Comparison of Log-Optimal Portfolio Selection Methods

Jieyun Wang
University of Illinois at Urbana-Champaign
jwang186@illinois.edu

Abstract

This project focuses on using machine learning knowledge in financial markets. We use existing historical data to find the log-optimal portfolio with the maximum asymptotic average growth rate. Next, we introduce semi-log-optimal selection, a smaller computational-complexity alternative of log-optimal. Finally, we combine kernel-based and nearest-neighbor-based strategy with semi-log-optimal selection. By comparing the results, semi-log-optimal with kernel-based strategy works the best among all four.

1 Introduction

In the financial market, the goal of some investors is to maximize their wealth in the long run. And now, we have a clear vision that when we consider multiple assets simultaneously, and also try to consider decisions over multiple period can lead to higher return. Thus, in order to construct such a portfolio, a best balance of all available assets is needed so that it maximizes the expected log-return. In the real world, however, an accurate statistical modeling of stock market behavior is known to be difficult to construct. On the other hand, by collecting the market's past data, using methods from nonparametric statistical smoothing and machine learning, we might derive an effective strategies based on the past experience.

This paper is organized in the following way: First, a portfolio selection problem is defined with assumptions. Then, strategies and algorithms are explained, namely log-optimal portfolio and semi-log-optimal portfolio selection, kernel-based and nearest-neighbor-based strategy, as well as gradient algorithm. Next, the implementation and results are presented followed by a discussion of what is found through the experiments. Finally, conclusion are made with proposal for future work.

2 Portfolio Selection Problems

As far as this project concerns, the goal of the investor is to maximize his wealth in the long run. To achieve this goal, one needs a portfolio, that is, a way to distribute his capital among available assets. This portfolio can be either static, which means one buys and holds without changing; or dynamic, which requires daily rebalancing.

2.1 Assumptions

Before we come to define our problem, assumptions must be made. In this project, we have the following assumptions:

a. Investor does not know the underlying distribution generating the stock price, but is allowed to use information collected from the market's past.

b. The underlying market process of the assets' relative price is memoryless, that is not affected by past information.

c. The assets are available for buying or for selling in unbounded quantities at the current price in any given trading period. But short selling is not allowed.

d. No transactions costs.

e. The behavior of the market is not affected by the actions of the investor.

2.2 Definitions

Instead of consider the whole market, we just consider a basket consisting of d stocks. The evolution of the market in time is represented by a sequence of price vector $\mathbf{s_1}$, $\mathbf{s_2}$ ..., where:

$$\mathbf{s_n} = (s_n^{(1)}, \ldots, s_n^{(d)})$$

with $s_n^{(j)}$ is the price of j-th stock on the n-th trading period. From this sequence, we can have the return vector $\{\mathbf{x_n}\}$, as $\mathbf{x_n} = (x_n^{(1)}, \ldots, x_n^{(d)})$, with

$$x_n^{(j)} = \frac{s_n^{(j)}}{s_{n-1}^{(j)}}$$

In addition, a portfolio vector is denoted by $\mathbf{b} = (b^{(1)}, \ldots, b^{(d)})$ for the static portfolio. $b^{(j)}$ of $\mathbf{b}$ denotes the proportion of the investors capital invested in j-th stock. Since no short selling is allowed, thus, we will have the set of portfolio vectors defined by:

$$\Delta_d = \{\mathbf{b} = (b^{(1)}, \ldots, b^{(d)}); \\ b^{(j)} \geq 0, \sum_{j=1}^{d} b^{(j)} = 1\}$$

3 Strategies and Algorithms

This part focuses on the theoretical definition of the algorithms used in this project.

3.1 Log-Optimal Portfolio

If the market process $\{\mathbf{X}_i\}$ is memoryless, i.e., it is a sequence of independent and identically distributed random return vector, then, the best constantly rebalanced portfolio is the log-optimal portfolio.

$$\mathbf{b}^* = \arg\max_{\mathbf{b}\in\Delta_d} E\{ln < \mathbf{b}, \mathbf{X}_1 >\}$$

with $< \mathbf{b}, \mathbf{X}_1 >$ denotes the inner product of the two vectors.

