DOI: 10.17323/2587-814X.2024.3.56.69

Long-term investment optimization based on Markowitz diversification

Alexander V. Kulikov
E-mail: avkulikov15@gmail.com

Dmitriy S. Polozov
E-mail: polozov.ds@phystech.edu

Nikita V. Volkov
E-mail: nikita.v.volkov@phystech.edu

Moscow Institute of Physics and Technology, Dolgoprudny, Russia

Abstract

The article introduces a long-term investment algorithm that identifies optimal solutions in lower dimensional spaces constructed through principal component analysis or kernel principal component analysis. Portfolio weights optimization is carried out using the Markowitz method. Hyperparameters of the model include window size, smoothing parameter, rebalancing period and the fraction of explained variance in dimensionality reduction methods. The algorithm presented incorporates weights regularization taking into account portfolio rebalancing transaction costs. Hyperparameters' selection is based on the Martin coefficient, which allows us to consider the maximum drawdown for the suggested algorithms. The results demonstrate that the proposed algorithm, trained from 1990 to 2016, shows higher returns and Sharpe ratios compared to the S&P 500 benchmark from 2017 to 2022. This indicates that weights optimization can improve the algorithm's performance through rebalancing.

Keywords: PCA, Kernel PCA, window size, Markowitz algorithm, Grid Search, Bayesian optimization

Citation: Kulikov A.V., Polozov D.S., Volkov N.V. (2024) Long-term investment optimization based on Markowitz diversification. *Business Informatics*, vol. 18, no. 3, pp. 56–69. DOI: 10.17323/2587-814X.2024.3.56.69

Introduction

The rapid development of financial markets is one of the most significant global trends nowadays. In the first two decades of the 21st century, market capitalization increased more than three times [1]. In the United States alone, from 2017 to 2021, the number of client investment accounts at the three largest brokerage firms – Charles Schwab, Fidelity Investments and Robinhood – nearly doubled [2]. This drive by private investors to actively seek promising investment opportunities has spurred researchers' interest in finding the optimal investment portfolio to maximize returns while accounting for financial risk.

Considerable progress in portfolio optimization was achieved with the advent of Markowitz's portfolio theory [3]. The classical Markowitz optimization problem involves finding an asset allocation in a portfolio that minimizes risk at a fixed expected return level. More generally, the search for an optimal asset allocation occurs with a given balance coefficient between risk minimization and return maximization, corresponding to the investor's degree of risk aversion.

The algorithm for finding the optimal asset allocation in the Markowitz problem is based on estimating the average returns of stocks and their covariance matrix from historical data. However, this algorithm has several limitation; in particular, the assumption that historical returns and risks of stocks will maintain their distribution in the future is not always accurate [4]. Moreover, Markowitz diversification is vulnerable to outliers due to the instability of average stock returns [5]. Outliers can also occur in the covariance matrix of returns used for risk assessment.

One approach to addressing the issues mentioned, as discussed in the literature, involves dimensionality reduction methods such as Principal Component Analysis (PCA) and Kernel Principal Component Analysis (Kernel PCA). These methods enhance the Markowitz algorithm by eliminating outliers in the covariance matrix and identifying key components to focus on when assembling a portfolio. PCA helps us to identify the most significant directions in the data space (principal components) that explain the highest variance. By reducing dimensionality, outliers have less impact on the principal components, as PCA considers the overall variability of the data. These approaches help to reduce noise in the data and improve portfolio optimization quality, as demonstrated in works [6, 7]. An example of using Kernel PCA in the Markowitz problem is presented in [8].

This paper addresses a generalized Markowitz problem that includes the commission an investor pays for rebalancing the portfolio and proposes an algorithm to solve this optimization problem. The algorithm preprocesses asset returns to remove noise, then constructs a portfolio using exponentially smoothed stock returns, and optimizes a linear combination of risk and return while accounting for the commission.

To overcome the limitations of the classical solution for the Markowitz problem, this study examines four variations of the proposed algorithm: without dimensionality reduction, with PCA, and with polynomial and Gaussian kernels in Kernel PCA.

In the aforementioned studies, the historical time horizon for determining expected returns and the covariance matrix was empirically determined. Studies [9–17] discuss approaches to predicting expected returns and the covariance matrix using machine learning methods and statistical models like GARCH. In this paper, to optimize the performance of the Markowitz algorithm, we propose optimizing the model's hyperparameters to determine the optimal portfolio based on historical data: portfolio rebalancing frequency,

window size, the number of selected companies for further analysis, and the dimensionality reduction parameter in Kernel PCA.

