

Multi-Criteria Method for Comparing the Effectiveness of Gradient Descent Modifications on Benchmark Functions

Viktor Morozov^{1,*}, Vladyslav Deineha^{1,†} and Danylo Kovalchuk^{1,†}

¹ Taras Shevchenko National University of Kyiv, 24, Bohdan Gavrilishin Str., Kyiv, 04116, Ukraine

Abstract

In this article, an analysis of optimization methods, in particular, first-order methods, which are actively used to minimize loss functions in various problems, is presented. Optimization is a key component in both business and research, particularly in machine learning, where it plays a critical role in training models and improving their performance. The main focus was on gradient descent and its modifications, such as Momentum, Heavy Ball, and Nesterov.

First-order methods, such as Standard Gradient Descent, are effective due to their simplicity of implementation, but can suffer from oscillation and slow convergence. Modifications, such as Momentum, Heavy Ball, and Nesterov, can in some cases significantly improve the optimization process.

A review of the recent publications has confirmed the importance of using these methods to solve applied problems requiring high accuracy and efficiency. Comparison of gradient descent modifications showed that each method has its own characteristics, and the choice of the optimal approach depends on the specifics of the case. In particular, the use of methods such as Nesterov Accelerated.

Gradient can significantly reduce the training time in real-world conditions.

In the practical part of this work, several optimization methods were tested on benchmark functions, including Himmelblau, Rosenbrock, Rastrigin, Ackley, and Beale. The methods implemented included Standard Gradient Descent, Momentum, Heavy Ball, and Nesterov. Applying the Analytic Hierarchy Process enabled a thorough evaluation of each method based on key criteria: the number of iterations, average time per iteration, total execution time, and the function value at the final iteration. This structured approach allowed for a clearer, more precise comparison, aiding in the selection of the most effective method for various optimization challenges. According to the experimental results, the Heavy Ball method demonstrated the best results on most functions, while the Standard Gradient Descent and other methods showed mixed results, depending on the properties of the functions.

Keywords

Optimization, Machine learning, Standard Gradient Descent, Momentum method, Heavy Ball method, Nesterov method, Analytic Hierarchy Process

1. Introduction

Optimization is an important aspect in many fields, ranging from economics and engineering to bioinformatics and artificial intelligence [1]. It allows to find the best solutions for complex problems by minimizing or maximizing a certain function depending on the task at hand. In the real world, the optimization process is used to find the most efficient ways to use resources, improve technological processes, and solve optimal management problems [2]. Optimization plays a particularly important role in machine learning, where it helps train models based on large amounts of data. In the context of machine learning, optimization plays a key role in tuning the parameters of models such as neural networks, regression models, support vector machines (SVMs), etc. These models are used for forecasting, classification, decision making, etc. [3]. For example, neural networks can be used to solve problems related to IT projects [4].


Information Technology and Implementation (IT&I-2024), November 20-21, 2024, Kyiv, Ukraine

* Corresponding author.

† These authors contributed equally.

✉ : viktor.morozov@knu.ua (V. Morozov); thedynkan@knu.ua (V. Deineha); goose@knu.ua (D. Kovalchuk)

ORCID: 0000-0001-7946-0832 (V. Morozov); 0009-0008-3123-2302 (V. Deineha); 0009-0008-4127-3001 (D. Kovalchuk)

 © 2024 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

One of the key optimization tools in machine learning is gradient descent methods, which are used to minimize loss functions. They allow quick and efficient training of machine learning models by gradually reducing the error based on the calculation of gradients. Due to their simplicity and efficiency, gradient descent methods have become the basis of many modern optimization algorithms. However, for more complex or non-uniform loss functions, standard gradient descent may not be fast enough or stable enough.

In this paper, several modifications of gradient descent, including Momentum, Heavy Ball, and Nesterov, are reviewed and compared. These methods offer different approaches to speeding up the optimization process and increasing its resistance to local minima. The main focus will be on comparing their effectiveness in terms of the number of iterations, time to reach the minimum, average time per iteration, and accuracy of the result.

