Stock Market Prediction with ARIMA and SVMs

Peiran Hu

March 16, 2015

Abstract

The main purpose of this paper is to compare ARIMA model and SVMs model in stock return prediction, and to generate a new model combining ARIMA and SVMs models. Microsoft, Citigroup and IBM, daily stock prices and return data are used as financial forecast application. Hit rate and normalized mean square error are used as metrics to measure prediction accuracy. The result is that SVMs model outperforms ARIMA model, since it has higher hit rate and lower normalized mean square error. However, they are not significantly different. Both SVMs model and ARIMA model don't have excellent performances in prediction. The hybrid ARIMA and SVMs model gives a better prediction result, because it can not only capture the linear trend, but also the non-linear trend of stock return. I also added an option in the hybrid model, choosing act or not, in order to improve the prediction accuracy.

Catalogeu

Key words

Stock market return prediction ARIMA SVMs

1 Introduction

Forecasting stock prices is one of the most challenging applications of modern time series forecasting, and a number of models have been developed to offer investors more precise predictions. Especially, several techniques are widely used in stock market prediction, including artificial neural networks, linear regression, multi-linear regression, genetic algorithms, and support vector machines. Here we used three models to predict stock return, ARIMA model, SVMs model, and a hybrid ARIMA and SVMs model.

ARIMA is short for autoregressive integrated moving average, and is a generalization of ARMA (autoregressive moving average) model. ARIMA is often used in time series forecasting, and it has become a pretty mature application. Support vector machine (SVM) developed by Vapnik is based on statistical learning theory, and is a branch of machine learning. SVMs are supervised learning models with associated learning algorithms, and they can help to analyze data and recognize patterns. SVMs can be used in classification and regression analysis.

ARIMA is a linear model, while SVMs with a kernel is a non-linear model. In theory, the hybrid ARIMA and SVMs model has both linear part and non-linear part, and in this case, it has features of both linear and non-linear models, which will lead to a more accurate prediction.

2 Literature Review

According to the statistics done by George S. Atsalakis and Kimon P. Valavanis (2009), the specific methods used to develop forecasting models are listed in Table 1.

Table 1: Comparative studies done by George S. Atsalakis and Kimon P. Valavanis (2009), classifying techniques used to predict subject stock markets. Statistics on seven models, ANNs, B&H, LR, ARIMA, RW, SVM and GA have carefully been made.

Article	ANNs	B&H	LR	ARIMA	RW	SVM	GA	Others
Andreou et al. (2000)	•							•
Armano et al. (2004)	•	•						
Atsalakis and Valavanis (2006a)		•						
Atsalakis and Valavanis (2006b)		•						
Baek and Cho (2002)		•						
Barnes et al. (2000)			•	•				•
Bautista (2001)					•			
Brownstone (1996)			•					

Cao et al. (2005)			•					•
Casas (2001)		•						
Chandra and Reeb (1999)		•						
Chaturvedi and Chandra (2004)			•					
Chen et al. (2003)		•			•			
Chen et al. (2005a, 2005b)	•							
Chenoweth and Obradovic (1996)	•	•						
Chun and Park (2005)					•			
Doesken et al. (2005)		•						
Donaldson and Kamstra (1999)				•				•
Dong et al. (2003)			•					
Dourra and Siy (2002)								•
Egeli et al. (2003)	•							
Fernandez-Rodriguez et al. (2000)		•			•			
Haiqin Y. et al. (2002)						•		
Harvey et al. (2000)		•	•					
Huang et al. (2005)	•					•		
Hui et al. (2000)	•							
Kanas and Yannopoulos (2001)			•					
Kim (1998)	•							
Kim and Han (1998)		•						•
Kimoto et al. (1990)			•					
Kosaka et al. (1991)								•
Koulouriotis (2004)								•
Koulouriotis et al. (2001)		•						
Koulouriotis et al. (2002)			•					
Koulouriotis et al. (2005)	•		•					•
Lam (2001)		•						
Leigh et al. (2002)		•						
Mizuno et al. (1998)	•							•
Motiwalla and Wahab (2000)		•	•					
Nishina and Hagiwara (1997)	•							
Oh and Kim (2002)	•	•						
Olson and Mossman (2003)	•							
Pai and Lin (2005)				•		•		
Pan et al. (2005)								•
Pantazopoulos et al. (1998)								•
Phua et al. (2001)			•					
Qi (1999)			•					
Quah and Srinivasan (1999)								•
Raposo et al. (2002)	•							•
Rast (1999)	•							
Rech (2002)				•				•
Refenes et al. (1993)			•					

Schumann and Lohrbach (1993)				•				
Setnes and Van Drempt (1999)		•	•					
Siekmann et al. (1999)		•	•		•			•
Thammano (1999)	•							
Thawornwong and Enke (2004)	•	•	•					•
Theodore B. Trafalis et al.						•		
Tristan Fletcher et al.	•					•		
Tsaih et al. (1998)		•						
Vanstone et al. (2005)		•						
Wah and Qian (2002)	•			•				•
Walczak (1999)			•					
Wang and Leu (1996)			•				•	•
Wittkemper and Steiner (1996)			•				•	•
Wu et al. (2001)	•							
Yumlu et al. (2004)	•							•
Yumlu et al. (2005)	•							
Zhang et al. (2002)	•							
Zhang et al. (2004)		•						
Zorin and Borisov (2002)	•			•				•

From table above, seven models are mentioned in stock market prediction, including ANNs (24 times), B&H (23 times), LR (20 times), ARIMA (7 times), RW (5 times), and SVMs (5 times), GA (twice). ANNs, LR, ARIMA, GA, RW, B&H, and SVMs are short for artificial neural networks, linear regression, autoregressive integrated moving average, genetic algorithms, random walk, buy and hold, and support vector machines, respectively.

Artificial neural networks (ANNs) are widely used in stock market prediction. And during these years, it has become one of the mature branches of machine learning. From statistics above, ANNs are the hottest field in stock market prediction research. ARIMA model is a widely used model in time series forecasting. According to papers above, ARIMA models are usually used to compare with other models, such as ANNs, SVMs, and are demonstrated underperforming other models. As for support vector machines models, the application of support vector machines in financial market prediction is a very new topic, however it is fast developing these years, and is becoming more and more mature.

Atiya A, Talaat N, and Shaheen S (1997) developed a method to forecast the stock market. The paper used novel aspects, which means the forecasts were based on fundamental company information, such as PE ratio, PB ratio, sales, profit margin, etc. And they found that the earnings related indicators were the prime propellers of a stock price.

Brownstone D (1996) used neural network predictions to obtain daily Market close 5 and 25 days ahead with sample selected from the Financial Times – Stock Exchange 100 Share Index, known as "The Footsie", and found no significant difference when measured by mean square error, root mean square error and percentage accuracy. When comparing with Multiple Linear Regression, Brownstone found that neural networks had advantages in the prediction of stock price moving

direction.

Cao Q, Leggio K B, Schniederjans M J (2005) aimed to compare univariate and multivariate neural network models with linear models in stock return prediction. They used stock price information of firms traded on the Shanghai Stock Exchange, and got a conclusion that the neural network models outperformed the linear models.

White H. used simple neural networks to predict IBM daily stock returns, and was disappointed by the failure of the simple neural networks model, leading to a conclusion that finding evidence against efficient markets with such simple networks would not be easy. However on the positive side, such simple neural networks could still capture extremely rich dynamic behavior.

Chen A S, Leung M T, and Daouk H predicted the moving direction of Taiwan Stock Exchange return using three strategies, probabilistic neural network (PNN), generalized methods of moments (GMM) with Kalman filter, and buy and hold (B&H). Better prediction on moving direction will lead to higher profits. Finally, empirical results showed that the PNN-based investment strategies obtained higher profits than other strategies used in this study.

Birgul E, Meltem O, and Bertan B aimed to predict Istanbul Stock Exchange market index with the application of neural networks. They also compared neural networks with moving average, and concluded that neural networks outperformed moving average.

Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang (2004) used support vector machines to forecast the weekly movement direction of NIKKEI 225 index. Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang also compared its performance with those of Linear Discrimination Analysis, Quadratic Discriminant Analysis and Elman Backpropagation Neural Networks. And the final result showed that SVMs overwhelmed other methods.

HaiqinYang, Laiwan Chan, and Irwin King (2002) used support vector regression machines to predict Hang Seng Index. Through comparing different margins of SVR, HaiqinYang, Laiwan Chan, and Irwin King gave out the conclusion that the application of standard deviation to calculate a variable margin gives a good forecasting result in the prediction of Hang Seng Index.

Theodore B. Trafalis and Huseyin Ince (2000) used daily stock prices data of IBM, Yahoo, and America Online as financial forecast application. The paper compared SVMs with other two methods, Backpropagation and RBF networks, and said that the result of SVMs was quite promising.

Kyoung-jae Kim (2003) compared SVM with back-propagation neural networks and case-based reasoning. The result showed that SVM offered a promising alternative to stock market prediction.

R. Glen Donaldson and Mark Kamstra used S&P 500 stock index to demonstrate that ANNs outperforms traditional forecasting models, such as GARCH model.

However, different viewpoints existed. Rech, Gianluigi (2002) found that the linear models outperforms the ANNs, and the statistical approach stands up well in comparison to other more sophisticated ANN models. Prof. Dr. Matthias Schumann, Dip1.-Kfm. Thomas Lohrbach (1993) couldn't give out a conclusion which method (ANN or ARIMA) was better.

Ping-Feng Pai and Chih-Sheng Lin (2004) used a hybrid ARIMA and support vector machines model to forecast stock prices. Ping-Feng Pai and Chih-Sheng Lin said that though ARIMA is

widely used, it cannot easily capture the nonlinear patterns. SVMs have been successfully applied in solving nonlinear regression estimation problems. Thus, the combination of the two methods could give out a more accurate prediction.

3 ARIMA

3.1 Introduction to ARIMA Model

For more than half a century, ARIMA has become one of the most popular models in time series analysis and prediction. It has been proved that ARIMA has wonderful predictive effect in many cases. However it also has disadvantages, which can be called Slutzsky's theorem, meaning that we can transform random noise into any time series we like by using ARIMA-kind computation, or by adding trend lines.

When applying ARIMA, we usually have several fixed steps. At the very beginning, there are three steps to build an ARIMA model, and then the original three-stage process of model selection, parameter estimation and model checking is developed into a five-stage process by adding a preliminary stage of data processing and a final stage of model application (or forecasting).

More specifically, the five stages can be described as follows.

The first stage, data preparation. Data preparation involves transformations and differencing. Transformations include square roots and logarithms, which can help stabilize the variance in a series. Differencing means taking the difference between consecutive observations, which can help eliminate trend or seasonality in the data.

The second stage, model selection. Number of lags are decided in this stage, mainly using some information criteria, such as AIC and BIC. The third stage, parameter estimation. It means finding the values of model coefficients.

The fourth stage, model checking. It means testing the assumptions of the model to identify whether any area of the model is inappropriate. If the model is found to be not appropriate, it is necessary to go back to stage two and select another more appropriate model.

The final stage, forecasting. And this is the ultimate purpose of developing a model[①]

① Valenzuela O, Rojas I, Rojas F, et al. Hybridization of intelligent techniques and ARIMA models for time series prediction[J]. Fuzzy Sets and Systems, 2008, 159(7): 821-845.

Autoregressive moving average (ARMA) model is a combination of autoregressive and moving average.

If ε_{t-i} (i $=$ 1, 2, 3, 4, … , q) is white noise, MA(q) can be defined as follows,

$$Y_t = \mu + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \theta_3\varepsilon_{t-3} + \cdots + \theta_q\varepsilon_{t-q}$$

where μ and θ_i(i $=$ 1, 2, 3, 4, … , q) are all constants.