Given the substantial number of hyperparameters in the algorithm, traditional parameter tuning methods (GridSearch and RandomSearch) may not yield optimal results. Therefore, we also use the Bayesian optimization method based on Gaussian processes for comparison [18]. Bayesian optimization is an algorithm for finding the optimal parameter values of a function when only limited information about the function and its behavior is available. The essence of the method lies in an iterative sequence of selecting subsequent trial points based on a probabilistic model that approximates the unknown function. This method allows for finding the optimal solution using a relatively small number of iterations, as the algorithm actively adapts to information obtained from previous iterations. In the context of this study, the function to be optimized is the Martin index, introduced in [19]. Thus, for each of the four algorithm variations, parameter tuning is conducted using both random search and Bayesian optimization.

The proposed algorithms were trained on the return data of approximately 300 stocks from 1990 to 2016. To evaluate the effectiveness of each of the eight algorithm variations, the study compared various performance metrics of the resulting portfolios and benchmarked them against the S&P 500 index. This index is a classic benchmark for developing algorithms as it is considered the best indicator of large-cap company stocks [20] and reflects the overall state of the US economy [21, 22]. It has been shown in [23] that the weights in the classical Markowitz algorithm are also related to this benchmark.

The research results indicate that the most effective algorithm is the variation using the Gaussian kernel in Kernel PCA tuned through Bayesian optimization. The portfolio optimized using this method shows the highest return and the Sharpe ratio, due to the lowest drawdown during the COVID-19 pandemic outbreak, also exhibited the highest volatility compared to other variations. Additionally, it is shown that from 2017 to 2022, all proposed algorithms achieve higher returns than the S&P 500 index.

The paper is structured as follows: Section 1 presents the formulation of the optimization problem, Section 2 provides a description of the algorithm for solving this problem and its variations, Section 3 details the metrics for comparing algorithm variations, Section 4 describes the data used, Section 5 presents an analysis of the research results.

1. Optimization problem

Consider a financial market model with N assets (stocks), whose returns are random variables r_i, $i = 1, \ldots, N$. The vector of expected stock returns is denoted as $\vec{\mu} = \left(\mathsf{E}r_1, \ldots, \mathsf{E}r_N\right)^\top$, and the covariance matrix of stock returns is denoted as

$$\Sigma = \left(cov(r_i, r_j)\right)_{(i,j=1)}^{N}.$$

An investor forms a portfolio from these assets with stock weights $w_i \geq 0$, $i = 1, \ldots, N$ where the condition $w_i \geq 0$ corresponds to the prohibition of short sales. Let α be the investor's risk aversion coefficient. Then the transformed Markowitz optimization problem for this investor takes the form:

$$\begin{aligned} \min_{\vec{w}} \quad & \frac{1}{2}\vec{w}^\top \Sigma \vec{w} - \alpha \vec{\mu}^\top \vec{w} \\ s.t. \quad & \vec{1}^\top \vec{w} = 1, \\ & \vec{w} \geq \vec{0}. \end{aligned} \tag{1}$$

According to [24], this problem is equivalent to the classical Markowitz problem, except for the fixed parameters: in the classical problem, the expected return parameter is fixed, while in the transformed problem, the balance coefficient between risk minimization and return maximization is fixed. In the algorithm considered $\alpha = 0.05$ however, in general, this parameter can be adjusted, with smaller α emphasizing risk minimization and larger α emphasizing return maximization.

In this paper, we consider a more general Markowitz optimization problem, which includes a rebalancing commission λ. Thus, if P is the rebalancing period and $\vec{w}_p$ is the stock weight distribution from the previous period, the optimization problem (1) takes the form:

$$\begin{aligned} \min_{\vec{w},P} \quad & \frac{1}{2}\vec{w}^{\top}\Sigma\vec{w} - \vec{\alpha}\mu^{\top}\vec{w} - \frac{\lambda}{P}\left\|\vec{w}-\vec{w}_p\right\|^2 \\ s.t. \quad & \vec{1}^{\top}\vec{w} = 1, \\ & \vec{w} \geq \vec{0}. \end{aligned} \tag{2}$$

We define the weight distribution vector at time 0 as $\vec{w}_p = \vec{0}$.

2. Algorithm

2.1. Algorithm stages

The optimization algorithm receives input data on stock prices over a specified period. The output of the algorithm is the weights w_i – the proportions of the corresponding assets in the investor's portfolio, which are rebalanced at a specified frequency, provided to the algorithm as a hyperparameter.

The operation of the algorithm, as presented below, can be broadly divided into two parts: hyperparameter tuning of the portfolio and selection of optimal weights corresponding to these hyperparameters. The hyperparameter tuning involves selecting parameters for data preprocessing (portfolio rebalancing frequency, window size factor, the number of selected companies for further analysis) and dimensionality reduction parameters (explained variance ratio and kernel hyperparameters in Kernel PCA).

Weight selection is performed using the fit method on a training dataset of stocks over a certain period and includes the following stages.

Data filtering. In the first stage, we filter data to keep for the analysis only observations within the specified time window.

