UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Portfolio Optimization Beyond Markowitz

Project internship work
in the Computer Science and Business Informatics
program Submitted by

Md Ziauddin Ridoy

Ludmilla Wagner

Emily Zerbe

Reviewer: Prof. Dr. Thomas Burkhardt, Abteilung Finanzierung, Finanzdienstleistungen und eFinance, Fachbereich 4: Institut für Management, Universität Koblenz-Landau
Second reviewer: Dipl. Inf. Heiko Neuhaus, Abteilung Finanzierung, Finanzdienstleistungen und eFinance, Fachbereich 4: Institut für Management, Universität Koblenz-Landau

Koblenz, October 2025

Kurzfassung

Hierhin kommt die deutsche Kurzfassung. Tipp: Erst am Ende schreiben.

Abstract

Hierhin kommt die englische Kurzfassung. Tipp: Erst am Ende schreiben.

Stichwrter: Wort 1, Wort 2, Wort 3

Erklrung

Ich versichere, dass ich die vorliegende Arbeit selbstndig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder hnlicher Form noch keiner anderen Prfungsbehrde vorgelegen hat und von dieser als Teil einer Prfungsleistung angenommen wurde. Alle Ausfhrungen, die wrtlich oder sinngem bernommen wurden, sind als solche gekennzeichnet.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. ja ☒ nein ☐

Der Verffentlichung dieser Arbeit im Internet stimme ich zu. ja ☐ nein ☒

Koblenz, 20. Oktober 2025

Table of Contents

Abkrzungsverzeichnis

ABC Abkrzung 1

DEF Abkrzung 2

Kapitel 1

Introduction

1.1 Background Motivation

Analyzing financial markets and optimizing portfolios is a key challenge affecting not only organizations such as banks and companies, but also private individuals [?]. Nevertheless, traditional models are reaching their limits due to the increasing availability of large amounts of high-frequency market data and unstructured information, such as that found in company reports or company news [?] [?].

One of the most important works of modern portfolio theory (MPT) is Harry Markowitz's 1952 essay, Portfolio Selection,for which he received a Nobel Prize. The essay deals with achieving a favorable balance between risk and return. Despite their enormous theoretical significance, numerous critics argue that the underlying assumptions of such models and their representation of financial markets do not correspond to reality. Critics argue, for example, that in the model assumptions, investors act rationally in an attempt to maximize profits while keeping risk low. In reality, investors are influenced and make irrational decisions, for example, when they invest hastily for fear of missing out. Further points of criticism are that investors in these models have unlimited capital and information at their disposal [?].

Against this backdrop, methods based on artificial intelligence (AI) are getting more attention. AI techniques offer a more accurate representation of the financial market. These tools can describe market structure, process large amounts of structured and unstructured information, and capture nonlinear relationships between different variables. AI tools assist portfolio managers throughout the entire process, including visualizing the market, identifying assets, compiling the portfolio, executing trades, and interpreting the results [?].

Kapitel 2

Fundamentals

2.1 Introduction to Artificial Intelligence Machine learning

There is no single definition of artificial intelligence (AI). The European Parliament defines AI as the ability of a machine to imitate human abilities, such as logical thinking, learning, planning, and creativity“[?, p. 98].

By improving forecasts, optimizing processes, and resource allocation, as well as by personalizing digital solutions for individuals and organizations. The use of AI can give companies a decisive competitive advantage and support socially and environmentally beneficial outcomes, for example, in the areas of healthcare, agriculture, etc. [?].

Machine learning is a subfield of artificial intelligence [?, p. 19]. It involves computer programs that automatically discover recurring patterns in data. These patterns are then used to categorize the data [?, p. 1].

Traditionally, three types of learning have been distinguished: supervised, unsupervised, and reinforcement. Supervised learning involves learning from sample data and linking new inputs with the correct outputs. For example, it uses historical data with target variables to improve the prediction model for stock returns. In contrast, unsupervised learning works with unlabeled data and attempts to find useful structures within it, such as through clustering or dimension reduction [?]. Reinforcement learning involves evaluating the outputs using rewards or punishments to improve the results [?].

