Algorithmic Trading Strategy Based On Massive Data Mining

Haoming Li, Zhijun Yang and Tianlun Li Stanford University

Abstract

We believe that there is useful information hiding behind the noisy and massive data that can provide us insight into the financial markets. Our goal in this project is to find a strategy to select profitable U.S stocks everyday by mining the public data. To achieve this we build models that predict the daily return of a stock from a set of features. These features are constructed based on quoted and external data that is available before the prediction date. When considering machine learning models we consider both regression and classification approaches and several supervised learning algorithms are implemented. In order to catch the dynamical nature of the financial market, we carefully design out-sample testing and cross validation procedures to ensure that our historical test results are reasonable and is achievable in the real market. Finally, we construct stock portfolios based on our forecast models and illustrate the performance of these portfolios to show that our strategy works indeed.

I. Introduction

ow can we discover stocks that will rise in the future? The general answer Lis to gather as much relevant and nontrivial information as possible. One possible way to get such information is mining the huge amount of financial and Internet data that cannot be easily understood. This data allows us to define various features for each individual stock. For example, we can distinguish different stocks by their historical performance, trading volume or sensitivity to external economical and financial variables. Then we can use machine learning models to discover the underlying relation between these features and actual performance of stocks. Finally, we can select those stocks that are predicted to have the highest returns.

The report is organized as follows. In part 2 we mainly discuss what data we are using and how we collected and processed the data. In part 3 we introduce our methodology of constructing features. Part 4 gives the machine learning models that we are implementing and the procedures of dynamically training and testing. Part 5 gives the results and the perfor-

mance of our daily selected portfolio as well as some discussions and analysis on the results we get. In Part 6 we draw the summary.

II. DATA DESCRIPTION

We collected daily trading data of 2666 U.S stocks trading (or once traded) at NYSE or NASDAQ from 2000-01-01 to 2014-11-10. This dataset includes each day's open price, close price, highest price, lowest price and trading volume of every stock. Data is collected from a free online database named Quandl.

Meanwhile, we also collected data that is not directly related to each stock but may contain additional information for forecasting purposes. These include the daily quotes of 5 commodity future contracts (gold, crude oil, nature gas, corn, cotton), 2 foreign currencies (EUR, JPY) and 1 interest rate (10-year treasury rate), all from 2000-01-01 to 2014-11-10. The aggregate size of all data files is 1.11 GB.

III. TARGETS AND FEATURES CONSTRUCTION

As our goal is to predict the daily return of each stock, then we naturally define our target as stock i's daily return on day t for all i and t:

$$Target_{i,t} = \frac{ClosePrice_{i,t}}{OpenPrice_{i,t}} - 1 \tag{1}$$

Note that we can also focus only on the direction in spite of the amplitude. Another way of defining or targets is:

$$Target'_{i,t} = sign\left(\frac{ClosePrice_{i,t}}{OpenPrice_{i,t}} - 1\right)$$
 (2)

For the first definition we have a regression problem and for the second we have a classification problem. Both of these two setups will be tried.

And then we have to construct features that help distinguish (or say, define) each stock every day. These features should be relevant to the performance and should be available before the trading day. It is well known that stock performance is correlated with dozens of things and our model will only employ a relative small amount of features in this paper for simplicity.

The features we constructed can be divided into two categories. The first category, which is named as direct features, contains some variables that are constructed by explicit (and lagged) market data of stocks, e.g., open, close, high, low, etc. The other category is named as indirect features and concerns about the information carried by external factors. That is, we construct one feature for each external variable to reflect how a specific stock can be affected at a certain day when the external variable changes. Now we will have a more detailed discussion on how we construct the features by category:

Direct Features:

Based on our raw data, we constructed 4 direct features:

$$RET_{i,t} = \frac{ClosePrice_{i,t}}{OpenPrice_{i,t}} - 1 = Target_{i,t-1}$$
 (3)

$$HL_{i,t} = \frac{HighPrice_{i,t}}{LowPrice_{i,t}} \tag{4}$$

$$VOL_{i,t} = ln (TradingVolume_{i,t-1})$$
 (5)

$$VOLCHNG_{i,t} = ln\left(\frac{TradingVolume_{i,t-1}}{TradingVolume_{i,t-2}}\right)$$
(6

These four features are properly lagged so that can be computed. And these are all relevant since they measure some trends or relative strength of each stock.

