A Stock Prediction System based on Hybrid ARIMA SVM Model

Peiran Hu

In this project, I use 3 models to predicate stock price.

# 1. ARIMA Model

## 1.1 Introduction to ARIMA Model

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

If $\varepsilon_{t-i}$ (i $=$ $1, 2, 3, 4, \ldots, q$) 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 $\mu$ and $\theta_i$(i $=$ $1, 2, 3, 4, \ldots, q$) are all constants.

If $\varepsilon_t$ is white noise, AR(p) can be defined as follows,

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

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

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

which can also be written as,

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

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

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

where

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

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

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

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

$$E(Y_1) = E(Y_2) = \cdots = E(Y_t) = \mu$$

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

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

$\mathrm{Cov}(Y_t, Y_{t-k}) = \gamma_k$ 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, $\mathrm{ACF} = \rho_k$ can be obtained from the above equation. And we can get $\mathrm{PACF} = \Phi_{ik}(i = 1,2,3,4, \dots)$ from the equation below,

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

# 2. SVM Model

## 2.1 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".

## 2.2 SVMs Classifier

### 2.2.1 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 1:

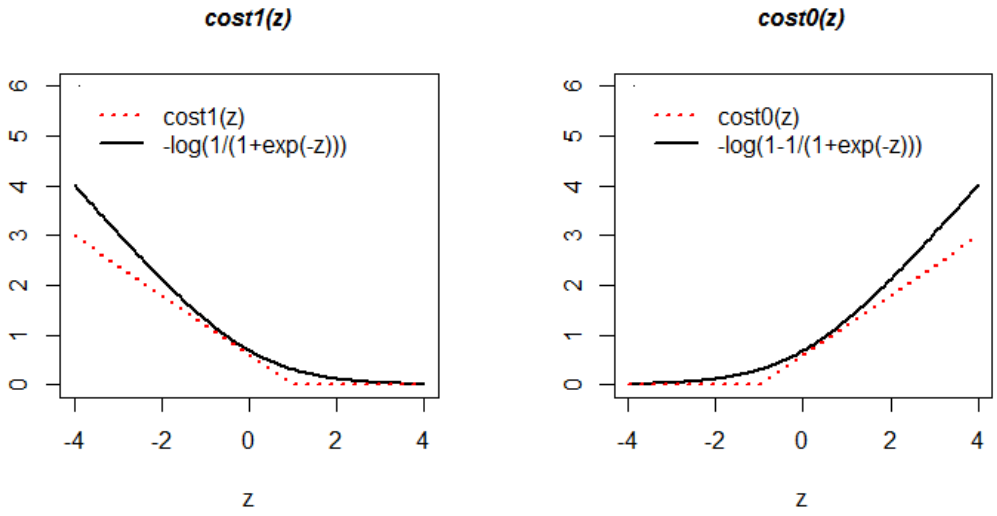


Figure 1: Graphs of cost functions, $cost_1(z)$ and $cost_0(z)$ in SVMs

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 2.

