Portfolio Optimization using Mean Absolute Deviation

Jamal Omosun, Angel Ortiz

Introduction

Risk and return are the foundation of any investment decision in finance. Investors, of course, want maximum return, but risk management is what characterizes a long-term strategy. Classic models, including Markowitz's mean-variance model, employ variance as a risk proxy. This, of course, adds in quadratic terms, and the optimizations are costly in terms of required computational power and can be difficult to interpret.
In this project, we consider an alternative: the Mean Absolute Deviation (MAD) model. While variance is measured in terms of risk as the squared deviation from expected returns, MAD is measured as the average of deviations from expected returns, resulting in a linear programming problem. This enables efficient calculation and more transparent financial interpretation. Using the Julia programming language and actual stock market data, we will build a MAD-based portfolio, evaluate its performance under various constraints, and compare it with conventional methods.

History and Motivators

Portfolio optimization as a field, emerged as previously mentioned, around 70 years ago with the work of Harry Markowitz in 1952. He found that maximizing the portfolio's return can be estimated using quadratic function of an investment's expected return and variance. This methodology leads to the mean-variance model and a quadratic optimization problem. From the 60s on, there was more work put into using the two parameter mean-variance model to find the choices of a utility maximizing investor and analyzing the quality of the results through the work of Nils Hakansson in 1972, Paul Samuelson in 1969, and James Tobin in 1967. The results were positive but a paper by Samuelson in 1970 proves that the mean-variance model works best when the risk is low.

Outside of our discussion of the mean-variance model, alternatives for measuring risk outside of variance started developing in the 70s and 80s. In the 1977 paper by Peter Fishburn, it established the foundations of downside risk measures and the 1978 paper by Vijay Bawa, the use of stochastic dominance was introduced to the world of portfolio optimization. Additionally, new risk measures were introduced and detailed, specifically types of measures that can be optimized through linear programming like the Gini mean difference model suggested by Shlomo Yitzhaki in 1982 and the mean-absolute deviation model of Hiroshi Konno and Hiroaki Yamazaki in 1991. Additional developments that changed the direction of the field of portfolio optimization include the formulation of multi-period and dynamic optimization approaches as well as the introduction of the multicriteria and multiobjective methodologies for portfolio selection.

In terms of motivators for the field of portfolio optimization, as well as this project, they can be split into two broad categories, mathematical and financial. The financial motivation for the field is partially what has been previously, maximizing return on investment (RoI) while minimizing risk factors given a level of capital as well as the investment options. The other major financial motivation is to best figure out how to diversify a portfolio, given investment options such as stocks, bonds, mutual funds, and commodities among others. The mathematical motivation comes down to the work that started with Markowitz and evolved over the years, which is how to best represent risk. Markowitz's work originally started, essentially by representing the problem as a quadratic program but by efforts of many mathematicians and economists over the years, is as a linear program using the Mean Absolute Deviation (MAD) as a financial risk measure. Our project attempts to implement a MAD-based portfolio optimization model using Julia and JuMP, applied to real financial data.

The MAD Model: Mathematical Formulation

This project is concerned with applying Mean Absolute Deviation (MAD) as a financial risk measure in the process of portfolio optimization. The MAD model minimizes the average absolute difference of the asset returns from their average, using some of the previously mentioned work proposed by Konno and Yamazaki, and this leads to a linear programming problem.

Let:

- w_j: proportion of total capital invested in asset j
- r_{jt}: return of asset j at time t
- $\bar{r}_j$: average return of asset j
- z_t: absolute deviation from expected return at time t
- T: total number of time periods
- R: minimum required portfolio return

The object which minimizes the absolute mean deviation we defined as:

$$\min \frac{1}{T} \sum_{t=1}^{T} z_t$$

Subject to:

$$\begin{aligned}
z_t &\geq \sum_{j=1}^{n} w_j (r_{jt} - \bar{r}_j), && \forall t \\
z_t &\geq -\sum_{j=1}^{n} w_j (r_{jt} - \bar{r}_j), && \forall t \\
\sum_{j=1}^{n} w_j \bar{r}_j &\geq R \\
\sum_{j=1}^{n} w_j &= 1 \\
w_j &\geq 0, && \forall j
\end{aligned}$$

The first two constraints define z_t For each time period t, z_t should be greater than or equal to both the positive and negative deviations of the portfolio return from the average return. Doing this ensures that z_t is equal to the absolute value of the deviation. Therefore using these constraints, minimizing z_t minimizes the sum of absolute deviations to model minimizing risk.