Indirect Features:

The intuition of constructing indirect features corresponding to some external economic indices is to compute the 'sensitivity' of each stock's return to these indices and multiply this sensitivity by the latest indices values. As mentioned above we have data of 8 external indices (5 commodity futures, 2 foreign currencies and 1 interest rate). For index j, we define the corresponding feature for stock i at day t as:

$$EXTRN_{j_{i,t}} = \beta_{-j_{i,t}} \times index_{j_{t-1}}$$
 (7)

Where:

$$(c, \beta_{-1,t}, \dots, \beta_{-8,t})^T = argmin_{\beta} ||X_{i,t}\beta - y_{i,t}||_2$$
(8)

Where:

$$X_{i,t} = \begin{pmatrix} 1 & index_1_{t-2} & \cdots & index_8_{t-2} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & index_1_{t-T-1} & \cdots & index_8_{t-T-1} \end{pmatrix}$$
(9)

$$y_{i,t} = \begin{pmatrix} Target_{i,t-1} \\ \vdots \\ Target_{i,t-T} \end{pmatrix}$$
 (10)

Here T is an arbitrary window period parameter to compute sensitivity. Intuitively, these 8 indirect features describe the relative change of stock i'th possible price change at time t with respect to the change of index j at time t-1. Since everything defined here is lagged, the values of these 8 indirect features are available.

Thus we construct 12 features for stocks. Cross-sectional averages of these features are shown as follows:

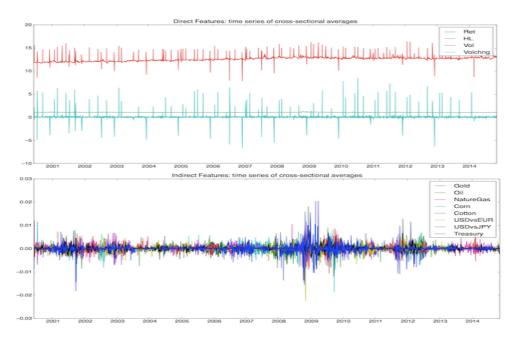


Figure 1: The upper figure indicates the cross-sectional averages of direct features, including RET, HL, VOL, VOLCHNG. The lower figure indicates the cross-sectional averages of indirect features, including gold, crude oil, nature gas, corn, cotton, USDvsEUR, USDvsJPY and Treasury

IV. LEARNING ALGORITHMS IMPLEMENTATION

Now that we have specified our targets and features, implementing specific machine learning algorithm is important.

As we have specified two ways of defining targets (numerical or categorical), we have two representations of predicting the performance of stocks: classification and regression. In the classification set-up we try to predict the trend of the stock in a specific day. Besides, we predict the exact return of a stock in regression set-up. For simplicity we first try linear models: logistic regression as the classification model and linear regression as the regression model. Then we implement SVM models (classifier and regression) to explore possible non-linear regularities utilizing kernels. Before feeding into models we also normalize and centralize our features to mean 0 and standard deviation 1. However our model is dynamic rather than fixed, depending on the date at which a return is to be predicted. Specifically, our procedure of training and testing models are as follows:

- 1. Specify a training window parameter W
- 2. To predict the performance of stocks on the date T_i , use the sample during $T_i 1, T_i 2, T_i 3, \dots, T_i W$ as the training set to train models.

After we generated predictions for every day we should measure how good our predictions are. We can compute every day's correction rate in the classification models and the mean square root error in the regression models but then it would then be abstruse to compare cross these two categories. We therefore give a more practical and visualizable way of measuring performance: testing the performance of stocks selected by our models. The methodology is as follows:

- 1. Specify a portfolio size (number of stocks to be picked) N
- 2. Every day choose N stocks according to the predictions generate by the models.