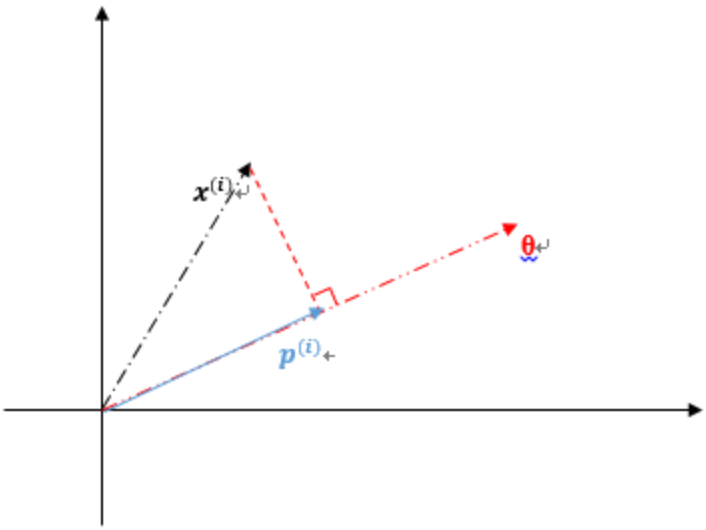


Figure 2: $p^{(i)}$ is the projection of $x^{(i)}$ on θ

We can get intuition from the above description. If $y^{(i)} = 1$, to let $p^{(i)}||\theta|| \geq 1$ and to minimize $\frac{1}{2}||\theta||^2$, $p^{(i)}$ should be as large as possible. And if $p^{(i)}$ is large, the margin between different classifications will be large, which can be seen in Figure 3.

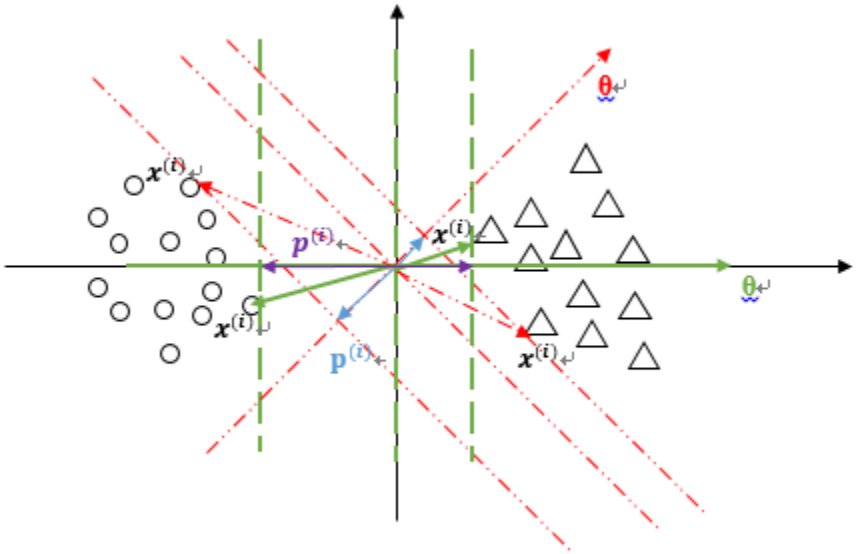


Figure 3: To get a correct classification, we should let the margin between different classifications as large as possible, so we should have $||p^{(i)}||$ large enough.

### 2.2.2 SVMs 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 4. 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.

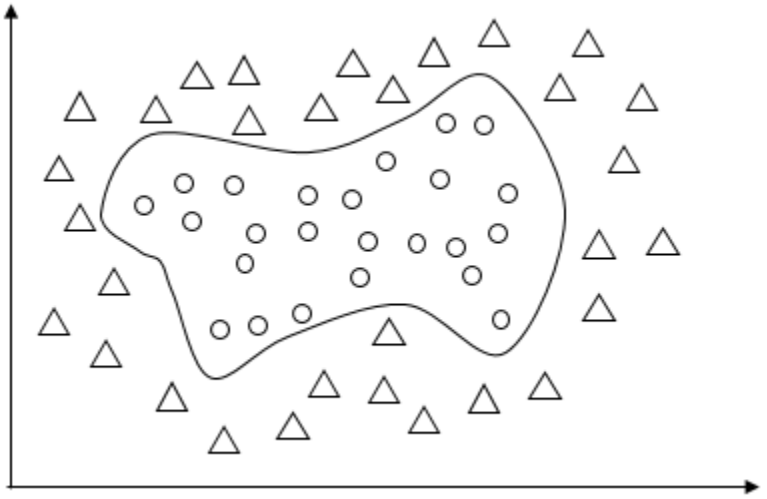

Figure 4: A simple classification problem

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**,

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

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

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

……

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

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

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

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

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

……

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

……

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

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**) =$(\boldsymbol{x}^T \boldsymbol{y} + c)^d$ (c and d are parameters here). Some other kernels are seldom used, such as string kernel, chi-squared kernel, and histogram intersection kernel.

