Combine Locality with Globality: Online Stock Selection with Multi-Scale and Multi-Entity Learning

grapeot

***Abstract*—Most algorithms to solve stock selection problem treat each stock either as entirely independent or with no discriminations. In this report, we provide a novel stock selection approach both responsive to recent changes and robust to noise, by combining the local and global views towards stock data. We first cluster the stocks, extract prices, volumes and their first derivatives as feature vectors in multiple temporal scales for each cluster, and then employ Support Vector Regression Machine to predict prices for each cluster as well as each scale, which are fused together for stock selection. According to the experiments, our approach can achieve state-of-the-art average annual return rates.**

I. Introduction

The problem of how to select stocks to form a promising portfolio to achieve a high return rate is interesting as well as useful, thus draws intensive research work in machine learning field. There are mainly two perspectives. One of them is to treat the stock prices as a time series, and try to predict future prices with regression approaches such as Gaussian Process [3] or Hidden Markov Model [5]. And the other is to learn the correlations between recent price changing patterns and future prices of stocks, thus to select stocks based on predicted prices. Typical approaches include Artificial Neural Network [6], Competitive Learning [10] and Support Vector Machine [2].

However, most of the conventional algorithms simply ignore the correlations among stocks or the long-term price changing trends, which causes them inaccurate or vulnerable to noise. In this report, we aim to provide a new approach for stock selection problem, both robust and accurate enough to noisy data in stock market, by taking the correlations among stocks and trends within a long-term interval into consideration. This report is organized as follows. The stock selection problem is first formulated in Sec. II, and our approach is provided in Sec. III. Related experiment methodology and results are shown in Sec. IV, followed by the conclusion in Sec. V.

II. Problem formulation

The stock selection problem is how to build a portfolio to get the best return rate after a given time. To make it clear in machine learning, we make some reasonable presuppositions. Since the trading fee varies from different brokers, we don't select a certain broker or method for fee calculation, but mainly concentrate on the return rates without trading fees. And we adopt a timely-rebuilding trading strategy here, i.e. maintaining a fixed-size portfolio from beginning to the end, selling out all the stocks and reselecting stocks to rebuild the portfolio after every time unit.

Therefore, the input of the problem is historical data of the stocks, including prices and volumes. And the output is a sequence of stock sets to select for each time unit, which is expected to gain the best return rate.

III. Approach

A. Overview

As Fig. 1 shows, the algorithm can be divided into two stages. It first tries to predict the return rate of each stock, and at second stage, it will build a portfolio based on the predictions.

To optimize the comprehensive return rate, the portfolio builder uses a greedy strategy. It takes the top n stocks with the largest expected return rate, divides the fund into n even parts, and uses each portion to buy one selected stock thus to form the portfolio, in which n is the portfolio size.

Therefore, the core part of the algorithm is how to predict the return rates. Here we try to solve this time series prediction problem based on Support Vector Regression (SVR) [7] framework. First we extract feature vectors from the historical data, and then use the feature vectors and corresponding return rates to train an SVR machine. When predicting, we use the feature vectors extracted from recent stock data to test, and get expected return rates from the SVR machine. The overview of the prediction framework is as Fig. 2 shows.

The reason why we choose SVR is, it is a powerful linear regression model, which can cover traditional linear models in time series prediction such as Autoregressive Moving Average or Kalman Filtering. In addition, it can be easily extended to nonlinear data by introducing kernels.

B. Local pattern features

Considering recent price changes have a larger effect on future stock prices, we take a local perspective on this prediction problem as the baseline, i.e. the future price is mainly determined by a recent sequence of prices and volumes. Based on this motivation, we use a sliding window to slice the historical data into segments, extract a feature vector from each segment, and train an SVR to discover the correlation between local patterns and return rates.

According to the experiment results in [4], first derivative of prices and volumes perform well in stock-related problems.