The next constraint deals with minimum return. This constraint is used to have the constraint's expected return, which is the weighted average of individual asset average returns or r_j is $\leq$ R or the minimum required portfolio return.

The next constraint uses portfolio weights (w_j) to certify that the sum of individual asset average returns sums to 1. This represents our total investment j in each individual asset adding up to the capital we put in.

Finally, the last constraint ensures that all portfolio weights (w_j) are ≥ 0. This, in terms of portfolio diversification, means that there is no short-selling allowed in our portfolio.

Data and Implementation

Now that we have our model mathematically defined, we need to do two major things. We need to collect some investment data and we need to convert this linear program into a computer program using Julia. For our investment data, we decided to collect stock price data to use as a basis for our model. We collected this data by scrapping stock price data off of the internet from popular finance websites and platforms. For our program using our previous MAD model, we implemented and tested a risk-aware portfolio optimization framework using this MAD model in Julia.

We use our collected stock price data and convert it into returns:

```
returns = diff(log.(prices), dims=1)
mean_returns = mean(returns, dims=1)
```

We then set up our optimization model in JuMP:

```
model = Model(GLPK.Optimizer)
@variable(model, w[1:n] >= 0)
@variable(model, z[1:T] >= 0)
@constraint(model, sum(w) == 1)
@constraint(model, sum(w .* mean_returns) >= R)
for t in 1:T
  @constraint(model, z[t] >= sum(w[j] * (returns[t,j] - mean_returns[j]) for j=1:n))
  @constraint(model, z[t] >= -sum(w[j] * (returns[t,j] - mean_returns[j]) for j=1:n))
end
@objective(model, Min, sum(z)/T)
optimize!(model)
```

Portfolio Optimization Results

After defining the MAD model and obtaining historical stock price data, we solved the optimizer in Julia using the JuMP modeling library with GLPK as a solver. We made the first scenario a baseline for comparison with imposing the following constraints:

- Minimum expected return: $R = 0.002$
- No short-selling: $w_j \geq 0$
- Diversification cap: $w_j \leq 0.2$

This setup resulted in the following optimal allocation:

Asset	**Weight**
AAPL	0.20
MSFT	0.20
GOOG	0.60

Table 1: Optimized portfolio weights (Base case: $R = 0.002$, $w_{ub} = 0.2$)

We see that the optimizer chose the highest permissible allocation to AAPL and MSFT (reaching 20% as a limit) and assigned the rest, 60%, to GOOG. This implies that GOOG must have had the best average return, or alternatively, was having the smallest mean deviation, and hence was best under the MAD criterion.

The approach meets the risk and return requirements while illustrating the interpretability and computational characteristics of MAD-based linear programming over standard quadratic ones.

Return Threshold Sensitivity

In order to examine how return expectations influence portfolio allocation, we implemented two alternative return constraints:

- Lower threshold: $R = 0.001$

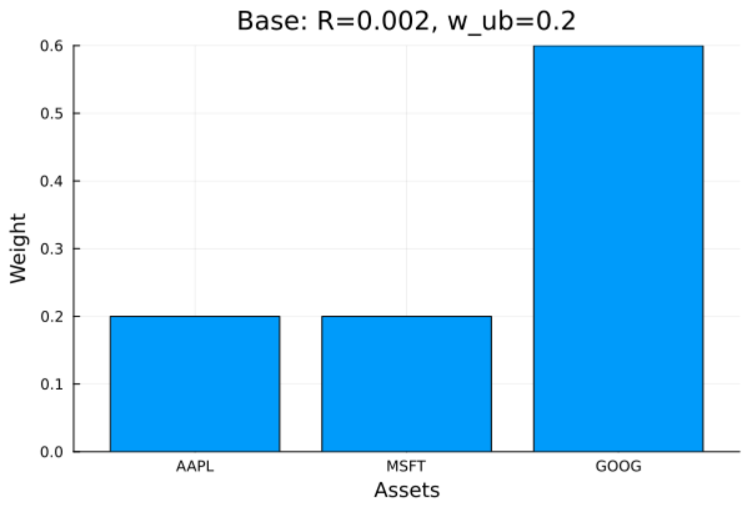

Figure 1: Optimized Portfolio Weights (Base Case)