In recent years, deep learning, i.e., the use of deep neural networks such as long short-term memory (LSTM), has made progress and proven to be powerful in predicting financial series, as it can recognize complex patterns in highly noisy market data [?].

For portfolio optimization, machine learning methods open up the possibility of mapping nonlinear dependencies between assets, improving forecasts of returns or risks, and modeling economic models. They therefore complement traditional financial mathematical models, which are often rely on linear assumptions and prove insufficient for such complex markets [?].

2.2 Genetic Algorithms

2.2.1 Introduction

Genetic algorithms are a proven method for optimizing complex problems. They are based on the basic principles of biological evolution, namely genetic variations resulting from mutation, the effects of which on subsequent generations can be positive, neutral, or negative. Through the recombination of individuals, a population gradually adapts to changing conditions. When implemented as an algorithm, different solution variants are evaluated using fitness functions in order to select and further develop the best options. This interactive process is continued until a defined termination criterion is reached. The end result is a solution that has the highest fitness in terms of the specified criteria and is therefore considered the optimal approximation to the target problem [?, p. 203].

2.2.2 Fundamental principles

A genetic algorithm can accommodate multiple input variables. In the financial sector, these variables could include asset prices, dividends, volatility, correlation parameters, and diverse asset breakdowns such as currencies, regions, asset classes, and credit ratings [?, p. 204]. Chromosomes represent potential solution candidates. In this context, a chromosome could be a portfolio allocation, with each gene describing a share of a particular asset [?, p. 204] [?, p. 7].

Each candidate solution is evaluated using a fitness function, which indicates how well the solution meets the optimization goal. One way to evaluate fitness is by combining expected return and variance [?, p. 206].

The population evolves through repeated applications of genetic operators:

Selection: Solutions with higher fitness are more likely to be selected for the next generation. This corresponds to the principle of ßurvival of the fittest.

Crossover: Two parent solutionsäre combined to produce new offspring. Parts of the chromosomes are exchanged to create new combinations of characteristics from both parent solutions.

Mutation: Individual genes are randomly altered to generate new solutions and prevent the population from becoming trapped in local optima.

[?, p. 8]

By repeatedly executing the genetic algorithm, a population is created over several generations that iteratively develops toward better solutions. The strength of this method lies in its ability to solve complex problems and demonstrate robustness [?, p. 27] [?, p. 213].

2.2.3 Variants and extensions

There are numerous extensions to the basic genetic algorithm that increase its efficiency and stability. One such extension is elitism, which involves preserving the best solutions of a generation to prevent the loss of knowledge [?].

There are numerous extensions to the basic idea of the actual genetic algorithm that increase its efficiency and stability. One of these is known as elitism, whereby the best solutions of a generation are retained so that their knowledge is not lost. Another option is steady-state selection, where only a few individual elements are replaced, for example, only the weakest individuals [?, p. 126-128].

Two other selection methods are Boltzmann selection and rank selection. With Boltzmann selection, the idea is to give weaker individuals a chance to be included in the next generation in order to increase diversity. Over time, selection becomes stricter, and the better individuals are favored. The advantage is that there is a balance between exploration and exploitation, i.e., a broad search at the beginning and a focus on the good solutions at the end [?, p. 126-128].

Rank selection solves the problem that, in normal fitness-proportional selection, individuals with extremely good fitness can dominate the population. The solution is to sort individuals by rank instead of directly using fitness as a selection criterion. The selection probability is then based on a linear distribution, for example. This prevents the strong dominance of individual individuals. These are just a few examples, there are numerous other variations [?, p. 126-128].

2.2.4 Advantages and limitations

The advantages of genetic algorithms are that they can solve complex and high-dimensional problems flexibly. They are well suited to nonlinear problems and can therefore be applied to a wide range of problem types, for example. Classic optimization methods can reach their limits in such cases [?].

