Trading Strategies that maximize return, sortino ration and minimize drawdown

Jie Bao

1. Introduction

For this project, we need to build a strategy to maximize return, sortino ratio and minimize drawdown in the meantime. Also, we have to implement some Machine Learning approach. We use Python to accomplish this project.

Generally speaking, we are going to trade on the first day of each month. Then we will calculate sortino ratio and drawdown for each month, using Machine Learning approach to predict the Treasury bond future price on the first day of next month in order to compute return of that month. Next, combining those three value to two numbers for every month, which are increment if long and increment if short. Then construct a binomial tree, or an oriented graph and use dynamic programming to find the largest path.

2. Data Selection

We choose front month US Treasury bond future as our trading instrument. For convenience, in this project we only select data from 2010-01-01 to 2018-12-31. Also we get the US Treasury bond yield data within the sample time interval, which we are going to use in the Machine Learning part. All the data are downloaded from quandl free resources.

Here is a screenshot of part of the data:

	Open	High	Low	Last	Change	Settle	Volume	Previous Day Open Interest
Date								
2010-01-04	115.03125	115.59375	115.03125	115.09375	NaN	115.09375	171137.0	667206.0
2010-01-05	115.00000	116.03125	115.00000	116.03125	NaN	116.03125	204871.0	658779.0
2010-01-06	115.71875	115.84375	115.31250	115.31250	NaN	115.31250	262691.0	659893.0
2010-01-07	114.96875	115.37500	114.96875	115.28125	NaN	115.28125	244454.0	651459.0
2010-01-08	115.03125	115.46875	115.00000	115.46875	NaN	115.46875	287775.0	666037.0
2010-01-11	115.25000	115.25000	115.09375	115.18750	NaN	115.18750	187928.0	652872.0
2010-01-12	116.03125	116.81250	116.03125	116.81250	NaN	116.81250	285992.0	656947.0
2010-01-13	116.50000	116.59375	115.93750	115.93750	NaN	115.93750	234650.0	653606.0

Table 1 US Treasury bond future price

	30 YR
Date	
2010-01-04	4.65
2010-01-05	4.59
2010-01-06	4.70
2010-01-07	4.69
2010-01-08	4.70
2010-01-11	4.74
2010-01-12	4.62
2010-01-13	4.71

Table 2 30yr US Treasury bond yield

3. Our approach

We have four major steps in our project, first, process the downloaded data; next, prepare all the input we need for the Machine Learning part; then, do the Machine Learning to predict the future price; finally, apply dynamic programming to find the optimized strategy.

3.1 Process the downloaded data

After we get the US Treasury bond future and bond yield data from quandl, we pick the data from 2010-01-01 to 2018-12-31 from the raw data as our research data.

3.2 Prepare data for Machine Learning

The input of our Machine Learning approach will be historical price, volume, treasury bond yield. The output should be future price, return, sortino ratio and drawdown. So here we need to compute drawdown, sortino ratio and return.

3.2.1 Compute drawdown

By definition, a drawdown is a peak-to-trough decline during a specific period for an investment, trading account, or fund.

Its formula as follows,

$$D(T) = \max\left\{0, \max_{t\in(0,T)} X(t) - X(T)\right\}$$

where X(t) is a stochastic process, here it's the price of the US Treasury bond future.

3.2.2 Compute Sortino ratio

The Sortino ratio takes an asset or portfolio's return and subtracts the risk-free rate, and then divides that amount by the asset's downside deviation. Downside deviation is a measure of downside risk that focuses on returns that fall below a minimum threshold or minimum acceptable return (MAR).

Its formula as follows,

$$SR = \frac{R_p - r_f}{\sigma_d}$$

where R_p is the return, r_f is the risk-free rate, σ_d is the downside deviation.

3.2.3 Compute return

This is simple, which is just the ending value minus the beginning value.

$$R = P_{end} - P_{start}$$

3.2.4 Process the input data

Since there are still some "nan" in our input data, we deal with that in different ways. For price and yield, we choose to fill using its previous value. While for sortino ratio, drawdown and return, we fill using 0. Then, everything is ready for Machine Learning.

3.3 Machine Learning

In order to improve the accuracy and effectiveness of our strategy, we need to know what the price will be in the future so that we can decide what action we should take. So here we use Machine Learning approach to do the prediction.

We implement neural network to do the prediction. Using treasury bond future price, volume and bond yield of last month to predict the treasury bond future price, return, sortino ratio and drawdown on the first day of next month.

$$f(P_{hist}, Vol_{hist}, Yield_{hist}) \xrightarrow{predict} (P_{fut}, ret_{fut}, SR_{fut}, DD_{fut})$$

We choose 70% data as our training data and the rest of 30% as test data.