The window size is determined by two hyperparameters: *period_change_portfolio* – the portfolio rebalancing period, and *size_of_window_rank* – the window size factor. The window size is expressed as the product of these parameters. This functional relationship allows the window size to vary in conjunction with changes in the portfolio rebalancing period.

Typically, the window size factor ranges from two to five, corresponding to a window of approximately 2–5 rebalancing periods. For instance, in [25], a 5-year window is used to analyze an asset's annual beta, whereas in [26], a moving window of 1000 days is used to evaluate portfolio rebalancing parameters every 250 days. Applying window filtering allows for the consideration of the most recent and relevant data while excluding outdated information.

Stock selection for portfolio construction. At this stage, the profitability of stocks is evaluated for inclusion in a portfolio consisting of the most profitable ones.

To assign greater weight to the most recent data, we apply exponential smoothing with a parameter $\alpha \in (0, 1)$. In the algorithm, the exponential smoothing parameter is set to $\alpha = 0.99$. If the application of filtering in the previous stage is equivalent to multiplying all observations within the window by 1, and outside it by 0, then exponential smoothing can be represented as multiplying returns by $\alpha^{\Delta t}$, where Δt represents the number of trading days that have passed from the date under consideration to the present moment. This approach ensures a higher weight for recent data while preserving the influence of earlier periods, with diminishing weight as one moves further back in time.

To select stocks to be included in the portfolio, we calculate the weighted average return for each stock using exponentially smoothed weights. The calculation is performed according to the formula:

$$\overline{r_{i\tau}} = \frac{\sum_{t=\tau-T+1}^{\tau} r_{it}\alpha^{\tau-t}}{\sum_{t=\tau-T+1}^{\tau} \alpha^{\tau-t}},$$

where T is the window size, r_{it} – return of the stock i at time t.

After computing the weighted average returns, we sort the stocks in descending order, and select the top *n_top_companies* for inclusion in the portfolio. The number of selected stocks is a hyperparameter of the algorithm.

Dimensionality reduction. This step is applied only for variations of the algorithm that use PCA or Kernel PCA. As noted above, these methods enhance

the Markowitz algorithm by removing outliers in the covariance matrix and isolating components. These components are subsequently used by the algorithm in portfolio formation.

When applying these dimensionality reduction methods, the number of components to include in the model is initially determined. This number is based on the explained variance of the first 1, 2, ..., *n_top_companies* components according to the hyperparameter *var_ratio*, which dictates the proportion of explained variance. The proportion of explained variance for d components is denoted as δ_d, which is the ratio of the sum of squared deviations of the observed data from their projection onto the principal components to the sample variance of the data. A high residual variance indicates that the principal components do not explain a sizable portion of the data variability, possibly due to additional unaccounted factors in the model. The number of components d is determined as the number where $\delta_d \geqslant$ *var_ratio*, but $\delta_{d-1} <$ *var_ratio*.

The identified number of components and hyperparameters are then passed to the dimensionality reduction method, which is trained on the training data. This process reduces the data dimensionality while preserving key characteristics and removing outliers.

Kernel PCA Hyperparameters:

- *kernel* – specifies the kernel type; the algorithm considered only the Gaussian kernel "rbf":

$$k(\vec{x},\vec{y}) = \frac{e^{-\|\vec{x}-\vec{y}\|^2}}{\gamma},$$

and polynomial kernel "poly":

$$k(\vec{x},\vec{y}) = \frac{(\vec{x}^\top \vec{y} + c)^q}{\gamma};$$

- *kerneldegree* – the degree of the polynomial kernel (q in the polynomial kernel formula);
- *kernelgamma* – the scale parameter (γ in the above formulas);
- *kernelcoef0* – the constant of the polynomial kernel (c in the polynomial kernel formula).

Stock allocation in the portfolio. After forming a set of stocks for inclusion in the portfolio, the algorithm determines their weights.

The algorithm solves the problem (2), where the last term essentially acts as regularization of the weights. We set the parameter λ, which accounts for the rebalancing commission of the portfolio, to be equal to 1%.

Additionally, if dimensionality reduction methods are used, it is necessary to transition from the latent space to the original space by inverse transformation $\vec{w} = \phi^{-1}(\vec{w}_q)$ for solving the problem. Let $\vec{\bar{r}}$ be the vector of average returns in the latent space, and C be the sample covariance matrix in the latent space. Then the transformed optimization problem takes the form:

$$\begin{aligned} \min_{\vec{w}_q} \quad & \frac{1}{2}\phi^{-1}(\vec{w}_q)^\top C \phi^{-1}(\vec{w}_q) - \alpha \vec{\bar{r}}^\top \phi^{-1}(\vec{w}_q) + \frac{\lambda}{P}\left\|\phi^{-1}(\vec{w}_q) - \vec{w}_p\right\|^2 \\ s.t. \quad & \vec{1}^\top \phi^{-1}(\vec{w}_q) = 1, \\ & \phi^{-1}(\vec{w}_q) \geq \vec{0}. \end{aligned} \tag{3}$$

In the case of PCA, the inverse transformation is performed by multiplying the weights by the component matrix $\phi^{-1}(\vec{w}_q) = Q\vec{w}_q$. This is a quadratic programming problem, which is effectively solved using the *cvxpy* package [27].