## 2.3 SVMs Regression

### 2.3.1 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:

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

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 variables$\xi_i$,$\xi_i^*$, and we call it soft margin formulation. It can be described by the following equation:

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

$$Subject\ to,$$

$$y_i - wx_i - b \le \varepsilon + \xi_i$$
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$
$$\xi_i\ , \xi_i^* \ge 0$$
$$\mathrm{C} > 0$$

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

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

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 $\lambda_i,\ \lambda_i^*,\ \ \eta_i,\ \eta_i^* \ge 0$.

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 $\lambda_i, \lambda_i^*$ cannot be zero at the same time. And finally, if $\lambda_i$ is in $(0, \mathrm{C})$ then the corresponding ξ is zero. Therefore b can be computed as follows.

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

### 2.3.2 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 $\Phi$: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

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

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

where $K(.,.)$ 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

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

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.

# 3.Hybrid ARIMA and SVMs Model

## 3.1 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, $\widetilde{Y_t}$. $\varepsilon_t$ represents the residual at time t,

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

$\varepsilon_t$ 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 $\xi_t$ is the random error. Suppose,$\widetilde{N_t} = \widetilde{\varepsilon_t}$, then we have

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

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

# 4. Evaluation 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 $\sigma^2$ 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 $|\Phi|$ 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. Other metrics may be used according to different trading strategies.

# 5. Forecasting

return = (today's stock price- yesterday's stock price)/ yesterday's stock price

thus, from the return prediction, we can calculate the predicated price as follows:

### 5.1 ARIMA Model

(1) AXP 3/16: (1-0.001197)*80.6=80.5 3/17: 1.003371*81.5=81.77

| | | |
|---|---|---|
| (2) T | 1.000237*32.76=32.77 | 1.000243*33.06=33.07 |
| (3)BA | 1.001606145*151.57=151.81 | 1.001875*153.67=153.96 |
| (4)GE | 1.001447*25.04=25.08 | (1-0.00106)*25.45=25.42 |
| (5)GS | 1.000337*189.34=189.4 | (1-0.001523)*191.90=191.61 |
| (6)IBM | 1.001517*154.28=154.51 | 1.00011*157.08=157.1 |
| (7)MSFT | 1.001405*41.38=41.44 | 1.003328*41.56=41.70 |
| (8)MCD | 1.000458*96.35=96.39 | 1.001868*97.15=97.33 |
| (9)NKE | 1.0013698*95.81=95.94 | (1-0.000145)*96.44=96.43 |

(10)WMT 1.002831*81.9=82.13 (1-0.000727)*83.29=83.23

**5.2 SVM Model**

(1)AXP 3/16: 1.001529*80.6=80.72 3/17: 1.00061*81.5=81.55

(3) T 1.001934*32.76=32.82 1.001458*33.06=33.11

(3)BA 1.00114*151.57=151.74 1.001088*153.67=153.84

(4)GE 1.000503*25.04=25.05 1.000482*25.45=25.46

(5)GS 1.000573*189.34=189.45 1.00057*191.9=192.00

(6)IBM 1.000336*154.28=154.33 1.000366*157.08=157.14

(7)MSFT 1.000078*41.38=41.38 1.000444*41.56=41.58

(8)MCD 1.001345*96.35=96.48 1.00215*97.15=97.36

(9)NKE 1.001361*95.81=95.94 1.000907*96.44=96.53

(10)WMT (1-0.000718)*81.9=81.84 1.000642*83.29=83.34

**5.3 Hybrid ARIMA & SVM**

(1)AXP 3/16: 1.008819*80.6=81.31 3/17: 1.001471*81.5=81.62

(2)T 1.005421*32.76=32.94 1.005281*33.06=33.23

(3)BA 1.010867*151.57=153.22 1.007952*153.67=154.89

(4)GE 1.000258*25.04=25.05 (1-0.00106)* 25.45=25.42

(5)GS 1.001050*189.34=189.54 1.003051*191.9=192.49

(6)IBM (1-0.0007)*154.28=154.17 (1-0.0052)* 157.08=156.26

(7)MSFT 1.003418*41.38=41.52 1.004558*41.56=41.75

(8)MCD (1-0.001224)*96.35=96.23 1.002856*97.15=97.43

(9)NKE 1.001571*95.81=95.96 1.000629*96.44=96.50

(10) WMT (1-0.000608)*81.9= 81.85 (1-0.002042)*83.29= 83.12

The three models' prediction results are listed here, and I plan to use the hybrid model to make the final prediction, because according to my evaluation metrics I compared the three model, the hybrid model is better than the other two models.