If ε_t is white noise, AR(p) can be defined as follows,

$$Y_t = \mathrm{c} + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \varphi_3 Y_{t-3} + \cdots + \varphi_p Y_{t-p} + \varepsilon_t$$

Hence, we can get the expression of ARMA(p, q) as follows,

$$Y_t = \mathrm{c} + \varphi_1 Y_{t-1} + \varphi_2 Y_{t-2} + \varphi_3 Y_{t-3} + \cdots + \varphi_p Y_{t-p} + \varepsilon_t + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \theta_3\varepsilon_{t-3} + \cdots + \theta_q\varepsilon_{t-q}$$

which can also be written as,

$$(1 - \varphi_1 L - \varphi_2 L^2 - \cdots - \varphi_p L^p) Y_t = \mathrm{c} + (1 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q)\varepsilon_t$$

$$Y_t = \frac{\mathrm{c}}{(1 - \varphi_1 L - \varphi_2 L^2 - \cdots - \varphi_p L^p)} + \frac{(1 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q)}{(1 - \varphi_1 L - \varphi_2 L^2 - \cdots - \varphi_p L^p)}\varepsilon_t$$

$$Y_t = \mu + \Psi(\mathrm{L})\,\varepsilon_t$$

where

$$\mu = \frac{\mathrm{c}}{(1 - \varphi_1 L - \varphi_2 L^2 - \cdots - \varphi_p L^p)}$$

$$\Psi(\mathrm{L}) = \frac{(1 + \theta_1 L + \theta_2 L^2 + \cdots + \theta_q L^q)}{(1 - \varphi_1 L - \varphi_2 L^2 - \cdots - \varphi_p L^p)}$$

$$\sum_{j=0}^{\infty} |\Psi_j| < \infty$$

According to Box-Jenkins methodology, ARMA model often uses two statistical functions for the graphical identification, the first one is autocorrelation function (ACF) and the other one is partial autocorrelation function (PACF). We have

$$\mathrm{E}(Y_1) = \mathrm{E}(Y_2) = \cdots = \mathrm{E}(Y_t) = \mu$$

$$\mathrm{Var}(Y_1) = Var(Y_2) = \cdots = \mathrm{Var}(Y_t) = \sigma^2$$

$$\mathrm{Cov}(Y_t, Y_{t-k}) = \mathrm{Cov}(Y_{t+l}, Y_{t-k+l}) = \gamma_k$$

$\mathrm{Cov}(Y_t, Y_{t-k}) = \gamma_k$ are called autocovariances, and it is independent on t, but dependent on the lag k. And we can easily derive the function of autocorrelation

$$\rho_k = \frac{\mathrm{Cov}(Y_t, Y_{t-k})}{\sqrt{\mathrm{Var}(Y_t)\mathrm{Var}(Y_{t-k})}} = \frac{E[(Y_t - \mu)(Y_{t-k} - \mu)]}{{\sigma_y}^k} = \frac{\gamma_k}{\gamma_0} \approx \frac{\sum_{t=1}^{n-k}(Y_t - \mu)(Y_{t-k} - \mu)}{\sum_{t=1}^{n-k}(Y_t - \mu)^2}$$

Hence, ACF $= \rho_k$ can be obtained from the above equation. And we can get PACF $= \Phi_{ik}(i =$ 1,2,3,4, …) from the equation below,

$$w_t = (Y_t - \mu) = \Phi_{1k} w_{t-1} + \Phi_{2k} w_{t-2} + \cdots + \Phi_{kk} w_{t-k} + \varepsilon_t$$

3.2 Experiment Design

3.2.1 Metrics

Two metrics are used to measure prediction accuracy. The first one is normalized mean square error (nMSE), defined as follows:

$$\text{nMSE} = \frac{1}{\sigma^2 N} \sum_{t=t_0}^{t_1} (A(t) - E(t))^2$$

where σ^2 is the variance of the true time series in period $[t_0, t_1]$, N is the number of patterns tested, and $A(t)$ and $E(t)$ are, respectively, the actual and expected stock prices at time t.

The second metric is the hit rate. Suppose, $\mathrm{D(t+h)} = \mathrm{sign(S(t+h) - S(t))}$ is the actual direction of change for S(t), and $\widehat{D}(t+h) = \mathrm{sign}(\hat{\mathrm{S}}(\mathrm{t+h}) - \hat{\mathrm{S}}(\mathrm{t}))$ is the predicted direction change. And if $\mathrm{D(t+h)} * \widehat{D}(t+h) > 0$, we call it a hit. Thus we define the hit rate as follows:

$$\mathrm{H(t)} = \frac{|\{t|\mathrm{D(t+h)} * \widehat{D}(t+h) > 0, t = 1, 2, \dots, n\}|}{|\{t|\mathrm{D(t+h)} * \widehat{D}(t+h) \neq 0, t = 1, 2, \dots, n\}|}$$

where |Φ| represents the number of elements in set E.

The hit rate is very useful when a trading decision is based solely on whether the predicted future price goes up or down as compared to the current prices (Saad et al. 1998). Other metrics may be used according to different trading strategies.

3.2.2 Data

Using closing prices of Microsoft (symbol MSFT), IBM (symbol IBM) and Citigroup (symbol Citi) from March 30, 2009, to March 28, 2014, 1259 observations, we get rate of change (ROC), so that there are 1258 rates of change in all.

We use US stocks because US stock market is a relatively mature stock market. And we use stock prices of three companies, Microsoft, IBM, and Citigroup, because they are all major companies in their own industry, having a long stock trade history. The supply of their stocks and demand for their stocks are relatively large, leading to a stock price changing pattern with less noise. With a great number of stocks traded in stock market, the probability of controlling stock prices is small.

We can see rate of return of three stocks, Microsoft, Citigroup, IBM, and their autocorrelation functions, partial autocorrelation functions in Figure 1

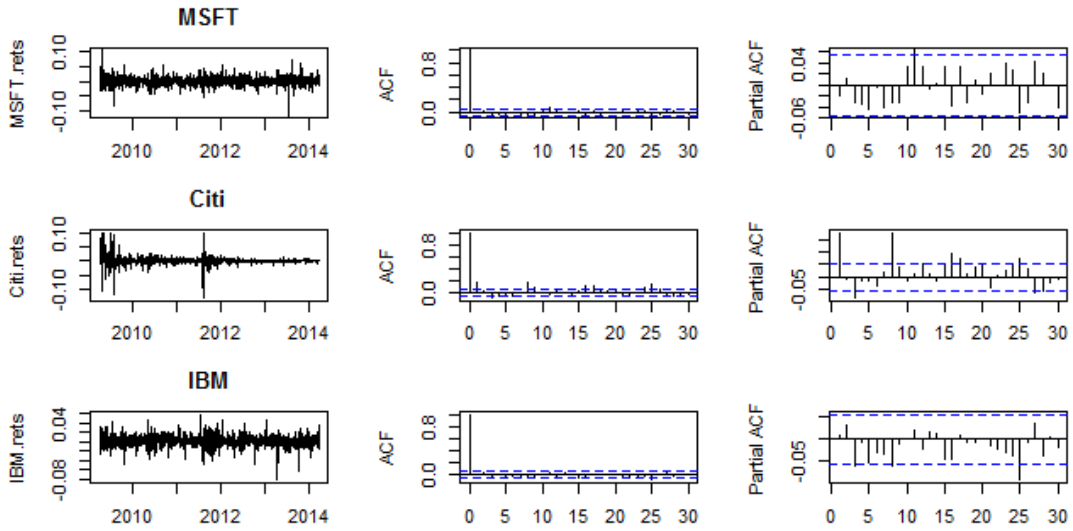

Figure 1: Rate of return of three stocks, Microsoft, Citigroup, IBM, and their autocorrelation functions, partial autocorrelation functions, using closing prices data of Microsoft, Citigroup and IBM from March 30, 2009, to March 28, 2014, 1259 observations in all.

3.2.3 Model

3.2.3.1 Data Preparation

Firstly, we should test whether the data we obtained is stationary. Augmented Dickey Fuller and Philips-Perron test are tests often used to test unit root. P value of Augmented Dickey Fuller Tests and Phillips–Perron Unit Root Test are all smaller than 0.01, meaning we should reject H_0: Not Stationary. The p value of ADF test and PP test are listed in Table 2.

Table 2: P value of Augmented Dickey Fuller (ADF) test, and Philips-Perron (PP) test, and results show the time series data, return of Microsoft, return of Citigroup, and return of IBM, we obtained are stationary

Variables	**P value**		**Conclusion**
	ADF Test	**PP Test**	
Return of Microsoft (MSFT.rets)	< 0.01	< 0.01	Stationary
Return of Citigroup (Citi.rets)	< 0.01	< 0.01	Stationary
Return of IBM (IBM.rets)	< 0.01	< 0.01	Stationary

3.2.3.2 Model Selection

Instead of the arbitrary way to select parameters for the $\mathrm{ARIMA}(p, d, q)$ model, we use AIC as a criterion. We select models with minimum AIC. As data shown above, the series is stationary, so we let d equal to zero.

We use iterated model selection method. That means every 30 days[②], we determine new parameters, p, d, q of the $\mathrm{ARIMA}(p, d, q)$ model based on the previous 500-day history.

3.2.3.3 Coefficient Estimation

With parameters selected above, we can estimate coefficients of our ARIMA model. We use iterated method for both parameter set (p, d, q) prediction and coefficient estimation. What we should pay attention to is that every time we finish a prediction and move on to another day's return prediction, we will change our training set, the history data, which are always the previous 500 days' return standing on the predicted day.

That means we will get hundreds of models in the prediction process, and we will also estimate model coefficients for hundreds of times.

3.2.3.4 Model Checking

As we mention in the last section, both of the methods require that we estimate coefficients for hundreds of times. It is almost impossible to check each model carefully, but I choose several models randomly[③] and test whether the residuals are white noise. And the test results are as follows:

For example, to predict the rate of day 661, we have known (p, d, q) = (2, 0, 2), and with rates from day 161 to day 660, we can get the ARIMA model.

$$\underset{}{\mathrm{MSFT}_t} = \underset{(0.0006)}{-0.0001} - \underset{(0.0371)}{1.3906}\mathrm{MSFT}_{t-1} - \underset{(0.0370)}{0.9311}\mathrm{MSFT}_{t-2} + \varepsilon_t + \underset{(0.0275)}{1.3974}\varepsilon_{t-1} + \underset{(0.0235)}{0.9753}\varepsilon_{t-2}$$

And then we analyze residuals. We depict residuals in Figure 2,

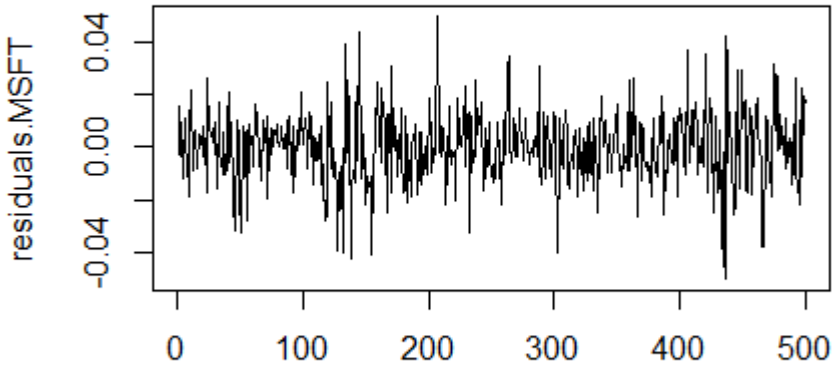

Figure 2: Residuals of the ARIMA function which predicts rate of return on day 661 using data from day 161 to day 660, with (p, d, q) = (2, 0, 2).

It might be hard to tell whether the residuals are normal distribution only based on the plot above.

② Here, 30 days is a variable, and can be altered.

③ I generated a series of uniform distribution random numbers as predicted days, and selected corresponding model to have a check.

So we do JB test, and the p value of the test is smaller than 0.0001, which means the distribution of residuals is significantly different from normal distribution.

In the same way, we have the following results in Table 3.

Table 3: Model checking results for ARIMA model of Microsoft, the eight days, day 661, day 734, ets, are selected from simple random sampling. None of the residuals are white noise, which means the ARIMA model is not appropriate in this case.

Day	(p, d, q)	prediction	real value	JB test (p value)	Conclusion
Day 661	(2, 0, 2)	0.004040	-0.035346	$4.122*10^{-7}$	Not a normal distribution
Day 734	(2, 0, 0)	0.000490	-0.004130	$5.983*10^{-10}$	Not a normal distribution
Day 786	(0, 0, 0)	0.000227	-0.000650	$4.09*10^{-13}$	Not a normal distribution
Day 807	(0, 0, 0)	0.000336	-0.025295	$5.718*10^{-14}$	Not a normal distribution
Day 824	(0, 0, 0)	0.000526	-0.001951	$6.507*10^{-13}$	Not a normal distribution
Day 832	(0, 0, 0)	0.000358	0.007473	$1.498*10^{-13}$	Not a normal distribution
Day 917	(0, 0, 0)	0.000172	-0.005251	$2.097*10^{-10}$	Not a normal distribution
Day 1061	(2, 0, 2)	0.005012	0.017442	$4.352*10^{-8}$	Not a normal distribution

From table above, I get a conclusion that models are not good, since none of the residuals follow normal distribution, meaning assumptions behind ARIMA model are invalid in this case. Since none of the models we randomly select are good, and none of the residuals from those models are white noise, we don't consider ARIMA model is a good application in stock market prediction here. And what's more, the model tends to select (0, 0, 0) as parameters, meaning the model hardly capture the time series information of data.

The model checking results of the other two companies, Citigroup and IBM, are in the following Table 4 and Table 5.

Table 4: Model checking results for ARIMA model of Citigroup, the eight days, day 661, day 734, ets, are selected from simple random sampling. None of the residuals are white noise, which means the ARIMA model is not appropriate in this case.