Here we only show a comparison of predicted Treasury bond future price and real price:

	pred	real
0	162.968984	161.93750
1	166.910685	165.18750
2	173.195776	174.34375
3	172.074503	172.68750
4	169.080890	171.93750
5	167.785376	167.96875
6	160.448696	162.78125
7	155.778627	150.65625

Table 3 Predicted value and real value

We can see that our predicted result are close to the real value.

3.4 Build the strategy

After we have the predicted value, we can use that to construct our strategy. First, we decide to trade on the first day of each month. Second, we must long or short on each trading day. Third, we have $200,000 at the beginning and each trade we will long or short 1 contract, or 100 shares.

However, we only have one goal for our strategy, which is to maximize return, sortino ratio and minimize drawdown. So our target function will be:

$$T = Max \sum_i (ret_i + SR_i - DD_i)$$

So we can compute $ret_i + SR_i - DD_i$ at the first day of each month:

if we take long position, $\Delta T_l = ret_i + SR_i - DD_i$

if we take short position, $\Delta T_s = -ret_i + SR_i - DD_i$

We can build a binomial tree, each node will represent a trading day. We assign each node with a value long_short_increment

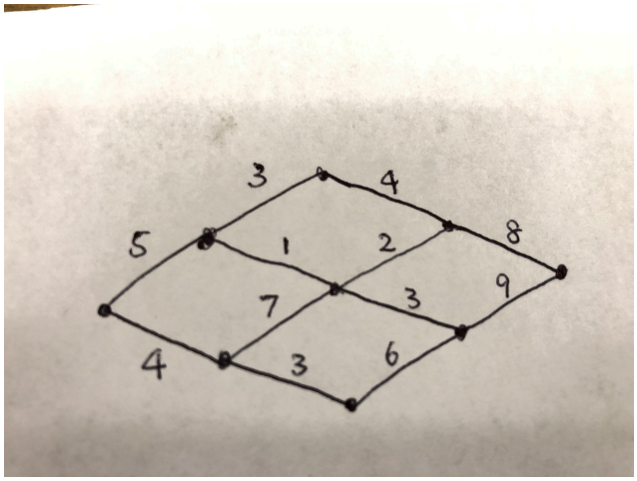
long_short_increment=$[\Delta T_l, \Delta T_s]$

and to create the value_mat:

	0	**1**	**2**	**3**
0	0	0	0	0
1	0	0	[0, 4]	0
2	0	[3, 1]	0	[0, 8]
3	[5, 4]	0	[2, 3]	0
4	0	[7, 3]	0	[9, 0]
5	0	0	[6, 0]	0
6	0	0	0	0

Table 5 Binomial tree (sample values)

And we can transform it to an oriented graph, as follows:



Pic 1 An oriented graph

So obviously, we can use dynamic programming to find the path with the maximum value.

We still need two more matrix to store result and path, called result_mat and path_mat. At the beginning, path_mat is all filled with 0. For result_mat, since we can't reach the nodes beyond those in Pic 1, so we fill them with -10000 thus they will never be touched.

We begin from the last column Using the following transition function:

$$\text{length}[i,j] = \max(\text{length}[:, j+1] + \text{value\_mat}[:, j])$$

where length[i,j] is the path with the largest value from last node to node [i,j] (In our project it's the result_mat), value_mat is the matrix in table 5.

Then goes backward and when we finally reached the first column, we fill the result_mat, as follows:

	0	1	2	3	4
0	-10000.0	-10000.0	-10000.0	-10000.0	-10000.0
1	-10000.0	-10000.0	12.0	-10000.0	-10000.0
2	-10000.0	15.0	-10000.0	8.0	-10000.0
3	23.0	-10000.0	12.0	-10000.0	0.0
4	-10000.0	19.0	-10000.0	9.0	-10000.0
5	-10000.0	-10000.0	15.0	-10000.0	-10000.0
6	-10000.0	-10000.0	-10000.0	-10000.0	-10000.0

Table 6 result_mat (sample value)

