Application of Neural Network and SVM in Designing Trading Strategies (MATH6380J Final Project)

Yue Jiang*
Zhenzhen Li†
Lizhang Miao‡

May 20, 2017

Abstract

With the growth of electronic exchanges around the world, the speed of trading has increased a lot. High-frequency traders rely on capturing trading opportunities in the market in a short period of time. In this project, we apply neural network and SVM methods to design high-frequency trading strategies. We also check the performance of strategies on Mainland futures market. Detailed numerical results will be presented as well as discussion on possible improvement.

1 Introduction

With the growth of electronic exchanges around the world, the speed of trading has increased a lot. More and more high-frequency traders are playing important roles

*Department of Mathematics, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. Email: yjiangan@connect.ust.hk.

†Department of Mathematics, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. Email: zlice@connect.ust.hk.

‡Department of Mathematics, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong. Email: lmiao@connect.ust.hk.

in the market. For high-frequency trading, the key is to capture the trading opportunities in a short period of time once a signal is observed in the market. Manually deciding when and how to trade may not be realistic and objective when coming to high-frequency circumstances. Hence designing trading strategies and using trading suggestions provided by algorithms becomes a hot direction. Machine learning as a popular research topic, it has also gained popularity in trading.

Financial data observed from the market are plenty and high-dimensional. The interactions between different prices and technical indicators are too complicated for simple models to capture the ground truth. Neural network is a commonly used technique that does not require any assumption on the dependence structure between features and output, and it can also deal with large amount of data. Besides, trading strategies based on statistical arbitrage do not need assumptions on the dynamics of asset price so SVM can be used here to classify upward and downward changes in asset price and provide suggestions on trading.

The report is organized as follows. Section 2 will describe the dataset used for the project. Section 3 will present some important definitions in this work. Section 4 and Section 5 will present the Neural network and SVM models for forecasting future price change. The proposed trading strategy algorithm will be described in Section 6. Section 7 presents the strategy performance. A brief summary will be given in Section 8.

2 Dataset Description

The dataset we use in this work is provided by Prof. Chen and his group in HKUST. As the trading strategy we are going to propose does not involve selecting from a pool of assets, we focus on one futures which is the most frequently traded in Mainland futures market in this work, i.e. the HS300 Stock Index futures. Ticker of HS300 Stock Index futures is "IF".

For IF futures, the trading time is from 9:15 a.m. to 3:15 p.m.. For every one minute, the closing price, open price, highest price, lowest price and trading volume in this minute are recorded. The pre-opening and post-close data are also included in the dataset and shown as data corresponding to 9:14 a.m. and 3:15 p.m.. The minute data of IF futures ranges from February 17, 2014 to February 3, 2017. For missing data in the dataset, we use the most recently recorded value in the past to replace

"NA".

3 Definitions

Let p_t^{close} be the current closing price, p_t^{high} be the current highest price, p_t^{low} be the current lowest price and p_t^{open} be the current open price. Below some important definitions are presented.

Definition 1. *Current typical price p_t is defined to be the average of current highest, lowest and closing prices.*

Definition 2. *Let p_{t-1} be the previous typical price. Then pseudo-log-return R_t is defined as* $\log(\frac{p_t}{p_{t-1}})$.

4 Neural Network Modelling

In this section, we present the neural network model that we use to predict the future changes in asset price.

4.1 Feature Selection

In total three groups of features are used: current time, last n pseudo-log-returns and last n standard deviations of transaction prices, where n is the window size. The current time includes two inputs: current hour and current minute. The number of inputs in total is $2n + 2$. In this project, we choose n to be 3.
We use the difference between highest and lowest prices divided by typical price to represent the standard deviation of transaction prices in one minute.

4.2 Output Selection

The deep neural network predicts the next one-minute pseudo-log-return. As we want to predict a variable that can best describe the market behavior, using closing, highest price or lowest price will bring much noise to the model. Besides, the objective of trading strategy is to learn the dynamics of the market in next one minute, and then take corresponding actions based on the prediction. As we do not have detailed tick

and trade data of IF futures, we decide to use the typical price to approximate the average price in one minute.

