Support Vector Machine-based Stock Prediction Analysis

Jingdong Yang

School of Management and Economic, The Chinese University of Hong Kong, Shenzhen 518000, China

Abstract. Changes in the stock market are closely related to the dynamics of the market development and economic research of the whole country. Correctly predicting the stock price trend is not only beneficial for investors to make correct investment management decisions but also of great importance to promote the effective allocation of resources and improve market effectiveness. To this end, the empirical prediction and analysis of a stock development market using a support vector machine show that the accuracy of stock prediction using a support vector classification machine is as high as about 90%, indicating that the accuracy of stock prediction based on support vector machine is ideal and meaningful.

Keywords: Support Vector Machines; Stock Forecasting.

1. Introduction

In recent years, artificial intelligence technology (big data, data mining, machine learning technology) has developed rapidly, and the resulting data analysis models have mapped low-dimensional indistinguishable data to high-dimensional distinguishable data through various spatial transformations and dimensional transformations, greatly compensating for the human brain's lack of analysis of non-linear laws. We can use historical asset prices, trading data, etc. as information input to the model, analyze the characteristics of the data, and use various classification and regression prediction algorithms in the field of machine learning to construct trading strategies. In actual secondary market securities trading, stocks have time-series characteristics, so recurrent neural networks are often used for stock price prediction. However, due to the weak ability to portray time series data with a long memory, the general recurrent neural model is prone to gradient explosion and overlearning, which makes training very difficult. In contrast, support vector machine algorithms do not increase the computational complexity when mapping to higher dimensional spaces because they use kernel functions to overcome the problems of dimensional catastrophe and nonlinear differentiability [1]. In addition to this, support vector machines overcome the problem that traditional neural networks can only find local optimal solutions and discover the global minimum of the objective function using known efficient algorithms, and also have better generalization capabilities.

2. Support Vector Machine Overview

Support vector machine is a supervised learning algorithmic model for analyzing data in classification and regression analysis in machine learning, proposed by the Soviet scholar Vladimir N. Vapnik in 1964 [2]. The core idea is to derive the maximum interval hyperplane that completely separates the two classes of samples, which is a global convex optimization problem. In practice, it is called linear support vector machine and nonlinear support vector machine for the reason of linear separability of training samples or not [3], while the optimal solutions of support models can be broadly classified into three categories: (1) hard interval maximization (2) soft interval maximization (3) kernel function and nonlinear interval maximization [4].

Where the strict mathematical definitions of linearly differentiable and hyperplane are and are two-point sets (sets of points) in an n-dimensional Euclidean space. If there exists an n-dimensional vector w and a real number b such that all points belonging to the points x have, and for all points belonging to the points y then we have. Then we call linearly divisible, that is, hyperplanes in n-dimensional Euclidean spaces of dimension equal to n-1 [5].

(1) Hard interval maximization

When the training samples are linearly divisible, assuming that the sample data set is $(x_i, y_i), i = 1,2,\dots,n, x_i \in R^d, y_i \in \{+1,-1\}$, then any hyperplane can be represented by $w^T X + b = 0$ to be represented. Then in the n-dimensional space, the distance from the point x to the line $w^T X + b = 0$ is the distance of $\frac{|w^T x+b|}{\|w\|}, w = \sqrt{w_1^2 + w_2^2 + \cdots + w_n^2}$. Assume that the hyperplane (w, b) can classify the training samples correctly, i.e., for$(x_i, y_i) \in D$, if $y_i = +1$, then we have $w^T x_i + b > 0$; if $y_i = -1$, then we have $w^T x_i + b < 0$. So far we have the upper and lower hyperplanes of the maximally spaced hyperplane, and the distance of each support vector to the hyperplane can be written as $d = \frac{|w^T x+b|}{\|w\|}$, then the maximization of the distance can be expressed as $\max_{w,b} 2\frac{|w^T x+b|}{\|w\|} = \max_{w,b} \frac{2}{\|w\|}$, thus constructing the Lagrangian function. $\min_{w,b} \max_{\alpha} L(w, b, \alpha) = \frac{1}{2}\|w\|^2 + \sum_{i=1}^m \alpha_i\big(1 - y_i(w^T x_i + b)\big)\ s.t.\ \alpha_i \geq 0$, and then use the strong duality transformation to solve. $\max_{\lambda} \left[\sum_{j=1}^n \lambda_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j \big(x_i x_j\big)\right]\ s.t. \sum_{i=1}^n \lambda_i y_i = 0, \lambda_i \geq 0$. [6]

