Trading Strategy with Time Series and Reinforcement Learning

Rutgers University, MSMF

Shao-Wen Lai

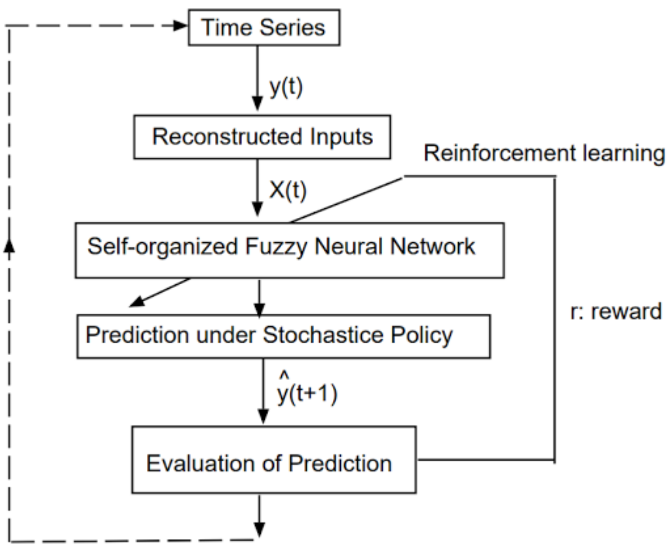
1. Introduction

With the trend of neural network, more and more people want to use deep learning with big data to predict the stock market. However, even if the accuracy of their model is good, those weights, ratios, and other combinations of numbers in neural networks are hard to be interpreted. So, it cannot help us understand the market. And models come from meaningless combinations can stand for long term and represent the market. Therefore, I want to use combination of different linear regression models on time series to simulate the neural networks. And take some useful skills in machine learning for time series to improve my model to predict the stock prices.

2. Literature Review

Forecasting Time Series by SOFNN with Reinforcement Learning

--by Takashi Kuremoto, Masanao Obayashi, and Kunikazu Kobayashi



Flow chart from the paper

In this paper, it only uses stock prices as inputs,

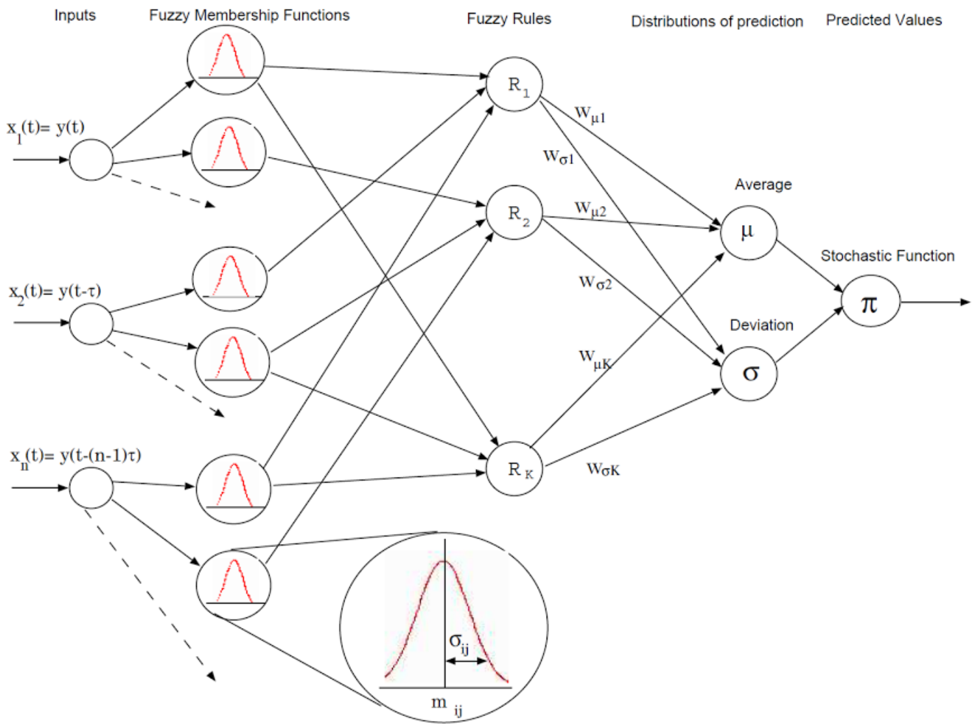
$$X(t) = (x_1(t), x_2(t), \cdots, x_n(t)) = (y(t), y(t-\tau), \cdots, y(t-(n-1)\tau)$$

$$x_i(t) := \textit{the stock price at time } (t-i+1)$$

$\mathrm{y(t)} := the\ stock\ price\ at\ time\ t$

$\hat{y}(t+1) := the\ predicted\ price\ at\ time\ t+1, with\ the\ distribution\ \pi \sim N(\mu, \sigma^2)$

, and use a simple neural network with kernel functions



Neural Network Architecture (SOFNN) from the paper

Where fuzzy membership functions are $\{B_{ij}(x_i(t))\}$

with $B_{ij}(x_i(t)) := exp\{-\frac{(x_i(t)-m_{ij})^2}{2\sigma_{ij}^2}\}$ is a kernel function

and $\mu(X(t), \omega_{\mu k}) = \frac{\sum_{k=1}^{K} \lambda_k w_{\mu k}}{\sum_{k=1}^{K} \lambda_k}$, $\sigma(X(t), \omega_{\sigma k}) = \frac{\sum_{k=1}^{K} \lambda_k w_{\sigma k}}{\sum_{k=1}^{K} \lambda_k}$

The creative skills are in later steps.

After getting predictions from this model, it takes the errors as time series of rewards and change the weights $\{w_{\mu k}\}$ and $\{w_{\sigma k}\}$, in every time t, according to the reward in time t.

Since stock price will go through kernel functions, so it is needed to use derivatives to know how much weight should be changed to modify the error, that is add or minus these rewards.

And in the process of computing rewards and modifying the error, the weights, $\{w_{\mu k}\}$ and $\{w_{\sigma k}\}$, would be improved to make better μ and σ, and then π would be more accurate. The method of updating the weights called stochastic gradient ascent(SGA). SGA is similar with the modified gradient descent applied on time series in the project, and it will be introduced in Methodology.