We use the strong law of large number to proof the optimality. First we use $W(\mathbf{b}) = E\{ln < \mathbf{b}, \mathbf{X}_i >\}$

$$\begin{aligned}\frac{1}{n} lnS_n &= \frac{1}{n}\sum_{i=1}^{n} ln < \mathbf{b}, \mathbf{X}_i > \\ &= W(\mathbf{b}) + \frac{1}{n}\sum_{i=1}^{n}(ln < \mathbf{b}, \mathbf{X}_i > \\ &\quad - E\{ln < \mathbf{b}, \mathbf{X}_i >\})\end{aligned}$$

From the law of large numbers, we will have

$$\frac{1}{n}\sum_{i=1}^{n}(ln < \mathbf{b}, \mathbf{X}_i > - E\{ln < \mathbf{b}, \mathbf{X}_i >\}) \to 0$$

Therefore,

$$\lim_{n\to\infty} \frac{1}{n} lnS_n = W(\mathbf{b})$$

almost surly, and similarly we have:

$$\lim_{n\to\infty} \frac{1}{n} lnS_n^* = W(\mathbf{b}^*) = \max_{\mathbf{b}} W(\mathbf{b})$$

with $S_n^* = S_n(\mathbf{b}^*)$ denote s the capital after day n achieved by $\mathbf{b}^*$.

Thus, the empirical log-optimal portfolio can be defined by

$$\mathbf{b}^* = \arg\max_{\mathbf{b}\in\Delta_d} \frac{1}{n}\sum_{i=1}^{n}\{ln < \mathbf{b}, \mathbf{x}_i >\}$$

with linear constraints:

$$\sum_{j=1}^{d} b^{(j)} = 1 \quad and \quad 0 \leq b^{(j)} \leq 1 \quad j = 1, \ldots, d$$

3.2 Semi-Log-Optimal Portfolio

Even though log-optimal algorithm could give us the best portfolio, the computational-complexity of it is too high. Thus, in order to overcome this, an alternative approximate solution, semi-log-optimal portfolio selection is introduced. Using the second-order Taylor expansion of the function $ln(z)$ at $z = 1$:

$$h(z) = z - 1 - \frac{1}{2}(z-1)^2$$

Thus, the empirical semi-log-optimal portfolio is defined by:

$$\mathbf{b}^* = \arg\max_{\mathbf{b}\in\Delta_d} \frac{1}{n}\sum_{i=1}^{n}\{h(< \mathbf{b}, \mathbf{x}_i >)\}$$

with the same linear constraints as log-optimal algorithm.

3.3 Kernel-Based Strategy

Then, we introduce kernel-based portfolio selection strategy. Define an infinite array of experts $\mathbf{B}^{(k,l)} = \{\mathbf{b}^{(k,l)}(\cdot)\}$, where k, l are positive integers. For one pair of k, l, choose the radius $r_{k,l} > 0$. Then, for $n > k+1$, define the expert $\mathbf{b}^{(k,l)}$ by

$$\mathbf{b}^{(k,l)}(\mathbf{x}_1^{n-1}) = \arg\max_{\mathbf{b}\in\Delta_d} \prod_{\{k<i<n:\|\mathbf{x}_{i-k}^{i-1}-\mathbf{x}_{n-k}^{n-1}\|\le r_{k,l}\}} < \mathbf{b}, \mathbf{x}_i >$$

And these experts are mixed as following:

$$S_n(\mathbf{B}) = \sum_{k,l} q_{k,l} S_n(\mathbf{B}^{(k,l)})$$

3.4 Nearest-Neighbor-Based Strategy

Nearest-neighbor-based strategy is similar to kernel-based strategy in definition, but instead of setting a radius, we choose a positive integer l, then, the expert searches for the l nearest-neighbor (NN) matches in the past. Now we introduce the set of l NN:

$$J_n^{(k,l)} = \{i; k+1 \le i \le n \ \ such \ \ that \ \ \mathbf{x}_{i-k}^{i-1} \ \ is \ \ among \ \ the \ \ l \ \ NN \ \ of \ \ \mathbf{x}_{n-k}^{n-1}\}$$

Then, we define the expert by:

$$\mathbf{b}^{(k,l)}(\mathbf{x}_1^{n-1}) = \arg\max_{\mathbf{b}\in\Delta_d} \prod_{i\in J_n^{(k,l)}} < \mathbf{b}, \mathbf{x}_i >$$

3.5 Gradient Algorithm

In this project, the gradient algorithm is used and can be described as below:

(1) Calculate:

$$W_n(P(\mathbf{b}_{k-1} + \delta \cdot \mathbf{e}_j)) \quad j = 1, \ldots, d$$

(2) If: $V_{k-1} \ge \max_j W_n(P(\mathbf{b}_{k-1} + \delta \cdot \mathbf{e}_j))$
then, stop looping and the result of the algorithm is $\mathbf{b}_{k-1}$
Else: set $V_k = \max_j W_n(P(\mathbf{b}_{k-1} + \delta \cdot \mathbf{e}_j))$
and

$$\mathbf{b}_k = P(\mathbf{b}_{k-1} + \delta \cdot \mathbf{e}_{j^*})$$

$$j^* = \arg\max_j W_n(P(\mathbf{b}_{k-1} + \delta \cdot \mathbf{e}_j))$$

Go to (1)

In the algorithm above, $P(\mathbf{b})$ is a projection of vector $\mathbf{b}$ to Δ_d. And $W_n(P(\mathbf{b})) = \frac{1}{n}\sum_{i=1}^n ln < P(\mathbf{b}), \mathbf{X}_i >$ for log-optimal algorithm, $W_n(P(\mathbf{b})) = \frac{1}{n}\sum_{i=1}^n h(< P(\mathbf{b}), \mathbf{X}_i >)$ for semi-log-optimal algorithm.