Day	(p, d, q)	prediction	real value	JB test (p value)	Conclusion
Day 661	(2, 0, 2)	-0.001070	-0.013549	$< 2.2*10^{-16}$	Not a normal distribution
Day 734	(3, 0, 3)	0.000124	0.002042	$< 2.2*10^{-16}$	Not a normal distribution
Day 786	(3, 0, 3)	0.001270	-0.000408	$< 2.2*10^{-16}$	Not a normal distribution
Day 807	(3, 0, 3)	-0.000508	0.004936	$< 2.2*10^{-16}$	Not a normal distribution
Day 824	(3, 0, 2)	0.001563	0.001204	$< 2.2*10^{-16}$	Not a normal distribution
Day 832	(3, 0, 2)	0.000924	-0.000400	$< 2.2*10^{-16}$	Not a normal distribution
Day 917	(3, 0, 1)	0.000095	0.004775	$< 2.2*10^{-16}$	Not a normal distribution
Day 1061	(3, 0, 1)	0.005855	0.001594	$< 2.2*10^{-16}$	Not a normal distribution

Table 5: Model checking results for ARIMA model of IBM, the eight days, day 661, day 734, ets, are selected from simple random sampling. None of the residuals are white noise, which means the ARIMA model is not appropriate in this case.

Day	(p, d, q)	prediction	real value	JB test (p value)	Conclusion

Day 661	(3, 0, 3)	0.001772	-0.026756	$< 2.2*10^{-16}$	Not a normal distribution
Day 734	(4, 0, 3)	0.001597	0.001163	$< 2.2*10^{-16}$	Not a normal distribution
Day 786	(3, 0, 3)	0.003837	0.003131	$< 2.2*10^{-16}$	Not a normal distribution
Day 807	(3, 0, 3)	0.004366	-0.013478	$< 2.2*10^{-16}$	Not a normal distribution
Day 824	(3, 0, 3)	0.000062	0.003266	$< 2.2*10^{-16}$	Not a normal distribution
Day 832	(3, 0, 3)	0.005816	-0.006169	$< 2.2*10^{-16}$	Not a normal distribution
Day 917	(3, 0, 4)	0.003097	0.005865	$< 2.2*10^{-16}$	Not a normal distribution
Day 1061	(1, 0, 3)	0.002336	0.004154	$< 2.2*10^{-16}$	Not a normal distribution

3.2.3.5 Forecasting

Though the model checking in previous section shows that ARIMA model is not a good application in this case, we still run the model and see the final prediction results.

With the ARIMA model trained, we can forecast stock price, and then compare the real stock price with the forecasted one. We can get "hit rate" and "nMSE" in each case. For stock Microsoft, the hit rate and nMSE are 0.494723 and 1.017788; for stock Citigroup, the hit rate and nMSE are 0.490765 and 1.093893; and for stock IBM, the hit rate and nMSE are 0.525066 and 1.026667.

We can also draw lines of forecast stock price and real stock price in Figure 3, Figure 4, and Figure 5 as follows.

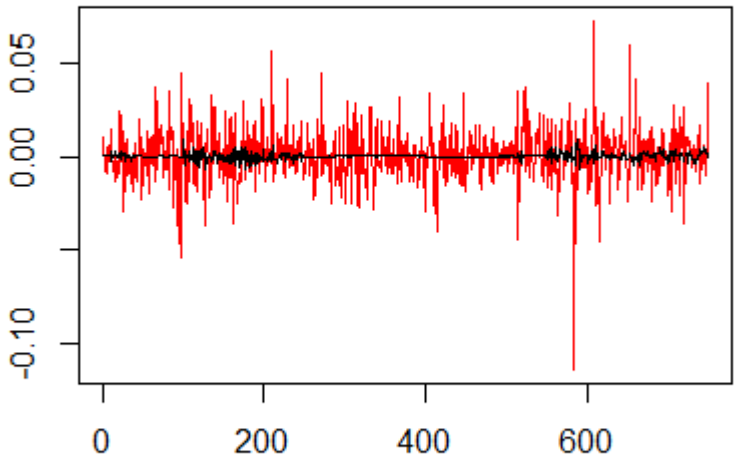

Figure 3: A line chart of Microsoft stock return, the red line is the real return and the black line is the return we predicted using our ARIMA model. And from statistics we do, the hit rate in this case is 0.494723, and the nMSE is 1.017788.

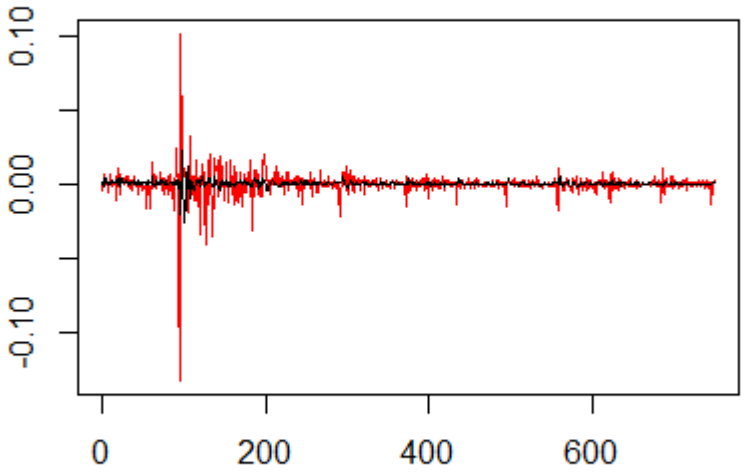

Figure 4: A line chart of Citigroup stock return, the red line is the real return and the black line is the return we predicted using our ARIMA model. And from statistics we do, the hit rate in this case is 0.490765, and the nMSE is 1.093893.

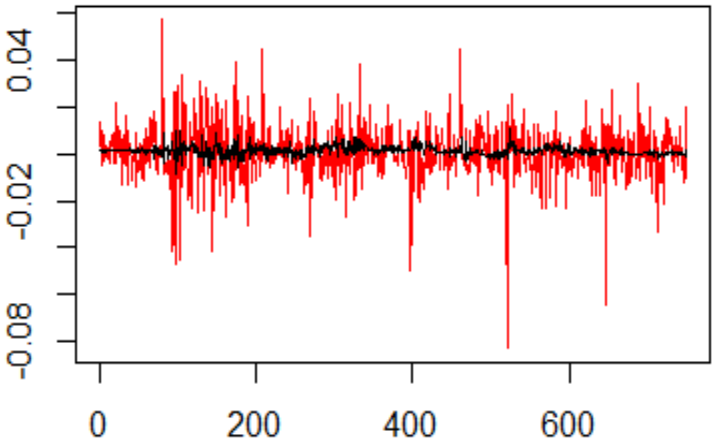

Figure 5: A line chart of IBM stock return, the red line is the real return and the black line is the return we predicted using our ARIMA model. And from statistics we do, the hit rate in this case is 0.525066, and the nMSE is 1.026667.

From three line charts above, we can see the trend of real return and predicted return. The predicted return tends to underestimate when return is positive, and to overestimate when return is negative. And what's more, the line chart of the predicted return of Microsoft even generates straight line for not a short period. Other two line charts of predicted return are also fairly flat.

We know from the result that the model tends to predict a wrong direction when the real value is negative, that is when the stock return is negative, the prediction often gives out a wrong stock moving direction, a positive one. It may cause great losses in real cases.

The number of return for each stock is 1258, and 500 rates are used as history data in the first regression. Hence we won't generate any prediction for those 500 rates. We will give out 758 rates of return and compare the forecast values with real ones. We use two metrics to evaluate the model, hit rate and normalized mean square error. Results can be seen in Table 6.

Table 6: Hit rate and nMSE of our return predictions on Microsoft stock, Citigroup stock, and IBM stock.

Stock	Hit Rate	nMSE
Microsoft (MSFT)	0.494723	1.017788
Citigroup (Citi)	0.490765	1.093893
IBM	0.525066	1.026667

If we allow a wider range of change of parameters we set in our programming, for example p and q can change from 0 to 8 instead of from 0 to 4, of course, results will change, but I think no big change can be made. Take Microsoft as an example, we change parameters of our model and see whether huge difference will occur. Results can be seen in Table 7.

Table 7: Hit rate and nMSE of our return predictions on Microsoft stock after changing parameters, such as p, d, q, length of history, and time span of parameter/model selection.

p	d	q	length of history	time span of parameter/ model selection (days)	Hit Rate	nMSE
(0, 4)	**0**	**(0, 4)**	**500**	**30**	**0.494723**	**1.017788**
(0, 4)	0	(0, 4)	500	20	0.477573	1.020596
(0, 4)	0	(0, 4)	500	60	0.490765	1.023264
(0, 8)	0	(0, 8)	500	30	0.480211	1.031616
(0, 4)	(0, 2)	(0, 4)	500	30	0.494723	1.017788
(0, 4)	0	(0, 4)	800	30	0.502183	1.008933

There is no need to try the same thing again on Citigroup and IBM. I can get a conclusion that ARIMA model is not an ideal one in this case of stock market prediction. And there are several reasons behind this result. Firstly, stock prices contain a lot of noises, which makes it hard to get precise predictions using past information. Secondly, too many factors drive the movement of the stock, and these factors are too complicated to be modeled, especially to be modeled by a single time series model.

4 Support Vector Machines for Forecasting Stock Movement Direction

4.1 Introduction to Support Vector Machines

The original SVMs algorithm was proposed by Vladimir N. Vapnik, and then was developed by Corinna Cortes and Vapnik in 1993. Support Vector Machine is an important part of Machine Leaning. SVM gives a clear and powerful way of solving complex small sampling size non-linear problems. SVMs model contains SVMs classifier and SVMs regression, which can help solve classification and regression problems.

More formally, support vector machines construct a hyper-plane or a set of hyper-planes in a

high-dimensional or infinite-dimensional space, and they can usually be used for classification, regression or other tasks. Intuitively, a good separation is created by the hyper-plane with the largest distance to the nearest training data point of any class, because generally, the larger the margin, the lower the generalization error of the classifier. Hence, SVMs sometimes are also called "large margin classifier".

4.1.1 SVMs Classifier

4.1.1.1 SVMs Classifier without a Kernel

For SVMs without a kernel (with a linear kernel), the optimization objective can be written as the following cost function:

$$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)}cost_1\left(\boldsymbol{\theta^T x^{(i)}}\right)+\left(1-y^{(i)}\right)cost_0\left(\boldsymbol{\theta^T x^{(i)}}\right)\right]+\frac{1}{2}\sum_{i=1}^{n}{\theta_j}^2$$

$$cost_1\left(\boldsymbol{\theta^T x^{(i)}}\right)\approx -\log\left(h_\theta\left(\boldsymbol{x^{(i)}}\right)\right)= -log\frac{1}{1+e^{-\boldsymbol{\theta^T x^{(i)}}}}$$

$$cost_0\left(\boldsymbol{\theta^T x^{(i)}}\right)\approx -\log\left(1-h_\theta\left(\boldsymbol{x^{(i)}}\right)\right)= -\log\left(1-\frac{1}{1+e^{-\boldsymbol{\theta^T x^{(i)}}}}\right)$$

Suppose z = $\boldsymbol{\theta^T x^{(i)}}$, we can plot $cost_0(z)$ and $cost_1(z)$ as the following graphs, see Figure 6:

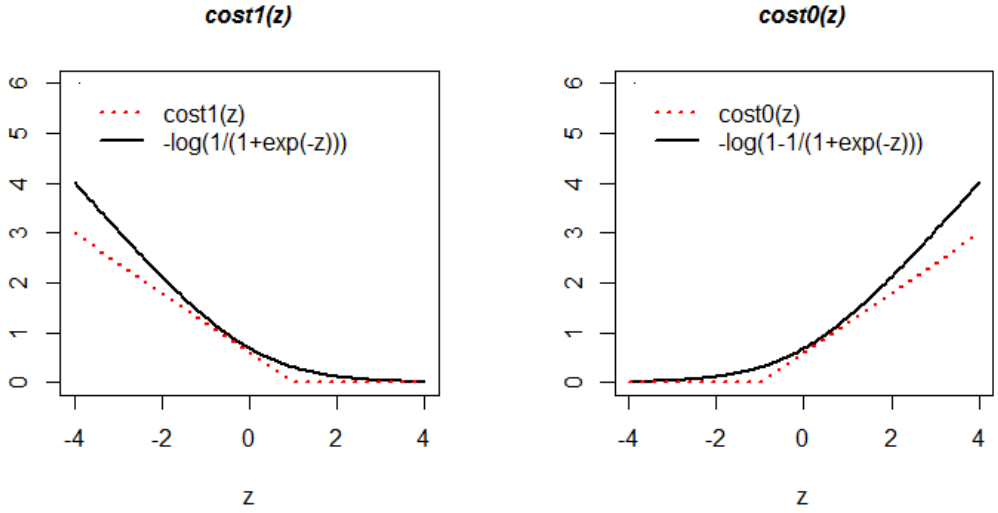

Figure 6: Graphs of cost functions, $cost_1(z)$ and $cost_0(z)$ in SVMs

Thus $cost_0(z)$ and $cost_1(z)$ have two segments respectively, as graphs shown above.

