Portfolio Optimization via Dimensionality Reduction and Sparsification

Shruti Sharma

Shuvodeep Dutta

Srishti Lakhotia

MSc Data Science

Chennai Mathematical Institute

Supervisor: Prof. Kavita Sutar

This report is inspired by the research conducted in ***Efficient Solution of Portfolio Optimization Problems via Dimension Reduction and Sparsification*** *by Cassidy K. Buhler and Hande Y. Benson, which explores portfolio optimization using linear algebra techniques.*

May 11, 2025

Abstract

The Markowitz mean-variance model is a cornerstone of modern portfolio theory, aiming to construct portfolios that optimally balance expected return and risk. However, when applied to large-scale financial datasets, the model suffers from computational inefficiencies due to the size and density of the covariance matrix. This report explores three techniques—dimension reduction using neural networks, reformulation using linear programming, and covariance matrix sparsification—to address this challenge. Dimension reduction focuses on identifying relevant assets using LSTM models and LP-based methods, significantly reducing problem size while maintaining portfolio quality. Sparsification replaces weak correlations with zeros while ensuring the resulting matrix remains positive semi-definite, thereby improving iteration speed. Experimental results demonstrate that these methods yield portfolios with comparable risk-return profiles to the classical approach but at a fraction of the computational cost, making them highly suitable for large-scale, real-time investment decisions.

Keywords: Portfolio Optimization, Machine learning, Mathematical programming, Large-scale optimization

Contents

1 Introduction

Portfolio optimization is a crucial problem in finance, where investors aim to construct portfolios that maximize returns while minimizing risk. The central challenge lies in determining how to allocate capital across a set of assets in a way that achieves the best possible trade-off between:

- **Expected return** - the average amount an investor anticipates earning, and
- **Risk** - typically measured by the volatility or standard deviation of returns.

An effective portfolio should not only target high returns but also ensure stability and resilience against market fluctuations.

One of the most influential approaches to the portfolio optimization problem is the **Markowitz Mean-Variance Model**, introduced by Harry Markowitz in 1952.

1.1 The Markowitz Model

The Markowitz model formulates portfolio optimization as a convex quadratic programming problem.

Let $p_{t,j}$ represent the (known) closing price for stock $j = 1, \ldots, N$ on day $t = 1, \ldots, (T-1)$. The return $x_{t,j}$ for stock $j = 1, \ldots, N$ on day $t = 2, \ldots, (T-1)$ is calculated as

$$x_{t,j} = \frac{p_{t,j} - p_{t-1,j}}{p_{t-1,j}}. \tag{1}$$

For portfolio weights $w \in \mathbb{R}^N$, the portfolio return at time $t = 2, \ldots, (T-1)$ is computed by

$$R_t = \sum_{j=1}^{N} w_j x_{t,j}. \tag{2}$$

Denoting the return matrix as $X \in \mathbb{R}^{(T-2)\times N}$, we can also write $R = Xw$. The portfolio return on day t, R_t, is a random variable with

$$\mathbb{E}[R_t] = \mu^T w, \quad \text{Var}[R_t] = w^T \Sigma w \tag{3}$$

where $\Sigma = \text{cov}(X)$ and $\mu_j = \mathbb{E}[x_{t,j}]$, $j = 1, \ldots, N$.

The Markowitz model is formulated as:

$$\begin{aligned} \max_{w} \quad & \mu^T w - \lambda w^T \Sigma w \\ \text{s.t.} \quad & e^T w = 1 \\ & w \geq 0 \end{aligned} \tag{4}$$

where e is a ones vector of appropriate size and $\lambda \geq 0$ is the risk aversion parameter. By varying the risk aversion parameter (λ), we obtain the efficient frontier - a curve representing the set of portfolios that offer the highest expected return for a given level of risk.

The Markowitz Model is conceptually simple yet widely applicable, extending beyond the domain of investing. Its structured approach to portfolio optimization makes it an efficient and valuable tool across various fields. In addition, advances in machine learning and data science techniques have significantly improved the accuracy of estimating parameters μ and Σ, thus improving the effectiveness and reliability of the model in practical applications.