In the Kernel PCA method, to find the vector $\phi^{-1}(\vec{w}_q)$ in the original space we search for the approximate preimage of the vector $\vec{w}_q$ by solving a minimization problem with Ridge regression (see [28]):

$$\phi^{-1}(\vec{w}_q) = \arg\min_{\vec{z}\in\mathbb{R}^D}\left\|\phi(\vec{z}) - \vec{w}_q\right\|^2.$$

This problem is not a quadratic programming problem and is solved less accurately using the *scipy.optimize* library [29].

The inverse transformation of the weights $\vec{w} = \phi^{-1}(\vec{w}_q)$ follows the formulas above, along with truncating weights based on a threshold value: if $w_i <$ *treshold*, then $w_i = 0$. We set the threshold value around 10^{-6} to eliminate weights that are too small to be practically achievable, as whole stock lots must be purchased.

Portfolio return estimation. After training the algorithm on the training dataset, we can construct a portfolio using the obtained weights and evaluate its performance on the test dataset. However, a portfolio with constant weights performs worse over time because the distribution of stock returns changes. Therefore, it is periodically necessary to rebalance the portfolio. Every *period_change_portfolio* trading day, we retrain the algorithm on both the training data and the known test data. The portfolio change period in the algorithm was chosen to be relatively long, ranging from three to four months to several years. This is because the Markowitz method performs well over long periods with infrequent portfolio changes. For instance, in [30], minimum-risk portfolios, i.e., classic Markowitz portfolios, were rebalanced 2–3 times a year.

We evaluate the portfolio return using the predict method, which takes the trained algorithm and test dataset as inputs. The output is a dataframe indexed by time and containing the calculated returns as a column. This allows for visualizing the results as a time series of portfolio returns and calculating economic indicators using the score method.

2.2. Hyperparameter tuning

In this study, we consider four variations of the initial algorithm: no dimensionality reduction, dimensionality reduction using PCA, and Kernel PCA with polynomial and Gaussian kernels. For each method, we optimize hyperparameters using two approaches: *RandomizedSearchCV* [31] and Bayesian optimization. This resulted in eight algorithms with different sets of hyperparameters.

The following hyperparameters were tuned in the methods for finding optimal weights:

- for the method without dimensionality reduction, only the hyperparameters of the pre-processing of the returns' matrix were selected. These parameters include:
  - the portfolio change period (ranging from three months to two and a half years);
  - the window size factor (from 2 to 5);
  - the number of top companies selected based on average returns (from 50 to 200);
- for the PCA method, in addition to the above parameters, the explained variance ratio was also tuned (from 80% to 95%);
- for Kernel PCA with a polynomial kernel, the above hyperparameters were tuned along with the polynomial kernel parameters:
  - degree (from 2 to 4);
  - constant term (from 0.5 to 1);
  - order parameter (from 0.02 to 0.04);
- for Kernel PCA with a Gaussian kernel, apart from the pre-processing parameters and the explained variance ratio, the order parameter was tuned (from 0.001 to 0.1). It is also noteworthy that, according to article [8], a portfolio formed using Kernel PCA with a Gaussian kernel is more risky. Therefore, we decided to diversify the risk less and optimized the number of *n_top_companies* based on returns in the range from 5 to 30 (for RBF and Bayesian RBF).

3. Portfolio comparison metrics

To compare portfolio optimization methods, we consider several metrics.

Portfolio value. The ratio of the current portfolio price to the initial price:

$$V_{\mathcal{P}}(\tau) = \prod_{t=1}^{\tau}\left(1+\vec{r}_t^{\top}\overrightarrow{w_t}\right).$$

Average rate of return (AR). The average investment profit earned per year. This measure is used for comparing the returns of different investment instruments. Let L be the total investment period in years, and T be the total number of trading days. The average annual return is then

$$AR_{\mathcal{P}} = \left(V_{\mathcal{P}}\left(T\right)\right)^{\frac{1}{L}} - 1.$$

VaR (value at risk) at level α. The inverse sample α-quantile of the portfolio returns

$$VaR_{\mathcal{P}}(\alpha) = -\widehat{q}_{\alpha}\left(\vec{r}_t^{\top}\vec{w}_t\right).$$

Risk premium (excess return). The gain from holding the portfolio compared to a risk-free asset. For a portfolio $\mathcal{P}$ with asset weights $\vec{w}_t$, relative returns $\vec{r}_t$, and risk-free rate rf_t at time $t = 1, ..., T$:

$$\mu_{\mathcal{P}} = \frac{1}{T}\sum_{t=1}^{T}\left(\vec{r}_t^{\top}\vec{w}_t - rf_t\right).$$