Another advantage with regard to portfolio optimization is that the fitness factor can be adjusted relatively easily, thereby changing the target [?, p. 206-207].

The ability of GAs to combine both exploration of the search space and refinement of solutions is another advantage. Through selection, crossover, and mutation, new solutions are generated that combine the characteristics of successful individuals, resulting in continuous improvement of the population. Crossover operations make it possible to control genetic diversity and explore different areas of the search space, while mutation reduces the risk of getting stuck in local optima [?, p. 8097â8101].

Despite these advantages, GAs also have clear limitations. They do not guarantee a global optimum, but at best an approximately optimal solution. The quality of the results depends on parameters such as population size, mutation rate, and number of generations. For very large or complex problems, the computational effort can be high, as many chromosomes must be evaluated over numerous generations in order to adequately cover the search space [?, p. 88] [?].

Despite the many advantages of genetic algorithms, there are some key limitations that must be taken into account when applying them. One important aspect is the selection of the initial population. The size and composition of the initial population has a significant impact on the quality of the solutions found. If the population is too small, only suboptimal solutions may be found, while a very large population greatly increases the computational effort [?, p. 8114â8116].

Another common problem is premature convergence. This occurs when the algorithm comes to a standstill too early at a local optimum, resulting in the loss of genetic diversity and the suppression of certain genes. This can lead to the global optimum being missed. To counteract this, the diversity of the population must be maintained, for example, through targeted control of selection pressure or combination with local search functions such as tabu search [?, p. 8114â8116].

In addition, selecting efficient fitness functions is a challenge. The fitness function controls the selection of the best individuals, and an unsuitable or computationally intensive function can make the algorithm inefficient or distort the results. This can increase the computation time, especially with a high number of iterations [?, p. 8114â8116].

Finally, the operators mutation and crossover also play a decisive role in the performance of genetic algorithms. Without mutation, no new genetic information flows into the population, while the absence of crossover limits the exploration of the search space and can lead to local optima. The right balance between these operations is crucial, but due to the stochastic nature of GAs, it is not possible to determine an exact, universal optimal measure [?, p. 8114â8116].

2.2.5 Conclusion

Overall, despite their limitations, genetic algorithms are a robust and versatile tool. Their ability to handle complex problems without strong assumptions about the solution structure, combined with flexibility and adaptability, makes them useful in many areas of application. At the same time, however, their use requires careful parameterization and critical evaluation of the results to ensure reliable and efficient solutions.

Author: Emily Zerbe

Kapitel 3

Methodology Algorithms

3.0.1 Introduction

Genetic algorithms (GAs) are among the most important methods of evolutionary computation and are increasingly being used in finance. They offer particular advantages in portfolio optimization, as they can solve nonlinear, high-dimensional problems with restrictions that cannot be adequately addressed using traditional methods such as Markowitz optimization [?] [?]. While Markowitz assumes a strict model with normal distribution assumptions and quadratic optimization, GAs can be used to map flexible, complex objective functions and more realistic market conditions [?], [?] [?, p. 206-207].

A genetic algorithm simulates the biological evolutionary process. Populations of âsolutionsâ (portfolios) are gradually improved through selection, crossover, and mutation. Fitness evaluation based on economic targets such as sharp ratio or risk means that advantageous portfolios are given preference in further development until the most optimized portfolio can be found [?, p. 203-213].

3.0.2 Chromosome representation in portfolio optimization

Chromosome representation is a key factor in the application of genetic algorithms (GA) to portfolio optimization problems, as it determines how investment decisions are modeled. In the following, we would like to introduce different methods. Binary encoding is the most historically known and well-researched method. Solutions are represented as bit strings, i.e., as a sequence of zeros and ones. The use of binary encoding can be justified by the fact that much of the existing theory on genetic algorithms is based on fixed bit strings of equal length. Practical guidelines, such as for the selection of suitable parameters such as crossover or mutation rates, have also mostly been developed in connection with binary coding. Despite these advantages, binary encoding is impractical and unnatural for many problems. This is where many-characters and reale-valued encoding come in. Here, a chromosome consists either of an alphabet with many characters or of real numbers. In many cases, many-characters and reale-valued encoding produce better results than binary encoding. However, performance depends heavily on the specific problem and the details of the genetic algorithm used [**?**, p. 117-118].