1.2 Problem

Despite its theoretical elegance and widespread application in asset management, the Markowitz model faces significant computational challenges when applied to large-scale datasets with hundreds or thousands of assets. These challenges arise primarily from the large, dense covariance matrix, which leads to extensive memory and processing requirements. Efficiently solving these large-scale portfolio optimization problems is essential for real-time investment decision-making.

1.3 Solution

The paper proposes two primary strategies to accelerate the computational process:

A. Dimension Reduction:
The objective of this approach is to reduce the size of the problem by identifying and focusing on a subset of assets that are most likely to be included in optimal portfolios. Two methods are suggested for this purpose:

- **Machine Learning (LSTM Networks):** Long Short-Term Memory (LSTM) networks, a type of recurrent neural network, are employed to analyze historical asset data and predict which assets are likely to be selected in the optimal portfolio.
- **Linear Programming (LP) Approximation:** A simplified version of the portfolio optimization problem is solved using linear programming techniques. The resulting solution provides insight into which assets are most likely to be included in the final portfolio, thereby enabling a reduction in the problem's dimensionality.

B. Sparsification:
This technique aims to improve computational efficiency by transforming the covariance matrix into a sparse form, where most of the off-diagonal elements are zero. The rationale is as follows:

- Asset pairs exhibiting weak correlations (i.e., correlations close to zero) are assumed to have negligible impact on portfolio risk. Consequently, their corresponding entries in the covariance matrix are set to zero.
- The resulting sparse matrix significantly reduces computational overhead, especially for large asset universes.
- Care is taken to ensure that the sparsified matrix retains essential mathematical properties—such as positive semi-definiteness—required for the optimization process to remain valid and solvable.

2 Methodologies

2.1 Reduction using LSTM

The Long Short-Term Memory (LSTM) network architecture employed in this study is particularly well-suited for financial and other time series data due to its ability to retain and utilize long-term temporal dependencies through its internal memory mechanisms. The network is structured as a sequence of five distinct layers:

1. **Input Layer:** Accepts the input sequence X_{train}.
2. **LSTM Layer:**
   - 150 hidden nodes
   - Hyperbolic tangent function for state activation
   - Sigmoid function for gate activation
   - Input and recurrent weights initialized with Glorot and Orthogonal strategies
3. **Fully Connected Layer:**
   - Takes in the output from the previous layer
   - Reshapes the data to prepare for the classification
4. **Softmax Layer:** Converts scores into class probabilities
5. **Weighted Cross-Entropy Output:**
   - Computes the loss between the predicted class probabilities (from the Softmax layer) and the true labels
   - Accounts for class imbalance using weights

The network is designed not to predict specific stock returns or prices, but rather to classify each asset into one of two categories:

- **Class 0:** Stocks that are never selected in any optimal portfolio across a range of risk aversion preferences.
- **Class 1:** Stocks that are selected in at least one optimal portfolio.

This classification framework facilitates dimensionality reduction by identifying and excluding assets that are unlikely to contribute to optimal portfolio construction. Consequently, a reduced covariance matrix can be constructed, significantly enhancing computational efficiency.

The training process begins by splitting the historical daily returns matrix into training and testing datasets. The LSTM model is then trained using the training subset to learn the classification task. The trained model is subsequently evaluated on the test set to to assess the model's ability to generalize to unseen data.

2.2 Reforming Markowitz as Linear Programming

The traditional Markowitz model employs a quadratic objective function due to its reliance on variance as a measure of portfolio risk, which is encapsulated in the covariance matrix Σ. However, this results in a computationally intensive optimization problem, particularly when dealing with a large number of assets.

To address this, the model is reformulated into a linear programming (LP) problem by introducing two key modifications:

1. Replacement of Quadratic Risk with Absolute Deviation

The covariance matrix Σ is expressed as:

$$\Sigma = A^T A, \quad \text{where } A = X - \bar{X}, \quad \bar{X}_{ij} = \frac{1}{T-1}\sum_{t=1}^{T-1} X_{t,j}$$

Here, matrix A represents the deviation of each asset return from its mean. While the traditional Markowitz model minimizes risk via the quadratic term $w^T \Sigma w$, this approach instead minimizes the L_1-norm of deviations:

$$\left\| A^T w \right\|_1$$

This captures the average magnitude of deviations in portfolio returns from their expected value, offering a linear alternative to variance.