Portfolio standard deviation. The sample standard deviation of the excess return, calculated as

$$\sigma_{\mathcal{P}} = \frac{1}{T-1}\sqrt{\sum_{t=1}^{T}\left(\vec{r}_t^{\top}\vec{w}_t - rf_t - \mu_{\mathcal{P}}\right)^2}.$$

Sharpe ratio. The ratio of the risk premium to the standard deviation of the portfolio:

$$ShR_{\mathcal{P}} = \frac{\mu_P}{\sigma_{\mathcal{P}}}.$$

Relative drawdown. The relative difference between maximum portfolio value up to the current moment and its current value. If $V_{\mathcal{P}}(t)$ is the portfolio price ratio at time t to the initial price, then

$$RD_{\mathcal{P}}(t) = 1 - \frac{V_{\mathcal{P}}(t)}{\max\limits_{\tau=1,\ldots,t} V_{\mathcal{P}}(\tau)}.$$

Martin ratio (Ulcer performance index, UPI) [10]. The ratio of the portfolio's excess return to the root mean square relative drawdown:

$$PF_{\mathcal{P}} = \frac{\mu_P}{\sqrt{\overline{RD_P^2}}}.$$

The Martin ratio is used for hyperparameter tuning during the validation stage.

4. Data

The study utilized data sourced from Yahoo's database using the *yfinance* library [32]. The analyzed assets included stocks of companies listed in the S&P 500 index, as well as 500 randomly selected stocks from Yahoo's database. This asset selection ensures that the weights obtained from the algorithm are independent of the weights built solely on the benchmark index. The constructed dataset includes the daily price dynamics of 1000 stocks over the period from 1990 to 2022.

We conducted preliminary data filtering to exclude stocks with more than 10% missing values across observations. For the remaining observations with missing values, the return was estimated as the average return for the entire preceding historical period. Additionally, we excluded from the analysis stocks with high volatility, whose price changed by more than twice compared to the previous day at least once.

After filtering, 300 stocks representing various sectors were selected. These included companies from the IT sector such as Apple and Microsoft; the financial sector including JPMorgan Chase and Citigroup; and the healthcare industry including Johnson & Johnson and Pfizer. Moreover, the list included consumer goods companies such as PepsiCo, The Coca-Cola Company, McDonald's, Procter & Gamble, and Walmart.

The data obtained was divided into two periods: train period (1990–2016) and test period (2017–2022). Hyperparameter tuning was performed using cross-validation for time series. This cross-validation method involves dividing the data into sequential blocks based on time. The model is trained on all preceding data blocks and validated on the subsequent block (see *Fig. 1*). We repeat this process multiple times and using the *TimeSeriesSplit* method from *sklearn* [33].

5. Results

The configurations of the selected hyperparameters for all methods are presented in *Table 1*. It is noteworthy that for most methods, the optimal portfolio rebalancing period is approximately six months (180 days), potentially indicating a global optimality for this time interval. Interestingly, the kernel scale factor for kernel models is roughly the same, around 0.04.

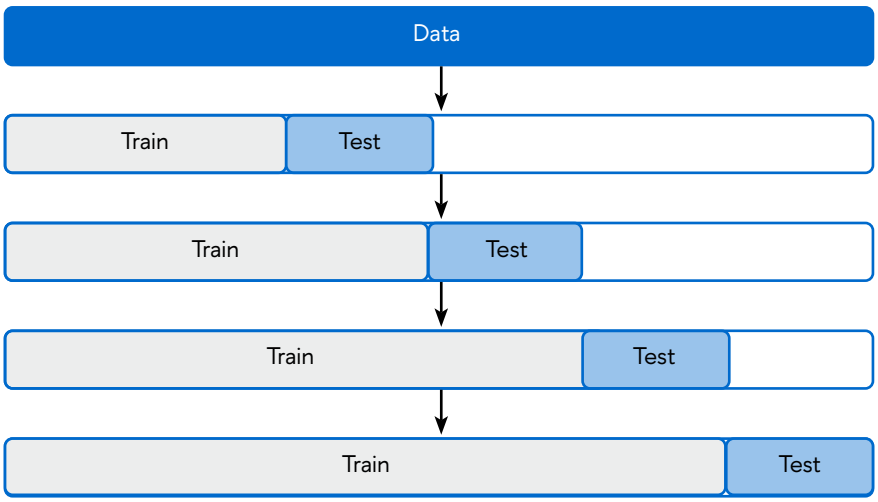

Fig. 1. Cross-validation scheme by *TimeSeriesSplit* from *sklearn*.

Table 1.