(2) Soft interval maximization

In practical applications, fully linearly divisible samples are rare, so soft spacing is introduced. Compared with the harsh conditions of hard spacing, soft spacing allows individual sample points to appear inside the spacing band, and this part of the sample points cannot satisfy the constraints$1 - y_i(w^T x_i + b) \leq 0$ .In order to measure the degree of soft spacing, we introduce a relaxation variable for each sample ξ_i , such that $\xi_i \geq 0$, the $1 - y_i(w^T x_i + b) - \xi_i \leq 0$, the optimization objective becomes $\min_{w} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^m \xi_i\ s.t. g_i(w, b) = 1 - y_i(w^T x_i + b) - \xi_i \leq 0 \quad , \quad \xi_i \geq 0 \quad , \quad i = 1,2,\dots,n$. Thus, the Lagrangian function is constructed as. $\min_{w,b,\xi} \max_{\lambda,\mu} L(w, b, \xi, \lambda, \mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^m \xi_i + \sum_{i=1}^n \lambda_i\big(1 - \xi_i - y_i(w^T x_i + b)\big) - \sum_{i=1}^n \mu_i \xi_i\ s.t. \lambda_i \geq 0, \mu_i \geq 0$,which can then be solved by using the strong duality transformation.

$\max_{\lambda} \left[\sum_{j=1}^n \lambda_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j \big(x_i x_j\big)\right]\ s.t. \sum_{i=1}^n \lambda_i y_i = 0$, $\lambda_i \geq 0$, $C - \lambda_i - \mu_i = 0$. [7]

(3) Kernel function and nonlinear interval maximization

Hard and soft spacing refers to the complete linear separability of samples or linear separability of most samples, but it will encounter sample points that are not linearly separable. For continuous samples, the solution to this situation is to map the samples that are linearly indistinguishable in the original space to a higher dimensional space, make the sample points linearly distinguishable in the higher dimensional space, and then construct the optimal classification hyperplane, which is a nonlinear SVM [8]. If the original sample points are denoted by x and $\phi(x)$ denotes the new vector obtained after mapping x to a higher dimensional feature space, then the segmentation hyperplane can be expressed as $f(x) = w^T \phi(x) + b$. Thus, the pairwise problem of nonlinear SVM is transformed into. $\max_{\lambda} \left[\sum_{j=1}^n \lambda_i - \frac{1}{2}\sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j \left(\phi(x_i)\phi\big(x_j\big)\right)\right]\ s.t. \sum_{i=1}^n \lambda_i y_i = 0, \lambda_i \geq 0, C - \lambda_i - \mu_i = 0$. The kernel function is a function that calculates the inner product of two vectors in the space after the implicit mapping, and it does this by first doing the inner product on the feature vector and then using the function k to change it, which helps to avoid the calculation in the high-dimensional space and greatly simplifies the problem solution, which is equivalent to doing the kernel mapping on the vector first and then doing the inner product, which shows that the introduction of the kernel function reduces our computation on the one hand, and reduces our storage data on the other hand memory usage. The common kernel functions are: linear kernel function ($k\big(x_i, x_j\big) = x_i^T x_j$), polynomial kernel function ($k\big(x_i, x_j\big) = \big(x_i^T x_j + R\big)^d$), and Gaussian kernel function ($k\big(x_i, x_j\big) = e^{\left(-\frac{\|x_i - x_j\|}{2\delta^2}\right)}$). [9]

3. Stock Data Analysis

The data in this paper are obtained from the CSMAR database, and representative sample stocks from three different industries are selected as the research objects, with a period from October 15, 2018, to October 14, 2022, with a total of 3,135 entries, and the data set is split into a training set and a test set with a ratio of 75% and 25%, respectively, and some of the raw data are shown in Table 1 below. The raw variables in the data set are transaction date, opening price, closing price, minimum price, maximum price, volume, price change, increase or decrease, etc. The data meaning of this paper TradingDate represents the date, Code for the stock code, OpenPrice for the opening price of the day, HighPrice for the highest price of the day, ClosePrice for the closing price of the day, LowPrice for the lowest price of the day, Volume for the volume of the day, diff for the difference between the closing price and the opening price that ClosePrice - Openprice is obtained, up for 1 represents up, up for -1 represents down.

