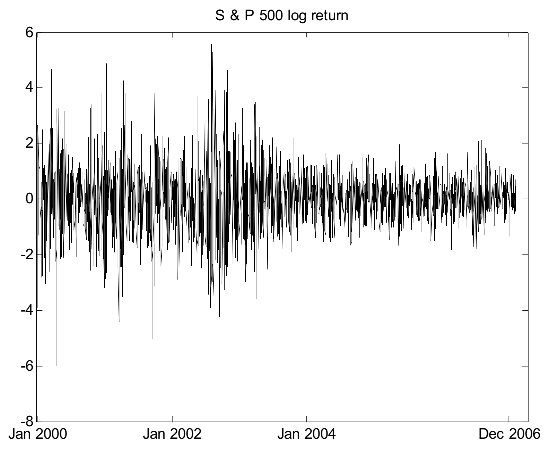
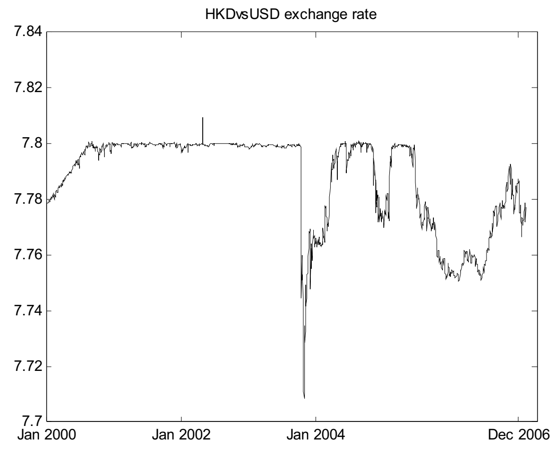


Figure 2. Daily closing prices (left) and Log return (right) of S & P 500 index



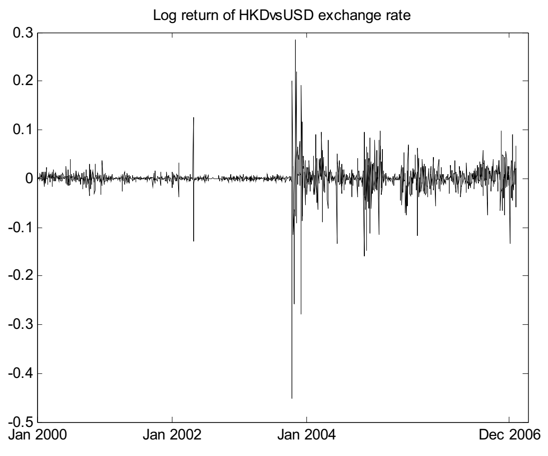


Figure 3. Daily prices (left) and Log return (right) of currency exchange rate between HKD and USD