MAFS6010Y HW1 - Stock Selection Using Bandit Algorithm

Wah Chi Shing

2022-03-11

**The code of this assignment is available in github via https://github.com/anthonywah/mafs6010y_hw1.

Problem Definition

In this assignment, the possibility of selecting stocks and building a self-learning strategy is examined. We will attempt to split the stocks in different clusters using several popular technical indicators, and investigate into which clusters of stocks shall outperform the market over a specific period of time. After each period, we will update our reward and value function, and let it proceed to the next period. In this task, our main target is to examine whether a bandit algorithm can learn on its own to select stocks which may beat the market. Also, we will investigate into how the exploration-exploitation trade-off decision (i.e. the ϵ parameter in the ϵ-greedy algorithm) and the learning rate parameter (step-size) will have an impact on our result.

Data Exploration

The sample data we will use are 10-year daily close price and volume of constituent stocks of SP500, as well as SP500 itself, from Yahoo finance via python package `yfinance`. For simplicity only stocks with data available from 2011-01-01 are picked.

Then, we will compute the technical indicators that will be used to classify stocks into different clusters. We will use the following indicators:

Price Z-score

This indicator is computed by:

$$\text{zprice}_t = \frac{P_t - \bar{P}}{\sigma(P)}$$

$$\text{where } \bar{P} = \frac{\sum_{i=1}^{14} P_{t-i}}{14} \text{ and } \sigma(P) = \sqrt{\frac{\sum_{i=1}^{14} (P_{t-i} - \bar{P})^2}{14}}$$

The formula is very much similar to the popular technical indicator "Bollinger Band", which may indicator potential reversal in price trends. Here we will use this indicator to classify stocks into 3 groups, namely:

- $\text{zprice} < -1$
- $-1 \leq \text{zprice} \leq 1$
- $1 < \text{zprice}$

Volume Z-score

This indicator is computed by:

$$\text{zvol}_t = \frac{\text{Vol}_t - \bar{\text{Vol}}}{\sigma(\text{Vol})}$$

$$\text{where } \bar{\text{Vol}} = \frac{\sum_{i=1}^{14} \text{Vol}_{t-i}}{14} \text{ and } \sigma(\text{Vol}) = \sqrt{\frac{\sum_{i=1}^{14} (\text{Vol}_{t-i} - \bar{\text{Vol}})^2}{14}}$$

This indicator is simply used to classify whether the stock is relatively being paid attention by trades and investors. We use this indicator to classify stocks into 2 groups, namely:

- zvol ≤ 0
- $0 <$ zvol

Relative Strength Index (RSI)

This indicator is computed by:

$$\text{Average Gain} = \text{Exponential Weighted Average}(+ve\Delta P)$$
$$\text{Average Loss} = \text{abs}(\text{Exponential Weighted Average}(-ve\Delta P))$$
$$\text{RSI} = 100 - \frac{100}{1 + \frac{\text{Average Gain}}{\text{Average Loss}}}$$

This is a popular momentum indicator used to indicate potential over-bought and over-sold signals in a particular stock. Here we will use this indicator to classify stocks into 3 groups, namely:

- RSI < 40
- $40 \leq$ RSI ≤ 60
- $60 <$ RSI

One sample price graph along with the indicators and one distribution graph of the 3 indicators in 10 years are shown below.