Selected hyperparameters for all 8 algorithms

	size_of_window_rank	period_change_portfolio	n_top_companies	var_ratio	kernel	kernelgamma	kerneldegree	kernelcoef0
No optimization	3.20	190	97	–	–	–	–	–
PCA	4.90	192	150	0.935	–	–	–	–
Poly KPCA	2.10	249	86	0.86	poly	0.04	4	0.77
rbf	4.90	170	27	0.91	rbf	0.04	–	–
Bayes	4.79	190	100	–	–	–	–	–
Bayes PCA	3.45	189	71	0.891	–	–	–	–
Bayes Poly KPCA	3.37	190	66	0.921	poly	0.034	2	0.844
Bayes rbf	2.44	188	17	0.764	rbf	0.044	–	–

Fig. 2. Returns on the test dataset for all 8 algorithms and the S&P 500.

After the hyperparameter tuning processes using Bayesian optimization and random search for the four methods, we compute the return on the test sample. The return graphs on the test dataset for all eight algorithms and the baseline S&P 500 benchmark are presented in *Fig. 2* and are also available on an interactive HTML chart [34].

In addition to the graphs, we calculate economic indicators from section 3. These are presented in *Table 2* for each portfolio and the S&P 500.

The study results show that the Bayes RBF method, using a Gaussian kernel in Kernel PCA, tuned with Bayesian optimization, is the most efficient algorithm. The portfolio optimized with this method had the lowest drawdown of 28.5% during the period of February–April 2020, corresponding to the first outbreak of the COVID-19 pandemic, which is 5 percentage points lower than the drawdown level of the S&P 500. This observation confirms the findings of study [8], demonstrating that algorithms with a Gaussian kernel perform better in market crisis situations. At the same time, the Gaussian kernel RBF and Bayes RBF algorithms show the highest volatility (27.7% and 28.5%, respectively). This result is due to the limited number of stocks considered, and thus, lower portfolio risk diversification.

Portfolios with the lowest volatility, apart from the S&P 500 index, were those with a polynomial kernel: Poly KPCA and Bayes Poly KPCA, with 22.5% and 23.1% volatility, respectively. This indicates the polynomial kernel's capability to effectively filter noise. Additionally, these portfolios are characterized by the best 5% VaR values (around 2.1%), which are slightly higher than the value for the S&P 500. However, the average annual returns for these portfolios are among the lowest – 12.9% and 13.0%.

PCA and Bayes PCA portfolios, based on the principal component method, showed average results in terms of risk and return. This indicates that even such classical algorithms with additional optimization outperform benchmarks on most metrics.

At the same time, the study results do not allow us to establish that Bayesian optimization or random search is more effective in hyperparameter selection.

Table 2.

Economic indicators on the test set for all 8 algorithms and the S&P 500

	S&P 500	Without optimization	PCA	Poly KPCA	rbf	Bayes	Bayes PCA	Bayes Poly KPCA	Bayes rbf
Profit factor	1.09	1.1	1.13	1.09	1.09	1.1	1.1	1.09	1.15
Sharpe ratio	0.41	0.57	0.65	0.46	0.63	0.58	0.53	0.45	0.85
Martin ratio	0.26	1.05	1.9	0.71	1.47	1.11	1.09	0.54	3.64
Annualized return	10.87%	18.75%	18.11%	12.9%	17.17%	18.71%	16.54%	13.01%	26.98%
Annualized volatility	20.26%	28.14%	23.97%	22.5%	27.71%	27.83%	26.08%	23.13%	28.51%
5% VaR	1.935%	2.815%	2.4%	2.113%	2.715%	2.817%	2.442%	2.151%	2.897%
Max drawdown	33.92%	44.41%	38.04%	43.51%	37.03%	42.99%	43.29%	34.1%	28.27%
Winning days	54.5%	54.2%	55.3%	53.6%	53.6%	54.1%	54.1%	53.4%	54.0%

While the S&P 500 is characterized by the lowest risk, its return is also low, which is reflected in the ratio indicators. Thus, the portfolios constructed, although more volatile than the S&P 500, show better results in other metrics, demonstrating that optimal portfolio rebalancing can provide additional return at the expense of risk.

To align our portfolios with the S&P 500's risk level, i.e. the average annual volatility of the S&P index, we consider the results of the Bayes RBF algorithm if part of the money is invested in this algorithm and the remainder in a risk-free asset (average annual return results are presented in *Table 3*). Note that these results allow the use of the considered algorithm over a long period, achieving higher returns than the S&P while maintaining the same risk level.

Conclusion

The study explored a long-term investment algorithm based on solving the Markowitz problem with an initial transition to a lower-dimensional space using various principal component analysis (PCA) variations. Periodic portfolio rebalancing allows for flexible responses to market changes, which is evidenced by drawdowns comparable to the benchmark in some models, despite higher risk indicators. However, portfolio rebalancing was not performed too frequently to allow the Markowitz method to demonstrate its effectiveness over sufficiently long intervals.