2. Using optimization in machine learning

Machine learning is a vast field of study that includes data analysis and model building techniques used to solve various problems, including time series forecasting [5]. Optimization in machine learning is a key element that determines the efficiency and accuracy of models. The process of training machine learning models is to find parameters that minimize the loss function, a mathematical representation of the errors between predicted and actual values. The goal of optimization is to find the model parameters that provide the best results on new, unknown data [6].

In most machine learning tasks, the goal is to minimize the loss function that represents the model error on the training data. To achieve this, various optimization methods are used to gradually adjust the model parameters to reduce the error. One of the most common optimization methods in machine learning is gradient descent and its modifications. Gradient descent uses the derivative of the loss function to determine the direction in which the parameters should be adjusted to reduce the value of the function. Classical gradient descent, as well as its advanced versions, such as Momentum, are among the tools used to optimize neural networks and many other models. During optimization, the choice of hyperparameters, such as learning rate, number of iterations, and other parameters that affect the speed and stability of the learning process, plays an important role. Automated methods, such as grid search, help to automate this process.

Optimization challenges in machine learning:

- High dimensionality of the parameter space. In complex models, such as deep neural networks, the number of parameters can reach millions or even billions. Optimization in such a large space is computationally challenging, and therefore first-order methods are the most appropriate due to their efficiency.
- The presence of local minima. In nonlinear models, local minima are often present, which can prevent the global minimum of the loss function from being reached. Gradient descent modifications can help solve this problem.
- Convergence speed. Optimization can be time-consuming for large models and data. Therefore, optimizers that use acceleration, such as Nesterov Accelerated Gradient, can significantly reduce model training time.

Thus, optimization is a fundamental component of machine learning that determines the success of a model in solving real-world problems. Effective use of optimization methods allows to create models capable of finding complex patterns in data and making accurate predictions.

3. Analysis of recent research and publications

This work [7] provides an overview of various gradient descent-based optimization algorithms and explains their strengths and weaknesses. The author sought to provide practical intuitions for understanding the behavior of these algorithms so that the user could apply them more effectively. The article discusses three main variants of gradient descent, of which the most popular is the mini-batch gradient descent method. The author also analyzes in detail the most common algorithms for optimizing stochastic gradient descent, including Momentum, accelerated Nesterov

gradient, and other methods, including adaptive ones. In addition, various algorithms for optimizing asynchronous SGD are investigated, as well as additional improvement strategies such as data shuffling, curriculum learning, batch normalization, and early stopping. The main conclusion of the article is that various variants and modifications of gradient descent can be adapted for certain machine learning tasks depending on the specifics of the data and model architecture. This review is useful for our article, as our paper also compares different modifications of gradient descent, including Momentum and Nesterov. The importance of choosing the right optimization strategy for a particular task is emphasized, which resonates with our analysis of the effectiveness of these methods in different settings.

This [8] article is devoted to improving the method of gradient descent with momentum, which is widely used to minimize loss functions in machine learning. The authors consider the method with the so-called Nesterov acceleration, where the gradient is calculated not at the current position in the parameter space, but at the expected position after one step. A new modification of this approach is proposed, which extends the concept of “acceleration” by estimating the gradient in positions that are several steps ahead, not just one. The degree of “superacceleration” is controlled by a new hyperparameter. The results show that the super-acceleration of the moment method is useful not only for the idealized problem, but also for the MNIST classification task using neural networks. An important conclusion is that this modification of the gradient descent with moment improves the convergence speed and efficiency of minimizing loss functions, which is especially relevant for large models in machine learning. In the context of our paper, this approach is relevant because our analysis also includes modifications of gradient descent, such as the momentum and Nesterov acceleration methods. The proposal to use the gradient from positions several steps ahead may provide additional advantages over standard methods, making this approach relevant to our study of optimization methods.