Table 1. Table of some raw data

TradingDate	OpenPrice	ClosePrice	HighPrice	LowPrice	Volume	Amount
2018/10/15	38.555	37.348	39.092	37.147	19564292	111620218
2018/10/16	37.281	36.141	37.751	35.94	21409682	117750310
2018/10/17	36.745	36.879	37.214	35.806	24031624	131320280
2018/10/18	36.544	35.739	36.678	35.538	17711684	95611246
2018/10/19	35.001	36.678	36.946	34.666	22024600	119053631
2018/10/22	37.08	39.226	39.896	36.879	39740925	229006742
2018/10/23	39.226	38.22	39.695	37.885	23898185	138072131
2018/10/24	38.22	38.22	39.159	38.086	17971668	103214803
2018/10/25	36.812	38.086	38.22	36.544	20079863	111420962
2018/10/26	38.421	37.885	39.025	37.616	17219956	98225670
2018/10/29	37.482	38.488	39.159	37.415	23351723	134120252

3.1 Pre-processing

Data pre-processing is a key step in the modeling process. In most cases, the raw data obtained from the data acquisition stage may have missing values or outliers, so preprocessing is required. Preprocessing data not only improves the prediction of the model, but also reduces the arithmetic power loss, shortens the computation time, and subsequently optimizes the model efficiency [10]. The basic process adopted in this paper includes: applying python's data preprocessing function IsNull() to detect missing values, and no missing values are found, and performing max-min normalization on data such as volume.

3.2 Modeling

The SVM model of the Sklearn library from the Python third-party library is called, and the linear kernel function, polynomial function, sigmoid function, and RBF function are called to construct the SVM model separately, to analyze the effect of different kernel functions on the stock prediction of the SVM model.

There are many ways to evaluate models, but both classification and regression problems can be broadly based on two methods. In stock price forecasting, the mean square error MSE and the coefficient of determination R2 can be used; in stock price trend forecasting, the F1 value and the ROC curve are used [11].

In this paper, firstly, different kernel functions are evaluated based on four dimensions: training set score, test set score, test set mean square deviation, and test set R2 score, and the specific scores are shown in Figure 1. From the comparison of these four data, the linear kernel function and RBF function outperform the polynomial function and sigmoid function.

Table 2. Some raw data scoring of the kernel functions constructed

Linear Kernel	
Training set scoring	0.997256515
Test set scoring	0.979423868
Standard Deviation of the test set	0.082304526
R2 of the test set	0.917694079
Gaussian kernel function	
Training set scoring	1
Test set scoring	0.699588477
Standard Deviation of the test set	1.201646091
R2 of the test set	-0.20166644
sigmoid Kernel	
Training set scoring	0.504801097
Test set scoring	0.489711934
Standard Deviation of the test set	2.041152263
R2 of the test set	-1.04118683
Polynomial kernel function	
Training set scoring	0.68861454
Test set scoring	0.674897119
Standard Deviation of the test set	1.300411522
R2 of the test set	-0.30043355

Secondly, this paper evaluates the model by the different areas under the ROC curve, the larger the area under the ROC curve, the higher the accuracy. And from the comparison of ROC curves, the inflection points of ROC curves of different kernel function SVM models are different, among which the SVM model with linear kernel function has the highest inflection point, and the specific curve is shown in Figure 1.