If C is very large, to minimize

$$C\sum_{i=1}^{m}\left[y^{(i)}cost_1\left(\boldsymbol{\theta^T x^{(i)}}\right)+\left(1-y^{(i)}\right)cost_0\left(\boldsymbol{\theta^T x^{(i)}}\right)\right]+\frac{1}{2}\sum_{j=1}^{n}{\theta_j}^2$$

we hope that the term

$$\sum_{i=1}^{m}\left[y^{(i)}cost_1\left(\boldsymbol{\theta^T x^{(i)}}\right)+\left(1-y^{(i)}\right)cost_0\left(\boldsymbol{\theta^T x^{(i)}}\right)\right]$$

could be equal to zero, that is, if $y^{(i)}=1$, we want $\boldsymbol{\theta^T x^{(i)}}\geq 1$, if $y^{(i)}=0$, we want $\boldsymbol{\theta^T x^{(i)}}\leq -1$. Thus, the optimization problem can be derived as the following:

$$\min_{\theta}\frac{1}{2}\sum_{j=1}^{n}{\theta_j}^2=\frac{1}{2}\left({\theta_1}^2+{\theta_2}^2+\cdots+{\theta_{n-1}}^2+{\theta_n}^2\right)=\frac{1}{2}\sqrt{\left({\theta_1}^2+{\theta_2}^2+\cdots+{\theta_{n-1}}^2+{\theta_n}^2\right)}$$

$$=\frac{1}{2}||\boldsymbol{\theta}||^2$$

s.t.

$\boldsymbol{\theta^T x^{(i)}}=\left|\left|\boldsymbol{\theta}\right|\right|||\boldsymbol{x^{(i)}}||\cos<\boldsymbol{\theta},\boldsymbol{x^{(i)}}>=p^{(i)}\left|\left|\boldsymbol{\theta}\right|\right|\geq 1$, if $y^{(i)}=1$

$\boldsymbol{\theta^T x^{(i)}}=\left|\left|\boldsymbol{\theta}\right|\right|||\boldsymbol{x^{(i)}}||\cos<\boldsymbol{\theta},\boldsymbol{x^{(i)}}>=p^{(i)}\left|\left|\boldsymbol{\theta}\right|\right|\leq -1$, if $y^{(i)}=0$

$p^{(i)}$ is the projection of $\boldsymbol{x^{(i)}}$ on $\boldsymbol{\theta}$ as the graph below shows, see Figure 7.

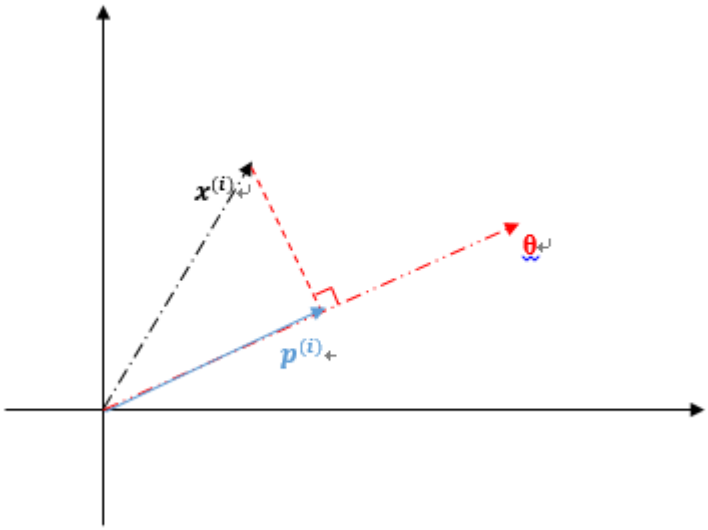

Figure 7: $\boldsymbol{p^{(i)}}$ is the projection of $\boldsymbol{x^{(i)}}$ on θ

We can get intuition from the above description. If $y^{(i)}=1$, to let $\boldsymbol{p^{(i)}}\left|\left|\boldsymbol{\theta}\right|\right|\geq 1$ and to minimize $\frac{1}{2}||\boldsymbol{\theta}||^2$, $\boldsymbol{p^{(i)}}$ should be as large as possible. And if $\boldsymbol{p^{(i)}}$ is large, the margin between different classifications will be large, which can be seen in Figure 8.

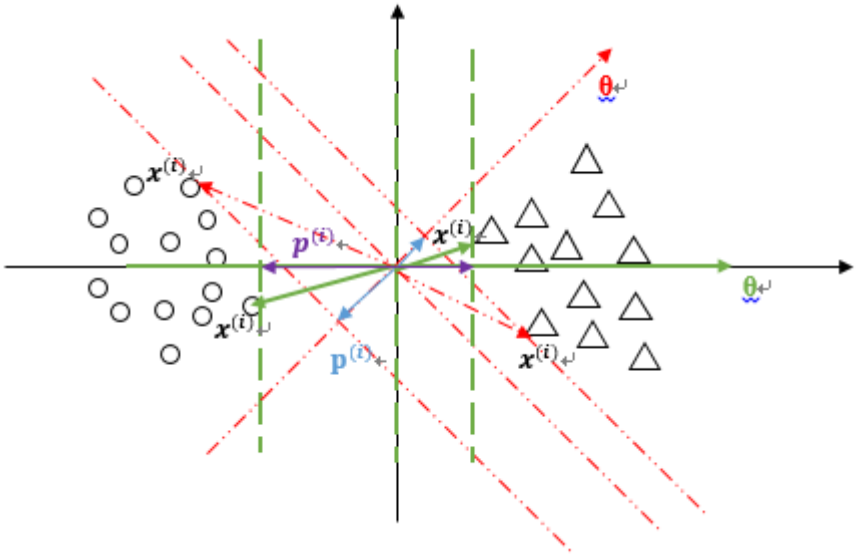

Figure 8: To get a correct classification, we should let the margin between different classifications as large as possible, so we should have $\|\boldsymbol{p}^{(i)}\|$ large enough.

4.1.1.2 SVM Classifier with a Kernel

The above part describes SVM without a kernel (with linear kernel), however sometimes the problem is non-linear and much more complex, which requires a kernel to solve.

Suppose here is a classification problem which clearly requires a non-linear boundary, which can be seen in Figure 9. Predict y = 1, if $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 {x_1}^2 + \theta_5 {x_2}^2 + \cdots \geq 0$. Suppose $f_1 = x_1, f_2 = x_2, f_3 = x_1 x_2, f_1 = {x_1}^2, f_1 = {x_2}^2$, then we have $\theta_0 + \theta_1 f_1 + \theta_2 f_2 + \theta_3 f_3 + \theta_4 f_4 + \theta_5 f_5 + \cdots \geq 0$. The features f_1, f_2, f_3, f_4, f_5 are kernels here.

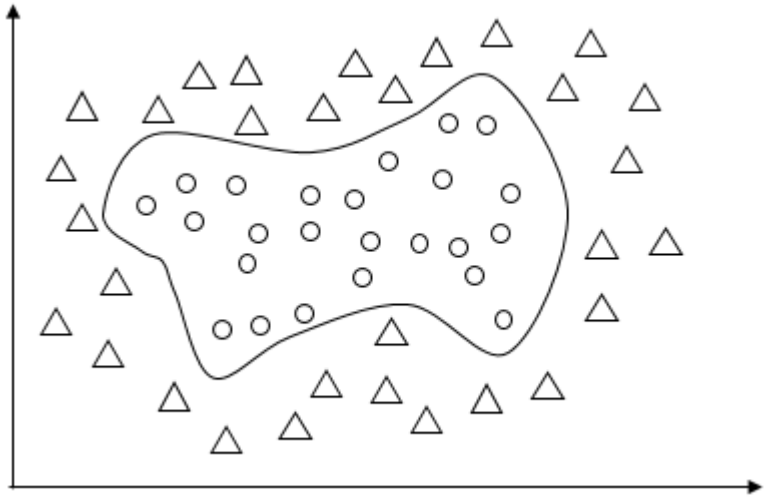

Figure 9: A simple classification problem

More generally, we often use landmarks $\boldsymbol{l}^{(1)}$, $\boldsymbol{l}^{(2)}$, $\boldsymbol{l}^{(3)}$… to define features f_1, f_2, f_3…
Given **x**,

$$f_1^{(i)} = similarity\left(\boldsymbol{x}^{(i)}, \boldsymbol{l}^{(1)}\right) = \exp\left(-\frac{\left\|\boldsymbol{x}^{(i)} - \boldsymbol{l}^{(1)}\right\|^2}{2\sigma^2}\right)$$

$$f_2^{(i)} = similarity\left(\boldsymbol{x}^{(i)}, \boldsymbol{l}^{(2)}\right) = \exp\left(-\frac{\left\|\boldsymbol{x}^{(i)} - \boldsymbol{l}^{(2)}\right\|^2}{2\sigma^2}\right)$$

$$f_3^{(i)} = similarity\left(\boldsymbol{x}^{(i)}, \boldsymbol{l}^{(3)}\right) = \exp\left(-\frac{\left\|\boldsymbol{x}^{(i)} - \boldsymbol{l}^{(3)}\right\|^2}{2\sigma^2}\right)$$

……

Here, the similarity functions $similarity\left(\boldsymbol{x}, \boldsymbol{l}^{(1)}\right)$, $similarity\left(\boldsymbol{x}, \boldsymbol{l}^{(2)}\right)$, and $similarity\left(\boldsymbol{x}, \boldsymbol{l}^{(3)}\right)$ are kernels, while $\exp\left(-\frac{\left\|x - \boldsymbol{l}^{(1)}\right\|^2}{2\sigma^2}\right)$, $\exp\left(-\frac{\left\|x - \boldsymbol{l}^{(2)}\right\|^2}{2\sigma^2}\right)$ and $\exp\left(-\frac{\left\|x - \boldsymbol{l}^{(3)}\right\|^2}{2\sigma^2}\right)$ are Gaussian Kernels, one of the most commonly used kernels.

As for landmarks $\boldsymbol{l}^{(1)}, \boldsymbol{l}^{(2)}, \boldsymbol{l}^{(3)}\ldots$ we often use $\boldsymbol{l}^{(i)} = \boldsymbol{x}^{(i)}$, i = 1, 2, 3, 4…,m, that is:

$$f_1^{(i)} = similarity\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(1)}\right) = \exp\left(-\frac{\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(1)}\right\|^2}{2\sigma^2}\right)$$

$$f_2^{(i)} = similarity\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(2)}\right) = \exp\left(-\frac{\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(2)}\right\|^2}{2\sigma^2}\right)$$

$$f_3^{(i)} = similarity\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(3)}\right) = \exp\left(-\frac{\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(3)}\right\|^2}{2\sigma^2}\right)$$

……

$$f_i^{(i)} = similarity\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(i)}\right) = \exp\left(-\frac{\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)}\right\|^2}{2\sigma^2}\right) = 1$$

……

$$f_m^{(i)} = similarity\left(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(m)}\right) = \exp\left(-\frac{\left\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(m)}\right\|^2}{2\sigma^2}\right)$$

SVM with kernel can be described as the following:

$$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)} cost_1\left(\boldsymbol{\theta}^T \boldsymbol{f}^{(i)}\right) + \left(1 - y^{(i)}\right) cost_0\left(\boldsymbol{\theta}^T \boldsymbol{f}^{(i)}\right)\right] + \frac{1}{2}\sum_{i=1}^{m}{\theta_j}^2$$

s.t.

$\boldsymbol{\theta}^T \boldsymbol{f}^{(i)} \geq 0$, if $y^{(i)} = 1$

$\boldsymbol{\theta}^T \boldsymbol{f}^{(i)} \leq 0$, if $y^{(i)} = 0$

Besides linear kernel and Gaussian kernel, the most often used kernels, we mentioned above, there is also some kind of kernel that is less often used, such as polynomial kernel, K(**x, y**) $=(\boldsymbol{x}^T \boldsymbol{y} + \mathrm{c})^d$ (c and d are parameters here). Some other kernels are seldom used, such as string kernel, chi-squared kernel, and histogram intersection kernel.

4.1.2 SVMs Regression

4.1.2.1 SVMs Regression without a Kernel

There are several kinds of SVMs regressions, and specifically, the ε-insensitive SVR (support vector regression) is mainly used in stock price prediction. So we just talk about the ε-insensitive SVR here. Our goal is to find a function f(x), which has a ε deviation from actually obtained target y_i for all training data, and at the same time is as flat as possible. f(x) can be expressed as follows:

$$\mathrm{f(x)} = wx + b \qquad w \in X, b \in R.$$

And we will solve the following problem:

$$\min \frac{1}{2}||w||^2$$

$$Subject\ to$$

$$y_i - wx_i - b \leq \varepsilon$$

$$wx_i + b - y_i \leq \varepsilon$$

If the constraints are infeasible, we can add slack variablesξ_i,ξ_i^*, and we call it soft margin formulation. It can be described by the following equation:

$$\min \frac{1}{2}||w||^2 + \mathrm{C}\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$

$$Subject\ to,$$

$$y_i - wx_i - b \leq \varepsilon + \xi_i$$

$$wx_i + b - y_i \leq \varepsilon + \xi_i^*$$

$$\xi_i\ , \xi_i^* \geq 0$$

$$\mathrm{C} > 0$$

where C shows the trade-off between the flatness of the f(x) and the amount up to which deviations larger than ε are tolerated. This is called ε-insensitive loss function $|\xi|_\varepsilon$:

$$|\xi|_\varepsilon = \begin{cases} 0, & if\ |\xi| \leq \varepsilon \\ |\xi| - \varepsilon, & if\ |\xi| > \varepsilon \end{cases}$$

The dual problem can be formulated by constructing the Lagrangian function, that is,

$$\mathrm{L} = \frac{1}{2}\left|\left|w\right|\right|^2 + \mathrm{C}\sum_{i=1}^{l}(\xi_i + \xi_i^*) - \sum_{i=1}^{l}\lambda_i(\varepsilon_i + \xi_i - y_i + wx_i + b)$$

$$-\sum_{i=1}^{l}\lambda_i^*(\varepsilon_i + \xi_i + y_i - wx_i - b) - \sum_{i=1}^{l}(\eta_i\,\xi_i + \eta_i^*\xi_i^*) \qquad (*)$$

and $\lambda_i,\ \lambda_i^*,\ \ \eta_i,\ \eta_i^*\ \geq 0$.