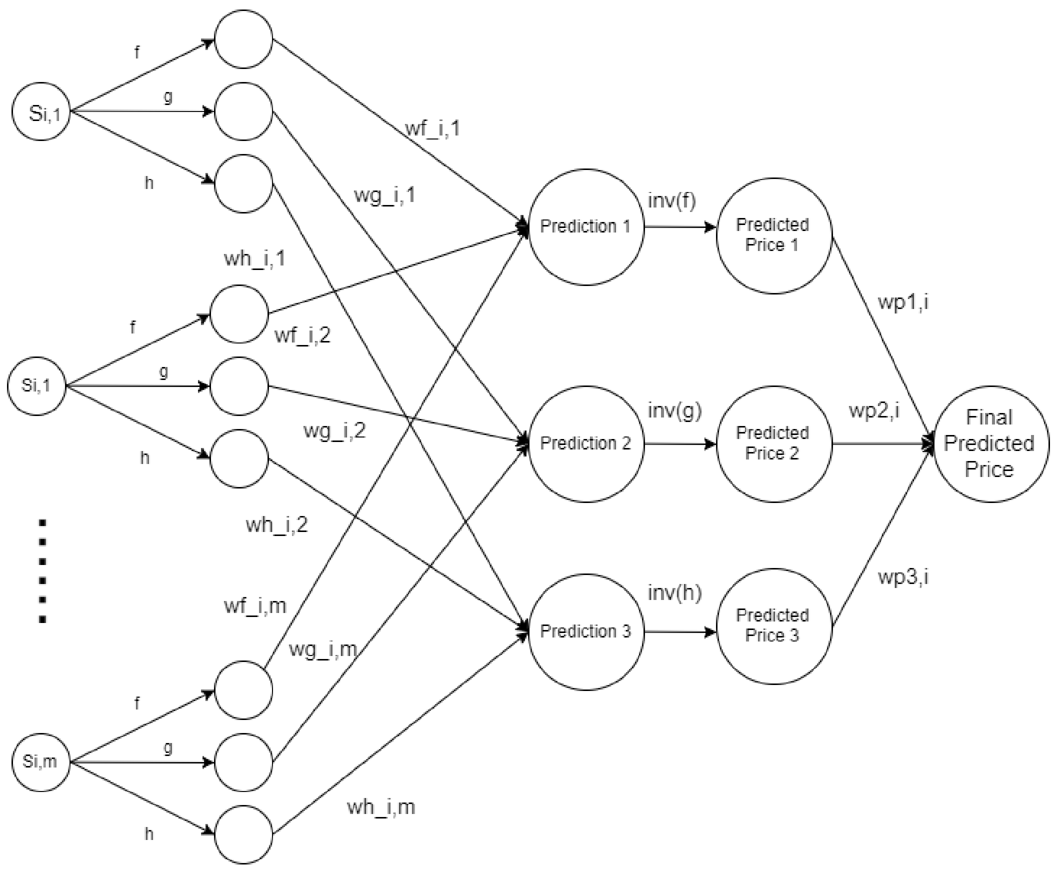
3. Methodology

To get the results, just follow "read.me" in the files, that is
Run GetData => run => run2 => run3 in MATLAB command window

The main purpose of the project is to

- **Simulate a 2-layer neural network**

  $S_{i,j} := the\ stock\ price\ of\ i^{th}\ stock\ on\ j^{th}\ day$



Neural Network Architecture of the project

In the figure f, g, h, inv(f), inv(g), inv(h) are given functions. The other values are the weights. That is, Prediction1, Prediction1, Prediction1, and Final Predicted Price are the weighted sum of the values in the previous layer. To make it clear, this figure is not the same as model of the project. In this project, these functions may take not only one stock price as their variables. However, after adjusting the time interval of the universe, there will be still the same number of {Preditction1}, {Preditction2}, and{Preditction3} for later process.

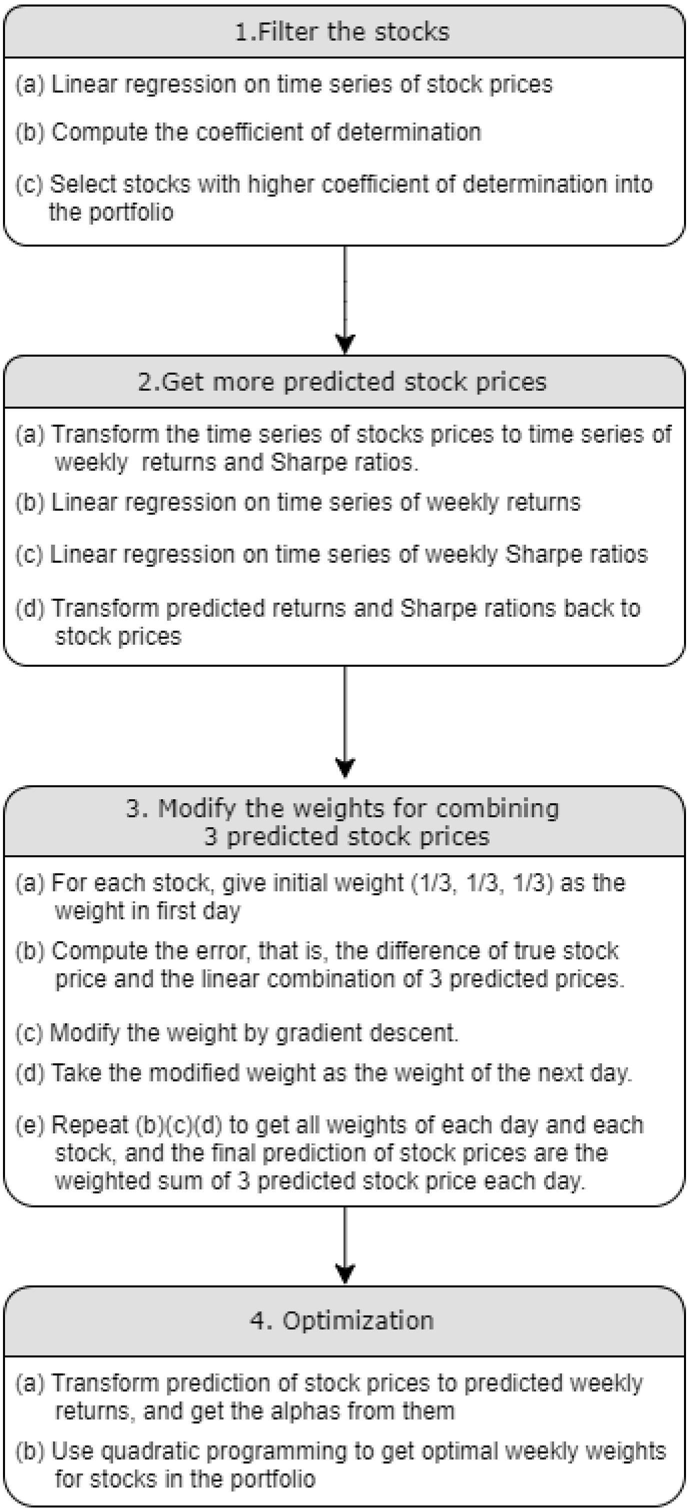
The neural network is to train the data the update the weights in every batch for getting the optimal weights and the optimal predictions.
Here, in the simulated neural networks, the set of $\{wf_{i,k}\}, \{wg_{i,k}\}$, and $\{wh_{i,k}\}$ are obtained by the linear regression in rolling windows
And $\{wp1_{i,}\}, \{wp2_i\}$, and $\{wp3_i\}$ are updated every day with following methods.