4.3 Network Architecture

The architecture is selected to have one input layer, five hidden layers and one output layer. The number of neurons in each layer depends on the number of inputs I. Each layer has $I, I, \lfloor 4I/5 \rfloor, \lfloor 3I/5 \rfloor, \lfloor 2I/5 \rfloor, \lfloor I/5 \rfloor$ and 1 neurons respectively. All neurons use a tanh activation function except that the output neuron uses a linear activation function (Arévalo et al.; 2016).

4.4 Neural Network Training

The dataset consists of 725 days of minute data for IF futures. After calculating the input features of neural network, we get 192850 observations. The first 483 days' data is chosen to be training set, which is roughly $\frac{2}{3}$ of the whole data. The remaining 242 days' data is regarded as test set. We first estimate the parameters in neural network using training set and then estimate the predicted next one-minute typical price on the whole test set.
In this project, we use the R package "H2O" to implement the neural network that we propose.

5 SVM Modelling

In this section, we present the SVM model that we use to predict the next one-minute typical price of the underlying futures contract. The input features are the same as those in Section 4. The output of SVM is also the next one-minute pseudo-log-return of the underlying asset.
In this project, we use the R package "e1071" to implement the SVM method that we propose.

6 Trading Strategy Algorithm

For each trading minute, we buy (sell) a share of underlying futures if the predicted pseudo-log-return of next minute is larger (smaller) than a threshold α,where α is

chosen by the investor. Usually this α should be larger than the transaction cost and it depends on the risk aversion of the investor. When the price exceeds the predicted typical price in this minute, we sell (buy) a share of futures. If the price never exceeds the predicted typical price until the end of this minute, we sell (buy) a share of futures with the closing price of the minute.

As we do not have detailed trade data of IF futures, we can not tell when the price will exceed the predicted typical price inside one minute. So when evaluating the performance of this trading strategy, if the predicted pseudo-log-return is in-between highest and lowest prices of next minute, then we assume that the trader is able to sell (buy) at the price of predicted typical price.

7 Strategy Performance

7.1 Prediction Accuracy

We first compare the prediction accuracy of neural network and SVM models that we propose before. We define the directional accuracy to be:

$$DA = 100 * \sum_{i=1}^{N} 1_{R_i * \hat{R}_i > 0}, \tag{1}$$

and the mean squared error to be:

$$MSE = \sum_{i=1}^{N} (R_i - \hat{R}_i)^2. \tag{2}$$

For both neural network and SVM methods, we estimate the model on training set and then calculate predicted next one-minute pseudo-log-return using the same set of parameters for each day in test set.

	Neural Network	SVM
MSE	2.852482e-07	2.762665e-07
DA	49.49512	50.47226

Table 1: Prediction Accuracy of Neural Network and SVM

The prediction accuracy of neural network and SVM methods are summarized in Table 1.

7.2 Evaluation of Strategy Performance

Typically when evaluating the performance of a trading strategy in finance, we need to running backtesting on historical data to check the return, Sharp ratio, max drawdown and hit rate etc. The length of historical data should be long enough to incorporate different market conditions but not too long at the same time to avoid degrading potential useful strategies.

There are some subtle issues when evaluating strategy performance, such as how to include transaction cost and how to determine if a limit buy (sell) order is filled when a market sell (buy) order arrives. Usually we need to construct an algorithm to match the incoming market orders with limit orders waiting in the limit order book according to price-time priority of order book. A detailed check of tick data and trade data of the asset is also require to ensure the accuracy of the algorithm.

However, in this project we only have 725 days of data in total and we only buy (sell) one share of the underlying futures contract at a time. Therefore we assume that our limit orders can always be filled by incoming market orders and we calculate the daily return on the test dataset to get the standard deviation and Sharp ratio. The transaction cost is set to be 0.0003 per trade.