To solve the optimal point, we have:

$$\frac{\partial \mathrm{L}}{\partial \mathrm{w}} = w - \sum_{i=1}^{l} (\lambda_i + \lambda_i^*)\, x_i = 0 \qquad (**)$$

$$\frac{\partial \mathrm{L}}{\partial \mathrm{b}} = \sum_{i=1}^{l} (\lambda_i + \lambda_i^*) = 0$$

$$\frac{\partial \mathrm{L}}{\partial \xi_i} = C - \lambda_i - \eta_i = 0$$

$$\frac{\partial \mathrm{L}}{\partial \xi_i^*} = C - \lambda_i^* - \eta_i^* = 0$$

By substituting (**) into (*), we get the following equation,

$$\max -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\lambda_i - \lambda_i^*)(\lambda_j - \lambda_j^*)x_i x_j - \varepsilon\sum_{i=1}^{l}(\lambda_i + \lambda_i^*) + \sum_{i=1}^{l} y_i(\lambda_i - \lambda_i^*)$$

$$\text{Subject to} \sum_{i=1}^{l}(\lambda_i - \lambda_i^*) = 0$$

$$\lambda_i, \lambda_i^* \in (0, C)$$

And from (**), we have

$$w^* = \sum_{i=1}^{l} (\lambda_i - \lambda_i^*)\, x_i$$

$$\mathrm{f(x)} = \sum_{i=1}^{l} (\lambda_i - \lambda_i^*)\, x_i x_j + b^*$$

We compute the optimal value of b from the complementary slackness conditions. Specifically,

$$\lambda_i(\varepsilon + \xi_i - y_i + w^* x_i + b) = 0$$

$$\lambda_i^*(\varepsilon + \xi_i + y_i - w^* x_i - b) = 0$$

$$(\mathrm{C} - \lambda_i)\xi_i = 0$$

$$(\mathrm{C} - \lambda_i^*)\xi_i^* = 0$$

One can make a useful conclusion from the above equations. The first conclusion is that only samples (x_i, y_i) with corresponding $\lambda_i = C$ lie outside the ε-insensitive range around f. The second conclusion is that λ_i, λ_i^* cannot be zero at the same time. And finally, if λ_i is in $(0, \mathrm{C})$ then the corresponding ξ is zero. Therefore b can be computed as follows.

$$b^* = y_i - w^* x_i - \varepsilon \qquad for\ \lambda_i \in (0, C)$$

$$b^* = y_i - w^* x_i + \varepsilon \qquad for\ \lambda_i^* \in (0, C)$$

4.1.2.2 SVMs Regression with a Kernel

To make the SVMs Regression nonlinear, we could, for example, simply preprocess the training patterns x_i by a map Φ:X→F into some feature space F, as described in Aizerman, Bracerman and Rozonoer (1964) and Nilsson (1965) and then try to find a hyper-plane in the feature space. We should solve the following QP problem,

$$\max -\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}(\lambda_i - \lambda_i^*)(\lambda_j - \lambda_j^*)K(x_i, x_j) - \varepsilon\sum_{i=1}^{l}(\lambda_i + \lambda_i^*) + \sum_{i=1}^{l} y_i(\lambda_i - \lambda_i^*)$$

$$\text{Subject to } \sum_{i=1}^{l}(\lambda_i - \lambda_i^*) = 0$$

$$\lambda_i, \lambda_i^* \in (0, C)$$

At the optimal point, we have

$$w^* = \sum_{i=1}^{l}(\lambda_i - \lambda_i^*)K(x_i)$$

$$\text{f(x)} = \sum_{i=1}^{l}(\lambda_i - \lambda_i^*)K(x_i, x) + b$$

where $\text{K}(.,.)$ is a kernel function.

Symmetric positive semi-definite function, satisfying Mercer's conditions can be used as a kernel function in SVM Regression. Mercer's conditions can be written as,

$$\iint K(x,y)g(x)g(y)dxdy > 0, \qquad \int g^2(x)dx < \infty$$

where

$$\text{K(x, y)} = \exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)$$

We usually use more than one kernel to map the input space into the feature space. The question is which kernel functions provide good generalization problem. People always use some validation techniques such as bootstrapping, and cross-validation to determine a good kernel. And what's more, even when we have decided a kernel function, we should decide the parameters of the kernel. For instance, RBF has a parameter σ and one has to decide the value of σ before the experiment. It is very important to select an appropriate kernel, and many algorithms are proposed to solve this problem.

4.2 Experiment Design[④]

4.2.1 Metrics and Data

Metrics are same with metrics mentioned above, that is normalized mean square error (nMSE) and hit rate. And the data are closing prices of IBM (IBM), Microsoft (MSFT) and Citigroup (Citi) from March 30, 2009, to March 28, 2014, 1259 observations. Through getting rate of change (ROC), we have 1258 observations now.

④ Coding about the SVMs Model can be seen in appendix

4.2.2 Model

4.2.2.1 Features

We put some features which we think is related with the movement of stock price into the SVMs component of the model. Features we add are concluded from stock price. Features include 1-day, 2-day, 3-day, 5-day, 10-day, 20-day and 50-day returns, mean, median, standard deviation, median absolute deviation, skewness, and kurtosis⑤. In fact some other features can be added, such as financial indicators, like PE ratio and PB ratio, and some trend indicators like BBI, DMI, etc.

4.2.2.2 Feature Selection

This step is optional. We could do feature selection, however, we could also choose not to do feature selection. That depends on our own choice. When we do feature selection, we think that not all features can be put into SVMs model, so that we select features that can improve error of models. There are two methods for feature selection, the first one is adding features and the second one is pruning features.

As for the first method, adding features, we first put one feature into the model, and this feature should have the model perform best. And then we add a second feature into the model, which can improve the error of SVMs model to the largest degree. We add features into the model until no improvement can be made anymore.

And as for the second method, pruning features, we first put all features into the model, and then we remove one feature that can best improve the performance of the SVMs model. We won't stop removing features until the performance of the model cannot be improved.

In fact, sometimes we don't do feature selection. We just put all features we believe appropriate into the model and ignore this step.

4.2.2.3 Training SVMs Model

I just jump last step, feature selection, and put all features I think appropriate into the model, go on the this step, training SVMs model.

We use iterated method for SVMs model training. We should note that every time we finish a prediction and move on to another day's prediction, the training set changes. We always use the 500 days' data following the predicted day.

So we will get hundreds of models in the prediction process, and we will also estimate model coefficients for hundreds of times.

⑤ These indicators are calculated from the last 21 days' rate of return.

4.2.2.4 Forecasting

With SVMs model we trained, we can forecast stock price, and then compare the real stock price with the predicted stock price. We use “hit rate” and “nMSE” as criteria to make a judgement. For stock Microsoft, the hit rate and nMSE are 0.502857 and 1.004239; for stock Citigroup, the hit rate and nMSE are 0.508571 and 1.010319; and for stock IBM, the hit rate and nMSE are 0.514286 and 1.003715.

When we set kernel as polynomial, we can plot them as follows. See Figure 10, Figure 11 and Figure 12.

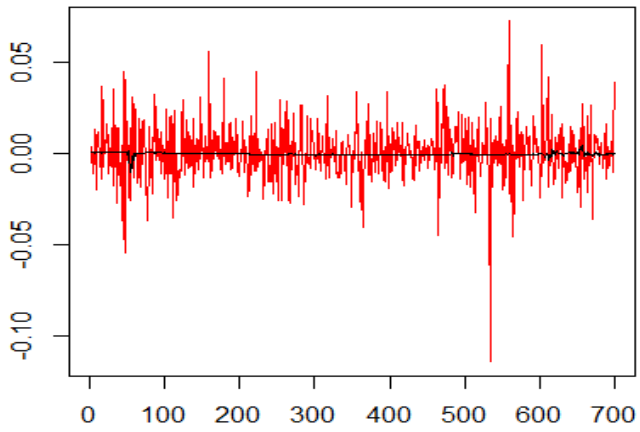

Figure 10: A line chart of Microsoft stock return, the red line is the real return and the black line is the return we predicted using our SVMs regression model. And from statistics we do, the hit rate in this case is 0.502857, and the nMSE is 1.004239.

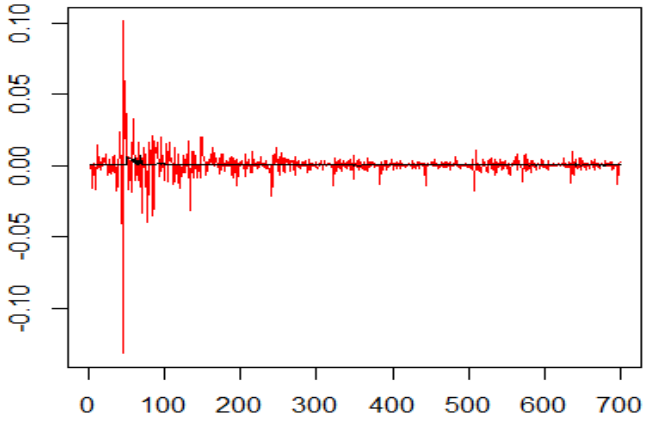

Figure 11: A line chart of Microsoft stock return, the red line is the real return and the black line is the return we predicted using our SVMs regression model. And from statistics we do, the hit rate in this case is 0.508571, and the nMSE is 1.010319.

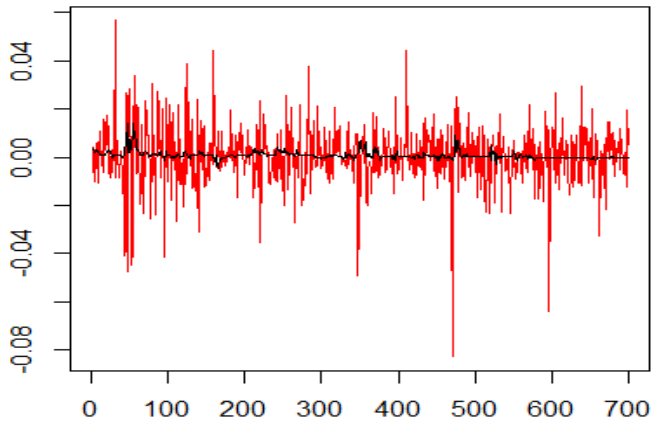

Figure 12: A line chart of Microsoft stock return, the red line is the real return and the black line is the return we predicted using our SVMs regression model. And from statistics we do, the hit rate in this case is 0.514286, and the nMSE is 1.003715.

From the results, we can see that the three black lines in graphs above are even much flatter than the black lines predicted using ARIMA. The three hit rates are all higher than 0.5, which is better than ARIMA model. However, the hit rate of IBM predicted with SVM is lower than the one predicted with ARIMA. In this case, we can see that there is no significant difference between the effects of the two methods, SVMs and ARIMA. Maybe such a simple SVMs regression is also not good enough to capture the complicated stock moving pattern.

When considering different kernels, we wonder whether different kernels will lead to significantly different results. So we compare hit rate and normalized mean square error of models using three different kernels, polynomial, radial and sigmoid. See Table 8, Table 9, and Table 10.

Table 8: Hit rate and nMSE of our return predictions on Microsoft stock using different kernels, polynomial, radial basis, and sigmoid.

Microsoft			
Kernel Type	Expression	Hit Rate	nMSE
Polynomial	(gamma*u'*v + coef0)^degree	0.502857	1.004239
Radial Basis	exp(-gamma*|u-v|^2)	0.491429	1.005152
Sigmoid	*tanh(gamma*u'*v + coef0)*	0.500000	1.002035

Table 9: Hit rate and nMSE of our return predictions on Citigroup stock using different kernels, polynomial, radial basis, and sigmoid.

Citigroup			
Kernel Type	Expression	Hit Rate	nMSE
Polynomial	(gamma*u'*v + coef0)^degree	0.508571	1.010319
Radial Basis	exp(-gamma*|u-v|^2)	0.510000	0.998489
Sigmoid	*tanh(gamma*u'*v + coef0)*	0.491429	1.009118

Table 10: Hit rate and nMSE of our return predictions on IBM stock using different kernels,

polynomial, radial basis, and sigmoid.

IBM			
Kernel Type	Expression	Hit Rate	nMSE
Polynomial	(gamma*u'*v + coef0)^degree	0.514286	1.003715
Radial Basis	exp(-gamma*|u-v|^2)	0.530000	0.997257
Sigmoid	*tanh(gamma*u'*v + coef0)*	0.500000	1.007406

From the three tables above, we don't see any significant differences when using three different kernels. And no kernel can perform really very well in all three cases.