Example:

Let's consider three stocks: A, B, and C.

A solution using binary encoding could look like this: [0, 1, 1]. This means that stock A is not included in the portfolio, while stocks B and C have been selected.

However, this representation does not provide information about the weight of each stock. This is where real-value encoding comes into place: [0, 0.4, 0.6]. In this proposed solution, it corresponds to the portfolio shares: Stock A = 0

Combining both encodings (hybrid encoding) would allow us to represent the selection of stocks and their weighting in a chromosome. This makes the solution decision-capable and directly interpretable for portfolio optimization.

3.0.3 Crossover Mechanisms

Crossover is one of the central genetic operators that plays a decisive role in creating new portfolios. It combines information from two parent solutionsto generate one or more child solutions,ïdeally with better risk-return characteristics [**?**, p. 8-9].

The classic method is single-point crossover. In this process, the chromosome is cut into two parts at a random point, and the sections are exchanged between the parents. This creates new child solutions that contain elements of both parents [**?**, p. 8-9]. However, there is one disadvantage to portfolio optimization: the weights of the assets do not automatically add up to one after the exchange. This is why additional normalization steps are necessary [?, p. 1750].

An extension of this is the two-point or multi-point crossover. Here, several intersection points are selected, which leads to greater mixing of the chromosomes and thus to greater diversity [?, p. 57].

Another option that is also used in the financial world is the arithmetic crossover. Instead of replacing the parents at a certain point, the child is formed as a weighted combination of the two parents [?, p. 10].

Example:

Parents A = [0.2, 0.3, 0.5]

Parent B = [0.4, 0.2, 0.4]

Child = 0.6 * A + 0.4 * B = [0.28, 0.26, 0.46]

Studies show that arithmetic crossover performs better and achieves better results in portfolio applications than single-point crossover and two-point crossover [?, p. 71] [?, p. 152].

3.0.4 Mutation

In addition to crossover, mutation is another essential genetic operator. While crossover combines existing information, mutation is responsible for introducing new genetic diversity. It prevents the algorithm from getting stuck in a local optimum and increases the chances of finding the global optimum [?, p. 129-130] [?, p. 471].

A simple form is uniform mutation, in which a randomly selected gene is changed within permissible limits. For example, the weight of share B in a chromosome could be increased from 0.3 to 0.32, while the other weights would have to be adjusted accordingly [?, p. 18].

The boundary mutation, on the other hand, places a weight directly on extreme values, such as the lower or upper limit of a permissible allocation (e.g., 0compiling the portfolio [?, p. 181].

In addition, there are adaptive mutation methods in which the mutation rate is dynamically adjusted. There are various possibilities here, for example, the mutation rate can be adjusted from generation to generation. Another option is to use a higher mutation rate at the beginning of the search, when the solution is still far from optimal, in order to create a broader spectrum of solutions. As the generations increase, the mutation rate is reduced in order to achieve more finer results [?, p. 181].

Mutation is therefore an important tool for ensuring that the search space is explored and genetic diversity is preserved. Choosing the right type of mutation can have a significant impact on the speed and quality of optimization [?], [?, p. 8 129-130] [?, p. 471].

3.0.5 Fitness function and fitness factors

In the context of portfolio theory and the application of genetic algorithms (GA), the choice of a suitable fitness function is of central importance, as it ultimately determines which solution is preferred for further development in the search process. A simple option is a return-based fitness function, where the goal is solely to maximize the expected portfolio return. In practice, however, more complex approaches that consider various goals and framework conditions are employed. The fitness functions are always applied to the entire portfolio that is to be optimized. The fitness factors used can be roughly divided into three groups: risk and return-related criteria, e.g., volatility limitation, minimization of fluctuations compared to a model portfolio, maximization of returns, or improvement of the Sharpe ratio. Portfolio restriction-related factors, e.g., diversification, reuse of existing investments to reduce reallocations, or consideration of preferred assets. As well as asset-specific criteria, e.g., adherence to a target allocation by asset class, sustainability requirements, regional preferences, or sentiment data. Each fitness factor is scaled to a value between 0.0 and 1.0 and assigned a weight that determines its relative importance in the optimization. Factors with a high weight have a greater influence on the result than those with a low weight. In this way, combining and weighting different factors can generate a single fitness value that enables the algorithm to objectively compare different portfolios [?, p. 206-207].