2. Introduction of Auxiliary Variables

Since absolute values are non-linear, auxiliary variables v_i are introduced to express the absolute deviation in a linear form. The reformulated optimization problem becomes:

$$\begin{aligned}
\max_{w} \quad & \gamma \mu^T w - \frac{1}{T}\sum_{i=1}^{T} v_i \\
\text{s.t.} \quad & -v \leq A^T w \leq v \\
& e^T w = 1 \\
& w, v \geq 0
\end{aligned}$$

This transformation preserves the essence of risk minimization while converting the problem into a tractable linear program.

This problem is then solved using the parametric self-dual simplex method with a specific pivot sequence that uncovers optimal portfolios for every value of $\gamma \geq 0$ in one pass of the method.

Advantages Over the Markowitz Model

1. **Speed:** Linear programming (LP) formulations are significantly faster to solve than quadratic programming (QP) problems. The proposed approach achieves solution times in microseconds, whereas the traditional Markowitz model typically requires milliseconds, particularly as the number of assets increases.

2. **Scalability:** The linear model demonstrates strong scalability, efficiently handling portfolios containing tens of thousands of assets. This makes it well-suited for large-scale investment universes and high-frequency trading applications.
3. **Stability:** By minimizing absolute deviation rather than variance, the model becomes less sensitive to extreme values in the return distribution. This enhances stability in the presence of noisy or anomalous data.

2.3 Sparsification by Correlation

The sparsification process involves replacing the covariance matrix in the quadratic formulation with a sparse matrix without losing essential information about the relationships between assets. The idea is to identify correlation values that fall below a predefined threshold and replace them with zero, thus simplifying the matrix.

Justification

Bühler and Benson argue that positive semidefinite (PSD) covariance matrices in financial applications are often diagonally dominant. The risk associated with individual assets (that is, their variances) typically has a more significant impact on the determination of optimal portfolio weights than the correlations between assets. In other words, asset-specific risk tends to outweigh the influence of interasset correlations when selecting portfolio weights.

Furthermore, large covariance entries, which correspond to strong and persistent correlations between assets, are likely to persist over time and have a meaningful impact on portfolio optimization. However, small correlation values can often be regarded as negligible and, therefore, replaced by zeros without significantly affecting the optimization outcome.

Thus, sparsification by correlation allows for a more efficient representation of the covariance matrix by removing insignificant correlations, leading to faster and more scalable portfolio optimization without compromising the quality of the solution.

Challenges

The sparsification method faces two major challenges:

- One of the key challenges is determining an appropriate threshold, denoted as τ, to identify insignificant correlations. The choice of τ is critical, as it directly influences which correlations are discarded and which are retained. An excessively low threshold may result in the inclusion of weak correlations that add unnecessary complexity, while a threshold that is too high may lead to the removal of important relationships between assets.
- Another challenge lies in ensuring that the sparsified covariance matrix remains positive semi-definite (PSD) after sparsification. Since the covariance matrix is crucial for maintaining the convexity of the optimization problem, any violation of the PSD condition could lead to suboptimal or undefined solutions.

Sparsification Procedure

Let θ be the dense $N \times N$ correlation matrix, and let τ be a correlation threshold such that $0 \leq \tau < 1$, derived from the unique values of θ. Define $\hat{\Sigma}(\tau)$ as the sparse $N \times N$ covariance matrix for the given threshold τ. The sparsification rule is as follows:

$$\hat{\Sigma}_{ij}(\tau) = \begin{cases} 0 & \text{if } -\tau \leq \theta_{ij} \leq \tau \\ \Sigma_{ij} & \text{otherwise} \end{cases} \qquad \forall i, j = 1, \ldots, N$$

Partial Matrix Completion Algorithm

To ensure that $\hat{\Sigma}(\tau)$ remains PSD, we apply a partial matrix completion algorithm. The goal is to fill certain entries in the matrix in a way that retains definiteness while preserving sparsity.