- **Update the weights in the second layer with the error of prior day with reinforcement learning skill, gradient descent in time series.**

  Since the prediction model of linear regression with rolling windows only apply the same set of coefficients "b" on predicting prices.
  ($b = (X'X)^{-1}X'Y$, $where\ X\ is\ a\ m\times n$ matrix from rolling windows,$Y\ is\ a\ m\times k$ matrix. So, the length of the coefficient is m, and N, the length of the data for regression, follows $\mathrm{N} \geq \mathrm{m} + \mathrm{n} - 1$)
  Even using rolling windows of data set (rolling N to get different b) to update the time interval of the data for linear regression. It only gets optimal coefficients in the time interval. For example, I regress the values of 5^{th} day from now against the values in past 50 days. Hence, N should be much larger than 50. But the trend of a stock may be very different from last month to this moth or even week to week. If I regress on a shorter time series (smaller m), the accuracy would decrease. And linear regression still gets the optimal b in past N days. It would not be useful with considering recent trend of stocks.
  Therefore, this part is to build a method can updates the coefficients by gradient descent every day for fitting the actual stock prices.

Flow chart of the methodology

There are 4 steps as the flow chart,



Steps in methodology

1)

The first step is using linear regression on time series of stock prices for predicting the stock price of next week (the 5th day from now), and then check the performance of prediction on each stock by the coefficient of determination. The stocks with higher coefficient of determination will be selected in the portfolio.

a) For i^{th} stock, $i = 1,2,\dots,200$, regress the stock prices of next week against the time series of the stock prices in past 50 days in rolling windows.

$$\text{Let } X_i = \begin{bmatrix} S_{i,1} & \cdots & S_{i,50} \\ S_{i,2} & \cdots & S_{i,51} \\ \vdots & \ddots & \vdots \\ S_{i,N_{train}-54} & \cdots & S_{i,N_{train}-5} \end{bmatrix}, Y_i = \begin{bmatrix} S_{i,55} \\ S_{i,56} \\ \vdots \\ S_{i,N_{train}} \end{bmatrix}$$

$where\ S_{i,j} := the\ stock\ price\ of\ i^{th}\ stock\ on\ j^{th}\ day$

$N_{train}\ is\ the\ length\ of\ training\ data = N - N_{test}$

$and\ N\ is\ the\ length\ of\ whole\ data, N_{test}\ is\ the\ length\ of\ test\ data$

$Then\ b_i = (X_i{}'X_i)^{-1}X_i{}'Y_i$,

And $PS_{i,j}$ is defined as the predicted stock price from timeseries of stock price, with $PS_{i,j} := [S_{i,j-54} \quad \dots \quad S_{i,j-5}] \times b_i$

b) Compared with using mean value directly, prediction by linear regression is a good fit for the stock prices in the period which we trained. Therefore, for each stock, there are 2 additional condition to check if the linear regression on time series is a suitable model to predict price.

- Divide the data into training data and test data. Then use b_i, got from training data, to get predicted stock prices on test data. And then get the coefficient of determination on test data to check if the performance of linear model is also good in the test data.

$$\begin{bmatrix} S_{i,N_{train}-53} & \cdots & S_{i,N_{train}-4} \\ S_{i,N_{train}-52} & \cdots & S_{i,N_{train}-3} \\ \vdots & \ddots & \vdots \\ S_{i,N-54} & \cdots & S_{i,N-5} \end{bmatrix} \begin{bmatrix} S_{i,N_{train+1}} \\ S_{i,N_{train+2}} \\ \vdots \\ S_{i,N} \end{bmatrix}$$

In the test data, we have similar matrices as training data, but we don't use linear regression. We use b_i directly to compute coefficient of determination.

- Since using mean value is a bad prediction for predicting stock prices, the coefficient of determination R^2, which is an index to compare the prediction with mean value, would be very closed to 1.

$$where\ R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

$$SS_{res} = \sum_{j=1}^{N_{test}} (S_{i,N_{train}+j} - PS_{i,N_{train}+j})^2$$

$$= \sum_{j=1}^{N_{test}} (S_{i,N_{train}+j} - [S_{i,N_{train}+j-54} \quad \cdots \quad S_{i,N_{train}+j-5}] \times b_i)^2$$

$$SS_{tot} = \sum_{j=1}^{N_{test}} (S_{i,N_{train}+j} - \mu_i)^2$$

$$\mu_i := average\ of\ \{S_{i,N_{train}+1}, S_{i,N_{train}+2}, \dots, S_{i,N}\}$$

Therefore, I invent a new coefficient of determination R_{pre}^2

with $R_{pre}^2 := 1 - \frac{SS_{res}}{SS_{pre}}$

$$SS_{pre} := \sum_{j=1}^{N_{test}} (S_{i,N_{train}+j} - S_{i,N_{train}+j-5} \times \mu_{return,i})^2$$

$$\mu_{return,i} := average\ of\ 5day\ return\ of\ i^{th}\ stock\ with\ rolling\ window$$

By the formula, SS_{res} is the L2-loss of $\{PS_{i,j}\}$, SS_{tot} is the L2-loss of $\{\mu_i\}$, and SS_{pre} is the L2-loss of $\{S_{i,j} \times \mu_{return,i}\}$. Hence, R^2 is the coefficient to compare the loss of our prediction $\{PS_{i,j}\}$ with the loss of using mean value of the targets as the prediction. And R_{pre}^2 is the coefficient to compare our prediction $\{PS_{i,j}\}$ with using today's stock price multiplied by the average return as the prediction. It is an intuitive and better prediction than using mean value of the target prices. More detailed explanations are in the "Empirical Session".

c) Without loss of generality, I can choose the stocks with higher SS_{pre}

into the portfolio. And we can say that they are more suitable for using linear regression to predict stock prices. Now, the stocks in portfolio has been selected. In the project, there are 46 stocks are selected. In the following parts, index would be reordered to the index in the portfolio.

$$S_{i,j} := the\ stock\ price\ of\ i^{th}\ stock\ of\ the\ proflio\ on\ j^{th}\ day$$

And N becomes N_{train} in the following steps.

