QR take home project

Yuhui Jin

February 9, 2024

1 General data preparation

The source code is in Google Colab. Here is the link: Google Colab

In this section, we will explain the preparation of the data for the analysis, including the following steps:

- Download the data from the GitHub repository
- Unzip the data
- Define the extraction folder
- Extract the data
- fill NaN values with 0
- Convert the is in trading universe to 1 and 0

We notice that there are some missing values in the data. Thus, for the sake of training the model, we only use the data are available in all days in the dataset, i.e. size is 2013 per security. We generate a new dataset with 109 securities and 2013 days. The selection criteria is as follows: 1. The security has group_id with first two digits 50. 2. The security has 2013 days information in the risk_factors and securitiy_reference_data_w_ret1d. This file is named as data_2013_50_nn.csv.

Although all of securities are in the category 50, we computed the correlation matrix for the securities, where each security's information is 2013 days ret_1d. We noticed that the mean for the correlation matrix is 0.3. Therefore, those securities are diverse enough for us to generate a balanced portfolio.

2 Data exploration

We start with a simple linear regression model. Then, a xgboost is employed. Notice that these two models did not use our data_2013_50_nn.csv file. Instead, we used all the 2019 securities with all 2013 days information in the risk_factors and securitiy_reference_data_w_ret1d. Also, we do not generate a portfolio. We use a feature selection method based on the regreession model. For linear regression, we used the lasso regression to find important features. For xgboost, we used the feature importance to find important features. One interesting thing is that xgboost always selects different features from the linear regression. And some time, the xgboost selections diverge.

The criteria for model selection is NOT the MSE loss score, but a simple sharpe ratio. Here, it is not a weighted portfolio or even based on date. It is as simple as our predicted return times the actual price, and use it as the actual daily return.

Although both methods are crude and straightforward, they are effective. We trained both models 200 times, with relative small parameters, and find the train/test Sharpe around 0.5, with std 0.03. The result after feature selections is better than the original ones by around 0.1. The reason that we have small training parameters is that the Xgboost overfits rapidly: When the n_estimators is greater than 5 or the max depth is greater than 5, the test Sharpe is around 0.

These results show that the Xgboost/linear regression is not the best model: 1. it can not predict the future events while, on a reasonably level, learn the training data. 2. The feature selection indicates that Xgboost does not use the security_id, i.e. the actual feature that distinguish each security. 3. The Xgboost overfits quickly.

Item	Total Return	Volatility	Sharpe	Max Drawdown	Positive Day	Sharpe (no trading cost)
Train	214.58	0.994	2.145	-13.10	0.5563	2.30
Valid	4.744	1.080	0.651	-7.285	0.504	0.797
Test	40.10	1.26	1.759	-10.35	0.491	1.885

Table 1: Portfolio result.

3 Neural Networks

The actual portfolio is constructed based on a neural network model. We construct two general models: one is a 3-layers fully connected network, and the other is a Lstm network. Similar to the Xgboost/linear regression, we use MSE as the loss function. And we use the predicted returns as the basis of the portfolio construction. Intuitively, if our predicted returns are high, then we have higher tendency to invest. Thus, to balance the portfolio, for one day t, we use all the predicted returns, normalized by their positive sum or negative sum, i.e. now the investment is balanced. After taken care of the trading cost, we compute the actual Sharpe and generate a cumulative return plot. The plot is shown below. All the features for this portfolio is listed in the notebook.

Data preparation:

- we use a time period of past 15 days for one datapoint.
- we have the ret_id row multiplied by 100. (to make easier to analyze all MSE loss: it's easier to read and understand 0.25 than 0.0025)
- we drop the closed price column.(duplicated data with ret_id)
- we drop the group_id column.(we already used data from 50 group.)
- Train is the first 80 percent of the data.
- Valid is the second 6 percent of the data.
- Test is the last 14 percent of the data.
- to avoid data leakage, we do not shuffle any data. This also makes it easier to compute all returns.
- we notice that training always fails with the 109 securities and 2013 days. We need to reduce the number of securities to 20.

Model selections of neural network:

- Due to time limit, we only tried two models: fully connected network and lstm network.
- The fully connected network does not perform well: a 2 layer network is too simple to learn the training data; a 3 layer network some time runs to the problem of vanishing gradient.
- The lstm network: we use a 3 layer lstm network and then connected a 3 layer fully connected network. All these networks have small hyperparameters to avoid overfitting. The connection part: we only used the last 2 or 4 layers generated by the lstm network. Instead of dropout, we used a batch normalization to avoid overfitting. When we used dropout, the training for the model always failed, i.e., the model's output is one fixed value. This maybe due to the training saturation caused by the activation function. For activation, we used relu, tahn, and sigmoid. We only did crude hyperparameter tuning. Although the actual loss function is MSE, we still used a sharpe ratio to help with final model selection. To avoid gradient explosion, which happens when we use lstm, we use a gradient clipping. For optim, we do not see a significant difference in terms of SGD, Adam, etc. In the model, we used security_id from 32 to 51th in the 109 securities.

Figure 1: Cumulative returns of training data (Orange curve is the one without trading cost).

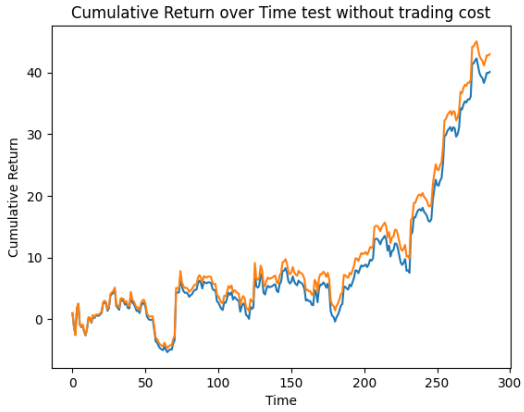

Figure 2: Cumulative returns of testing data (Orange curve is the one without trading cost).

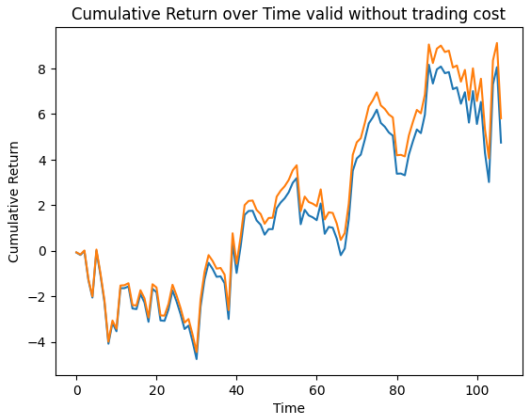

Figure 3: Cumulative returns of validing data (Orange curve is the one without trading cost).