Author: Emily Zerbe

Kapitel 4

4. Portfolio Optimization Implementation

4.0.1 Markowitz Solver

Markowitz optimization is based on the assumption of rational investors who compile a portfolio according to the risk-return principle [?]. The aim is to find portfolios that deliver the maximum expected return for a given level of risk, or conversely, to minimize risk for a given level of return [?, p. 77-91].

Calculation of expected return and covariance matrix

First, the historical daily returns of the selected assets (AAPL, MSFT, GOOGL, AMZN, TSLA) are loaded and annualized to calculate the expected returns:

$\text{expected}_returns[ticker] = returns.mean() * 252$

The covariance matrix is created in the same way to capture the risk structure between the assets:

$\text{cov}_matrix = returns_df.cov() * 252$

This matrix provides the basis for all further calculations of portfolio risk.

Portfolio key figures

Key figures are calculated for any portfolio with weights w:

Expected portfolio return:

portfolio$_reeturn = np.dot(weights_array, list(self.expected_reeturns.values()))$

This results in the following formula:

$$R_p = w^T \mu$$

Risk (volatility):

portfolio$_risk = np.sqrt(np.dot(weights_array.T, np.dot(self.cov_matrix.values, weights_array)))$

This results in the following formula:

$$\sigma_P = \sqrt{(}w^T \sum w)$$

Sharpe Ratio:

sharpe$_ratio = (portfolio_return - self.risk_free_rate)/portfolio_riskifportfolio_risk > 1e - 10else0$

This results in the following formula:

$$S = (R_p - r_f)/\sigma_p$$

These functions allow the evaluation of each portfolio [?, p. 225-310].

Minimum variance portfolio (MVP)

The minimum variance portfolio is determined by solving a linear equation system via matrix inversion of the covariance matrix:

$$W_M VP = (\sum^{(} -1)\ 1)/(1^T \sum^{(} -1)\ 1)$$

This formula can be seen in the following code:

This is the inverse of the covariance matrix:

inv$_cov = np.linalg.inv(self.cov_matrix.values)$

This is vector 1:

ones = np.ones(self.num$_a ssets$)

This corresponds to the vector:

denominator = np.dot(ones.T, np.dot(inv$_c ov, ones$))

This is where the actual calculation takes place:

mvp$_w eights = np.dot(inv_c ov, ones)/denominator$

Here, $\Sigma isthecovariancematrixand 1 isavectorofones.Thisweightingminimizesportfoliovariancewi$

Tangency portfolio

The tangency portfolio maximizes the Sharpe ratio and is calculated by optimizing the excess return $(\mu - r_f)[?, p.225 - 310] : w_T angeny = (\sum^{(} -1)\ (\mu - r_{\ 1))/(^T \sum^{(-1)(-r_1))}}$

This formula can be seen in the following code:

This is the inverse of the covariance matrix:

inv$_c ov = np.linalg.inv(self.cov_m atrix.values)$

Here, the excess return is calculated:

excess$_r eturns = np.array(list(self.expected_r eturns.values())) - self.risk_f ree_r ate$

A vector containing only ones:

ones = np.ones(self.num$_a ssets$)

This is the numerator:

numerator = np.dot(inv$_c ov, excess_r eturns$)

This is the denominator:

denominator = np.dot(ones.T, numerator)

The tangency is calculated here:

tangency$_w eights = numerator/denominator$

This portfolio represents the efficient portfolio on the efficient frontier that offers the highest excess return per risk.