| | ARIMA | SVM | Hybrid ARIMA&SVM |
|---|---|---|---|
| **AXP(American express)** | 3/16 80.5/real:81.5 | 80.72 | 81.31 |
| | 3/1781.77/real:80.91 | 81.55 | 81.62 |
| **T (AT&T)** | 32.77/real:32.97 | 32.82 | 32.94 |
| | 33.07/real:33.01 | 33.11 | 33.23 |
| **BA (Boeing)** | 151.81/real:153.32 | 151.74 | 153.22 |
| | 153.96/real:153.59 | 153.84 | 154.89 |
| **GE (general electric)** | 25.08/real:25.36 | 25.05 | 25.05 |
| | 25.42/real:25.15 | 25.46 | 25.42 |
| **GS (Goldman Sachs)** | 189.4/real:191.70 | 189.45 | 189.54 |
| | 191.61/real:189.94 | 192.00 | 192.49 |
| **IBM** | 154.51/real:155.93 | 154.33 | 154.17 |
| | 157.1/real:156.13 | 157.14 | 156.26 |
| **MSFT (Microsoft)** | 41.44/real:41.38 | 41.38 | 41.52 |
| | 41.7/real:41.37 | 41.58 | 41.75 |
| **MCD (McDonald's)** | 96.39/real:97.32 | 96.48 | 96.23 |
| | 97.33/real:96.24 | 97.36 | 97.43 |
| **NKE (Nike)** | 95.94/real:96.36 | 95.94 | 95.96 |
| | 96.43/real:95.7 | 96.53 | 96.5 |
| **WMT (Walmart)** | 82.13/real:83.27 | 81.84 | 81.85 |
| | 83.23/real:82.35 | 83.34 | 83.12 |

【Note: the orange box is the predicted price for 3/16, the white box is the predicted price for 3/17】

# Appendix:

## 1. ARIMA Model

**arima_main.r** (screenshot of arima_main.r)

```
require(quantmod)
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 = 1258
day = 50
company.rets = table.rets
ahead = 50

## 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)

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))
```

I use **load_data.R** to load historical stock data, use **model_selection.r** to select the optimal parameters in arima model, use **arima_prediction.r** to make the return prediction, then, use **utilities.r** to make the evaluation.

## 2. SVM Model

**svm_main.r** (screenshot of svm_main.r)

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

source("svm_load_data.r")
source("svm_training_model_prediction.r")
source("svm_model selection_feature selection.r")

#Configuration area
tt = table
historylen = 1200
corenum = 1
modelprd = "days"
featureslct = "all"

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

data = svmFeatures( tt )

rets = rets[index(data)]
data = data[index(rets)]

stopifnot( NROW( rets ) == NROW( data ) )

startDate = "2010-3-17"
endDate = "2015-3-17"

fore = svmComputeForecasts(
                data = data,
                history = historylen,
                response = rets,
                cores = corenum,
                trace = T,
                modelPeriod = modelprd,
                startDate = startDate,
                endDate = endDate,
                featureSelection = featureslct
                )

real = tt[paste(startDate, "/", endDate, sep = "")]

real.y = na.trim(ROC(Cl(real), n = 1))
pred.y = fore$Forecasts

print(sprintf("Hit Rate = %f (%d / %d),
                nMSE = %f ",
                hit.rate, pred.correct, test.examples, nmse))
```

## 3. Hybrid Model

**hybrid_main.r** (screenshot of hybrid_main.r)

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

##Configuration area 1

start = 500
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(table)
rets = table.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_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))
```