2)

The second step is to get predicted stock prices for each stock every day. After filtering the stock, all stocks in the portfolio have their actual stock prices $\{S_{i,j}\}$, and their predicted prices $\{PS_{i,j}\}$, which is predicted from stock price. There will be 2 more predicted prices set from weekly returns and Sharpe ratios, in following steps.

a) Here we get weekly returns $\{R_{i,j}\}$ in rolling window

where $R_{i,j} = \frac{S_{i,j}}{S_{i,j-5}} - 1$

and get weekly Sharpe ratio $\{SR_{i,j}\}$ in rolling windows

where $SR_{i,j} = \frac{R_{i,j} - rf_j}{\sigma_i}$

and $rf_j :=$ *the risk-free rate on* j^{th} *day*

$\sigma_i := the\ variance\ of\ weekly\ returns\ of\ i^{th}\ stock$

The covariance matrix of weekly returns of stocks is obtained with weekly returns (without rolling window)

Since $R_{i,j} = \frac{S_{i,j}}{S_{i,j-5}} - 1$, returns are undefined when j<=5. Because of it and similar situations happened in later steps, in the following parts, the start day of the time series would become some days later. Therefore,

$new\ S_{i,j} := orignal\ S_{i,j+20}$

$R_{i,j}$, $SR_{i,j}$ are computed from $new\ S_{i,j}$

b) Let $\mathrm{XR}_i := \begin{bmatrix} R_{i,1} & \cdots & R_{i,50} \\ R_{i,2} & \cdots & R_{i,51} \\ \vdots & \ddots & \vdots \\ R_{i,N-54} & \cdots & R_{i,N-5} \end{bmatrix}$, $\mathrm{YR}_i := \begin{bmatrix} R_{i,55} \\ R_{i,56} \\ \vdots \\ R_{i,N} \end{bmatrix}$

And $\mathrm{XR} := \begin{bmatrix} XR_1 \\ XR_2 \\ \vdots \\ XR_{np} \end{bmatrix}$, $\mathrm{YR} := \begin{bmatrix} YR_1 \\ YR_2 \\ \vdots \\ YR_{np} \end{bmatrix}$

where np $:= number\ of\ stocks\ in\ portfolio$

Regress YR against XR, and get $\mathrm{bR} = \left(XR^{'}XR\right)^{-1}XR^{'}YR$

Then we get the set of predicted returns $\{pR_{i,j}\}$ with

$$pR_{i,j} := [R_{i,j-54} \quad \dots \quad R_{i,j-5}]\times bR$$

Here, all returns use the same coefficient bR obtained from linear regression on all stocks in portfolio. Since returns are ratios of prices, the relationship in time series between different stocks should more similar than prices. And generally, we would compare the returns of different stocks instead of the difference of stock prices in a period.

c) Similarly, for the same reason, use the same way to get the set of predicted Sharpe ratios $\{pSR_{i,j}\}$ with the set of Sharpe ratio $\{SR_{i,j}\}$

d) Transform set of predicted returns $\{pR_{i,j}\}$ and predicted of Sharpe ratio $\{pSR_{i,j}\}$ to sets of predicted stock prices $\{PR_{i,j}\}\ and\ \{PSR_{i,j}\}$.
Where $PR_{i,j} := S_{i,j-5}\times pR_{i,j}$ and $PSR_{i,j} := S_{i,j-5}\times(pSR_{i,j}\times\sigma_i + rf_j)$
Now we have 3 different sets of predicted stock prices

$\{PS_{i,j}\}, \{PR_{i,j}\}, \{PSR_{i,j}\}$ to predict actual stock prices $\{S_{i,j}\}$

In the later step, the final version of the set of predicted prices would be the linear combination of them.

3)

In the third step, for the i^{th} stock, the predicted price on j^{th} day would be a linear combination (or say the weighted sum) of $PS_{i,j}$, $PR_{i,j}$, and $PSR_{i,j}$
And the weight would be decided by the prediction, weight, and the actual stock price of the previous day. Since weight can be updated without using future data, if the accuracy of the prediction is improved with the method, it would be a practical way to use in the real world. Therefore, it has to show if the new prediction is better than the any of 3 sets of predicted prices we have obtained in the empirical session.

a) For each stock, let the initial weight $[\frac{1}{3}\ \frac{1}{3}\ \frac{1}{3}]'$ be the weight on the first day.

b) Compute the error of the weighted sum and the actual stock price. Define the error function F for gradient descent.

Let $F\left(weight_{i,j}\right)$ be th function of predtiction error, then

$$F\left(weight_{i,j}\right) := P_{i,j}^{com} - S_{i,j}$$
$$= [\,PS_{i,j}, PR_{i,j}, PSR_{i,j}]\times weight_{i,j} - S_{i,j}$$

$$weight_{i,j} := \textit{the weight of linear combination for } i^{th} \textit{ stock on } j^{th} \textit{ day},$$
$$\textit{and it is an } 3\times 1 \textit{ matrix}$$

$$P_{i,j}^{com} := \textit{the weighted sum of } 3 \textit{ predicted stock prices}, PS_{i,j}, PR_{i,j}, PSR_{i,j}$$
$$= [\,PS_{i,j}, PR_{i,j}, PSR_{i,j}]\times weight_{i,j}$$

c-e) Use gradient descent to obtain the weight on $(j+1)^{th}$ day. And the difference of weights on $(j+1)^{th}$ and j^{th} day is to make $P_{i,j}^{com}$ closed to $S_{i,j}$, the actual stock price. And we hope it would also make $P_{i,j+1}^{com}$ a better prediction than the weighted sum without updating weights.

Since it is a linear combination, it is convenient to get the gradient.

$$\nabla F\left(weight_{i,j}\right) = [PS_{i,j}, PR_{i,j}, PSR_{i,j}]$$

$$weight_{i,j+1} = weight_{i,j} - \mathrm{sign}(P_{i,j}^{com} - S_{i,j})\times \alpha \times \nabla F\left(weight_{i,j}\right)$$

α is the learning rate, and it should be tried with different value to get the optimal mean L1-loss $= \frac{1}{N}\times\sum_{j=1}^{N}|R_{i,j}^{com} - R_{i,j}\,|$[1],

where $R_{i,j}^{com} = \frac{P_{i,j}^{com}}{P_{i,j-5}^{com}} - 1$

In the process, $weight_{i,j+1}$ is modified from $weight_{i,j}$ in the direction of making $P_{i,j}^{com}$ closed to $S_{i,j}$. This way works without loss of generality.

Then the final version of the set of predicted prices $\{P_{i,j}^{com}\}$ is obtained.

Example of weights for summing up the predictions

For stock CBSH, the weights in first 16 days are

0.333	0.331	0.333	0.331	0.333	0.336	0.333	0.335	0.338	0.340	0.342	0.344	0.342	0.340	0.338	0.335
0.333	0.331	0.333	0.331	0.333	0.335	0.333	0.335	0.337	0.339	0.341	0.343	0.341	0.339	0.337	0.335
0.333	0.331	0.333	0.331	0.333	0.335	0.333	0.335	0.337	0.339	0.341	0.343	0.341	0.339	0.337	0.335

These weights move slowly, so the weights in the early days of the data set are supposed to be inaccurate.