Efficient frontier and random portfolios

The efficient frontier is constructed by calculating the risk-minimizing portfolio for a set of target returns R*. This yields a curve of efficient portfolios. The input is the range of target returns between MVP and tangency returns. The solution is the quadratic optimization problem using Lagrange multipliers. The output is the arrays of (sp, Rp) and the corresponding weighting [?, p. 1851-1872].

Random portfolios

Thousands of random portfolios are generated for visualization purposes. These serve as a comparison and highlight which portfolios are inefficient because they may lie above the frontier [?, p. 225-310].

Visualization

The system creates various plots. The efficient frontier (blue) shows the optimal risk-return combination. The random portfolios are displayed in gray; these are the comparison set of random weightings. The MVP and tangency portfolios are marked in red and green, respectively. The weighting diagrams are bar charts that serve to illustrate the composition of the individual portfolios. The individual assets are the positioning of the individual stocks according to risk and expected return. All graphics are saved.

Results and storage

The final results MVP and tangency portfolios with their key figures and weightingsâare stored in a JSON file. This enables structured reuse, especially for comparison with alternative optimization methods such as genetic algorithms.

Author: Emily Zerbe

4.0.2 Genetic Algorithm (GA)

The genetic algorithm is an alternative optimization method. Its aim is to find a portfolio structure that maximizes the risk-return ratio, measured using the Sharpe ratio. Based on historical return data, an evolutionary optimization process is used to gradually find better portfolio compositions. In contrast to classic Markowitz optimization, this approach is not based on analytical equations, but on a heuristic search that replicates biological evolutionary principles such as selection, crossover, and mutation [?, p. 203].

Parameters

First, the global parameters that control the behavior of the algorithm are defined. These include the population size, the number of generations, the initial and final mutation rates, and the crossover rate. The HA$_{M}AX_{W}EIGHT_{P}ER_{A}SSETparameterisparticularlyimportant,asitspe$
$freeinterestrate(RISK_{F}REE_{R}ATE = 0.02)servesasareferencevalueforcalculatingtheSharperatio.$

Initialization

During initialization, the historical return data is read in, from which two key figures are calculated: the expected returns and the covariance matrix of the assets. The expected return of each asset is calculated as the mean of the daily returns, extrapolated to 252 trading days. The covariance matrix is calculated analogously from the daily returns and also scaled on an annual basis. These two variables form the mathematical basis for all further calculations in the optimization process.

Various auxiliary functions are then defined to evaluate the performance of a portfolio. As in the Markowitz Solver, the calculate$_{p}ortfolio_{m}etricsmethodcalculatestheexpectedportfolioreturn,$

Fitness function

The core of the optimization is the fitness$_{f}unctionmethod.Itcalculatesthefitnessvalueforeachindividu$

Initialization of the population

The first phase of the genetic algorithm is the initialization of the population. This involves generating a certain number of random portfolios whose weights are chosen at random but then normalized so that they add up to one:

weights = np.random.random(self.num$_{a}ssets)weights/ = np.sum(weights)$

This ensures that each portfolio is fully invested and does not contain any short positions. In the subsequent evolution phase, the population is further developed over a specified number of generations. Each generation consists of several steps that follow the biological principles of natural selection.

Selection

In the selection step, the best individuals are selected based on their fitness to serve as parents for the next generation. The code uses a so-called tournament selection for this purpose. Several individuals (in this case, three) are randomly selected from the current population, and the individual with the highest fitness wins the tournament. This method offers a good balance between randomness and performance, as it favors fitter individuals while still maintaining a certain degree of genetic diversity [**?**, p. 193].

Crossover

The next step is crossover, i.e., the combination of the genetic information of two parent portfolios. With a probability of 80two parents are selected at random and âcutâ at a random position. The first part comes from the first parent, the second from the second. The resulting offspring is then normalized so that the weights again add up to one. This step creates a new generation of portfolios that combine the characteristics of both parents and potentially offer better risk-return combinations [**?**, p. 8-9].