For regression model we simply choose the N stocks that are predicted to have the highest return, and for classification models we choose N stocks that are best classified, i.e., with the largest scores in the classification

3. Compute the actual return of every day's stock basket. Compare this time series to the market index such as S&P 500. Furthermore we can denote one specific day as 'successful' if the portfolio we selected has greater return than the market index and as 'unsuccessful' if that didn't happen. Then we can compute the successful rate for each model:

$$SR = \frac{N \ of \ succesful \ days}{N \ of \ total \ days} \tag{11}$$

Then we compare different models and present the model with the best successful rate. Results see Figure 2 and Figure 3

V. Discussions and Analysis

The linear models behave well. Actually from the portfolio return graphs we can see that linear classification model and linear regression model give similar results here. The regression approach and classification approach both can capture the underlying regularities.

The supporting vector machines don't behave well enough. One possible reason is that for these extremely noisy data linear simple models can behave better. SVM classifier's performance is especially bad. One possible reason is that the 'confidence' of the classification, or say decision function that we are sorting as an indicator of potential success doesn't make much sense in the non-linear case. Using more data to train each day's model may improve the performance of SVM, but the computational cost will increase significantly since we retrain our model for each trading day.

Another interesting question to think is whether our models behave stably over time. From the graph we find that the 2 linear models behave relatively stably before the end of 2008. To quantify this we compute SR every 80 days and plot these time series showed in Figure 4

VI. CONCLUSION

To conclude: we derived an approach to predict daily returns of U.S stocks based on their trading data and external financial indices. Our linear models work well in both regression framework and classification framework. The best model turns out to be linear classifier: logistic regression. It gives 56.65% successful rate and 2000% cumulative return over 14 years. However as time pass by the models tend to behave less stably especially after 2008.

In the future we can try to find methods that give more stable predictions. From the perspective of machine learning we can try mixing different models our train models with more data every day. From the perspective of investment we can make predictions on 'alphas' rather than returns. Also we can try to get more information from text data. News and social networks can be excellent information resources for predicting stocks returns.

REFERENCES

[Abu-Mostafa and Atiya, 1996] Abu-Mostafa, Y. S., & Atiya, A. F. (1996). Introduction to financial forecasting. *Applied Intelligence.*, 6(3):205-213.

[Smola and Schölkopf, 2004] Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression *Statistics and computing.*, 14(3): 199-222.





Figure 2: *left figure is the result of Logistic Regression modelIn the implementation we specify our window parameter* W=5 and portfolio size N = 100. We first implement linear models. The logistic regression gives SR = 56.65%. The cumulative return of stocks selected by this model. The right figure shows linear regression model giving SR = 55.82%. The cumulative return of the portfolio selected.As we can see that for linear models, both classification setup and regression setup behave well. \$1 invested in 2000 became about \$20 in 2014 if we have continually implemented the trades suggested by these 2 models. This success indicates that our approach did get information from the data we collected.

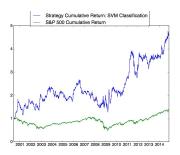




Figure 3: The left is SVM. We use Gaussian Kernel and since the financial data is highly noised we set the parameter C as 0.85. Surprisingly SVM give worse results than linear models. For SVM classifier we have: SR = 49.56%. The return of selected portfolio. The rightis SVM regression we get SR = 53.18%. After we adjusted the window parameter W and the regularization parameter W this doesn't improve much.

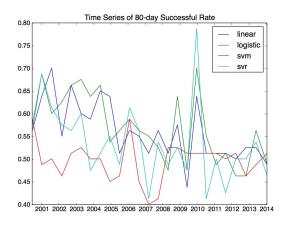


Figure 4: It can be shown that all models tend to behave less well as time goes by especially after 2008. We believe that from that time daily stock prices change depend on more factors than we have recovered.