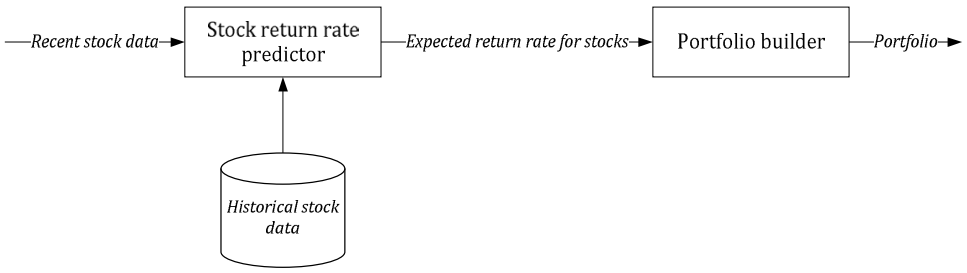

Fig. 1. Algorithm overview

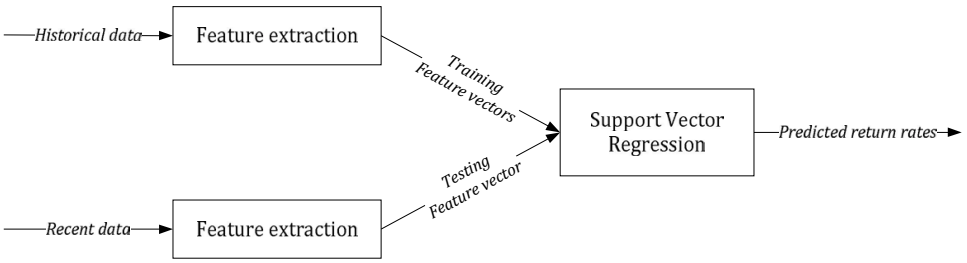

Fig. 2. Overview of return rate predictor

So for each local window $(p_1, ..., p_m; v_1, ..., v_m)$, we first normalize the prices, volumes and their first derivatives separately, and then concatenate them together as the feature vector. The return rate is also calculated as $(p_{m+1} - p_m)/p_m$, where p_i and v_i stand for the prices and volumes respectively, and m is the window length. In prediction process, we only adopt the most recent window and extract the feature vector with the same method to pass to SVR machine.

C. Multi-scale learning

Although local pattern features introduced above can describe how stock data changes, its performance is highly related to the length of the sliding window. When the window length gets larger, the feature vector reveals price changes in a longer period, thus the noise in the data will be smoothed and we can get a more stable regression result. However, this result is not responsive enough. No matter how the stock price changes recently, this change can only affect a small portion of the feature vector, and is usually ignored in the regression. Similarly, if we use a small window, the regression result will be responsive enough, and also sensitive to noise in stock market.

This requires us to adopt a global view in temporal space. To enable our predictor to respond the market changes timely, and also robust to noises, we introduce multi-scale learning. The motivation is to fuse the regression result with different window lengths together to get better result. We first train n SVR machines with different window lengths $l_1, ..., l_n$, use them to predict future return rates of a stock as $r_1, ..., r_n$, and calculate the final result by fusing them linearly with different weights $w_1, ..., w_n$, i.e. $r = \sum_{i=1}^{n} r_i w_i$. With this multi-scale learning, we can extend the window length to a large value to overcome the inherent sensitivity to noises of local perspective while still preserve its responsiveness to latest data dynamics.

D. Multi-entity learning

Until now, we are simply collecting feature vectors from all the stocks, and put them together into one single SVR machine. Although this approach can capture the common relations between historical and future prices, it lacks discrimination towards each stock, thus will have errors for each stock more or less.

A characteristic can help reduce this error, i.e. the stocks can be divided into several typical clusters. Fig. 3 shows the normalized prices of two stock clusters. We can observe that the price changing patterns are similar inside one cluster, but apparently different between clusters. Therefore this structure information can be used to improve prediction accuracy. If we split all the stocks into clusters, and train SVR machines separately, we can expect the SVR machines be more pertinent to corresponding cluster, thus can make more accurate predictions. For clustering, we employ Spectral Clustering [9] here, and use the correlations among normalized stock prices and volumes as the similarity metrics.

E. Online update

Within the process of stock trading, the market is also changing itself. On the one hand, our prediction model is getting out of date, on the other hand, we are gathering more and more information. So it is reasonable to update the regression model timely with the newly-came stock data. Here we maintain a training window. With time passing, we discard out-dated data, and adopt newly-fetched data to form a fixed-length data set which will be used to retrain the SVR machines.

IV. Experiments and discussion

A. Dataset and methodology

To evaluate our stock selection algorithm, we use the stocks from Standard&Poor 500 Index. The data includes