	Neural Network	SVM
Mean Return by Trade	0.0001	0.0001
Std of Return by Trade	0.0009	0.0009
Sharp Ratio	1.6186	1.1672

Table 2: Strategy Performance of Neural Network and SVM Methods with $\alpha = 0.0003$

	Neural Network	SVM
Mean Return by Trade	0.0002	0.0001
Std of Return by Trade	0.0011	0.0010
Sharp Ratio	2.5216	2.3036

Table 3: Strategy Performance of Neural Network and SVM Methods with $\alpha = 0.0004$

	Neural Network	SVM
Mean Return by Trade	0.0002	0.0002
Std of Return by Trade	0.0012	0.0012
Sharp Ratio	2.9268	3.0159

Table 4: Strategy Performance of Neural Network and SVM Methods with $\alpha = 0.0005$

The performance of strategies based on neural network and SVM methods are summarized in Table 2, Table 3 and Table 4 with different values of α.

8 Summary

In this project, we implement both neural network and SVM on a real dataset of IF futures to predict the next one-minute pseudo-log-return and check the performance of a trading strategy. For both method, the performance of strategy are close on the test dataset and the Sharp ratio is larger than 2, which indicates that this strategy may be useful in practice. Yet to apply this strategy into real trading, we still need to do detailed backtesting and look at the long-term performance and stability of the strategy.

When implementing neural network and SVM to fit the real dataset, we did not work a lot on parameter tuning as tuning too much will make the strategy more close to data mining, or a specific model that works on only a few assets and generate overfitting. When designing strategy, discovering some important features or models is of more significance. Besides, good trading strategies are usually composed of several strategies that have normal performance. Combining several useful features is also a key step.

The models presented in this project are simple examples and there may be improvements in the future. For example, we can use a moving window on the test dataset to reestimate the parameters on a rolling basis as history in the near past may be more important compared to history long time ago. Moreover, we may try using two neural networks or SVMs at the same time, one for predicting the next one-minute return while the other one for predicting the next two-minute return, and designing more robust strategies.

Thank Prof. Yao for introducing lots of methods in data analysis to us and encour-

aging us to practice. We also hope this report may be helpful to classmates who are interested in trading strategies.