The weights in 100th to 115th day are

0.335	0.337	0.340	0.342	0.340	0.337	0.335	0.333	0.330	0.328	0.330	0.333	0.335	0.333	0.330	0.328
0.333	0.335	0.338	0.340	0.338	0.335	0.333	0.331	0.328	0.326	0.328	0.331	0.333	0.331	0.328	0.326
0.333	0.335	0.338	0.340	0.338	0.335	0.333	0.331	0.328	0.326	0.328	0.330	0.333	0.330	0.328	0.326

We can find that the weights of PS are higher than others, and such differences exist until to the end. It seems that the weights move in a correct direction. And it has showed by the accuracy of the final prediction (weighted sum of 3 prediction) increased after getting rid of the first 100 daily final prediction. (showed in Empirical Session)

4)

Finally, the last step is the optimization.

a) Convert predicted prices $\{P_{i,j}^{com}\}$ to predicted weekly returns $\{pR_{i,wj}^{com}\}$

With $pR_{i,wj}^{com} = \frac{P_{i,wj\times 5}^{com}}{P_{i,wj\times 5-4}^{com}}$

b) On wj^{th} week, we have predicted excess return

$$\alpha_{wj} := \begin{bmatrix} pR^{com}_{1,wj} \\ pR^{com}_{2,wj} \\ \vdots \\ pR^{com}_{np,wj} \end{bmatrix} - \begin{bmatrix} \mu w_1 \\ \mu w_2 \\ \vdots \\ \mu w_{np} \end{bmatrix}$$

, where μw_i is the average weekly return of i^{th} stock.

In addition, $w_{A,wj}$ and $w_{A,wj}$ are np×1 matrices, where $w_{A,wj}$ is the active weight we are finding, $w_{B,wj}$ is the bench mark weight (by proportion of market capital in the portfolio).

Σ is the np×np covariance matrix of stock returns, and β is the np×1 matrix of betas of stocks.

$$\max_{w_A} \; w'_{A,wj}\alpha_{wj} - \frac{\lambda}{2} w'_{A,wj}\Sigma w_{A,wj}$$

$$\Rightarrow \min_{w_{A,wj}} \; -w'_{A,wj}\alpha_{wj} + \frac{\lambda}{2} w'_{A,wj}\Sigma w_{A,wj}$$

with constraints
$$\begin{gathered} -0.3 \le w'_A\beta \le 0.3 \\ -w_{B,wj} \le w_{A,wj} \le 1 - w_{B,wj} \\ w'_{A,wj}1 = 0 \end{gathered}$$

The first term is to make activate investment can get more access return by the alpha predicted in the model. The latter term is to ensure active risk would not be too high when we pursue higher access return. Therefore, the lambda here should be tried with different value to a get suitable value for the portfolio and the model[2]. These constraints are to limit the beta, and make sure there is no short and beta bet.

β can be computed by returns of stocks in portfolio with the benchmark returns (calculate by the stocks' returns with market capital weighted).

After optimization, the whole weights of stocks of portfolio $W_P := W_A + W_B$ is obtained by the weekly active and bench mark weights ($W_A = [w_{A,1}\ w_{A,2}\ w_{A,3} \ldots.]$, $W_B = [w_{B,1}\ w_{B,2}\ w_{B,3} \ldots.]$). The trading strategy is completed.

4. Data

Same as the paper inspired me, I only use the stock prices to predict stock prices and alpha. In the first step, I get the stock prices of Top 200 stocks in NASDAQ in the period 31/12/2008 to 31/12/2016. And all the stock prices are retrieved from Yahoo Finance with the API in MATLAB.

Since I want to show that the strategy can be applied on different universe after my filter, the data of stock prices are not saved, and it would take some time to retrieve data in “GetData.mat”.

After the first filter, there are 46 stocks have been selected. And for testing and improving accuracy, some data in the beginning and the end of the period has to be abandoned.

For example, I give up 100 days data in step3. Because the learning rate is small, and the initial data is given intuitively (mean value of different predicted value is usually better than each of them), the weights need some time to converge to the better values.

	mean	STD	min	max
MSFT	31.71487	11.09298	15.44419	59.26732
AMZN	305.4595	182.4409	73.6	844.36
NVDA	18.76827	11.26848	7.787691	71.77251
TXN	35.09213	13.54153	13.74392	69.43372
MDLZ	27.43853	9.495235	11.94942	45.32643
ADI	39.13249	12.48692	15.26203	64.49027
LRCX	51.38313	18.19862	22.52643	100.489
AAL	20.90747	15.3581	1.973181	54.31586
MCHP	33.969	10.1513	14.45877	61.34662
WLTW	97.02566	18.17544	57.48243	130.7977
MYL	33.71855	15.57392	12.1	76.06
SBAC	70.66005	32.48693	22.2	128.01
MXIM	24.18294	7.231275	9.839958	39.69069
MELI	90.22229	34.47454	19.72838	190.562
ACGL	46.5328	17.38918	18.38667	83.15
HAS	44.32045	17.60829	17.67942	84.22901
SNPS	34.68753	10.44979	18.2	60.56
AMD	4.893758	2.247129	1.62	10.16
RYAAY	44.36844	20.56426	19.17966	87.64
ASML	60.21291	32.3938	13.61188	111.3574
SHY	81.86581	1.447705	77.79339	84.29572

JBHT	57.35327	19.57358	23.37973	89.43139
TTWO	19.02788	9.508263	7.52	46.34
CDNS	13.69034	5.647197	4.59	26.25
QVCB	18.41137	6.915321	3.878069	31.4
VRSN	46.73896	20.21598	15.22528	93.12
EXPD	40.00323	5.904301	26.64289	51.55796
TRMB	23.28326	6.970922	9.25	39.96
GRMN	32.46389	9.11254	13.50025	52.49079
CGNX	12.40318	6.765853	2.896751	27.17946
MRVL	12.58512	2.813313	6.506254	20.21112
SGEN	28.0673	12.97981	8.19	57.25
UHAL	176.0112	118.3107	28.44403	431.4254
IONS	25.58668	18.61745	6.47	77.08
SBNY	83.73813	39.52471	24.98	160.73
NDSN	56.19683	19.72497	16.01019	100.9993
ERIE	61.70572	19.29414	24.47529	102.006
LAMR	36.10596	13.22581	12.16004	65.10173
COHR	54.67894	19.18976	17.25146	113.37
ICLR	39.27683	20.16632	14.83	85.04
LECO	45.11188	16.57304	14.32194	70.75474
ESLT	52.43271	18.56687	26.12317	100.5177
PBCT	11.60999	1.940888	7.960409	15.56285
FIZZ	18.39187	11.8753	5.997961	60.09102
OZRK	21.61082	13.24042	4.352711	53.0877
CBSH	32.88676	7.381951	18.64336	49.79721

There are 46 stocks in 1892 days totally.

And after the last abandon, and transformation. The universe becomes 46 stocks in 358 weeks.

5. Empirical Session

The key point of my alpha model is the skill to combine different predicted prices to get a better prediction. And take the advantage of the change from prices to different ratio to make prices data can be used in many ways.