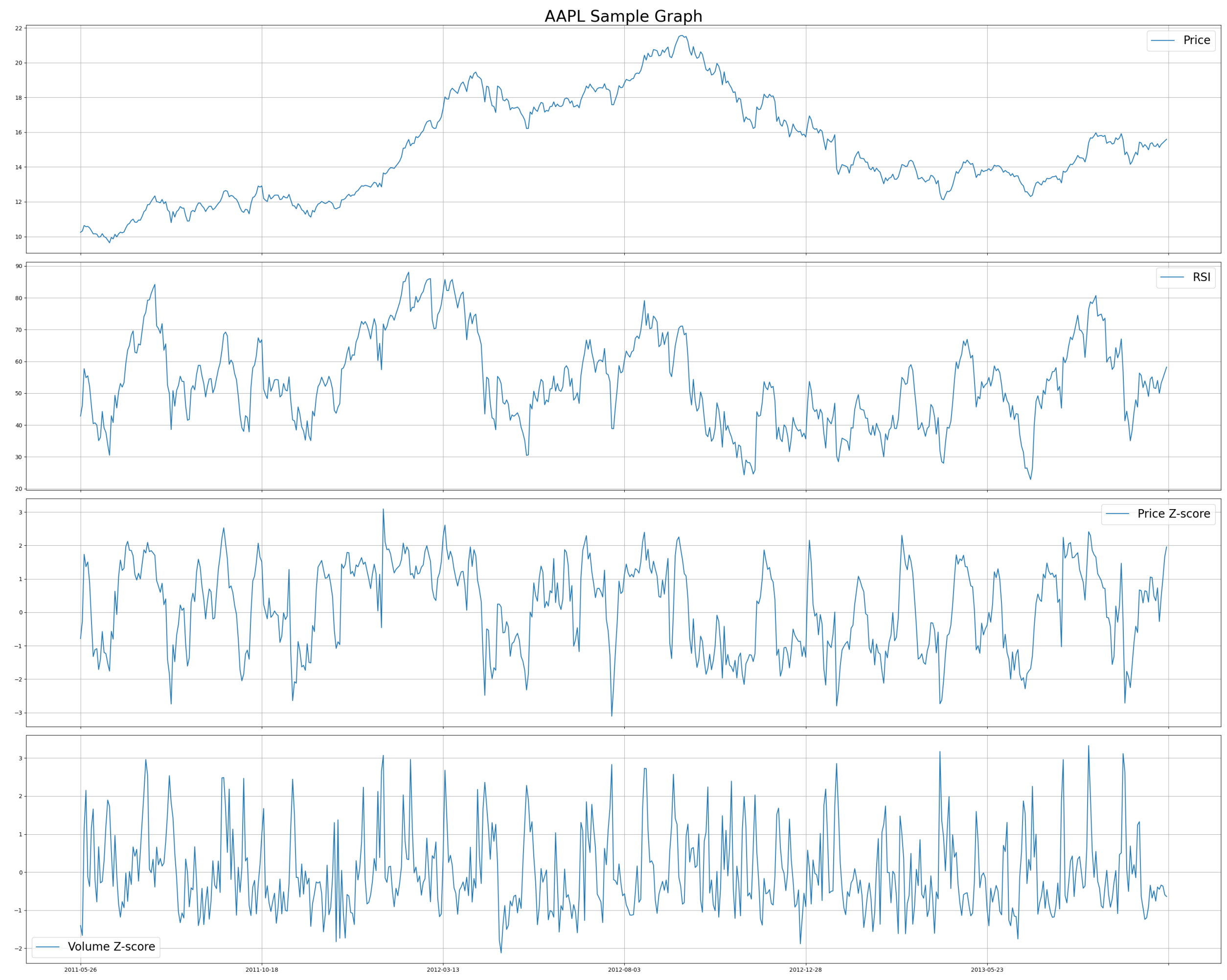

Figure 1: AAPL Price graph & Indicators

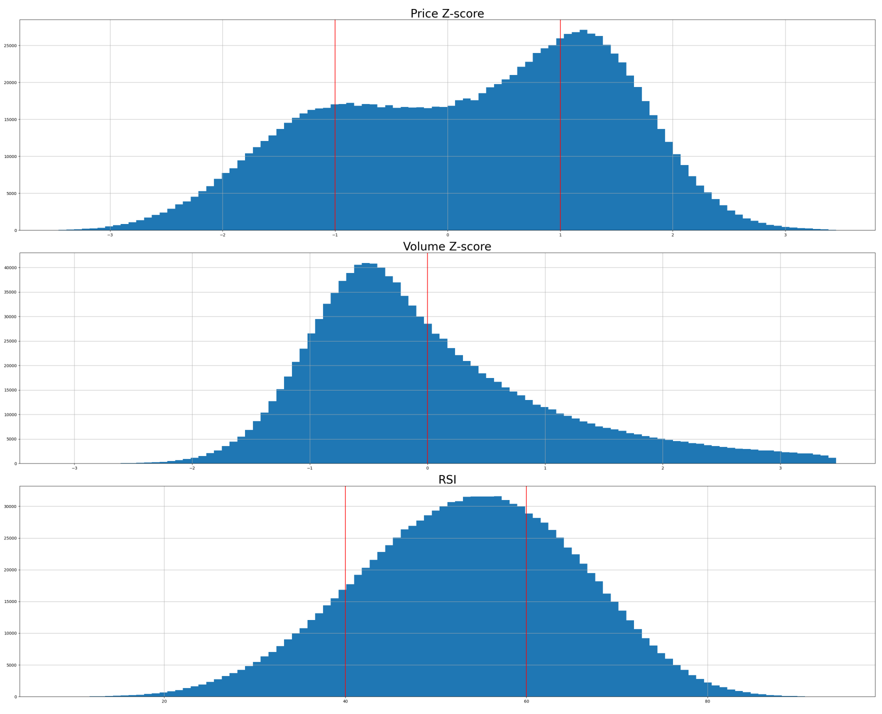

Figure 2: Indicator Distribution of SP500 Constituents over 10 years

Methodology

As mentioned, we have different classifications using the 3 different indicators. Therefore, we will have $3 \times 2 \times 2 = 18$ clusters in total. These 18 clusters of everyday represent the 18 arms we may choose in each period. Let's define the i^{th} cluster to be C_i, where $i = 0, 1, ...17$.

We define one period t to be 3 trading days, meaning that an arm (1 of the 18 clusters, C_i) will be picked every 3 days. In each period, we will long the C_i portfolio with equal weights to each stock, short SP500 index futures at the close price of 1^{st} trading day and close the whole position on the 3^{rd} trading day close, to measure how much this C_i will outperform the market in that period. The reward R_t of period t, which is the return of this trade, will be observed. Note that if there is no stocks included in the cluster, the reward will be observed as return of shorting SP500 index futures over 3 days.

We will use a ϵ-greedy algorithm to decide the probability for switching between exploration and exploitation. Firstly we will set $\epsilon = 0.1$, and we will see how the result will change with this parameter.

The Gradient Bandit Algorithm will be used to determine the preference and probability of selecting each cluster. The formula is given by:

$$\Pr\{C_t = a\} = \pi_t(a) = \frac{e^{H_t(a)}}{\sum_{b=0}^{17} e^{H_t(C_b)}}$$

We will use the average of all rewards $\bar{R}$ as the baseline, $\alpha = 0.1$ as the step-size parameter to update the cluster preferences by the below formula:

$$H_{t+1}(C_t) = H_t(C_t) + \alpha(R_t - \bar{R})(1 - \pi_t(A_t))$$
$$H_{t+1}(a) = H_t(a) - \alpha(R_t - \bar{R})\pi_t(a) \text{ for all } a \neq A_t$$

Result & Conclusion

First of all, we will use the default settings $\epsilon = 0.1$ and $\alpha = 0.1$ to simulate the bandit over 1000 runs, and examine the reward and total rewards (total cluster return - market return). Initial result is described in figure 3.

We can see that the mean reward is always revolving around 0 over time. However, the total reward, meaning the total return less market performance, is gradually increasing with time. Despite several draw-downs, the bandit algorithm is able to select correct stocks to trade and beat the market performance. The mean total reward is slightly above 45%.