Mutation

Mutation also plays a central role, introducing random small changes in the portfolio weights. It prevents the algorithm from getting stuck in a local optimum too early and ensures that new areas of the solution space continue to be explored [**?**, p. 129-130] [**?**, p. 471]. The mutation is applied with a probability that gradually decreases over the course of generations, a concept known as the adaptive mutation rate [**?**, p. 181]. The mutation rate starts at 0.2 and decreases linearly to 0.05 as the algorithm progresses. This means that early in the search, there is a lot of exploration, followed later by a targeted refinement of the best solutions [**?**, p. 181]. Mathematically, the mutation rate for generation t is calculated as:

$$m_t = \ m_0 - (m_0 - \ m_f \ ) \bullet t/T$$

Where m$_0 represent the initial and m_f final mutation rates, respectively, and T is the total number of ger$

Main loop

In the main loop of the genetic algorithm (run$_{g}enetic_{a}lgorithm),thiscycleofselection,crossover,andmu$

Progress and results

The algorithm logs its progress using the Python logger and stores key intermediate results in lists such as fitness$_{h}istoryandbest_{i}ndividual_{h}istory.Thisinformationisthenusedtographicallyrepres$

In addition, the plot$_{p}ortfolio_{w}eightsmethodvisualizestheweightingstructureofthefinalportfolioas$
$orunderweightedintheoptimalportfolio.Finally,anadditionalfunction,compare_{w}ith_{m}arkowitz,allow$

Summary

In summary, this code combines the fundamentals of modern portfolio theory with a heuristic optimization method. While the Markowitz approach is based on solving a closed system of equations, the genetic algorithm uses an evolution-based search that can also be applied to nonlinear or constrained problems. The adaptive mutation rate and weight restriction attempt to strike a balance between diversification, return optimization, and realism. The implementation is modular, so parameters such as population size, mutation rate, or crossover probability can be easily adjusted to test different scenarios.

Overview

The presented implementation of portfolio optimization combines classic methods of Markowitz theory with a heuristic procedure based on a genetic algorithm (GA). The goal of both approaches is to determine efficient portfolios that maximize the ratio of return to risk as measured by the Sharpe ratio.

Markowitz optimization is based on the assumption of rational investors and uses historical return data to calculate expected returns and the covariance matrix. This allows key figures such as expected portfolio return, risk (volatility), and Sharpe ratio to be determined. On this basis, the minimum variance portfolio (MVP), which minimizes risk for a given return, and the tangential portfolio, which maximizes the Sharpe ratio, are calculated analytically. The efficient frontier is formed by a large number of possible portfolios, each of which offers an optimal combination of risk and return. Randomly generated portfolios are used for visualization and comparison.

The genetic algorithm represents an alternative, evolution-based optimization method. It simulates the principles of natural selection, crossover, and mutation to gradually improve the portfolio composition over many generations. Random portfolios are generated, evaluated, and further developed based on their fitness. An adaptive mutation rate ensures a balance between exploration and fine-tuning. In addition, a weighting restriction is introduced so that no single asset can account for more than 25its progress, stores interim results, and displays the development of fitness values. In the end, the portfolio with the highest fitness is identified as optimal. A comparison function also allows the GA results to be compared with the results of Markowitz optimization.

Author: Emily Zerbe

Kapitel 5

Experiments Results

5.0.1 Comparative Analysis

This section presents and evaluates the results of the comparison between the classic Markowitz approach and the genetic algorithm (GA). In addition, it examines how the use of AI-based forecasts affects portfolio optimization.

5.0.2 Markowitz vs. Genetic Algorithm: Methodological Differences

The Markowitz approach is considered the basis of modern portfolio theory. It pursues the goal of achieving the highest possible expected return for a given risk [?]. To this end, an optimization problem is solved using quadratic programming. This allows efficient portfolios to be determined using the minimum variance portfolio (MVP) or the tangential portfolio with the best Sharpe ratio [?].