Before the combination, mean L1-loss of predicted returns from $\{PS_{i,j}\}, \{PR_{i,j}\}$, and $\{PSR_{i,j}\}$ (the predicted prices by prices, returns and Sharpe ratios) are 0.0432, 0.0313, and 0.0320

After the combination, mean L1-loss of predicted returns $R_{i,j}^{com}$ is 0.0277, smaller than all mean L1-loss before. And after the abandon of returns in first 100 days, it decreases to 0.0268.

Another strength of my model is that I replace R^2 with my new R_{pre}^2. Since the in the 8 years of the period I chose, the stock price increased a lot. Most of mean stock prices in past 50 days are much lower than the price of 5 days after. So, it is obvious that today's price is a much better prediction of the stock price 5 days later. And today's price multiplied by average weekly return is even better. Hence, in my first step I invent a new coefficient of determination:

$$R_{pre}^2 := 1 - \frac{SS_{res}}{SS_{pre}}$$

$$SS_{pre} := \sum_{j=1}^{N_{test}} (S_{i,N_{train}+j} - S_{i,N_{train}+j-5} \times \mu_{return,i})^2$$

It is a coefficient to compare SS_{res}, L2 loss of our prediction, and SS_{pre}, L2-loss of using today's price multiplied by mean return as the prediction. By the formula, if $R_{pre}^2 > 0$, then $SS_{pre} > SS_{res}$, and the loss of prediction from average weekly return is larger than the loss of predictions from linear regression.

Hence, I select the stocks with $R_{\text{pre}}^2 > 0.05$ on the test data[3].

The last row of the diagram[3] is the mean of R^2, R_{pre}^2 on training data, and R_{pre}^2 on test data.

R^2	R_{pre}^2 on training data	R_{pre}^2 on test data
0.986825	0.056395	-0.11697

All R^2 are very closed to 1 because compared with mean value of target prices, the model is much more accurate. However, the purpose of the model is to find a better way to predict the stock prices, so it must be better than an intuitive method. In addition, no one would take the mean stock price in a long period as the prediction of the stock price next week. If the stock prices increase or decrease a lot in the time interval we choose, R^2 would be very close to 1, but it doesn't mean that the model works better in this stock than others. Hence, R^2 can't be regarded as a coefficient to check the effectiveness of the model in this problem.

Mean R^2_{pre}on training data is 0.056. It represents that the loss of the model is a little less than the intuitive method. The mean value of R^2_{pre} on test data is negative, but we can still find 46 stocks with $R^2_{\text{pre}} > 0.05$ on test data. Selecting the stocks suitable for linear regression is important for continuing to work on more linear method.

Although it is not a very good prediction, it has been proved that using linear regression is better than using average return to predict the stock prices with $R^2_{\text{pre}} > 0$ on the test data. Hence, the linear regression works on the stocks in my portfolio.

Interestingly, in step2, it is found that using regression on returns and Sharpe ratios can get better prediction (showed by mean L1-loss before), although I choose stocks by the R^2_{pre}. It presents that selected stocks are suitable to use linear regression on time series for prediction, instead of only having good performance on linear regression on time series of prices.

Final Results

TE = 0.0289

IR = 1.9068

IC = 0.2644

The result is surprising. The return, IR and IC of the portfolio are higher, than my expectation. And the portfolio can earn much more if there is no constraint of Tracking Error. When I knew that the L1-loss of return is about o.o27, I don't think is really good because if the prediction of a stock return is 1% this week, it may be negative with not low possibility. However, in the process of optimization of the portfolio. It can distribute more weights in 1 to 5 stocks of 46 stocks, and distribute negative weights in others. That makes my portfolio get access return even if my predictive alpha model doesn't have high accuracy.

It is a problem that the lambda is too high when I want to optimize the portfolio and control the tracking error. A possible cause may be the inaccurate beta. Although there is a constraint $-0.3 \leq w'_{A,j}\beta \leq 0.3$ in the quadratic programming, the inaccurate beta may make it not useful for controlling risk. 5-year beta and 3-year beta are popular for portfolio management. Beta in this project comes from the stock prices in 358 weeks, about 7 years. Hence, using rolling windows to get beta for such frequent investment may be a better way to control risks.

6. Summary

In this project, the application of reinforcement learning, in step3, is successful, and it is what I focus on. Using the today's error (or say reward and plus it, as the method in the paper) to modify the weight really improve the prediction of next week. And the way to select stock for the portfolio is helpful, too. Separating the time series to training data and test data for validation can find the stocks suitable for linear regression. And it has been proved when it also gets good performance by regression on other ratios instead of stock prices. Besides, compared with normal R^2, the modified R^2, R^2_{pre}, is better to check the effectiveness of the predictive model in this problem.

However, to find a really feasible investment strategy and prediction of alpha, I should make sure all my predictions are validated by the data outside of the period I trained (in stock market, it'd better use data later than the period we work on for validation), instead of using back test only. In the project, I use the same period for linear regression and develop the strategy, so it may not work in real world. Generally, there should not be a so good strategy developed by such limited data. I think the really feasible timeseries strategy only happen in very small time steps, like a second, with high frequency trading. Nonetheless, the step3 only use past data for prediction except the way to find best learning rate. In my viewpoint, the learning rate is not very variable for different time interval. When the scale of learning rate is represented in exponential function [4], it is not difficult to find a good learning rate (L1-loss < 0.0313, the smallest of three L1-loss before linear combination). For deep learning, the learning rate is also represented in $k \times 10^{-n}$, and n would be modified to $n \pm 1$ when the error can't converge to the tolerance. So, the step3 should be somehow useful in the real world.

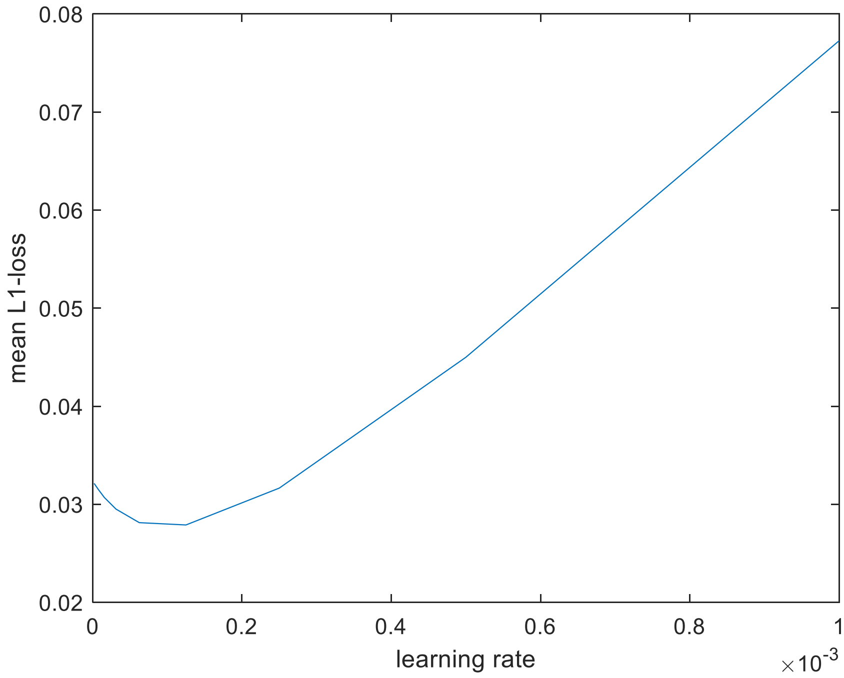
For time series model, if I continue working on linear model, I should not add data other than stock prices. Since the coefficient of them would be much different than parameter made from prices. It probably needs other ratios, multiplication, and even kernel functions to deal with them. My step3 may not still work.

