Optimization in Portfolio Allocation for an Arbitrary Number of Assets

Richard S., Raafi R., Nancy S., Jeremy B.

Team 1
December 25, 2022

1 Introduction

This paper examines portfolio management using techniques from stochastic optimization. In particular we look at a portfolio comprised of a N number of assets and optimize the percentage breakdown of our initial investment into each asset to maximize returns at a future time T. We define our optimization parameters as a vector of decimals under the constraints that each component must be positive and the sum of components must equal 1. These decimals represent the percentage of the initial investment into each of our assets and the vector has the same length, N, as the number of assets being invested into. In our problem, we implement the generalized gradient as discussed by Vazquez-Abad (1999) to construct the target vector field. We then implement a batch approach to make approximations of our stochastic gradient under the assumption of a martingale difference noise model and simulations of Black-Scholes pricing model. In initial approaches we also included a bound on the risk of our portfolio measured by the portfolio's variance. The removal of that constraint will be touched on further in our Conclusion. The result of our approach matches that of the trivial problem and pushes our portfolio to be placed into assets with highest return regardless of variance.

2 Problem Formulation

2.1 Optimization Problem

We are considering n assets for our portfolio. Let the return of asset i be R_i. Let the expected return of asset i at time T be $\mathbf{E}[R_i] = \frac{E[S_i(T)]-S_i(0)}{S_0}$. Suppose our total capital is 1 monetary unit. This allows us to view allocations as proportions and can be scaled to fit any amount of capital. The allocation for each asset will be denoted as θ_i where $\sum_{i=1}^{n} \theta_i = 1$. Your return at time T is $J(\theta) = \sum_{i=1}^{n} \theta_i \mathbf{E}[R_i]$. Our goal is to maximize total return while staying within our constraints. The problem formulation is below.

$$\max J(\theta) = \sum_{j=1}^{n} \theta_i \frac{E[S_i(T)] - S_i(0)}{S_i(0)} \tag{1}$$

Constraints:

$$c(\theta) = \sum_{i=1}^{n} \theta_i = 1, \theta_i \geq 0$$

2.2 Simulating with Black-Scholes Model

In order to model the path of an asset, we turn to the Black-Scholes model. In doing so, we are able to model a stock path using the following equation:

$$S(T) = S(0)e^{(\mu - \frac{1}{2}\sigma^2)T + \sigma\sqrt{T}Z} \tag{2}$$

Where the parameters are μ and σ for a fixed t. Of course, these means that $S(T)$ will follow a log-normal distribution with an expectation that is dependent on μ and σ.

2.3 Allocation Problem

The question of interest is, how do we construct a portfolio of n assets that has the highest expected return. As stated previously, we are considering a θ vector to be a vector of allocations (or weights). Let $S(T)$ denote a vector of assets and θ_i will denote the allocation of asset i, $S_i(T)$..
Our goal is to find the optimal $\theta = \theta^*$ such that $J(\theta^*)$ is maximized.

3 Methodology and Convergence Analysis

3.1 Vector Field

In order to meet the constraints of the problem, we decided to use a Generalized Gradient approach. The general formulation for this is given by

$$\mathcal{G}_i(J(\cdot)) = \theta_i \left(\frac{\partial J(\cdot)}{\partial \theta_i} - \frac{1}{V} \sum_{j=1}^{V} \frac{\partial J(\cdot)}{\partial \theta_j} \right) \tag{3}$$

Using our constructed cost function, $J(\theta)$, the generalized gradient works out to be

$$J(\theta) = \sum_{j=1}^{n} \theta_i \frac{E[S_i(T)] - S_i(0)}{S_i(0)} \tag{4}$$

$$\frac{\partial J(\theta)}{\partial \theta_i} = \frac{E[S_i(T)] - S_i(0)}{S_i(0)}$$

$$\mathcal{G}_i[J(\cdot)] = \theta_i \left(\frac{\partial J(\cdot)}{\partial \theta_i} - \frac{1}{V} \sum_{j=1}^{V} \frac{\partial J(\cdot)}{\partial \theta_j} \right)$$

$$\mathcal{G}_i[J(\theta)] = \theta_i \left(\frac{\partial J(\theta)}{\partial \theta_i} - \frac{1}{V} \sum_{j=1}^{V} \frac{\partial J(\theta)}{\partial \theta_j} \right)$$

$$\begin{aligned} \mathcal{G}_i[J(\theta)] &= \theta_i \left(\frac{\partial J(\theta)}{\partial \theta_i} - \frac{1}{V} \sum_{j=1}^{V} \frac{\partial J(\theta)}{\partial \theta_j} \right) \\ &= \theta_i \left(\frac{\partial J(\theta)}{\partial \theta_i} - \frac{1}{V} \sum_{j=1}^{V} \frac{S_j(T) - S_j(0)}{S_j(0} \right) \\ &= \theta_i \left(\frac{S_i(T) - S_i(0)}{S_i(0)} - \frac{J(\theta)}{V} \right) \end{aligned} \tag{5}$$