The bad performance of SVMs model in this case may be because of the noisy stock price, or because of the not so appropriate feature input. Or maybe the stock return pattern cannot be adequately captured by the simple SVMs regression, but a more complicated model, like the model we will discuss later.

5 A Hybrid ARIMA and SVMs Model in Stock Price Forecasting

ARIMA model is one of the most widely used linear models in time series forecasting. Though the application of SVMs model in stock price forecasting is a newly developed area, it has the advantage of solving nonlinear regression estimation problems. ARIMA and SVMs could be viewed as complementary techniques, and the combination of them can help us exploit the unique strength of the ARIMA model and the SVMs model in stock price prediction.

From the test results above, we can see that though SVMs Regression outperforms ARIMA, SVMs alone still don't perform so well. The hybrid model might be a new way to solve the prediction problem.

5.1 Introduction to the Hybrid Methodology

The behavior of stock prices cannot be easily captured. Hence a hybrid strategy with both linear and nonlinear modeling abilities might be good for forecasting stock prices. Model in this section has two components, linear ARIMA component and nonlinear SVMs component. And the hybrid model (Z_t) can then be represented as follows

$$Z_t = Y_t + N_t$$

where Y_t is the linear part and N_t is the nonlinear part. Firstly, we estimate the value of the linear ARIMA component, $\widetilde{Y_t}$. ε_t represents the residual at time t,

$$\varepsilon_t = Z_t - \widetilde{Y_t}$$

ε_t can be forecasted by SVMs model, and can be represented as follows,

$$\varepsilon_t = \Phi(f_{1t} + f_{2t} + f_{3t} + \cdots + f_{nt}) + \xi_t$$

where f_{1t}, f_{2t}, f_{3t}, …, f_{nt} are features added to the SVMs model, and ξ_t is the random error. Suppose,$\widetilde{N_t} = \widetilde{\varepsilon_t}$, then we have

$$\widetilde{Z_t} = \widetilde{Y_t} + \widetilde{N_t}$$

5.2 Experiment Design

5.2.1 Metrics and Data

Metrics are same with metrics mentioned above, that is normalized mean square error (nMSE) and hit rate. And the data are closing prices of IBM (IBM), Microsoft (MSFT) and Citigroup (Citi) from March 30, 2009, to March 28, 2014, 1259 observations. Through getting rate of change (ROC), we have 1258 observations now.

5.2.2 Model

5.2.2.1 Data Preparation

From the conclusion above, we know that the data are stationary and don't need to be transformed.

5.2.2.2 Model Selection

Using history data, we decide (p, d, q) of the model based on minimum AIC. Consider the conclusion above, we set d equal to zero. We also use iterated method here, that is we choose a new parameter set (p, d, q) every 30 days[⑥], we determine new parameters, p, d, q of the $\mathrm{ARIMA(p, d, q)}$ model based on the previous 500-day history.

5.2.2.3 Coefficient Estimation of ARIMA component

Using previous 500 days' return as history data, we could get coefficients of ARIMA regressions. And thus, we get a new ARIMA model every time we predict a new day.

5.2.2.4 Residuals

We get residuals of ARIMA, which should be saved and will be used in the next step.

5.2.2.5 Features

We use features, including 1-day, 2-day, 3-day, 5-day, 10-day, 20-day and 50-day returns, mean, median, standard deviation, median absolute deviation, skewness, and kurtosis. Actually, some

[⑥] Here, 30 days is a variable, and can be altered.

other features can be added, such as financial indicators, like PE ratio and PB ratio, and some trend indicators like BBI, DMI⑦, etc.

5.2.2.6 Training SVMs Model

I put all features selected into the model and then train SVMs model. We use iterated method for SVMs model training. We should note that every time we finish a prediction and move on to another day's prediction, the training set changes. We always use the 500 days' data following the predicted day. So we will get hundreds of models in the prediction process, and we will also estimate model coefficients for hundreds of times.

With the SVMs model trained, we could forecast residuals, the non-linear component in stock return.

To improve the prediction accuracy, I add an option to the hybrid model, choosing act or not. We can get ensemble mean of the original ARIMA model, and SVMs model, and then compare them with the hybrid one. If the ensemble and the hybrid model give out the same stock moving direction, we accept the prediction of the hybrid model, or we won't accept the prediction, and choosing not to take an action.

5.2.2.7 Forecasting

We first forecast the linear component with ARIMA model, and the residual part with SVMs model. Then we add them up, getting the final forecast. Finally, comparing with the ARIMA and SVMs, we choose to accept the prediction or not. If the three models give out the same prediction about stock movement direction, we choose to accept the prediction, if not, we reject the prediction.

We have the final result in Table 11:

Table 11: Hit rate and nMSE of the three stocks, Microsoft, Citigroup and IBM using hybrid model. "Agree" means the number of predictions where the three models, ARIMA, SVMs and hybrid model give out the same direction. "Correct" means the number of predictions having the same movement direction with the real stock return in the collection of "Agree".

Company	Hit Rate	nMSE	Agree	Correct
Microsoft	0.683333	≈1	120	82
Citigroup	0.625954	≈1	262	164
IBM	0.684426	≈1	244	167

We can see from the above table, the hybrid model performs significantly better (we compare the hit rate mainly) than other two models, ARIMA model and SVMs model.

⑦ BBI, Bull And Bear Index; DMI, Directional Movement Index.

6 Problem Analysis

As the conclusion I get from previous stock market predictions that the stock market prediction problem is quite difficult. Here are several reasons. Firstly, stock prices contain a lot of noises, which means that the time series is noisy and non-stationary, and it makes it hard to use past information to generate precise predictions. In this case, it might be better to remove noises, which will lead to the prediction of a smoothed time series that may lag behind actual price changes. Secondly, many factors leading to stock price fluctuations cannot be adequately captured or maybe too numerous and too difficult to be modeled. And thus, the price of a single stock represented by a single time series model may not be enough to accurately predict its future prices.

7 Conclusions

I use three methods, ARIMA, SVMs Regression, and a hybrid method of ARIMA and SVMs, to predict rate of return of three stocks, MSFT, Citi, and IBM. And the results are not so ideal, consider the pretty low hit rate and high normalized mean square error. However, the hybrid method has a better performance compared with the other two methods. It is because that the combination of the linear model, ARIMA and the non-linear model, SVMs with a kernel, are complementary techniques, which will help to capture both the linear and the non-linear trend of the stock market. Adding an option to the hybrid model also improves the effects of the prediction to some extent.

Appendix

1 ARIMA Model Coding

1.1 “arima_main.r”

```
require(quantmod)
require(Defaults)
require(xts)
require(zoo)
require(TTR)
```

```
source("load_data.R")
source("model_selection.r")
source("arima_prediction.r")
source("utilities.r")

## Configuration area
p.max = 4
d.max = 0
q.max = 4
start = 501
day = 30
company.rets = MSFT.rets

## Code
back_test_result = back_test.arima(     company.rets,
                                        p.max=p.max,
                                        d.max=d.max,
                                        q.max=q.max,
                                        day = day,
                                        start=start)

fig3 = dev.new()
plot(back_test_result$real.values, type="l", col='red')
lines(back_test_result$predictions)

fig4 = dev.new()
pred.compare = (back_test_result$real.values > 0) - (back_test_result$predictions > 0)
barplot( pred.compare )

hr = hit_rate(back_test_result$predictions, back_test_result$real.values)
nmse = nMSE(back_test_result$predictions, back_test_result$real.values)

print(sprintf("Hit rate = %f, nMSE = %f", hr, nmse))
```

1.2 "load_data.R"

```
#read MSFT and get ROC
load("MSFT.Rda")
MSFT.close<-Cl(MSFT)
MSFT.rets <- na.trim(ROC(MSFT.close,type = "discrete",n = 1))

#read Citi and get ROC
load("Citi.Rda")
Citi.close<-Cl(Citi)
```

```
Citi.rets <- na.trim(ROC(Citi.close,type = "discrete",n = 1))

#read IBM and get ROC
load("IBM.Rda")
IBM.close<-Cl(IBM)
IBM.rets <- na.trim(ROC(IBM.close,type = "discrete",n = 1))

#read GSPC and get ROC
load("GSPC.Rda")
GSPC.close<-Cl(GSPC)
GSPC.rets <- na.trim(ROC(GSPC.close,type = "discrete",n = 1))

op<-par(mfrow=c(3,3),mar=c(2,4,3,2)+.1)
MSFT.Date<-index(MSFT)[-1]
plot(MSFT.Date,MSFT.rets,main="MSFT",xlab="time",ylab="MSFT.rets",type="l")
acf(MSFT.rets,main="")
pacf(MSFT.rets,main="")

Citi.Date<-index(Citi)[-1]
plot(Citi.Date,Citi.rets,main="Citi",xlab="time",ylab="Citi.rets",type="l")
acf(Citi.rets,main="")
pacf(Citi.rets,main="")

IBM.Date<-index(IBM)[-1]
plot(IBM.Date,IBM.rets,main="IBM",xlab="time",ylab="IBM.rets",type="l")
acf(IBM.rets,main="")
pacf(IBM.rets,main="")

par(op)
```

1.3 "model_selection.r"

```
#select best (p, d, q), which result in the smallest AIC

arma.model.selection <- function (rets, p.max, d.max, q.max)
{
    best.aic <- 1e9
    best.model <- NA
    best.p <- NA
    best.d <- NA
    best.q <- NA

    aics <- array(NA, dim = c(p.max+1, d.max+1, q.max+1) )
    for (p in 0:p.max) {
```

```
        for (d in 0:d.max) {
            for (q in 0:q.max) {
                print(sprintf("Model selection... p = %d, d = %d, q = %d", p, d, q))
                flush.console()

                r <- list(aic=1e9)

                try( r <- arima(rets, order=c(p,0,q)) )

                aics[p+1, d+1, q+1] <- r$aic

                if (r$aic < best.aic)
                {
                    best.aic <- r$aic
                    best.model <- r
                    best.p <- p
                    best.d <- d
                    best.q <- q
                    print(sprintf("aic=%f, p=%d, d=%d, q=%d", r$aic, p, d, q))
                }
            }
        }
    }

    list(aics = aics, best.aic = best.aic,
        best.model = best.model,
        best.p = best.p, best.d = best.d, best.q = best.q)
}
```

1.4 "arima_prediction.r"

```
back_test.arima<-function(company.rets, day, p.max, d.max, q.max, start, last = length(company.rets) ){
    predictions <- NULL
    real.values <- NULL

    wnd_length <- start - 1

    for(i in start:last) {
        #view the past 500 days of the predicted day as history data
        company.history.data<-company.rets[(i-wnd_length) : (i-1)]
        # select best (p, d, q) the first day and every n days
        if ((i %% day == 0) | (i == start))
        {
```

```
            model_selection_result = arma.model.selection(company.history.data,
                        p.max, d.max, q.max)
        }

        try({
            company.arima<-arima(company.history.data,
                                 order=c(model_selection_result$best.p,
                                         model_selection_result$best.d,
                                         model_selection_result$best.q))
            company.pred<-predict(company.arima)$pred[1]
        })
        # Using trained ARIMA model to get prediction of the predicted day
        predictions <- c(predictions, company.pred)
        real.values <- c(real.values, company.rets[i])

        print(sprintf("Day %d: p = %d, d = %d, q = %d, prediction = %f, real value=%f",
                        i,
                        model_selection_result$best.p,
                        model_selection_result$best.d,
                        model_selection_result$best.q,
                        company.pred, company.rets[i]))

        if (i %% 10 == 0)
        {
            plot(real.values, type="l", col='red')
            lines(predictions)

            hr = hit_rate(predictions, real.values)
            nmse = nMSE(predictions, real.values)

            print(sprintf("Hit rate = %f, nMSE = %f", hr, nmse))
        }

        flush.console()
    }

    list(predictions=predictions, real.values=real.values)
}
```

1.5 "utilities.r"

```
hit_rate <- function(   predictions,
                        real.values){
    correct = sum(
        sum((predictions > 0)&(real.values > 0)),
```

```
        sum((predictions == 0)&(real.values == 0)),
        sum((predictions < 0)&(real.values < 0))
                )
        correct/(length(predictions))
    }
nMSE <- function( predictions, real.values){
            distance_pre_real = predictions - real.values
            distance_square = distance_pre_real*distance_pre_real
            mean(distance_square)/var(real.values)
        }
```

2 SVMs Model Coding

2.1 “svm_main.R”

```
require(quantmod)
require(e1071)

source("load_data.R")
source("model_selection.r")
source("svm_prediction.r")
source("utilities.r")
source("e1071.r")

## Configuration area
start = 501
day = 10

data = svmFeatures(MSFT)
rets = MSFT.rets[index(data)]

## Code
back_test_result = back_test.svm(  data, rets,
                                   day = day,
                                   start=start)

plot(back_test_result$real.values, type="l", col='red')
lines(back_test_result$predictions)

hr = hit_rate(back_test_result$predictions, back_test_result$real.values)
nmse = nMSE(back_test_result$predictions, back_test_result$real.values)

print(sprintf("Hit rate = %f, nMSE = %f", hr, nmse))
```