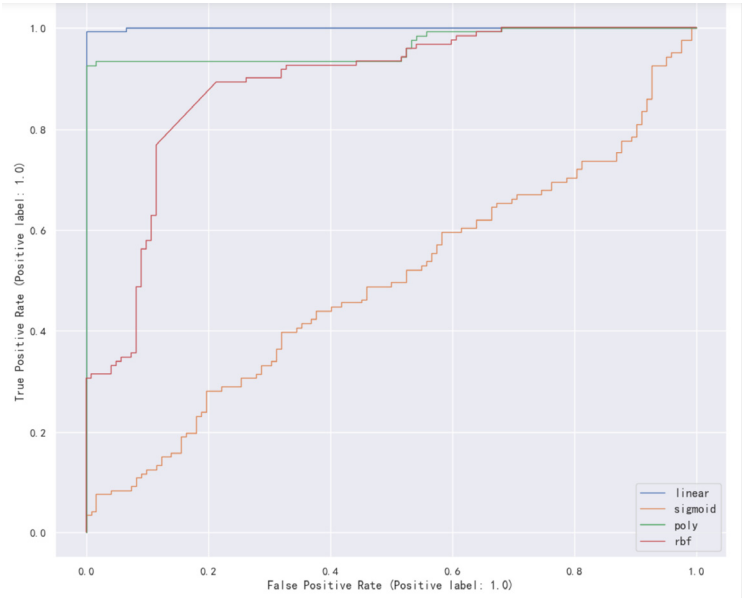


Figure 1. ROC curves of different kernel functions constructed

Finally, from the overlap curves between the predicted and true values of each model, the SVM model with linear as the kernel function has the highest degree of agreement between the predicted and true values, as shown in Figure 2.

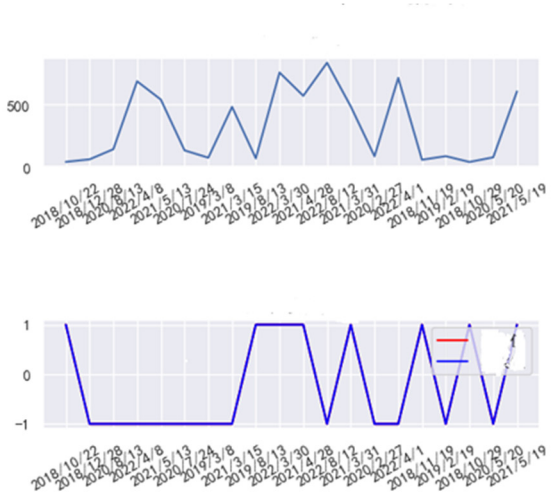


Figure 2. Overlap curve between predicted and true values constructed

The model evaluation code logic is as follows.

```
def evaluate(X_train, X_test, y_train, y_test):
    # Model name
    #Linear kernel functions
    clf3 = svm.SVC(kernel='linear')
    clf3.fit(X_train, y_train)
    y_pred3 = clf3.predict(X_test)
    print("Linear kernel function:")
    print("Training set score:", clf3.score(X_train, y_train))
    print("Test set score:", clf3.score(x_test, y_test))
    print("Test set mean squared error:", metrics.mean_squared_error(y_test,
y_pred3.reshape(-1, 1)))
    print("Test set R2 score:", metrics.r2_score(y_test, y_pred3.reshape(-1, 1)))
    # Gaussian kernel function
    clf1 = svm.SVC(kernel='rbf', C=10, gamma=0.1, coef0=0.1)#kernel function type,
gamma kernel coefficient, number of degree polynomial kernel functions, penalty factor for C error
term
    clf1.fit(X_train, y_train)
    y_pred1 = clf1.predict(X_test)
    print("Gaussian kernel function:")
    print("Training set score:", clf1.score(X_train, y_train))# The best possible score is 1.0,
which can be a negative number.
    print("Test set score:", clf1.score(x_test, y_test))
    print("Test set mean squared error:", metrics.mean_squared_error(y_test,
y_pred1.reshape(-1, 1)))
    print("Test set R2 score:", metrics.r2_score(y_test, y_pred1.reshape(-1, 1)))

    # sigmoid kernel function
    clf2 = svm.SVC(kernel='sigmoid', C=2)
    clf2.fit(X_train, y_train)
    y_pred2 = clf2.predict(X_test)
    print("sigmoid kernel function:")
    print("Training set score:", clf2.score(X_train, y_train))
    print("Test set score:", clf2.score(x_test, y_test))
    print("Test set mean squared error:", metrics.mean_squared_error(y_test,
y_pred2.reshape(-1, 1)))
```

```
        print("Test set R2 score:", metrics.r2_score(y_test, y_pred2.reshape(-1, 1)))

        # Polynomial kernel functions
        clf = svm.SVC(kernel='poly', C=2)
        clf.fit(X_train, y_train)
        y_pred = clf.predict(X_test)
        print("Polynomial kernel function:")
        print("Training set score:", clf.score(X_train, y_train))
        print("Test set score:", clf.score(x_test, y_test))
        print("Test set mean squared error:", metrics.mean_squared_error(y_test,
y_pred.reshape(-1, 1)))
        print("Test set R2 score:", metrics.r2_score(y_test, y_pred.reshape(-1, 1)))
    #The larger the area under the ROC curve, the higher the accuracy
        fig, ax = plt.subplots(figsize=(12, 10))
        lsvm_roc = plot_roc_curve(estimator=clf3, X=X_test, y=y_test, ax=ax, linewidth=1,
label='linear')
        ssvm_roc = plot_roc_curve(estimator=clf2, X=X_test,
                                  y=y_test, ax=ax, linewidth=1, label='sigmoid')
        psvm_roc = plot_roc_curve(estimator=clf, X=X_test,
                                  y=y_test, ax=ax, linewidth=1, label='poly')
        rsvm_roc = plot_roc_curve(estimator=clf1, X=X_test,
                                  y=y_test, ax=ax, linewidth=1, label='rbf')
        ax.legend(fontsize=12)
        plt.show()
        #print(y_pred1)
        # Draw the closing price trend and the true and predicted value charts
        plt.subplot(3, 1, 1)
        # Closing price chart
        plt.title("Closing Price Trend")
        plt.plot(X_test1['TradingDate']. iloc[-20:], X_test1['ClosePrice']. iloc[-20:])
        plt.xticks(rotation=30)
        plt.subplot(3, 1, 3)
        plt.title("Forecast-Real Graph")
        plt.plot(X_test1['TradingDate']. iloc[-20:], y_pred3[-20:], color='red', label='predicted
value')
        plt.plot(X_test1['TradingDate']. iloc[-20:], X_test1['up']. iloc[-20:], color='blue',
label='true value')
        plt.xticks(rotation=30)
        plt.legend()

        plt.show()

        return 0
```