The algorithm we use to solve this problem is the following:

$$\theta_{n+1} = \theta_n + \epsilon_n Y_n \tag{6}$$

3.2 Gradient Estimation

In this section, we discuss the technique used to estimate the gradient. We settled on the Infinitesimal Perturbation Analysis (IPA) approach. In order to do so, we defined $X_i(\theta) = \theta_i S_i(T)$ and $h(x) = \frac{x - \theta S_i(0)}{S_i(0)}$. With these equations, we were able to prove that our problem is sufficient for IPA as follows;

(i) $\frac{d(X(\theta))}{d\theta_i}$ exists at θ_0 with probability 1.

$$X(\theta) = \theta_i S_i(T)$$
$$\frac{d(X(\theta))}{d\theta_i} = S_i(T)$$

(ii) The mapping $h : R \to R$ is differentiable.

$$h_i(X(\theta)) = \frac{x - \theta_i S_i(0)}{S_i(0)}$$
$$\frac{d(h_i(X(\theta))}{d\theta_i} = \frac{1}{S_i(0)}$$

(iii) $h(x)$ is Lipschitz Continuous on Θ.

Let Θ be the set of all θ vectors which meet the constraints mentioned previously. Then $h(x)$ is a composition of continuous functions and parameters, hence $h(x)$ is continuous.

Note: This method was not implemented in the algorithm. This section exists to show that we attempted to use IPA before switching to the generalized gradient and removing the variance constraint.

3.3 Convergence Analysis

In this section, we show that the algorithm converges using our constructed $G(J(\theta))$. We aim to prove the conditions put forth by Theorem 4.1 In order to use the algorithm $\theta_{n+1} = \theta + \epsilon_n Y_n$ for $\epsilon_n = 1/n$. First is to show that We can generate a Y_n to approximate our target vector field. In this paper, we will define our S^* to be a vector of log-normals with each component having a unique distribution $logNorm(\mu_i, \sigma_i^2)$

$$Y_{n,i} = \theta_i \left(\frac{S_i^* - S(0)}{S(0)} - \frac{J(\theta; S^*)}{V} \right) \tag{7}$$

Since S_i^* has the same distribution as $S_i(T)$, and the randomness of Y_n and $G(J(\theta_n))$ depend only on S_i^* and $S_i(T)$ respectively, $E[Y_n] = E[G(J(\theta_n))]$. Hence Y_n is an unbiased estimator of $G(J(\theta_n))$. The is import because it implies that Y_n follows a Martingale Difference Noise model such that,

$$\delta M_n = Y_n - \mathrm{E}[Y_n | \mathcal{F}_{n-1}]$$
$$\Rightarrow Y_n = G((\theta_n)) + \delta M_n + \beta_n$$

with

$$E[\delta M_n] = E[\beta_n] = 0$$

There are four conditions needed to apply theorem 4.1. The conditions, along with their proofs, are shown below (note that we will consider);

(a1) $\sum_{i=0}^{\infty} \epsilon_i = \infty$:
This condition is set because $\sum_{i=0}^{\infty} \frac{1}{i+1} = \sum_{j=1}^{\infty} \frac{1}{j}$ is a harmonic series.

(a2) the error terms are asymptotically negligible, in the sense that $\sum_{i=0}^{\infty} \epsilon_i \|\beta_i\| < \infty$ w.p.1:
This condition is met by $E[\beta_n] = 0$.

(a3) the variance term, $V = E[(\delta M_n)^2] = 0$, satisfies $\sum_{i=0}^{\infty} \epsilon_i^2 V_n < \infty$w.p.1:
This condition is met because V_n can be considered a constant because the values are random and are not

a function of n. Since the ϵ_i^3 terms converge in sum, it stands that the above sum will also converge.

(a4) The ODE

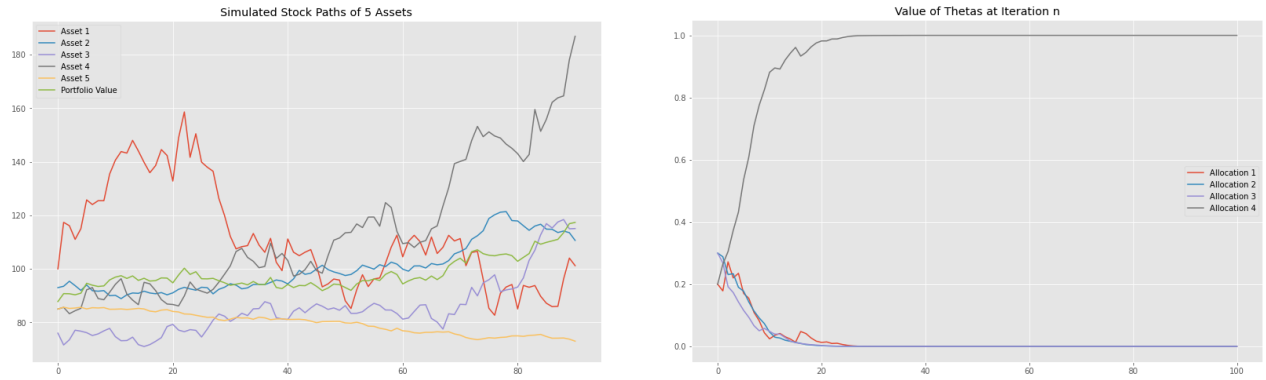
$$\frac{\delta v(t)}{\delta t} = G(v(t)) \tag{8}$$