2.2 "load_data.R"

```
#read MSFT and get ROC
load("MSFT.Rda")
MSFT.close<-Cl(MSFT)
MSFT.rets <- na.trim(ROC(MSFT.close,type = "discrete",n = 1))

#read Citi and get ROC
load("Citi.Rda")
Citi.close<-Cl(Citi)
Citi.rets <- na.trim(ROC(Citi.close,type = "discrete",n = 1))

#read IBM and get ROC
```

```
load("IBM.Rda")
IBM.close<-Cl(IBM)
IBM.rets <- na.trim(ROC(IBM.close,type = "discrete",n = 1))

#read GSPC and get ROC
load("GSPC.Rda")
GSPC.close<-Cl(GSPC)
GSPC.rets <- na.trim(ROC(GSPC.close,type = "discrete",n = 1))

op<-par(mfrow=c(3,3),mar=c(2,4,3,2)+.1)
MSFT.Date<-index(MSFT)[-1]
plot(MSFT.Date,MSFT.rets,main="MSFT",xlab="time",ylab="MSFT.rets",type="l")
acf(MSFT.rets,main="")
pacf(MSFT.rets,main="")

Citi.Date<-index(Citi)[-1]
plot(Citi.Date,Citi.rets,main="Citi",xlab="time",ylab="Citi.rets",type="l")
acf(Citi.rets,main="")
pacf(Citi.rets,main="")

IBM.Date<-index(IBM)[-1]
plot(IBM.Date,IBM.rets,main="IBM",xlab="time",ylab="IBM.rets",type="l")
acf(IBM.rets,main="")
pacf(IBM.rets,main="")

par(op)
```

2.3 "model_selection.r"

```
svm.model.selection <- function(   x, y,
                                   gamma=10^(-3:0),
                                   cost=10^(-3:0),
                                   sampling="cross", cross=5,
                                   kernel="radial")
{
    newSvm = tune( svm,
                   train.x=x,
                   train.y=y,
                   ranges=list( gamma=gamma, cost=cost ),
                   tunecontrol=tune.control( sampling=sampling, cross=cross ),
                   kernel=kernel )

    newSvm
}
```

2.4 "svm_prediction.r"

```
back_test.svm<-function(company.data, company.rets, start=501, last=length(data), day = 10) {
    predictions <- NULL
    real.values <- NULL

    wnd_length <- start - 1

    for(i in start:last) {
        company.history.data <- company.data[(i-wnd_length) : (i-1)]
        company.history.rets <- company.rets[(i-wnd_length) : (i-1)]
        company.today.data   <- company.data[i]
        company.today.ret    <- company.rets[i]

        train.x  <- as.matrix( coredata( company.history.data ) )
        train.y  <- as.matrix( coredata( company.history.rets ) )

        test.x   <- as.matrix( coredata( company.today.data ) )
        test.y   <- as.matrix( coredata( company.today.ret ) )

        if ((i %% day == 0) | (i == start))
        {
            model_selection_result = svm.model.selection(
                train.x, train.y)
        }

        try({
            company.svm<-svm(train.x, train.y,
                gamma  = model_selection_result$best.parameters$gamma,
                cost = model_selection_result$best.parameters$cost,
                kernel   = "radial")

            company.pred <- predict(company.svm, test.x)
        })

        predictions <- c(predictions, company.pred[1])
        real.values <- c(real.values, test.y[1])

        print(sprintf("Day %d: gamma = %f, cost = %f, prediction = %f, real value=%f",
                      i,
                      model_selection_result$best.parameters$gamma,
                      model_selection_result$best.parameters$cost,
                      company.pred[1], test.y[1]))

        if (i %% 10 == 0)
```

```
        {
            plot(real.values, type="l", col='red')
            lines(predictions)

            hr = hit_rate(predictions, real.values)
            nmse = nMSE(predictions, real.values)

            print(sprintf("Hit rate = %f, nMSE = %f", hr, nmse))
        }

        flush.console()
    }

    list(predictions=predictions, real.values=real.values)
}
```

2.5 "utilities.r"

```
hit_rate <- function(   predictions,
                        real.values){
    correct = sum(
        sum((predictions > 0)&(real.values > 0)),
        sum((predictions == 0)&(real.values == 0)),
        sum((predictions < 0)&(real.values < 0))
            )
        correct/(length(predictions))
    }

nMSE <- function( predictions, real.values){
            distance_pre_real = predictions - real.values
            distance_square = distance_pre_real*distance_pre_real
            mean(distance_square)/var(real.values)
        }
```

2.6 "e1071.r"

```
svmFeatures = function(series)
{
    require(PerformanceAnalytics)

    close = Cl(series)

    rets = na.trim(ROC(close, type="discrete"))

    # 1-day, 2-day, 3-day, 5-day, 10-day, 20-day and 50-day returns
    res = merge(na.trim(lag(rets, 1)),
```

```
                    na.trim(lag(ROC(close, type="discrete", n=2), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=3), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=5), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=10), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=20), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=50), 1)),
                    all=FALSE)

    # Add mean, median, sd, mad, skew and kurtosis
    res = merge(res,
                    xts(na.trim(lag(rollmean(rets, k=21, align="right"),1))),
                    xts(na.trim(lag(rollmedian(rets, k=21, align="right"),1))),
                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=sd),1))),
                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=mad),1))),
                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=skewness),1))),
                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=kurtosis),1))),
                    all=FALSE)

    # Add volume with a lag of two
    res = merge(res, xts(na.trim(lag(Vo(series),2))), all=FALSE)

    colnames(res) = c("ROC.1", "ROC.2", "ROC.3", "ROC.5", "ROC.10", "ROC.20", "ROC.50",
                      "MEAN", "MEDIAN", "SD", "MAD", "SKEW", "KURTOSIS",
                      "VOLUME")

    return(res)
}
```

3 Hybrid ARIMA and SVMs Model Coding

3.1 “hybrid_main.r”

```
require(quantmod)
require(e1071)
require(Defaults)
require(xts)
require(zoo)
require(TTR)

##Configuration area 1

start = 501
day = 30

source("load_data.R")
source("hybrid_svm_model_selection.r")
source("hybrid_svm_prediction.r")
source("hybrid_utilities.r")
source("ensemble_main.r")

#Configuration area 2
#ARIMA parameters
p.max = 4
d.max = 0
q.max = 4

data = svmFeatures(MSFT)
rets = MSFT.rets[paste(index(data)[1], "/")]

#Code
back_test_result_hybrid = back_test.hybrid( data, rets,
                                    day = day,
                                    start = start)

back_test_result_hybrid_predictions = back_test_result_hybrid$predictions
back_test_result_hybrid_real.values = back_test_result_hybrid$real.values
print(back_test_result_hybrid_predictions)
print(sprintf("hybrid predictions = %f", back_test_result_hybrid_predictions))
```

```
# hr = hit_rate(back_test_result_hybrid_predictions, back_test_result_ensemble_predictions,
back_test_result_hybrid_real.values)
# nmse = nMSE(back_test_result_hybrid_predictions, back_test_result_ensemble_predictions,
back_test_result_hybrid_real.values)

hr = hit_rate3(back_test_result_hybrid_predictions, back_test_result_arima$predictions,
back_test_result_svm$predictions, back_test_result_hybrid_real.values)
nmse = nMSE3(back_test_result_hybrid_predictions, back_test_result_arima$predictions,
back_test_result_svm$predictions, back_test_result_hybrid_real.values)

print(sprintf("Hit rate = %f, nMSE = %f", hr, nmse))
```

3.2 “load_data.r”

```
require(zoo)       #na.trim
require(TTR)       #ROC
require(quantmod)

#read MSFT and get ROC
load("MSFT.Rda")
MSFT.rets<-na.trim(ROC(Cl(MSFT),type="discrete",n=1))

#read Citi and get ROC
load("Citi.Rda")
Citi.rets<-na.trim(ROC(Cl(Citi),type="discrete",n=1))

#read IBM and get ROC
load("IBM.Rda")
IBM.rets<-na.trim(ROC(Cl(IBM),type="discrete",n=1))



op<-par(mfrow=c(3,3),mar=c(2,4,3,2)+.1)

plot(MSFT.rets)
acf(MSFT.rets,main="")
pacf(MSFT.rets,main="")

plot(Citi.rets)
acf(Citi.rets,main="")
pacf(Citi.rets,main="")
```

```
plot(IBM.rets)
acf(IBM.rets,main="")
pacf(IBM.rets,main="")

par(op)
```

3.3 "hybrid_svm_model_selection.r"

```
svmFeatures = function(series)
{
    require(PerformanceAnalytics)

    close = Cl(series)

    rets = na.trim(ROC(close, type="discrete"))

    # 1-day, 2-day, 3-day, 5-day, 10-day, 20-day and 50-day returns
    res = merge(na.trim(lag(rets, 1)),
                    na.trim(lag(ROC(close, type="discrete", n=2), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=3), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=5), 1)),
#                     na.trim(lag(ROC(close, type="discrete", n=10), 1)),
#                     na.trim(lag(ROC(close, type="discrete", n=20), 1)),
#                     na.trim(lag(ROC(close, type="discrete", n=50), 1)),
                    all=FALSE)

    # Add mean, median, sd, mad, skew and kurtosis
    res = merge(res,
#                     xts(na.trim(lag(rollmean(rets, k=21, align="right"),1))),
#                     xts(na.trim(lag(rollmedian(rets, k=21, align="right"),1))),
 #                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=sd),1))),
                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=mad),1))),
                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=skewness),1))),
                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=kurtosis),1))),
                    all=FALSE)

    # Add volume with a lag of two
#     res = merge(res, xts(na.trim(lag(Vo(series),2))), all=FALSE)

#     colnames(res) = c("ROC.1", "ROC.2", "ROC.3", "ROC.5", "ROC.10", "ROC.20", "ROC.50",
#                                 "MEAN", "MEDIAN", "SD", "MAD", "SKEW", "KURTOSIS",
```

```
#                               "VOLUME")

    return(res)
}



#SVMs model selection
hybrid.model.selection <- function(     x, y,
                                        gamma=10^(-3:0),
                                        cost=10^(-3:2),
                                        sampling="cross", cross=5,
                                        kernel="radial")
{
    p = 2
    d = 0
    q = 2

    company.arima<-arima(y, order=c(p, d, q))

    # Get it residual
    company.residual<-company.arima$residual

    newSvm = tune( svm,
                    train.x=x,
                    train.y=company.residual,
                    ranges=list( gamma=gamma, cost=cost ),
                    tunecontrol=tune.control( sampling=sampling, cross=cross ),
                    kernel=kernel )

    newSvm$best.p = p
    newSvm$best.d = d
    newSvm$best.q = q
    newSvm
}
```

3.4 “hybrid_svm_prediction.r”

```
back_test.hybrid<-function(company.data, company.rets, start=501, last=dim(data)[1], day = 10) {
    predictions <- c()
    real.values <- c()
```

```
arima.preds <- c()

wnd_length <- start - 1

for(i in start:last) {
    company.history.data <- company.data[(i-wnd_length) : (i-1)]
    company.history.rets <- company.rets[(i-wnd_length) : (i-1)]
    company.today.data   <- company.data[i]
    company.today.ret    <- company.rets[i]

    # Do model selection
    if ((i %% day == 0)  | (i == start))
    {
        #model_selection_result = hybrid.model.selection(company.history.data ,
company.history.rets)
        model_selection_result = list(best.p = 2, best.d = 0, best.q = 2, best.parameters =
list(cost = 1, gamma = 0.07))
    }

    try({
        # Train an arima model
        company.arima<-arima(company.history.rets,
                             order=c(model_selection_result$best.p,
                                     model_selection_result$best.d,
                                     model_selection_result$best.q))

        # Get it residual
        company.residual<-company.arima$residual

        # Train a SVM model
        company.svm<-svm(company.history.data, company.residual,
            gamma  = model_selection_result$best.parameters$gamma,
            cost = model_selection_result$best.parameters$cost,
            kernel   = "radial")

        # Make prediction
        arima_prediction <- predict(company.arima)$pred[1]
        svm_prediction <- predict(company.svm, company.today.data)

        company.pred <- arima_prediction + svm_prediction
```

```
        })

        predictions <- c(predictions, as.numeric(company.pred))
        real.values <- c(real.values, as.numeric(company.today.ret[1]))
        arima.preds <- c(arima.preds, as.numeric(arima_prediction))

         print(sprintf("Day %d: p = %d, d = %d, q = %d, gamma = %f, cost = %f, prediction = %f,
real value=%f",
                       i,
                       model_selection_result$best.p,
                       model_selection_result$best.d,
                       model_selection_result$best.q,
                       model_selection_result$best.parameters$gamma,
                       model_selection_result$best.parameters$cost,
                       company.pred, company.today.ret[1]))

        # if (i %% 10 == 0)
        # {
            # r = as.xts(zoo(cbind(return=real.values), index(company.data)[start:i]))
            # p = as.xts(zoo(cbind(return=predictions), index(company.data)[start:i]))
            # a = as.xts(zoo(cbind(return=arima.preds), index(company.data)[start:i]))
            # plot(r, type="l")
            # lines(p, col="red")
            # lines(a, col="blue")

            # hr = hit_rate(predictions, real.values)
            # nmse = nMSE(predictions, real.values)

            # print(sprintf("Hit rate = %f, nMSE = %f", hr, nmse))
        # }

        flush.console()
    }

    list(predictions=predictions, real.values=real.values)
}
```