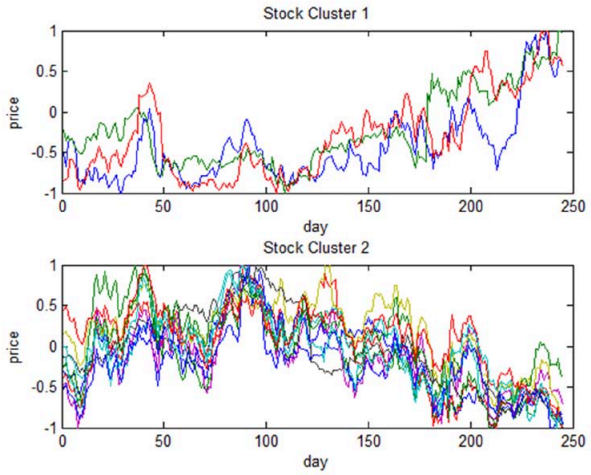

Fig. 3. Two stock clusters

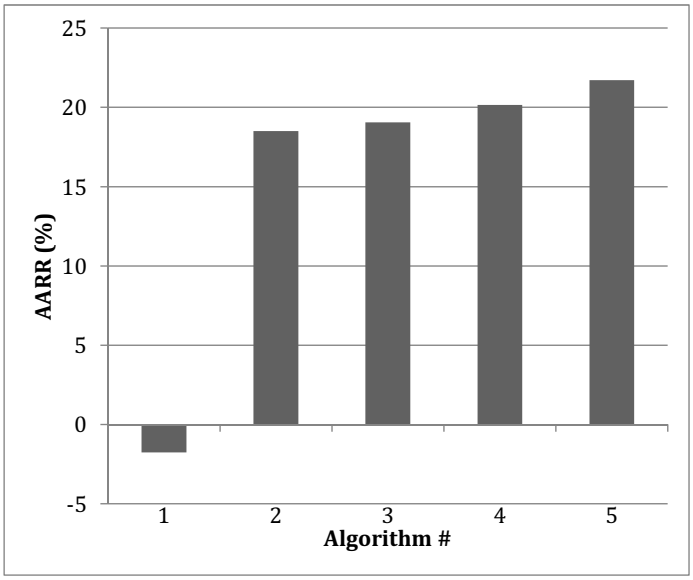

Fig. 4. Evaluation results for stock selection algorithms. Please refer to Table I to get corresponding description for each algorithm #.

weekly prices $\{p_i\}_{i=1}^{400}$ and volumes $\{v_i\}_{i=1}^{400}$ in the latest 400 weeks until Dec. 2010. We take that in first 200 weeks as initial training data, and execute our trading algorithm on the following 200 weeks. To make the evaluation clearer, we set the initial fund as \$100, and allow the algorithm to buy less than one share (for example, it can buy 0.2 share of a stock). Therefore, from the 201st week, the algorithm will determine which stocks to buy. In the next week, it will know the the prices and volumes of all the stocks last week (i.e. 201st week) and how much money it holds now, based on which it makes another decision. This process will repeat until hitting the end of the data. Thus we can get a sequence of how much money the algorithm holds in each week $\{m_i\}_{i=1}^{200}$, and the annual return rate starting from week i can be calculated as $(m_{i+52} - m_i)/m_i$ (one year is treated as 52 weeks). The performance of the algorithm is then evaluated by the average annual return rate (AARR)

$$\text{AARR} = \frac{1}{200-52} \sum_{i=1}^{200-52} \frac{m_{i+52} - m_i}{m_i}$$

We employed LibSVM [1] to implement the SVR machine and SpectralLIB [8] for spectral clustering.

B. Experiment results

With the methodology mentioned above, we test all our algorithms, from the baseline SVR with pure local pattern features to the algorithm with online updating, multi-scale and multi-entity learning. The algorithms are first executed with different settings of parameters, including the local window length, portfolio size, and cluster number in multi-entity learning, and the best performance is reported for each algorithm. We also adopt a random stock selector for comparison here. The performance of each algorithm is listed in Table I and Fig. 4.

From the result, we can see the simplest baseline outperforms the random selector obviously, and by combining online updating, multi-scale and multi-entity learning, we can get a even better result, average annual return rate as 21.72%. Considering 1.32% as the annual return rate reported by Robert et. al. in KDD '07 [10], we have got a state-of-the-art result.[1]

V. Conclusion

From the previous analysis and experiments, we can draw the conclusion that the motivation to combine locality and globality of stock data is effective in stock selection problems. The future work may be two-fold. One is to do more visualization and analysis work on stock data, trying to find more properties which can be used in price prediction and stock selection. The second is to introduce nonlinearity with kernels in the SVR framework and observe whether this will improve the performance.

References

[1] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[2] A. Fan and M. Palaniswami. Stock selection using support vector machines. In *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*, 2001.

[3] M. Farrell and A. Correa. Gaussian Process Regression Models for Predicting Stock Trends. pages 1–9, 2007.

[4] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani. Mining The Stock Market: Which Measure Is Best? In *In proceedings of the 6 th ACM Int'l Conference on Knowledge Discovery and Data Mining*, pages 487–496, 2000.

[5] M. Hassan and B. Nath. Stock market forecasting using hidden markov model: a new approach. In *Intelligent Systems Design and Applications, 2005. ISDA '05. Proceedings. 5th International Conference on*, pages 192 – 196, 2005.

[1] I have checked the code carefully to avoid over-fitting, and will keep tracking the return with real data in the following several months to ensure the reported result is correct.

TABLE I
EVALUATION RESULTS FOR STOCK SELECTION ALGORITHMS.

#	Algorithm	AARR
1	Random	-1.76%
2	SVR	18.50%
3	Online updating SVR	19.06%
4	Online updating SVR with multi-entity learning	20.16%
5	online updating SVR with multi-scale and multi-entity learning	21.72%

[6] T. Kimoto, K. Asakawa, M. Yoda, and M. Takeoka. Stock market prediction system with modular neural networks. In *Neural Networks, 1990., 1990 IJCNN International Joint Conference on*, pages 1 –6 vol.1, June 1990.

[7] A. J. Smola and B. Scholkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[8] D. Verma and M. Meila. Spectralib - package for symmetric spectral clustering, 2001. http://www.stat.washington.edu/spectral/.

[9] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007.

[10] R. J. Yan and C. X. Ling. Machine learning for stock selection. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*, page 1038, 2007.