has a unique limit for each initial condition. Let S denote the set of stable points of this ODE and assume that $S \neq \emptyset$.

We know that S in this case is non-empty because of the trivial solution of letting $\theta_k = 1$ for the $S_k(T)$ with the largest expected return. Now observe that due to the formulation of the gradient, $G_i(J(\theta)) < 0$ if the return of asset i is lower than the average return. This would then lower the allocation of that stock for the next iteration. The opposite happens when a stock has a return that is greater than the average. Thus, the vector field will always move in the direction of the highest return.

4 Experimental Results

For our experiment, we simulated the stock path of 4 different assets using the Black-Scholes model for each quarter. Each asset has a different initial value $S(0) = (100, 92, 76, 10)$ and variance $\sigma = (1.1, 0.25, 0.5, 0.01)$. To show the convergence of our algorithm, we present the results of where three of the four assets have the same mean and asset number 4 has the highest mean $\mu = (1, 1, 1, 3)$



(a) Stock path simulated using Black-Scholes Method

(b) Performance of algorithm after 100 algorithms

Figure 1: Experimental Results

Observe that for this problem, the optimal portfolio is found after only nearly 30 iterations.

5 Conclusion

We used an optimization algorithm introduced in Chapter 1, a convergence algorithm found in Chapter 4, the basis of exercise 6.6 from Chapter 6, and Generalized Gradients from Vazquez-Abad (1999) in order to use noisy asset observations to create the best performing portfolio given N assets. Although our initial plans included constraining our possible portfolios by the variance of the portfolio, we were unable to find a meaningful way to incorporate this constraint. Originally we tried using standard IPA, but found the constraints would not be accounted for in our model. Thus we switched to generalized gradient for its mass-preserving properties and the ability to keep all components of θ between 0 and 1 inclusive. We found these properties to be very desirable, even though variance could not be accounted for. This leaves room for future work where constraints on portfolio risk are included to account for lower risk investors.

6 Code

```
from random import gauss, seed
from math import sqrt, exp
import numpy as np

import matplotlib.pyplot as plt
plt.style.use('ggplot')

import time as t

def generate_value(s0, mu, sigma, n, t):

  stocks = [s0]
  st = s0
  for i in range(n):
    st *= exp((mu - 0.5 * sigma ** 2) * (t / n) + sigma * sqrt(t / n) * gauss(mu=0, sigma=1))
    stocks.append(st)

  return stocks

def port_val(thetas, portfolio):
  return np.dot(thetas, portfolio)

def J(thetas, portfolio):
  rets = (portfolio[:,1] - portfolio[:,0])/portfolio[:,0]
  return np.multiply(thetas, rets).sum()


def G(thetas, portfolio):
  jay = J(thetas, portfolio) / len(thetas)

  S_t = portfolio[:,1]
  S_0 = portfolio[:,0]
  rets = (S_t - S_0)/S_0 - jay
  g = np.multiply(thetas, rets)

  return g

  T = 0.25 # quarter
theta = np.array([0.2, 0.3, 0.3, 0.2])
theta_display = [[theta[0]], [theta[1]], [theta[2]], [theta[3]]]
n=1 #number days were breaking up the intervals
N=1

start = t.time()

for i in range(100):
  epsilon = 1/(n+1)
  Yn = np.zeros(len(theta))

  for i in range(N):
    S1 = generate_value(100,1.5,1.1,n,T)
    S2 = generate_value(93,1,0.25,n, T)
```

```
    S3 = generate_value(76,1,0.5,n, T)
    S4 = generate_value(10,3,0.01,n, T)
    S_t = np.array([S1,S2,S3,S4])

    Yn = np.add(Yn, np.array(G(theta, S_t)))

  Yn = np.divide(Yn, N)
  theta = np.add(theta, epsilon * Yn)
  theta = np.divide(theta, theta.sum())
  for i in range(len(theta)):
    theta_display[i].append(theta[i])
```

References

[1] F. Vázquez-Abad. "Strong points of weak convergence: a study using RPA gradient estimation for automatic learning," in: Autom. 35 (1999), pp. 1255–1274.