For example, if I add time series of volume as new variables in the data, we can't apply linear regression on it directly. We know there is some relationship between stock prices and volume, but it would not be that the predicted stock prices can be modified by plus x% of volume. It should be more a complicated relationship. If I can't find the right function for the new variable. The predicted prices in step2 would be less meaningful and step 3 may not work.

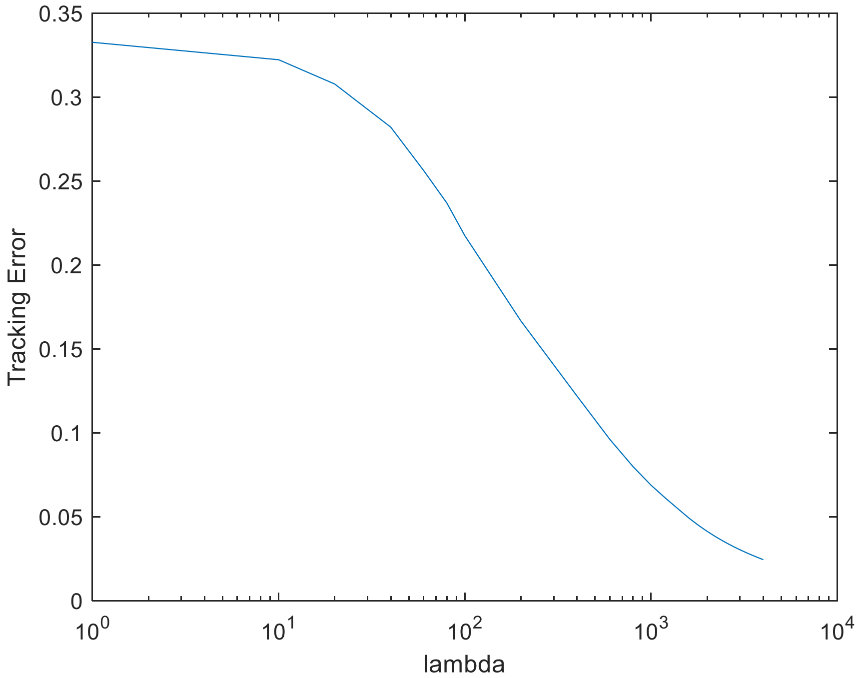
Therefore, I prefer adding more ratios made from stock prices to get more predicted prices. For example, $\frac{R_{i,j}-R_{M,j}}{\sigma_i}$, where $R_{M,j} := market\ return\ at\ j^{th}\ day$, is a good candidate, because it is a modified Sharpe ratio and maybe more related to alpha. Afterwards, in step3, there are more predicted prices, and more parameters of weights to control. I believe it would get better results.

7. Appendix

[1]



[2]



[3]

	R^2	R^2_{pre} on training data	R^2_{pre} on test data
AAPL	0.992848	0.031169	-0.02576
GOOGL	0.993108	0.086919	-0.1774
GOOG	0.992807	0.084755	-0.08607
MSFT	0.991104	0.067887	0.093632
AMZN	0.994047	0.06509	0.0789
CMCSA	0.996491	0.063019	-0.65975
INTC	0.986752	0.030827	-0.03758
CSCO	0.976564	0.044465	-0.03031
AMGN	0.993981	0.064292	-0.34806
CELG	0.993334	0.086183	-0.0471
GILD	0.994615	0.046365	0.019906
NVDA	0.991686	0.072687	0.17811
PCLN	0.992953	0.074306	-0.02981
WBA	0.993363	0.039559	-0.01408
TXN	0.993432	0.034132	0.108755
NFLX	0.988364	0.061427	-0.12356
SBUX	0.996749	0.095269	-0.42755
ADBE	0.993705	0.069811	0.031918
QCOM	0.977778	0.042438	-0.15759
COST	0.99646	0.03722	-0.28984
BIIB	0.991518	0.051294	0.046831
BIDU	0.983039	0.068395	-0.03968
MDLZ	0.993469	0.051723	0.125206
AMOV	0.955066	0.046487	-0.11441
AABA	0.990179	0.062111	-0.10673
QQQ	0.996009	0.088988	-0.02268
TMUS	0.983478	0.048015	-0.0705
ATVI	0.993699	0.036459	-0.01852
AMAT	0.98556	0.020525	-0.04448
FOX	0.992109	0.040883	-0.1958
ADP	0.996037	0.036934	-0.48977
CSX	0.984883	0.053686	-0.40202
REGN	0.994244	0.102085	-0.13653
CME	0.990951	0.034324	-0.37914
CTSH	0.988302	0.057891	-0.11845

EBAY	0.988072	0.044534	-0.42937
MAR	0.993097	0.051681	-0.05227
VRTX	0.978364	0.045601	-1.10333
ISRG	0.980591	0.043522	0.028364
MU	0.987481	0.061528	0.006591
EA	0.993969	0.052751	0.001372
ESRX	0.983745	0.022002	-0.21705
INTU	0.993722	0.04973	-0.55092
EQIX	0.994984	0.045126	0.002388
ALXN	0.990767	0.059624	-0.00686
FOXA	0.992834	0.050142	-0.27004
MNST	0.994402	0.04598	-0.04695
ILMN	0.98933	0.048493	-0.06713
ADI	0.989226	0.038151	0.116945
LRCX	0.986681	0.033343	0.08284
LBTYA	0.990673	0.075401	-0.02039
LBTYK	0.991603	0.077526	-0.10901
FISV	0.997629	0.097564	-0.46375
ADSK	0.978938	0.075666	-0.13147
INCY	0.990048	0.145062	0.001173
WDC	0.988272	0.052718	-0.12169
DISH	0.992216	0.042854	-0.89279
SIRI	0.994128	0.073984	0.021005
PCAR	0.979668	0.035953	-0.06744
ROST	0.996073	0.058772	0.007701
CTRP	0.980763	0.056305	-0.00311
CERN	0.992351	0.036027	-0.79534
AMTD	0.987949	0.036286	-0.12279
AAL	0.991298	0.078706	0.251116
EXPE	0.989215	0.065566	-0.09079
PAYX	0.994441	0.047816	-0.54195
NTES	0.992342	0.053741	0.003165
MCHP	0.987122	0.038213	0.253752
WLTW	0.98068	0.04141	0.082254
SYMC	0.972511	0.012087	0.046488
TROW	0.98444	0.052428	-0.0813
NTRS	0.978803	0.052276	-0.09007