Algorithm 1 Partial Matrix-Completion for Threshold τ

Set $\hat{\Sigma}(\tau)$ according to Equation (7)
$\hat{n} \leftarrow$ column(s) where there exists an off-diagonal non-zero entry in $\hat{\Sigma}$
for each $i \in \hat{n}$ and $j \in \hat{n}$ **do**
    $\hat{\Sigma}_{ij}(\tau) \leftarrow \Sigma_{ij}$
end for

This procedure selectively recovers entries in $\hat{\Sigma}$ to maintain structural and mathematical integrity.
The sparsification method enhances computational efficiency without decreasing the matrix dimensions, as reducing the number of non-zero elements proves to be far more effective than simply shrinking the matrix size. The algorithm takes advantage of sparse structures by skipping over the zero entries, leading to reductions in both memory consumption and computational time.

3 Implementation

To gain practical insights into the approaches presented in the research paper, we examine both the results from the paper's implementation and the outcomes of our own implementation of the proposed methods. The primary objective of the implementation is to evaluate the computational efficiency and accuracy of these techniques when applied to a financial dataset.

Here is a comparison of tools and libraries used by Benson and Buhler and us:

	Benson and Buhler	**Us**
Programming Language(s)	Python, Matlab	Python
Optimization Solver	Gurobi Optimizer v10.0.02	HiGHS
Libraries	Matlab Deep Learning Toolbox	Scikit-Learn

Table 1: Comparison of Tools and Libraries Used

3.1 Results in Paper

The financial data used in the paper was collected from Yahoo! Finance over the dates January 23rd 2012 to December 31st 2019 for 374 firms selected from the S&P 500 index. The returns were computed using the percentage change in the closing price.

LSTM and LP Reduction Results

For reduction by neural networks, Buhler and Benson partitioned the data into the first 1499 days for training and the subsequent 500 days for testing i.e. they chose a 75/25 split. Their argument was that it would give them a large enough testing set such that the covariance matrix would not be rank-deficient.

For every solution on the efficient frontier, they recorded the CPU time, total number of iterations, and CPU time per iteration (TPI). Given that CPU time can vary on each run for reasons external to the numerical testing, they computed the efficient frontier 20 times and recorded the mean for the CPU times.

In addition, each solution on the efficient frontier also gave an expected risk, expected return and actual return.

Metric	No Reduction	LSTM Reduction	LP Reduction
Matrix Size	374×374	128×128	82×82
Total Iterations	168	71	95
Avg CPU Time	0.0183s	0.0162s	0.0165s
Avg TPI (CPU Time/it)	0.0688s	0.1439s	0.1097s
Assets Selected	65	43	30
Expected Return	0.00161	0.00145	0.00128
Actual Return	0.01490	0.00104	0.00379
Expected Risk	0.00029	0.00017	0.00017

Table 2: LSTM and LP Reduction Results

We observe that both the LSTM and LP reduction techniques result in significantly fewer iterations required for convergence. Although, on average, the CPU time per iteration was smaller for the unreduced model, the overall computational time experienced a significant reduction for the reduced models.

The dimension reduction process effectively reduces the associated risk but, in doing so, also leads to a decrease in the return, both in terms of expected and actual outcomes.

Both the reduced and unreduced models demonstrated a discrepancy between actual and expected returns, a finding that is not uncommon, as the Markowitz model is known to be sensitive to estimation errors, particularly in larger portfolios.

Among the two reduction methods, the LP-based reduction outperformed the LSTM reduction in terms of predictive accuracy, achieving 70.9% compared to 61.2% for the LSTM model.

Sparsification Results

The paper reveals that for low threshold values ($0.4 \leq \tau \leq 0.7$), partial matrix completion as a means to reestablish positive semidefiniteness requires a substantial trade-off in terms of sparsity.

However, for thresholds greater than 0.8 $\tau > 0.8$, the method successfully preserves only the most significant relationships in the risk measure, promoting both sparsity and positive semidefiniteness simultaneously.