The genetic algorithm, on the other hand, does not work analytically, but is based on biological principles such as selection and mutation [?, p. 8-9]. In the analysis, 100 possible portfolios were âevolutionarilyâ improved over 50 generations.

5.0.3 Results Comparison

The quantitative results show that both approaches lead to different but overall competitive results. The Markowitz minimum variance portfolio achieves an expected return of 17.04a risk of 27.78and a Sharpe ratio of 0.640.

These results show that GA can achieve similar efficiency to the Markowitz approach (ga$_{v}s_{m}arkowitz_{c}omparison.png$).

5.0.4 Differences in portfolio composition

There are also clear differences in the weighting of the individual stocks. The minimum variance portfolio distributes the shares rather conservatively: Microsoft (54.1positions to reduce risk (minimum$_{v}ariance_{p}ortfolio_{w}eights.png$).

The tangential portfolio, on the other hand, is significantly more risk-tolerant. It focuses heavily on Google (51.6The strong short position in Amazon (-39.1

The genetic algorithm does not use negative weighting and thus generates a balanced portfolio with 39.8

5.0.5 Convergence and efficiency of the genetic algorithm

The behavior of the genetic algorithm during optimization shows a typical development. The initial fitness was 0.627 and rose rapidly to 0.640 in the first ten generations, after which the value stabilized. This indicates an efficient search for solutions. However, there is a risk that the algorithm will âget stuckâ too early in a so-called local optimum, i.e., it will no longer find the best possible solution (ga$_{o}ptimization_{p}rogress.png$).

5.0.6 With vs. without AI forecasts

The second part of the analysis examined the influence of AI models on portfolio optimization. Two forecasting models were used for this purpose: the Chronos model for time series analysis and the TTM (time-to-maturity) model. Both models were combined using an ensemble method to obtain more robust predictions.

The comparison between historical and forecast volatility shows clear differences. For Apple, the expected volatility rises from 29Google is expected to see slightly higher volatility at 36

These results show that AI forecasts change the assessment of risk. This results in new weighting strategies that better reflect future market conditions (forecast$_r$*eturns*$_a$*nalysis.png*).

5.0.7 Change in correlations

The correlations between individual stocks are also shifting as a result of AI forecasts. Historically, correlations between technology stocks ranged from 0.43 to 0.68, indicating strong joint performance. Tesla was somewhat more independent (0.27-0.60). The integration of forecast data slightly changes these structures, which in turn opens up new diversification opportunities (correlation$_m$*atrix.png*).

5.0.8 Impact on performance

Overall, the use of AI forecasts improves portfolio optimization on several levels. First, more accurate volatility predictions lead to better risk assessments. Second, the models enable dynamic allocation that is not only based on historical data but also takes expected developments into account. Third, combining different models increases the stability of the results because outliers or distortions in individual forecasts are balanced out.

In practical terms, the combination of genetic algorithms and AI forecasts offers a good balance between performance and feasibility. While the Markowitz tangential portfolio delivers the highest Sharpe ratio, its implementation involves short selling and higher transaction costs. The genetic algorithm delivers similarly good results but does not require short positions.

Author: Emily Zerbe

Kapitel 6

Explainable Equity Research Reports

Kapitel 7

Discussion

Kapitel 8

Conclusion   Outlook

Kapitel 9

Appendix References

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Die02] Dietterich, Thomas G.: Ensemble learning. In: *The handbook of brain theory and neural networks* 2 (2002), S. 110–125

[Fam65] Fama, Eugene F.: The behavior of stock-market prices. In: *The journal of Business* 38 (1965), Nr. 1, S. 34–105

[KT13] Kahneman, Daniel ; Tversky, Amos: Prospect theory: An analysis of decision under risk. In: *Handbook of the fundamentals of financial decision making: Part I.* World Scientific, 2013, S. 99–127

[Pal19] Paley, William: *Natural Theology; Or Evidences of the Existence and Attributes of the Deity, Collected from the Appearances of Nature by William Paley, Dd...* FC and J. Rivington, 1819