Moreover, optimizing the Martin coefficient during hyperparameter selection showed that the portfolios

Table 3.

Comparison of the best algorithm and S&P 500

	S&P 500	Bayes rbf	S&P-risk Bayes rbf	0% Bayes rbf	50% Bayes rbf	75% Bayes rbf	90% Bayes rbf
Annualized return	10.87%	26.98%	19.93%	2.61%	14.8%	20.89%	24.54%
Annualized volatility	20.26%	28.51%	20.26%	0.0%	14.26%	21.38%	25.66%
Sharpe ratio	0.41	0.85	0.85	0.0	0.85	0.85	0.85

achieved high returns and relatively moderate drawdowns.

Portfolios with trajectories more profitable than the benchmark, albeit slightly riskier, were also obtained. It is important to note that portfolio risk can be managed through the risk aversion coefficient, allowing investors to choose portfolios based on their risk preferences. Risk can also be regulated by the selected methods: a method with Kernel PCA and a polynomial kernel selects a less risky but also less profitable portfolio, while a method without dimensionality reduction takes on more risk with the potential for higher returns. A balanced approach in this dilemma could be the simple PCA method, which maintains a balance between risk and return.

Notably, the algorithm using Kernel PCA with a Gaussian kernel exhibited the best economic indicators, mainly due to its modest drawdown during the pandemic compared to other algorithms. It can be concluded that during crisis periods, the application of a Gaussian kernel to stock returns for transitioning to a lower-dimensional space is most effective. ■

References

1. The World Bank Group (2023) *Market capitalization of listed domestic companies. World Federation of Exchanges database*. Available at: https://data.worldbank.org/indicator/CM.MKT.LCAP.CD (accessed 20 November 2023).

2. Abramov A.E., Kosyrev A.G., Radygin A.D., Chernova M.I. (2021) Behavior of private investors in the stock markets of Russia and the US. *Russian Economic Developments*, vol. 18, pp. 11–16 (in Russian).

3. Markowitz H. (1952) Portfolio selection. *The Journal of Finance*, vol. 7, no. 1, pp. 77–91. https://doi.org/10.2307/2975974

4. Durall R. (2022) Asset allocation: From Markowitz to deep reinforcement learning. *arXiv:2208.07158*. https://doi.org/10.48550/arXiv.2208.07158

5. Best M.J., Grauer R.R. (1991) On the sensitivity of mean–variance-efficient portfolios to changes in asset means: Some analytical and computational results. *The Review of Financial Studies*, vol. 4, no. 2, pp. 315–342. https://doi.org/10.1093/rfs/4.2.315

6. Stefatos G., Hamza A.B. (2007) Cluster PCA for outliers detection in high-dimensional data. Proceedings of the *2007 IEEE International Conference on Systems, Man and Cybernetics, Montreal, QC, Canada*, pp. 3961–3966. https://doi.org/10.1109/ICSMC.2007.4414244.

7. Saha B.N., Ray N., Zhang H. (2009) Snake validation: A PCA-based outlier detection method. *IEEE Signal Processing Letters*, vol. 16, no. 6, pp. 549–552. https://doi.org/10.1109/LSP.2009.2017477

8. Peng Y., Albuquerque P.H.M., do Nascimento I.F., Machado J.V.F. (2019) Between nonlinearities, complexity, and noises: An application on portfolio selection using kernel principal component analysis. *Entropy*, vol. 21, no. 4, 376. https://doi.org/10.3390/e21040376

9. Ma Y., Han R., Wang W. (2021) Portfolio optimization with return prediction using deep learning and machine learning. *Expert Systems with Applications*, vol. 165, 113973. https://doi.org/10.1016/j.eswa.2020.113973

10. Heaton J.B., Polson N., Witte J. (2016) Deep learning for finance: Deep portfolios. *Applied Stochastic Models in Business and Industry*, vol. 33, no. 1, pp. 3–12. https://doi.org/10.2139/ssrn.2838013

11. Chen K., Zhou Y., Dai F. (2015) A LSTM-based method for stock returns prediction: A case study of China stock market. Proceedings of the *2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, USA*, pp. 2823–2824. https://doi.org/10.1109/BigData.2015.7364089

12. Yun H., Lee M., Kang Y.S., Seok J. (2020) Portfolio management via two-stage deep learning with a joint cost. *Expert Systems with Applications*, vol. 143, 113041. https://doi.org/10.1016/j.eswa.2019.113041

13. Chong E., Han C., Park F. (2017) Deep learning networks for stock market analysis and prediction: Methodology, data representations, and case studies. *Expert Systems with Applications*, vol. 83, pp. 187–205. https://doi.org/10.1016/j.eswa.2017.04.030

14. Fischer T., Krauss C. (2018) Deep learning with long short-term memory networks for financial market predictions. *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669. https://doi.org/10.1016/j.ejor.2017.11.054