Appendix A R Code of Neural Network Method

```
AVG<-(close+high+low)/3
AVG_lag1<-pushback(AVG,L=1)
pseudo_log_return<-log(AVG/AVG_lag1)
pseudo_log_return_lag_1<-pushback(pseudo_log_return,L=1)
pseudo_log_return_lag_2<-pushback(pseudo_log_return,L=2)
pseudo_log_return_lag_3<-pushback(pseudo_log_return,L=3)

RANGE<-(high-low)
STD<-RANGE/AVG
STD<-pushback(STD,L=1)
STD_lag_1<-pushback(STD,L=1)
STD_lag_2<-pushback(STD,L=2)
STD_lag_3<-pushback(STD,L=3)

HOUR<-as.numeric(substr(rownames(RANGE),1,2))
HOUR<-pushback(HOUR,L=1)
MINUTE<-as.numeric(substr(rownames(RANGE),3,4))
MINUTE<-pushback(MINUTE,L=1)

#calculate target value on each day
target<-log(AVG/pushback(AVG,L=forward))
target<-target[c(-1:-(forward+3)),]
target<-c(as.matrix(target))


pseudo_log_return_lag_1<-pushback(pseudo_log_return_lag_1,L=(forward-1))
pseudo_log_return_lag_2<-pushback(pseudo_log_return_lag_2,L=(forward-1))
pseudo_log_return_lag_3<-pushback(pseudo_log_return_lag_3,L=(forward-1))
STD_lag_1<-pushback(STD_lag_1,L=(forward-1))
STD_lag_2<-pushback(STD_lag_2,L=(forward-1))
STD_lag_3<-pushback(STD_lag_3,L=(forward-1))
HOUR<-pushback(HOUR,L=(forward-1))
MINUTE<-pushback(MINUTE,L=(forward-1))

#put them into vector form
```

```
pseudo_log_return_lag_1<-pseudo_log_return_lag_1[c(-1:-(forward+3)),]
pseudo_log_return_lag_1<-c(as.matrix(pseudo_log_return_lag_1))
pseudo_log_return_lag_2<-pseudo_log_return_lag_2[c(-1:-(forward+3)),]
pseudo_log_return_lag_2<-c(as.matrix(pseudo_log_return_lag_2))
pseudo_log_return_lag_3<-pseudo_log_return_lag_3[c(-1:-(forward+3)),]
pseudo_log_return_lag_3<-c(as.matrix(pseudo_log_return_lag_3))
STD_lag_1<-STD_lag_1[c(-1:-(forward+3)),]
STD_lag_1<-c(as.matrix(STD_lag_1))
STD_lag_2<-STD_lag_2[c(-1:-(forward+3)),]
STD_lag_2<-c(as.matrix(STD_lag_2))
STD_lag_3<-STD_lag_3[c(-1:-(forward+3)),]
STD_lag_3<-c(as.matrix(STD_lag_3))
HOUR<-HOUR[c(-1:-(forward+3)),]
HOUR<-c(as.matrix(HOUR))
MINUTE<-MINUTE[c(-1:-(forward+3)),]
MINUTE<-c(as.matrix(MINUTE))

day.no<-ncol(close)
ob.perday<-nrow(close)
ob.perday2<-nrow(close)-(3+forward)

HOUR<-rep(HOUR,day.no)
MINUTE<-rep(MINUTE,day.no)

#combine them into dataframe
dt<-data.frame(target,pseudo_log_return_lag_1,pseudo_log_return_lag_2,
pseudo_log_return_lag_3,STD_lag_1,STD_lag_2,STD_lag_3,HOUR,MINUTE)
day.no.train<-floor(day.no*2/3) #take 2/3 as training set

IN<-dt[1:(ob.perday2*day.no.train),]
OUT<-dt[(ob.perday2*day.no.train+1):(ob.perday2*day.no),]

library(h2o)
h2o.init(nthreads = -1)
h2o.no_progress()
h2oIN<-as.h2o(IN)
h2oOUT<-as.h2o(OUT)
y<-"target"

dl_fit<- h2o.deeplearning(y = y,training_frame = h2oIN,
hidden = hidden_nodes,activation=activation_func,
```

```
rho=0.9999,epsilon=1E-10)
dl_fit_perf <- h2o.performance(model = dl_fit, newdata = h2oOUT)

predict<-h2o.predict(dl_fit, h2oOUT)
da<-sum(predict*h2oOUT[,1]>0)*100/nrow(predict) #directional accuracy

predictreturn.train<-c(as.matrix(h2o.predict(dl_fit, h2oIN)))
actualreturn.train<-target[1:(ob.perday2*day.no.train)]
predicterror.train<-actualreturn.train-predictreturn.train

predictreturn.test<-c(as.matrix(predict)) #it's predicted return on test set
actualreturn.test<-target[(ob.perday2*day.no.train+1):(ob.perday2*day.no)]
predicterror.test<-actualreturn.test-predictreturn.test
```

Appendix B R Code of SVM Method

```
in2=IN
library(e1071)
svm_fit<-svm(target~.,data=in2)
summary(svm_fit)

testset=OUT[,-1]
#prediction
svm_pre<-predict(svm_fit,testset)

da_svm<-sum(svm_pre*OUT[,1]>0)*100/length(svm_pre)
predicterror.svm<-OUT[,1]-svm_pre
```

References

Arévalo, A., Niño, J., Hernández, G. and Sandoval, J. (2016). High-frequency trading strategy based on deep neural networks, *International Conference on Intelligent Computing*, Springer, pp. 424–436.