Optimizing Portfolio Allocation Using Steepest Descent on the Sharpe Ratio

Shubhang Tyagi
MATH 132A Winter Quarter 9:30AM
Instructor: Paul Atzberger
March 17th, 2025

1 Problem Statement

The goal of this project is utilize a gradient descent method to provide a porfolio recommendation. We do this by utilizing and important ratio in stock investments, the Sharpe Ratio. The Sharpe Ratio is used to compute the risk-free return of a portfolio. In investing, investors want to allocate their assets to maximize returns while minimizing any unneccesary risk. Portfolio optimization is a key problem in finance. We maximize the Sharpe Ratio to find the optimal asset weights using the Steepest Descent Method. The algorithm that is discussed in this study holds weight in the subject of optimization. In this paper, the implementation, mathematical background and analysis of such results are discussed.

2 Background and Methodology

In this section, the mathematical principles that this project is based on is discussed and the procedure is explained. The Sharpe Ratio is computed from R_p is the portfolio return, R_f is the risk-free rate, and σ_p is the standard deviation of the portfolio returns. The risk-free rate is a constant that was acquired using a simple Google search of what a Treasury Bond gets you. The standard deviation of the portfolio returns are calculated using the covariance matrix as shown below. The function that computes the Sharpe Ratio is also stated below.

$$S(w) = \frac{R_p - R_f}{\sigma_p},$$

The function is the one that is maximized for the Sharpe Ratio. Maximizing the Sharpe Ratio involves solving a nonlinear optimization problem. We implement the steepest descent method.

Gradient-based methods, such as Steepest Descent, use first-order derivatives to iteratively improve the solution. At each step, starting from x^k, a line search is conducted in the direction $-\nabla f(x^k)$ until a maximizer x^{k+1}, is found. In other words,

$$\alpha_k = \text{arg max } f(x^k) - \alpha \nabla f(x^k)$$

the Steepest Descent method follows the direction of the negative gradient to reach an optimal point:

For Sharpe Ratio optimization, the function $f(x)$ represents the gradient of the Sharpe Ratio with respect to stock weights.

$$w_{n+1} = w_n - \alpha \nabla S(w_n),$$

where α is the step size. For Sharpe Ratio optimization, the function $f(x)$ represents the gradient of the Sharpe Ratio with respect to stock weights.

The language in which the mathematical backbone is coded in is Python. Below are the functions defined in the code as well as and explanation for what each of them does.

1. Data Retrieval:
   The data calls the data from the YFinance API and put into a data frame for further calling.

2. Computing Returns for the portfolio:
   The .pct change function returns the daily price changes of the portfolio in terms of a percentage. The dropna() function removes any NAN values from the data set, which is a form of preprocessing the data.

3. Computing the Sharpe Ratio:
   This function serves two purposes. It is a helper function for the incoming descent algorithm and its role is also outputting the expected returns, volatility and Sharpe Ratio. From the above formula, the Sharpe Ratio is calculated using this function. The derivative of the formula is used to calculate the gradient at a given weight

4. Steepest Descent on the Sharpe Ratio:
   The objective function is the Sharpe Ratio formula and the decision variable is the weight of the portfolio, i.e, the percentage of each stock in the portfolio. This function goes through 1000 iterations of computing the Sharpe Ratio at a given set of weights, finding the gradient and step size values, and adjusting the new weights with the new step size and gradient values. After the iterations are complete, the function returns the optimal weights, daily returns, daily volatility and the maximum Sharpe Ratio.

```python
def fetch_data(tickers, start_date, end_date, interval='1d'):
    data = yf.download(tickers, start=start_date, end=end_date, interval=interval)['Close']
    return data

def compute_returns(data):
    return data.pct_change().dropna()

def compute_sharpe_ratio(weights, returns, risk_free_rate):
    expected_return = np.dot(weights, returns.mean())
    volatility = np.sqrt(np.dot(weights.T, np.dot(returns.cov(), weights)))
    sharpe_ratio = (expected_return - risk_free_rate) / volatility if volatility > 0 else 0
    return expected_return, volatility, sharpe_ratio

def steepest_descent_sharpe(returns, risk_free_rate, alpha=0.01, max_iters=1000, tol=1e-6, max_weight=0.2):
    n_assets = returns.shape[1]
    weights = np.ones(n_assets) / n_assets
    iterations = 0
    for _ in range(max_iters):
        expected_return, volatility, sharpe_ratio_value = compute_sharpe_ratio(weights, returns, risk_free_rate)
        gradient = (returns.mean() - risk_free_rate) / volatility - (expected_return - risk_free_rate) * (returns.cov() @ weights) / (volatility ** 3)
        alpha = np.argmax([compute_sharpe_ratio(weights + step * gradient, returns, risk_free_rate)[2] for step in np.linspace(0.001, 0.1, 10)]) * 0.001
        new_weights = weights + alpha * gradient
        new_weights = np.clip(new_weights, 0, max_weight)
        new_weights /= new_weights.sum()
        iterations += 1
        if np.linalg.norm(new_weights - weights) < tol:
            break
        weights = new_weights
    return weights, compute_sharpe_ratio(weights, returns, risk_free_rate)

tickers = ['AAPL','NVDA', 'META', 'SNAP', 'GME', 'ZM', 'AMZN']
start_date = '2015-03-14'
end_date = '2025-03-14'
risk_free_rate = 0.04/252
stock_data = fetch_data(tickers, start_date, end_date)
returns = compute_returns(stock_data)
optimal_weights, (opt_return, opt_volatility, opt_sharpe) = steepest_descent_sharpe(returns, risk_free_rate, max_weight=0.3)
```

```python
print("Optimal Portfolio Weights:", (optimal_weights*100).round(3))
print("Optimal Expected Daily Return:", (opt_return*100).round(3))
print("Optimal Daily Volatility:", (opt_volatility*100).round(3))
print("Optimal Sharpe Ratio:", (opt_sharpe*100).round(3))
```

3 Results

The results are given as the final weights of the portfolio, as well as the expected daily returns, daily volality and Sharpe Ratio not only to enrich the user with the risk-to-reward information but also as a means for quick debugging of the source code. Below are the final results.

```
Optimal Portfolio Weights: Ticker
AAPL    30.032
AMZN     5.063
GME     17.169
META    17.703
NVDA    30.032
SNAP     0.000
ZM       0.000
Optimal Expected Daily Return: 0.232
Optimal Daily Volatility: 2.612
Optimal Sharpe Ratio: 8.285
```

As one can see, the portfolio weights are diversified. This is partly due to the maximum allocation limit that was set in the steepest descent function. By trading standards, a daily return of 0.232 and a daily volatility of 22.612 is higher than expected but not unheard off. While these minor fluctuations from the normal could be due to the choice of tickers, i.e, highly volalite stocks such as AMZN and META are offsetting the values, this means that the implementation is resulting in a high-risk to high-reward behavior. What these results most definitely portray is that the above percentages will result in the highest possible Sharpe Ratio. Furthermore, this portfolio recommendation outperforms the Treasury Bonds by around 4 percent, which means it is worthwhile to invest using this strategy for the long term.

4 Further Work and Reflection

A comparsion of various gradient descent would be enriching. Other ratios such as , the Calmar Ratio, can be maximized to find optimal trading patterns such as when to buy and where to invest one's assets during a tumulutous period of trading. To further improve the current project, I would adjust the computation of the expected returns holistically, using volume and implied volality instead of solely the mean of the daily percentage changes. Overall, this project serves to solve a real world problem among amateur investors and provide an example in implementation of a gradient descent method.