15. Hoseinzade E., Haratizadeh S. (2019) CNNpred: CNN-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, vol. 129, pp. 273–285. https://doi.org/10.1016/j.eswa.2019.03.029

16. Kim J., Lee M. (2023) Portfolio optimization using predictive auxiliary classifier generative adversarial networks. *Engineering Applications of Artificial Intelligence*, vol. 125, 106739. https://doi.org/10.1016/j.engappai.2023.106739

17. Siaw R., Ofosu-Hene E., Tee E. (2017) Investment portfolio optimization with GARCH models. *Elk Asia Pacific Journal of Finance and Risk Management*, vol. 8, no. 2. Available at: https://ssrn.com/abstract=2987932 (accessed 04 August 2024).

18. Bardenet R., Bengio Y., Bergstra J., Kégl B. (2011) Algorithms for hyper-parameter optimization. Proceedings of the *Advances in Neural Information Processing Systems 24 (NIPS 2011)*.

19. Martin P.G., McCann B.B. (1989) *The investor's guide to fidelity funds*. John Wiley & Sons.

20. S&P Global (2023) *S&P 500. S&P Dow Jones Indices*. Available at: https://www.spglobal.com/spdji/en/indices/equity/sp-500/#overview (accessed 20 November 2023).

21. Beneish M.D., Whaley R.E. (1997) A scorecard from the S&P game. *Journal of Portfolio Management*, vol. 16, no. 2, 23.

22. Latham S., Braun M. (2010) Does short-termism influence firm innovation? An examination of S&P 500 firms 1990–2003. *Journal of Managerial Issues*, vol. 22, no. 3, pp. 368–382.

23. Zhang Z. (2022) Study of portfolio performance under certain restraint comparison: Markowitz Model and Single Index Model on S&P 500. Proceedings of the *2022 7th International Conference on Social Sciences and Economic Development*, pp. 1930–1938. https://doi.org/10.2991/aebmr.k.220405.323

24. Lien G. (2002) Non-parametric estimation of decision makers' risk aversion. *Agricultural Economics*, vol. 27, no. 1, pp. 75–83. https://doi.org/10.1016/S0169-5150(01)00063-9

25. Fama E.F., MacBeth J. (1973) Risk, return, and equilibrium: Empirical tests. *Journal of Political Economy*, vol. 71, pp. 607–636.

26. Paolella M.S. (2017) The univariate collapsing method for portfolio optimization. *Econometrics*, vol. 5, no. 2, 18. https://doi.org/10.3390/econometrics5020018

27. The CVXPY authors (2023) *CVXPY 1.4 Manual*. Available at: https://www.cvxpy.org/index.html (accessed 20 November 2023).

28. Bakir G., Weston J., Schölkopf B. (2003) Learning to find pre-images. Proceedings of the *Advances in Neural Information Processing Systems 16 (NIPS 2003)*.

29. The SciPy community (2023) *SciPy v1.11.4 Manual*. Available at: https://docs.scipy.org/doc/scipy/tutorial/optimize.html (accessed 20 November 2023).

30. Drenovak M., Rankovic V. (2014) Markowitz portfolio rebalancing with turnover monitoring. *Economic Horizons*, vol. 16, no. 3, pp. 207–217. https://doi.org/10.5937/ekonhor1403211D

31. scikit-learn. Machine Learning in Python (2023) *API Reference. sklearn.model_selection. RandomizedSearchCV*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html (accessed 20 November 2023).

32. GitHub (2023) *yfinance documentation*. Available at: https://yfinance.readthedocs.io/en/documentation/ (accessed 20 November 2023).

33. scikit-learn. Machine Learning in Python (2023) *API Reference. sklearn.model_selection. TimeSeriesSplit*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html (accessed 20 November 2023).

34. *Interactive chart of research results*. Available at: https://disk.yandex.ru/d/5nF7RiXKIWp1ig (accessed 20 November 2023).

About the authors

Alexander V. Kulikov

Dr. Sci. (Phys.-Math.);

Associate Professor, Moscow Institute of Physics and Technology, 9, Institutskiy Ln., Dolgoprudny 141701, Moscow Region, Russia;

E-mail: avkulikov15@gmail.com

ORCID: 0000-0002-3963-0814

Dmitriy S. Polozov

Master's Student, Moscow Institute of Physics and Technology, 9, Institutskiy Ln., Dolgoprudny 141701, Moscow Region, Russia;

E-mail: polozov.ds@phystech.edu

ORCID: 0009-0008-1108-4955

Nikita V. Volkov

Doctoral Student, Moscow Institute of Physics and Technology, 9, Institutskiy Ln., Dolgoprudny 141701, Moscow Region, Russia;

E-mail: nikita.v.volkov@phystech.edu

ORCID: 0009-0007-8434-9822