3.5 “hybrid_utilities.r”

```
hit_rate3 <- function(      predictions1, predictions2, predictions3,
                            real.values){
```

```
    agree = sum(
        sum((predictions1 > 0)&(predictions2 > 0)&(predictions3>0)),
        sum((predictions1 == 0)&(predictions2 == 0)&&(predictions3==0)),
        sum((predictions1 < 0)&(predictions2 < 0)&&(predictions3<0))
                )
    print(agree)

    agree.and.correct = sum(
        sum((predictions1 > 0)&(predictions2 > 0)&(predictions3>0)&(real.values>0)),
        sum((predictions1 == 0)&(predictions2 == 0)&(predictions3>0)&(real.values==0)),
        sum((predictions1 < 0)&(predictions2 < 0)&(predictions3>0)&(real.values<0))
                )
    print(agree.and.correct)

    agree.and.correct / agree
    }


nMSE3 <- function( predictions1, predictions2, predictions3, real.values){
            ifagree = (((predictions1 > 0) == (predictions2 > 0)) && (predictions1 > 0) ==
(predictions3>0))
            predictions1 = predictions1[ifagree]
            predictions2 = predictions2[ifagree]
            predictions3 = predictions3[ifagree]
            real.values = real.values[ifagree]

            distance_pre_real = predictions1 - real.values
            distance_square = distance_pre_real*distance_pre_real
            mean(distance_square)/var(real.values)
        }
```

3.6 “ensemble_main.r”

```
require(quantmod)
require(Defaults)
require(xts)
require(zoo)
require(TTR)
require(e1071)

source("load_data.R")
```

```
source("arima_model_selection.r")
source("arima_prediction.r")
source("svm_model_selection.r")
source("svm_prediction.r")
source("e1071.R")

## Configuration area
p.max = 4
d.max = 0
q.max = 4
start = 501
day = 30
company.rets = MSFT.rets
data = svmFeatures(MSFT)
rets = MSFT.rets[index(data)]

## Code
back_test_result_arima = back_test.arima(  rets,
                                           p.max=p.max,
                                           d.max=d.max,
                                           q.max=q.max,
                                           day = day,
                                           start=start)

back_test_result_svm = back_test.svm(      data, rets,
                                           day = day,
                                           start=start)

print(sprintf("ensemble predictions = %f", back_test_result_ensemble_predictions))
```

3.7 "arima_model_selection.r"

```
#select best (p, d, q), which result in the smallest AIC

arma.model.selection <- function (rets, p.max, d.max, q.max)
{
    best.aic <- 1e9
    best.model <- NA
    best.p <- NA
    best.d <- NA
    best.q <- NA
```

```
    aics <- array(NA, dim = c(p.max+1, d.max+1, q.max+1) )
    for (p in 0:p.max) {
        for (d in 0:d.max) {
            for (q in 0:q.max) {
                print(sprintf("Model selection... p = %d, d = %d, q = %d", p, d, q))
                flush.console()

                r <- list(aic=1e9)

                try( r <- arima(rets, order=c(p,0,q)) )

                aics[p+1, d+1, q+1] <- r$aic

                if (r$aic < best.aic)
                {
                    best.aic <- r$aic
                    best.model <- r
                    best.p <- p
                    best.d <- d
                    best.q <- q
                    print(sprintf("aic=%f, p=%d, d=%d, q=%d", r$aic, p, d, q))
                }
            }
        }
    }

    list(aics = aics, best.aic = best.aic,
        best.model = best.model,
        best.p = best.p, best.d = best.d, best.q = best.q)
}
```

3.8 "arima_prediction.r"

```
back_test.arima<-function(company.rets, day, p.max, d.max, q.max, start, last =
length(company.rets) ){
    predictions <- NULL
    real.values <- NULL

    wnd_length <- start - 1
```

```
for(i in start:last) {
    #view the past 500 days of the predicted day as history data
    company.history.data<-company.rets[(i-wnd_length) : (i-1)]
    # select best (p, d, q) the first day and every n days
    if ((i %% day == 0) | (i == start))
    {
        model_selection_result = arma.model.selection(company.history.data,
                    p.max, d.max, q.max)
    }

    try({
        company.arima<-arima(company.history.data,
                            order=c(model_selection_result$best.p,
                                    model_selection_result$best.d,
                                    model_selection_result$best.q))
        company.pred<-predict(company.arima)$pred[1]
    })
    # Using trained ARIMA model to get prediction of the predicted day
    predictions <- c(predictions, company.pred)
    real.values <- c(real.values, company.rets[i])

    print(sprintf("Day %d: p = %d, d = %d, q = %d, prediction = %f, real value=%f",
                    i,
                    model_selection_result$best.p,
                    model_selection_result$best.d,
                    model_selection_result$best.q,
                    company.pred, company.rets[i]))

    # if (i %% 10 == 0)
    # {
        # plot(real.values, type="l", col='red')
        # lines(predictions)

        # hr = hit_rate(predictions, real.values)
        # nmse = nMSE(predictions, real.values)

        # print(sprintf("Hit rate = %f, nMSE = %f", hr, nmse))
    # }

    flush.console()
```

```
    }

    list(predictions=predictions, real.values=real.values)
}
```

3.9 "svm_model_selection.r"

```
svm.model.selection <- function(   x, y,
                                   gamma=10^(-3:0),
                                   cost=10^(-3:0),
                                   sampling="cross", cross=5,
                                   kernel="radial")
{
    newSvm = tune( svm,
                   train.x=x,
                   train.y=y,
                   ranges=list( gamma=gamma, cost=cost ),
                   tunecontrol=tune.control( sampling=sampling, cross=cross ),
                   kernel=kernel )

    newSvm
}
```

3.10 "svm_prediction.r"

```
back_test.svm<-function(company.data, company.rets, start=501, last=dim(data)[1], day = 10) {
    predictions <- NULL
    real.values <- NULL

    wnd_length <- start - 1

    for(i in start:last) {
        company.history.data <- company.data[(i-wnd_length) : (i-1)]
        company.history.rets <- company.rets[(i-wnd_length) : (i-1)]
        company.today.data   <- company.data[i]
        company.today.ret    <- company.rets[i]

        train.x  <- as.matrix( coredata( company.history.data ) )
        train.y  <- as.matrix( coredata( company.history.rets ) )

        test.x   <- as.matrix( coredata( company.today.data ) )
        test.y   <- as.matrix( coredata( company.today.ret ) )
```

```
    if ((i %% day == 0) | (i == start))
    {
        model_selection_result = svm.model.selection(
            train.x, train.y)
    }

    try({
        company.svm<-svm(train.x, train.y,
            gamma  = model_selection_result$best.parameters$gamma,
            cost = model_selection_result$best.parameters$cost,
            kernel    = "radial")

        company.pred <- predict(company.svm, test.x)
    })

    predictions <- c(predictions, company.pred[1])
    real.values <- c(real.values, test.y[1])

    print(sprintf("Day %d: gamma = %f, cost = %f, prediction = %f, real value=%f",
                  i,
                  model_selection_result$best.parameters$gamma,
                  model_selection_result$best.parameters$cost,
                  company.pred[1], test.y[1]))

    # if (i %% 10 == 0)
    # {
        # plot(real.values, type="l", col='red')
        # lines(predictions)

        # hr = hit_rate(predictions, real.values)
        # nmse = nMSE(predictions, real.values)

        # print(sprintf("Hit rate = %f, nMSE = %f", hr, nmse))
    # }

    flush.console()
}

list(predictions=predictions, real.values=real.values)
```

```
}
```

3.11 "e1071.r"

```
svmFeatures = function(series)
{
    require(PerformanceAnalytics)

    close = Cl(series)

    rets = na.trim(ROC(close, type="discrete"))

    # 1-day, 2-day, 3-day, 5-day, 10-day, 20-day and 50-day returns
    res = merge(na.trim(lag(rets, 1)),
                    na.trim(lag(ROC(close, type="discrete", n=2), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=3), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=5), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=10), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=20), 1)),
                    na.trim(lag(ROC(close, type="discrete", n=50), 1)),
                    all=FALSE)

    # Add mean, median, sd, mad, skew and kurtosis
    res = merge(res,
                    xts(na.trim(lag(rollmean(rets, k=21, align="right"),1))),
                    xts(na.trim(lag(rollmedian(rets, k=21, align="right"),1))),
                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=sd),1))),
                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=mad),1))),
                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=skewness),1))),
                    xts(na.trim(lag(rollapply(rets, width=21, align="right", FUN=kurtosis),1))),
                    all=FALSE)

    # Add volume with a lag of two
    res = merge(res, xts(na.trim(lag(Vo(series),2))), all=FALSE)

    colnames(res) = c("ROC.1", "ROC.2", "ROC.3", "ROC.5", "ROC.10", "ROC.20", "ROC.50",
                        "MEAN", "MEDIAN", "SD", "MAD", "SKEW", "KURTOSIS",
                        "VOLUME")

    return(res)
}
```

References

[1] Atiya A, Talaat N, Shaheen S. An efficient stock market forecasting model using neural networks[C]//Neural Networks, 1997, International Conference on. IEEE, 1997, 4: 2112-2115.

[2] Atsalakis G S, Valavanis K P. Surveying stock market forecasting techniques–Part II: Soft computing methods [J]. Expert Systems with Applications, 2009, 36(3): 5932-5941.

[3] Birgul Egeli A. Stock Market Prediction Using Artificial Neural Networks [J]. Decision Support Systems, 22: 171-185.

[4] Brownstone D. Using percentage accuracy to measure neural network predictions in stock market movements [J]. Neurocomputing, 1996, 10(3): 237-250.

[5] Cao Q, Leggio K B, Schniederjans M J. A comparison between Fama and French's model and artificial neural networks in predicting the Chinese stock market [J]. Computers & Operations Research, 2005, 32(10): 2499-2512.

[6] Chen A S, Leung M T, Daouk H. Application of neural networks to an emerging financial market: forecasting and trading the Taiwan Stock Index [J]. Computers & Operations Research, 2003, 30(6): 901-923.

[7] Constantinou E, Georgiades R, Kazandjian A, et al. Regime switching and artificial neural network forecasting of the Cyprus Stock Exchange daily returns[J]. International Journal of Finance & Economics, 2006, 11(4): 371-383.

[8] Glen Donaldson R, Kamstra M. Neural network forecast combining with interaction effects [J]. Journal of the Franklin Institute, 1999, 336(2): 227-236.

[9] Huang W, Nakamori Y, Wang S Y. Forecasting stock market movement direction with support vector machine [J]. Computers & Operations Research, 2005, 32(10): 2513-2522.

[10] Kim K. Financial time series forecasting using support vector machines [J]. Neurocomputing, 2003, 55(1): 307-319.

[11] Müller K R, Smola A J, Rätsch G, et al. Predicting time series with support vector machines[M]//Artificial Neural Networks—ICANN'97. Springer Berlin Heidelberg, 1997: 999-1004.

[12] Pai P F, Lin C S. A hybrid ARIMA and support vector machines model in stock price forecasting [J]. Omega, 2005, 33(6): 497-505.

[13] Rech G. Forecasting with artificial network models[R]. SSE/EFI Working Paper Series in Economics and Finance, 2002.

[14] Rojas I, Valenzuela O, Rojas F, et al. Soft-computing techniques and ARMA model for time series prediction [J]. Neurocomputing, 2008, 71(4): 519-537.

[15] Schumann M, Lohrbach T. Comparing artificial neural networks with statistical methods within the field of stock market prediction[C]//System Sciences, 1993, Proceeding of the Twenty-Sixth Hawaii International Conference on. IEEE, 1993, 4: 597-606.

[16] Trafalis T B, Ince H. Support vector machine for regression and applications to financial forecasting[C]//Neural Networks, IEEE-INNS-ENNS International Joint Conference on. IEEE

Computer Society, 2000, 6: 6348-6348.
[17] Vapnik V, Golowich S E, Smola A. Support vector method for function approximation, regression estimation, and signal processing [J]. Advances in neural information processing systems, 1997: 281-287.
[18] Wah B W, Qian M. Constrained formulations and algorithms for stock-price predictions using recurrent FIR neural networks[C]//AAAI/IAAI. 2002: 211-216.
[19] White H. Economic prediction using neural networks: The case of IBM daily stock returns[C]//Neural Networks, 1988. IEEE International Conference on. IEEE, 1988: 451-458.
[20] Yang H, Chan L, King I. Support vector machine regression for volatile stock market prediction[M]//Intelligent Data Engineering and Automated Learning—IDEAL 2002. Springer Berlin Heidelberg, 2002: 391-396.