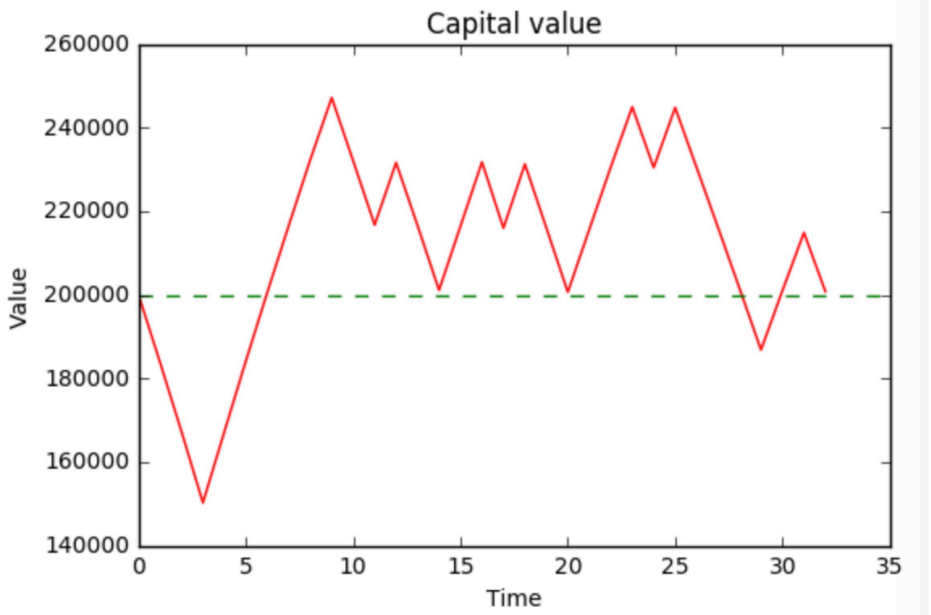
Moreover, we use a matrix to store path:

	0	1	2	3	4
0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	1.0	0.0	0.0
2	0.0	-1.0	0.0	1.0	0.0
3	1.0	0.0	1.0	0.0	0.0
4	0.0	-1.0	0.0	-1.0	0.0
5	0.0	0.0	-1.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0

Table 7 path_mat (sample values)

Beginning from the first column. If path_mat[i,j]=1, then it's a short position, next goes to the next column and next row; If path_mat[i,j]=-1, then it's a long position, next goes to the next column and previous row. Do this until we reach the last column.

Currently, we have our optimized strategy. Here is the plot of the change of our capital value:




```
Current capital: 200761.744758
```

Graph 1 Capital value

We can see that this running we get $761.7 profit, which is acceptable. However, we are not able to ensure profit each time.

4. Conclusion

This is the end of the project. I spent about 3 days, it's really rewarding. I learned a lot and also enhanced myself in my programming. Machine Learning strategies will outperform those traditional ones, like strategies based on technical indicators I did before. Here I would like to summarize my work and share my feelings.

For the data part, I choose the data after the financial crisis, which is from 2010 to 2018. Market is steady during these years, and it will represent most of the situation. That's why I choose this time interval.

Next is how to implement the Machine Learning tools. You know, speaking of ML, the first thing come to my mind is "prediction". So what should I predict? Thinking of price is a stochastic process and it's not determinant in the future. Also, if I don't know the future price, I can't build my strategy, or you can only wait until the price is known, but it's late at that time. Therefore, I decide to apply Machine Learning to predict the future treasury bond future price.

Because I have to predict a price as accurate as possible, so I can't use SVM. Naive Bayesian seems to be less effective according to my experience. That's why I select neural network as my Machine Learning tool because it can predict more features and is quite efficient. Also, it's easy to implement, just few demands and you get the result. It has excellent performance as you can see in table 3 that the predicted results and real values are close.

In order to maximize return, sortino ratio and minimize drawdown. For convenience I put them together into two values for long and short at each trading day. At each trading day I have two choices, long or short. So it's like a binomial tree with different values connecting two nodes. Now what I encounter is an oriented graph, a typical problem in graph theory. The most effective method to solve is dynamic programming. And then, we get the finally strategy, which turns out to be quite good.

Since we only need to visualize the movement of our capital, and it's only one line. Thus I choose "Matplotlib" to do the plot. It's simple and clear.

I spent a lot on the dynamic programming part because we don't use too much in Financial Engineering. In order to properly use dynamic programming, I refer to the textbook of my operational research course, researching it and try to implement the model. Another difficulty is the datestamp object. Actually the index of the data downloaded from quandl is neither string nor easily transformed to datetime object, but a new data type called "datestamp". I struggled with it for like 30 minutes and try to transform it to datetime object on my own but no good. Then I turn to google and it turned out that datestamp objects are actually a float number. I didn't know that and that's why I can't make it.

This project is beneficial and interesting. However, I think there are still some parts that I can improve. For example, next time I can try other Machine Learning approach instead of neural network, or maybe Reinforcement Learning, if possible. Anyway, I hope I can increase the profit of my strategy in the future. One important thing I get from this project is that Machine Learning strategy is more reliable than traditional indicator-based ones, at least you are less likely to lose much money.

Reference

https://www.investopedia.com/
https://scikit-learn.org/stable/