4. Main Conclusion

Stock prices fluctuate frequently and, in many cases, quite dramatically. For a more stable stock market, it is necessary to reduce uncertainty for investors and funders and to enable stock forecasts to be applied more flexibly between investors and funders in order to achieve an optimal resource allocation market.

In this paper, SVM is applied to stock market forecasting. SVM is used to select the kernel function and optimize the parameters, and then find the optimal model to measure the stock trend. Based on the fast convergence and high accuracy of the SVM model, the SVM model can predict the stock data well and make the prediction results very close to the actual values. Kernel function selection and kernel parameter selection have a very important impact on the learning and prediction performance of SVM. Different kernel functions and kernel parameters are directly related to the accuracy of the operation results. svm has good accuracy in predicting stock prices and provides a meaningful analysis tool for general investors.

5. Summary

Stock data are chaotic and disorderly. The original variables of the dataset used in this paper are trading date, opening price, closing price, minimum price, maximum price, volume, and increase/ decrease rate. The SVM algorithms using different kernel functions for stock prediction are different. From the experimental analysis results, the LINEAR kernel function is more suitable for stock prediction analysis.

References

[1] Li, Hang. Statistical learning methods. Beijing: Tsinghua University Press,2012.

[2] Xixi Xu. Research on stock price prediction and investment strategy based on support vector machine. Xi'an: Northwest University Academic, 2018.

[3] Shahi Tej Bahadur, Shrestha Ashish, Neupane Arjun, Guo William. stock Price Forecasting with Deep Learning: A Comparative Study. Mathematics,2020,8(9).

[4] Zhang, X.-G. On statistical learning theory and support vector machines. Journal of Automation, 2000,01:36-46.

[5] Gu Yaxiang. Ding S. F. Advances in support vector machine research. Computer Science, 2011,02:14-17.

[6] Tang, Inv. Research on support vector machine algorithm based on statistical learning theory. Huazhong University of Science and Technology, 2005.08.

[7] Chen, Yi-Bing. Zhang L. L., Nie G. L. et al. financial time series forecasting based on improved support vector regression machine. Mathematical Practice and Perceptions, 2012, 42(04): 38-44.

[8] Jun Wei. An efficient support vector machine parameter optimization algorithm. Computer Technology and Development. 2015,25(12):97-104.

[9] Liu M, Zhou S, Wu H. A new hybrid kernel function support vector machine. Computer Applications, 2009, 29(S2):167-168.

[10] Yang XW, Hao ZF. Algorithm design and analysis of support vector machines. Beijing: Science Press, 2013.

[11] Ren, Y. F. Some research on SVM model improvement. Nanjing University of Posts and Telecommunications, 2013.