- Higher threshold: $R = 0.004$

All other limitations were kept constant. They obtained resulting portfolio weights as:

Scenario	**AAPL**	**MSFT**	**GOOG**
$R = 0.001$	0.20	0.20	0.60
$R = 0.004$	0.20	0.20	0.60

Table 2: Portfolio weights under varying return thresholds

Notably, weights were consistent across every level tested. That outcome implies the base solution was optimized for a group of return constraints, probably due to a dominance on the part of GOOG on both expected return as well as deviation. The optimizer wasn't required to change weights, particularly given diversification limits were already in place.

Diversification Constraint Analysis

To examine how diversification constraints impact composition in a portfolio, we altered the upper bound on individual asset weights. We had a base version with a 20% investment allowance on an asset. For this, we made the constraint more rigid as follows:

- Diversification constraint: $w_j \leq 0.1$

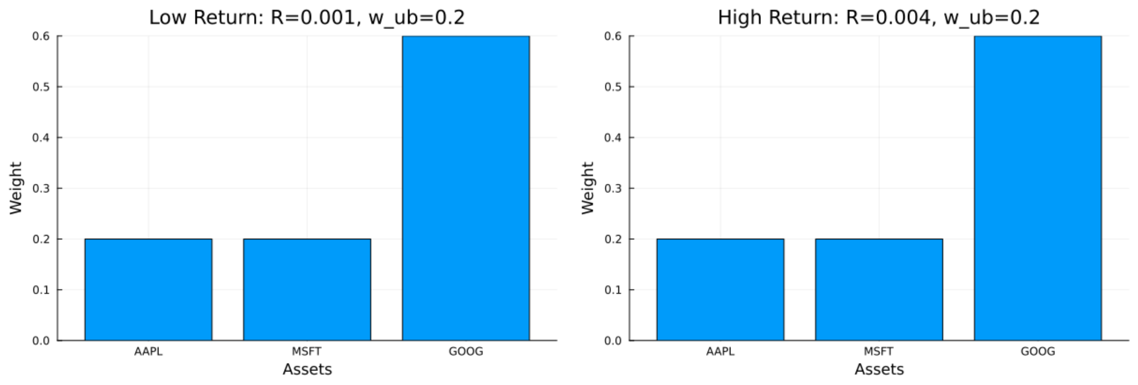

Figure 2: Left: Portfolio with $R = 0.001$; Right: Portfolio with $R = 0.004$

- Minimum expected return: $R = 0.002$

The resulting portfolio weights were:

Asset	**Weight**
AAPL	0.10
MSFT	0.10
GOOG	0.80

Table 3: Portfolio weights with tighter diversification cap ($w_{ub} = 0.1$)

Relative to the benchmark case, the optimizer invested only a minimum (10%) in MSFT and AAPL, placing all other 80% in GOOG. This is what one would expect from before: GOOG has highest return-per-unit deviation, and the optimizer takes advantage whenever it can.

Tightening the diversification constraint would cause the optimizer to allocate more thinly, but as one asset still leads, it would be allocated as much as possible of what remains. In more balanced data, it would encourage more even distribution between assets, increasing exposure and reducing concentration risk.

Discussion and Interpretation

Our results from scenarios present several important findings regarding the MAD model under different levels of financial constraints.