The Markowitz model was tested under various levels of sparsification.

$$\text{Sparsity} = \frac{\text{Number of zero-valued elements}}{\text{Total number of elements}}$$

Sparsity Level	**0% (Dense)**	**50% Sparse**	**90% Sparse**	**99% Sparse**
Avg Time per Iteration	0.069 s	0.033 s	0.023 s	0.019 s
Assets Selected	65	150	286	356
Expected Return	0.0016	0.0018	0.0018	0.0018
Actual Return	0.0149	0.0182	0.0198	0.0208

Table 3: Sparsification Results

Sparsification significantly reduced the run-time per iteration. As the covariance matrix becomes sparser, the optimal solution invests in a larger number of assets.

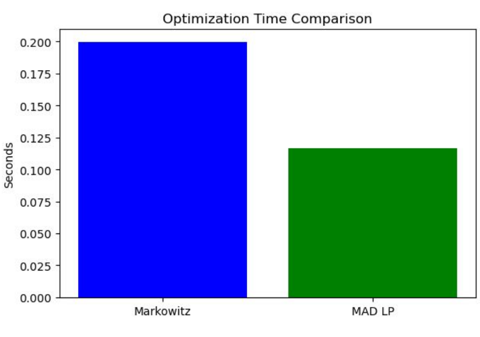
In terms of portfolio performance, the expected risk remained comparable to that of the dense portfolio, while the expected return increased as sparsity increased. This suggests that sparsification encourages greater diversification, particularly in cases where most stocks are strongly positively correlated.

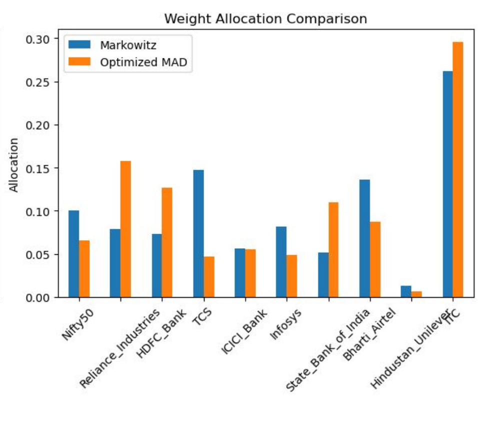
Overall, sparsification improved the performance of the optimizer without compromising on either risk or return.

3.2 Our Results

We have implemented one of the techniques proposed in the paper: Reduction by LP.

For this, we have used data of 10 stocks including Nifty50, Reliance Industries, TCS, etc. from the dates 09-05-2023 to 09-05-2025. We have found the optimal portfolio using both the classical Markowitz Model, as well as the LP reformulation. We have tabulated the running time of both models along with the weight allocation provided by each of the models. The following graphs summarise our findings:





Inferring from these, we clearly see that the LP version – Optimised MAD LP (green bar) – takes significantly less time for computation compared to the quadratic Markowitz problem (0.12 seconds vs 0.2 seconds).

Both models seem to allocate weights to various stocks in different ways.

The linearity of the LP formulation gives the model adequate reduction in time complexity theoretically, which is verified by our implementation as well. We expect this difference in computation time to be amplified with an increment in the number of stocks considered for the optimal portfolio.

4 Future Improvements

The following points highlight key areas for future research and exploration that could further enhance the effectiveness of portfolio optimization techniques:

- **Exploring Complex Sparsification Techniques:** There is potential to experiment with more complex sparsification methods that, in conjunction with techniques aimed at restoring positive semidefiniteness, if lost. Such a method is proposed in H. Liu, L. Wang, T. Zhao, *Sparse Covariance Matrix Estimation With Eigenvalue Constraints* (2014)
- **Hyperparameter Tuning in Neural Networks:** Further research can delve into hyperparameter tuning within neural network models, which could improve performance and generalizability.
- **Use of Alternate Neural Network Structures/Transformer Models:** The investigation of alternate neural network architectures, such as Transformer-based models, presents an opportunity for improving portfolio optimization strategies.
- **Testing on Other Financial Assets:** The techniques employed in this study can be extended and tested on a variety of other financial assets, including cryptocurrencies, stock derivatives, and asset mixtures. This testing would assess the robustness of these methods across different financial markets and asset classes.