DLTR	0.993523	0.054593	-0.0851
SWKS	0.992157	0.080512	-0.21899
FITB	0.983832	0.06382	-0.21493
CHKP	0.990839	0.071304	-0.19129
PFF	0.995403	0.094342	-0.00396
ORLY	0.997285	0.083177	-0.23946
MYL	0.987611	0.049349	0.15953
SBAC	0.994962	0.110562	0.228102
IBKR	0.991305	0.053692	-0.19985
DVY	0.996797	0.064561	-0.27537
XLNX	0.985447	0.028673	-0.08004
BMRN	0.989596	0.088508	-0.38069
VIA	0.987874	0.03765	-0.12257
KLAC	0.990633	0.030668	-0.15979
WYNN	0.985355	0.093016	-0.2087
ALGN	0.991659	0.051469	-0.43524
CTAS	0.99714	0.067857	-0.83333
CA	0.982389	0.042217	-0.19365
HBAN	0.985598	0.054782	-0.01395
ULTA	0.993769	0.044421	-0.06309
IDXX	0.991314	0.032938	-0.09768
XRAY	0.986387	0.077334	-0.54294
HSIC	0.995856	0.045355	-0.35848
MXIM	0.98558	0.046111	0.135659
MELI	0.978378	0.044912	0.155126
ACGL	0.996842	0.045438	0.106829
FAST	0.986071	0.04716	-0.08191
VOD	0.987324	0.042615	-0.95134
SHPG	0.989659	0.046033	-0.05394
NDAQ	0.995337	0.057388	-0.67204
CINF	0.996783	0.036486	-0.58786
EMB	0.993425	0.02898	-0.03019
HAS	0.993224	0.055435	0.251353
SNPS	0.992756	0.047696	0.086186
CSJ	0.997793	0.087166	-0.44934
CTXS	0.959488	0.04394	-0.24918
AMD	0.969835	0.054995	0.182114

RYAAY	0.992496	0.049318	0.103462
ASML	0.993883	0.069681	0.17573
ETFC	0.970142	0.082532	-0.35319
ANSS	0.988728	0.051008	-0.6924
SHY	0.995733	0.090376	0.060801
SNI	0.984778	0.066878	-0.02583
JBHT	0.992165	0.041901	0.179651
LKQ	0.991277	0.03019	-0.52515
MBB	0.997171	0.04249	-0.1107
NTAP	0.96094	0.028617	-0.29334
HOLX	0.987611	0.034769	-0.16589
TTWO	0.99188	0.056006	0.149261
CDNS	0.993318	0.087912	0.220636
QVCB	0.988666	0.088475	0.23212
QVCA	0.98962	0.054038	-0.17787
VRSN	0.9935	0.054283	0.234905
EXPD	0.952285	0.038167	0.056152
CHRW	0.952613	0.045286	-0.01592
TRMB	0.97811	0.054565	0.165838
GRMN	0.979681	0.020631	0.190363
IBB	0.994485	0.076935	0.02038
VIAB	0.989447	0.043132	-0.18541
IPGP	0.983254	0.045469	0.034794
CGNX	0.992027	0.054722	0.112165
STX	0.988741	0.055904	-0.02886
IAC	0.989704	0.037273	0.022849
DOX	0.994474	0.080163	-0.42336
JAZZ	0.993574	0.148738	-0.0549
SCZ	0.987473	0.046898	-0.0505
DISCB	0.973473	0.224398	-0.26464
CSGP	0.993352	0.04167	-0.11681
SIVB	0.987525	0.053139	-0.10244
SEIC	0.992641	0.076856	-0.64952
IEP	0.985284	0.050976	-0.38102
FLEX	0.981081	0.028366	-0.26156
MRVL	0.949665	0.025207	0.215464
ZION	0.955882	0.034816	-0.07539

OTEX	0.989916	0.039267	-0.04158
LULU	0.980509	0.048018	-0.67754
ODFL	0.995204	0.051019	-0.00328
TLT	0.988744	0.038158	0.02068
EWBC	0.990439	0.055412	-0.04675
GT	0.982702	0.037714	-0.03098
STLD	0.965623	0.035049	0.006991
ALKS	0.983971	0.08544	0.005214
DISCA	0.987047	0.038986	-0.05012
AKAM	0.976214	0.035776	-0.36222
EXEL	0.947228	0.049352	0.020648
JKHY	0.997581	0.060259	-0.87936
TSCO	0.994369	0.061535	-0.51658
IEI	0.995017	0.040836	0.041935
ACWI	0.988206	0.060217	-0.03818
SGEN	0.979449	0.045703	0.104105
IEF	0.993653	0.032643	0.041261
UHAL	0.996326	0.058442	0.095587
DISCK	0.987554	0.049211	0.000993
CPRT	0.990918	0.027426	0.043096
AGNC	0.989722	0.071827	-0.04613
FFIV	0.963921	0.024038	-0.01014
ALNY	0.983221	0.083232	-0.11856
QGEN	0.957274	0.027963	0.021485
CIU	0.998056	0.043389	-0.02976
PPC	0.990953	0.09445	-0.03181
IONS	0.980693	0.036735	0.083538
MIDD	0.994574	0.060497	-0.73885
ON	0.929552	0.030984	0.012784
ABMD	0.99228	0.064175	-0.10005
ARCC	0.991862	0.066967	-0.00499
MKTX	0.996975	0.061188	-1.54888
SBNY	0.994287	0.083636	0.216527
NDSN	0.988161	0.067638	0.278976
DXCM	0.993037	0.062953	-0.60558
PTC	0.985985	0.066489	0.017047
JBLU	0.991788	0.070681	-0.12466

ERIE	0.993786	0.054961	0.138226
LAMR	0.990111	0.066167	0.09653
COHR	0.983564	0.047608	0.123062
OLED	0.960953	0.042918	-0.14333
SRCL	0.986246	0.029547	-0.31518
ICLR	0.992023	0.047759	0.105326
LOGI	0.974826	0.023847	-0.05617
SHV	0.986574	0.176708	-0.00213
RGLD	0.945459	0.042518	-0.17375
LECO	0.99078	0.056235	0.053474
ESLT	0.990827	0.049522	0.185215
PBCT	0.977587	0.067022	0.168452
VEON	0.97339	0.043087	-0.25614
MSCC	0.978873	0.032969	-0.03106
FIZZ	0.992082	0.062387	0.201807
OZRK	0.994848	0.085994	0.108636
CBSH	0.986977	0.069297	0.149388
	0.986825	0.056395	-0.11697

Data in blue means stocks are selected into the portfolio (R^2_{pre} on test data>0.05)

[4]