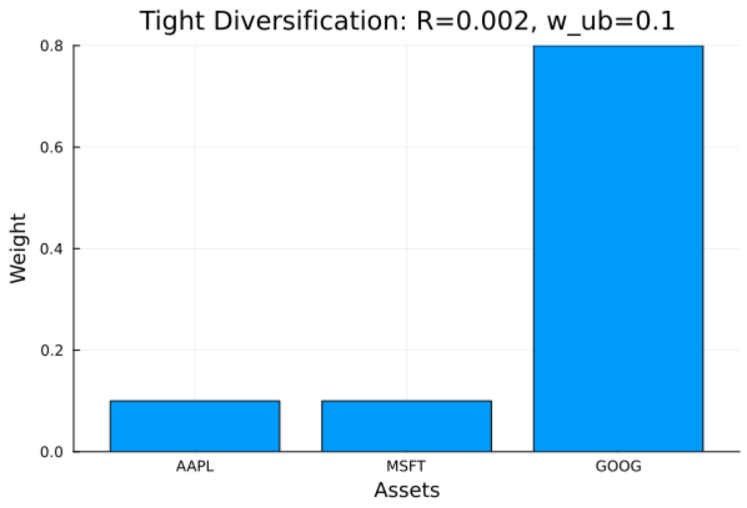

Figure 3: Optimized Portfolio Weights with $w_{ub} = 0.1$

To start, the optimizer always gravitated towards a focused allocation in GOOG. This is a clue that, in our sample, GOOG offered a great mix of high mean return with low deviation, making it, from a risk-minimization standpoint, the best asset. Even under an escalated expected return constraint (from $R = 0.001$ to $R = 0.004$), the best solution was still exactly the same. This is an indication of how the MAD objective — optimising average deviation from mean returns — tends to result in stable solutions whenever one asset significantly dominates.

Second, diversification limits contributed significantly towards changing the structure of the portfolio. When lowering the top limit per asset to 10%, the optimizer still allocated the most it could to the weaker assets (AAPL and MSFT), but assigned the rest 80% to GOOG. As it should in a linear program, if there is a significantly stronger objective contribution, one would take all weight not otherwise limited.

These findings highlight some practical benefits of using MAD-based optimization:

- **Interpretability:** Unlike mean-variance models that rely on covariance matrices and quadratic programming, MAD provides a linear and intuitive framework.
- **Efficiency:** The linear structure makes the model computationally simple, even for larger portfolios.

- **Stability:** The optimizer's tendency to maintain similar weights across return thresholds suggests robustness in portfolio construction.

Nevertheless, our findings also disclosed a shortcoming. While most restrictions acted as predicted, one situation—the loose diversification model with $w_{ub} = 0.5$—produced invalid negative weights. This, despite the deliberate enforcement of non-negativity constraints, suggests numerical instability in the GLPK solver or ill-conditioning in data for our model.

Practically, this would imply that solver selection as well as data scaling may have a subtle but significant influence on outcomes. Changing to a robust solver like Gurobi or CPLEX would potentially prevent these issues. Nevertheless, for scenarios tested with modest bounds, the MAD model behaved well and provided useful findings.

Conclusion

In this project, we optimized a portfolio with a Mean Absolute Deviation (MAD) model, using real stock market data in Julia. We reframed portfolio risk as the average deviation from expected returns, reducing the optimization problem into a linear program — making it easy both to interpret and computationally efficient.

Our base-case and alternative scenarios illustrated the consistent, interpretable allocations generated by the MAD model. Weight was focused in the most profitable, stable asset, yet still with regard for return, diversification, and non-negativity constraints. When modifying the expected return constraint, leaving everything else equal, nothing changed at optimum, showing robustness in the MAD approach under some conditions. We also demonstrated how diversification constraint enforces more widespread distribution, whereas strong assets still get preference whenever allowed.

One significant issue we faced was numerical instability in loose diversification, with GLPK generating invalid negative weights under constraints. That highlighted a practical concern with solver trustworthiness and underscores the necessity for verification when employing open-source optimization software.

In short, the MAD model is a useful alternative to mean-variance optimization, especially in those situations where it is important to be computationally efficient as well as give clear financial interpretation. Though it certainly cannot be expected to supplant more sophisticated methods in all situations, its linearity as well as performance on constrained problems make it a useful tool for portfolios sensitive to risk.

References

- Konno, H., and Yamazaki, H. (1991). "Mean-absolute deviation portfolio optimization model and its applications to Tokyo stock market." *Management Science*, 37(5), 519–531.

- Ahti Salo, Michalis Doumpos, Juuso Liesiö, Constantin Zopounidis, Fifty years of portfolio optimization, European Journal of Operational Research, Volume 318, Issue 1, 2024, Pages 1-18, ISSN 0377-2217

- JuMP.jl Documentation. `https://jump.dev/JuMP.jl/stable/`

- GLPK Solver Documentation. `https://www.gnu.org/software/glpk/`

- Historical price data collected from public sources (Yahoo Finance, Google Finance)

- Course-provided materials on financial optimization and risk modeling