5 Conclusion

In real-world investing, being able to quickly update and optimize portfolios is important, especially as the number of possible investments grows. The proposed methods let us handle much bigger problems in less time, without sacrificing much in terms of portfolio quality.

First, it is often unnecessary to evaluate every available asset, as many do not contribute meaningfully to the optimal portfolio. By focusing only on the most relevant assets, the computational burden can be substantially reduced.

Second, weak inter-asset relationships can be disregarded, which further simplifies the underlying mathematical computations and accelerates processing.

Third, machine learning techniques and simplified analytical models can assist in identifying the assets and relationships that most significantly impact portfolio performance.

Overall, these strategies allow for faster and more scalable portfolio optimization for large asset universes with little compromise in terms of risk-adjusted returns.

6 References

1. *Efficient Solution of Portfolio Optimization Problems via Dimension Reduction and Sparsification* by Cassidy K. Buhler and Hande Y. Benson.

2. *Sparse Covariance Matrix Estimation With Eigenvalue Constraints* by H. Liu, L. Wang, and T. Zhao.

3. *Portfolio Selection* by H. Markowitz. The Journal of Finance, 7, 77–91 (1952).

4. *Addressing 2030 EU Policy Framework for Energy and Climate: Cost, Risk and Energy Security Issues* by F. deLlano Paz, P.M. Fernandez, and I. Soares. Energy, 115, 1347–1360 (2016).

5. *What is the Optimal Power Generation Mix of China? An Empirical Analysis Using Portfolio Theory* by S. Zhang, T. Zhao, and B.C. Xie. Applied Energy, 229, 522–536 (2018).

6. *Extension of Portfolio Theory Application to Energy Planning Problem – The Italian Case* by M. Arnesano, A. Carlucci, and D. Laforgia. Energy, 39(1), 112–124 (2012).

7. *Risk-Based Optimal Bidding Patterns in the Deregulated Power Market Using Extended Markowitz Model* by B. Ostadi, O.M. Sedeh, and A.H. Kashan. Energy, 191, 116516 (2020).

8. *Sustainability in Supplier Selection and Order Allocation: Combining Integer Variables with Markowitz Portfolio Theory* by F. Kellner and S. Utz. Journal of Cleaner Production, 214, 462–474 (2019).

9. *Machine Learning and Portfolio Optimization* by G.Y. Ban, N. El Karoui, and A.E. Lim. Management Science, 64(3), 1136–1154 (2018).

10. *Machine Learning and Financial Planning* by J.M. Mulvey. IEEE Potentials, 36(6), 8–13 (2017).

11. *Applying Least Squares Support Vector Machines to Mean-Variance Portfolio Analysis* by J. Wang and J. Kim. Mathematical Problems in Engineering, 2019.

12. *Data Mining with Markowitz Portfolio Optimization in Higher Dimensions* by M. Bennett. SSRN 2439051 (2014).

13. *Linear Programming: Foundations and Extensions* by R.J. Vanderbei. Springer Nature, Vol. 285 (2020).

14. *Long Short-Term Memory* by S. Hochreiter and J. Schmidhuber. Neural Computation, 9(8), 1735–1780 (1997).

15. *Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions* by T. Fischer and C. Krauss. European Journal of Operational Research, 270(2), 654–669 (2018).

16. *Portfolio Optimization-Based Stock Prediction Using Long-Short Term Memory Network in Quantitative Trading* by V.D. Ta, C.M. Liu, and D.A. Tadesse. Applied Sciences, 10(2), 437 (2020).

17. *Portfolio Selection: The Effects of Uncertain Means, Variances, and Covariances* by G.M. Frankfurter, H.E. Phillips, and J.P. Seagle. Journal of Financial and Quantitative Analysis, pp. 1251–1262 (1971).

18. *Estimation for Markowitz Efficient Portfolios* by J.D. Jobson and B. Korkie. Journal of the American Statistical Association, 75(371), 544–554 (1980).