4 Implementations and Results

4.1 Data Collection

In this implementation,stocks in the basket was chosen from S&P 500, based on the size of market capitalization. There are altogether 25 stocks involved, but these 25 stocks are not the top 25 stocks ranked by market capitalization. This is because the time scope of this project is 10 years, and some of the high market capitalization stocks were only launched in the recent years, like Facebook, Inc. (FB). The following table shows the list of stocks chosen for this project:

Asset No.	Ticker	Asset No.	Ticker
1	XOM	14	ORCL
2	MSFT	15	T
3	JNJ	16	BAC
4	WFC	17	MRK
5	GE	18	INTC
6	WMT	19	C
7	CVX	20	GILD
8	JPM	21	DIS
9	PG	22	AMZN
10	VZ	23	CMCSA
11	IBM	24	PEP
12	PFE	25	SLB
13	KO		

Table 1: Stocks in the Basket.

We used Bloomberg to collect the daily close price of all these 25 stocks from Jan 2nd, 2004 to Dec 31st, 2013. From these close price, using the definition above $x_n = S_n/S_{n-1}$, we can get the historical return vector. In this project, these data are divided into two groups: from Jan 2nd, 2004 to Dec 31st, 2012, these part of data are used as training data to get the portfolio weight vector; from Jan 2nd, 2013 to Dec 31st, 2013, these part of data are used as testing data to compare the portfolio return of different strategies.

4.2 Results and Discussion

4.2.1 Log-Optimal and Semi-Log-Optimal

In this part of experiment, we use the learning rate $\delta = 0.0001$, and have the following results:

Asset No.	Log-Optimal	Semi-Log-Optimal
20	0.532	0.537
22	0.468	0.463

Table 2: Portfolio Vector.

This tells for both algorithm, apart from asset 20 (GILD) and 22 (AMZN), the rest are not in the portfolio. And from Table 2, we could also see that the difference of the two portfolio is not much.
Then we use the historical data in year 2013 to check the returns of the two portfolio. Since the returns are close, we use the following as y-axis: $r_{log}/r_{semi-log}$

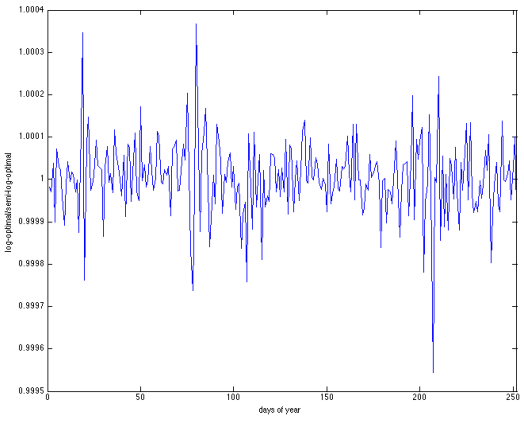

Figure 1: Comparison of Returns

At the first glance of Figure 1, it doesn't tell much, except that the ratio moves around 1, which means the two portfolios give almost the same return. Then, consider the mathematical properties of this sequence, we can see that the mean of this sequence is 1.0000049. From this, we can say that log-optimal algorithm gives slightly better return on average than semi-log-optimal. This can be expected since the second is a smaller computational-complexity alternative of the first, thus, in the same time, some information must have been left behind. However, if we take the efficiency into account, when $\delta = 0.0001$, log-optimal algorithm takes 130.91 seconds to find the optimal solution, while semi optimal algorithm only takes 81.32 seconds. Thus, with the returns almost the same, semi-log-optimal is a better choice for computation.