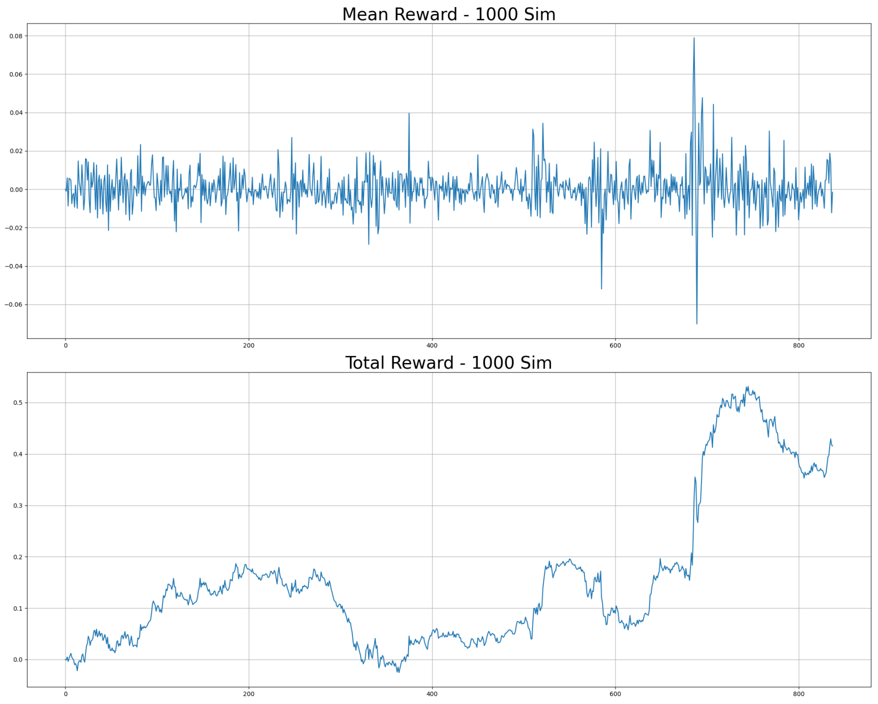

Figure 3: Base Case - 1000 simulations, $\epsilon = 0.1$, $\alpha = 0.1$

Then, we will try to alter the parameter ϵ to $\{0.0001, 0.01, 0.1, 0.2, 0.5\}$. The result is shown in figure 4.

We can see that there is not much difference in both rewards and total rewards when we change the parameter ϵ. Even if there are some gaps between $\epsilon = 0.0001$ and $\epsilon = 0.5$, the difference is merely around $\pm 1\%$. There is no observable evidence to prove that greediness can have an impact on the algorithm performance under this circumstance.

Lastly, we will alter the parameter α to $\{0.0001, 0.01, 0.1, 0.2, 0.5\}$. The result is shown in figure 5.

Again, we can see that not much difference in both rewards and total rewards is observed when we change the parameter ϵ. There are some gaps between $\epsilon = 0.5$ and $\epsilon = 0.2$ when it approaches the end of simulation, but it fails to explain any relationship with the change in parameter α.

To conclude, we can observe that given around 800 rounds and 18 arms for a bandit algorithm to learn over time, the algorithm is able to learn a strategy to pick stocks that may outperform the market. No observable changes can be concluded if we make the algorithm to be more "greedy" (i.e. larger ϵ) make it learn faster (i.e. larger α). Still, we may conclude that a bandit algorithm **can** be used to choose which stocks to invest.

Improvements

Several potential improvements can be made to improve the accuracy of the results. To name a few:

- Slippage and transaction cost should be considered if we want to run a full backtest after training a stock selection bandit algorithm;
- We may use the open price of the trading day as the entry price of the portfolio to make the return more realistic;
- Lengthen the trading days to 5 or 10 days in each round so that indicators may catch longer-term performance;
- RSI and zprice may have higher correlation due to the similarity in the formula. We may consider using another momentum technical indicators to replace RSI;
- We may consider long - portfolio to see if there may be any positive return, regardless of the market performance;
- Different criteria may be used such that less clusters will be available. In this case the algorithm may be able to learn a more concise decision-making policy;