4.2.2 Kernel-Based Strategy

Next, we show experiment for combining the kernel-based portfolio selection and the semi-log-optimal algorithm to give the kernel-based semi-log-optimal portfolio. Some numerical results obtained are presented below. An infinite array of experts is suggested by theory, however, in practice, we have to take a finite array of size $K \times L$. In this experiments, we choose $K = 5, L = 10$ with the values listed in Table 3. Then, we choose the uniform distribution $\{q_{k,l}\} = 1/(KL)$, and the radius:

$$r_{k,l}^2 = 0.0008 * d * k(1 + l/10)$$

And Table 3 summarizes the annual yield achieved using test data by each experts.

k / **l**	50	100	150	200	250
1	0.6186	0.5845	0.6073	0.6951	0.7154
2	0.6036	0.5824	0.6480	0.6709	0.7154
3	0.6262	0.6376	0.6445	0.6446	0.7154
4	0.6115	0.6253	0.6369	0.6214	0.6331
5	0.6351	0.6349	0.6547	0.6126	0.5709
6	0.6137	0.6730	0.6703	0.5680	0.5986
7	0.6137	0.6718	0.6359	0.5643	0.5736
8	0.6220	0.6735	0.6165	0.5539	0.5691
9	0.6327	0.6625	0.6165	0.5622	0.5462
10	0.6390	0.6651	0.6263	0.5424	0.5541

Table 3: Annual Yields of the Individual Experts for the Kernel Strategy, using Semi-Log-Optimal Algorithm.

And the annual yield of the kernel-based semi-log-optimal portfolio is 0.6252. In the meantime, the annual yield obtained by semi-log-optimal portfolio is 0.6066. If we consider the daily return and using $r_{kernel}/r_{semi-log}$ as y-axis, we will have the following graph.

This Figure 2 looks quite similar to the graph shown in Figure 1 in sense of moving trend. However, if we take a closer look, we might notice that the magnitude in Figure 2 is larger than that in Figure 1. This implies that kernel-based strategy will

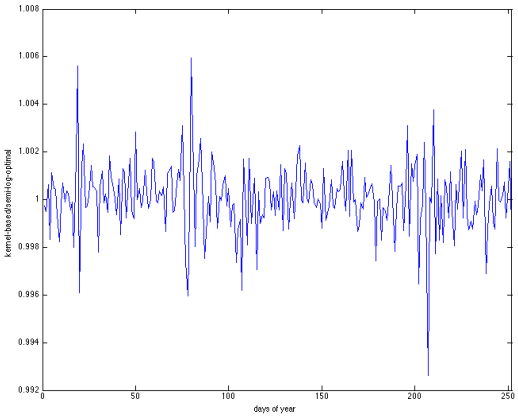

Figure 2: Comparison of Returns

give higher return than log-optimal and semi-log-optimal in the long run, which is consistent with the annual yield we get above.

4.2.3 Nearest-Neighbor-Based Strategy

Apart from the kernel-based strategy, we also combine semi-log-optimal with nearest-neighbor-based strategy. Again, we take a finite array of size $K \times L$, with $K = 5, L = 10$ and $\{q_{k,l}\} = 1/(KL)$. In this experiment, we take l as the number of nearest neighbors. Then, we could have the table of the annual yield achieved using test data by each experts:

k **l**	50	100	150	200	250
70	0.1564	0.5706	0.7154	0.7154	0.4635
140	0.1442	0.7154	0.2009	0.3257	0.5745
210	0.7154	0.7154	0.7154	0.6386	0.6467
280	0.7154	0.7154	0.7154	0.4806	0.6931
350	0.2626	0.7154	0.4512	0.7154	0.7154
420	0.2626	0.7154	0.7154	0.6748	0.6673
490	0.2626	0.7154	0.7154	0.4635	0.6648
560	0.3589	0.6580	0.7154	0.4635	0.5787
630	0.4334	0.6918	0.5531	0.4635	0.3721
700	0.5048	0.2715	0.3806	0.2848	0.5081

Table 4: Annual Yields of the Individual Experts for the Nearest-Neighbor Strategy, using Semi-Log-Optimal Algorithm.

However, the annual yield of the nearest-neighbor-based semi-log-optimal portfolio is only 0.5613, which is the worst among the four.

5 Conclusion

From the results we get, we could come to the following conclusions:

a. As a computational-complexity alternative of log-optimal, semi-log-optimal algorithm gives a portfolio that is quite similar to the one given by log-optimal with less computing time.

b. Among all four algorithms and strategies, the kernel-based semi-log-optimal gives the portfolio with the highest return.

As for the extension of this project, we could put short selling and transaction fees in to consideration.

References

Laszlo Gyorfi,Gyorgy Ottucssak and Harro Walk. 2012. *Machine Learning for Financial Engineering*, chapter 1 *On the History of the Growth-Optimal Portfolio*. Imperial College Press, London,UK.

Laszlo Gyorfi,Gyorgy Ottucssak and Harro Walk. 2012. *Machine Learning for Financial Engineering*, chapter 2 *Empirical Log-Optimal Portfolio Selections: A Survey*. Imperial College Press, London,UK.

Laszlo Gyorfi,Gyorgy Ottucssak and Harro Walk. 2012. *Machine Learning for Financial Engineering*, chapter 5 *Nonparametric Sequential Prediction of Stationary Time Series*. Imperial College Press, London,UK.