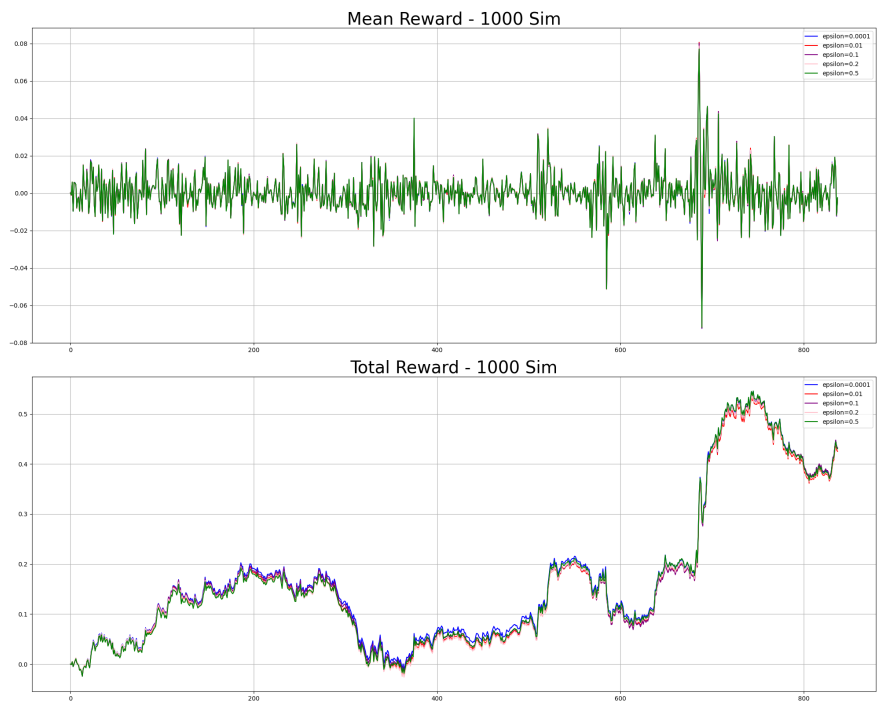

Figure 4: Alter ϵ Case - 1000 simulations, $\epsilon \in \{0.0001, 0.01, 0.1, 0.2, 0.5\}$, $\alpha = 0.1$

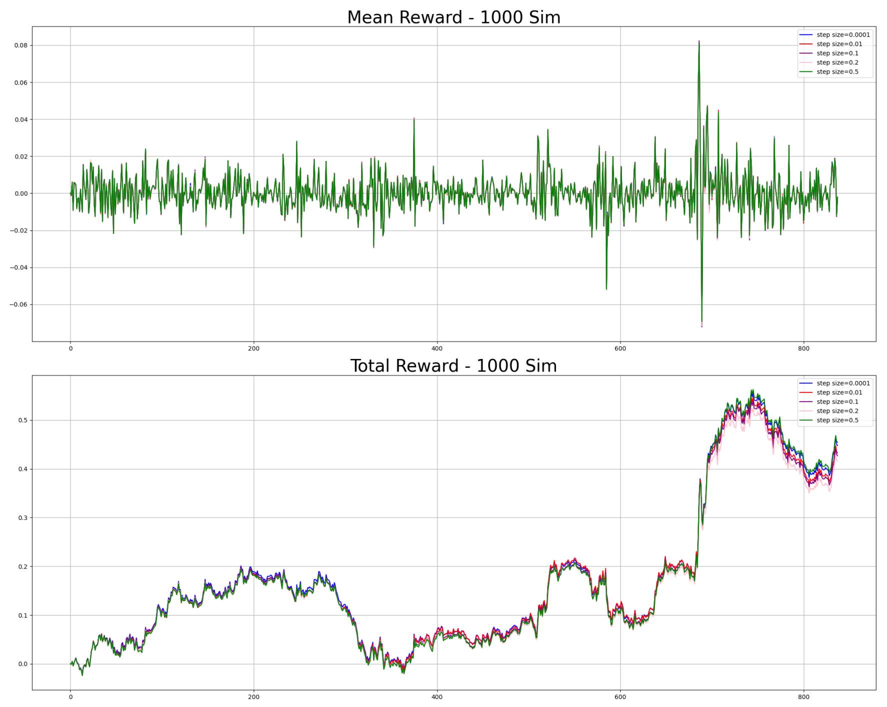

Figure 5: Alter α Case - 1000 simulations, $\epsilon = 0.1, \alpha \in \{0.0001, 0.01, 0.1, 0.2, 0.5\}$