Work [9] is devoted to the use of the heavy ball moment to accelerate gradient descent in optimization problems. The authors first explain the concept of pathological curvature arising in different regions of a function and give an overview of standard gradient descent. They demonstrate the problems associated with applying gradient descent to the function given as an example. The main idea is that without a moment, the gradient descent may converge too slowly due to the characteristics of the function. To solve this problem, the moment is used to adjust the current step in the direction of the previous one, speeding up the convergence process. Using the same example, the author shows that using the moment improves the learning process and converges to the minimum much faster. This article is important for our topic because it demonstrates the heavy ball method, which is also analyzed in our study.

Paper [10] presents a new optimization method based on control theory called Controlled Gradient Descent (CGD). This approach is aimed at overcoming the shortcomings of optimization algorithms, in particular, the problems associated with the choice of an appropriate geometric structure. The effectiveness of CGD is demonstrated using various test functions, such as the Rosenbrock benchmark function, as well as a non-planar objective function and a semi-convex objective function, which are often encountered in machine learning problems. This approach is suitable for solving large-scale problems and shows promise for further development of optimization methods. The Rosenbrock function, which will also be used in our practical part of the paper, is an important tool for demonstrating the effectiveness of the method.

Paper [11] is devoted to the use of stochastic gradient descent with momentum (SGDM) for training deep neural networks (DNNs) and recurrent neural networks (RNNs), which was previously considered a difficult task due to problems with optimizing such models. The authors show that with proper initialization and careful use of parameters, both DNNs and RNNs can be trained successfully, achieving results that were previously only possible with complex second-order methods such as Hessian-Free (HF). An important aspect of the study is that improperly initialized networks cannot be trained effectively using momentum, and that the absence or poor tuning of momentum significantly reduces performance. The researchers also proved that a well-tuned momentum can successfully solve problems in deep and recurrent network training tasks that previously required the use of second-order methods. This is directly related to our topic, as our work also considers various modifications of gradient descent, including moment and Nesterov methods. The article emphasizes that even first-order methods, such as SGD with moment, can achieve optimization performance similar to second-order methods, which is especially important

for training complex models. This study confirms the importance of careful tuning of the moment parameters, emphasizing the benefits of the moment to speed up convergence and improve optimization quality.

Article [12] is devoted to the use of artificial intelligence (AI) and transfer learning techniques to automate e-waste sorting in smart cities. The authors emphasize the importance of digitalization in the context of the circular economy and consider automated e-waste processing as one of the key steps towards sustainable development. The study uses the AlexNet model with the transfer learning technique. Particular attention is paid to tuning the gradient descent optimizer and selecting the learning rate, which is directly related to our topic, since various modifications of gradient descent are also analyzed. The results show that using SGDM with a properly tuned learning rate yields an accuracy of almost 98%, which emphasizes the effectiveness of this approach. The paper also addresses overfitting issues and applies data augmentation techniques to improve model generalization, which is also useful for our study. This study demonstrates that the use of optimization algorithms such as gradient descent can improve the efficiency and accuracy of processing systems, contributing to the development of circular smart cities.

In this article [13], a new method of accelerated gradient descent is proposed that combines Taylor expansion and conjugate direction with the Nesterov accelerated gradient method. The goal was to increase the speed of convergence of optimization processes on the example of optimizing the thickness of an oil film to minimize the friction coefficient on a textured surface. Nesterov method is known for its faster convergence than standard first-order methods, but the authors improved it by including additional terms through the Taylor expansion, which allows for a more accurate approximation of the solution. The use of conjugate directions makes the method more efficient for large-scale problems, where it has advantages over the gradient descent method and is less memory intensive than Newton's method. The results of numerical experiments conducted using the finite element method in FreeFEM++ show that the proposed method has faster convergence than the Nesterov method and is capable of finding deeper solutions. Moreover, the method is easy to implement and suitable for large-scale continuous optimization problems. The useful conclusions relate to the improvement of gradient descent methods, in particular the Nesterov method. The proposed method demonstrates that a combination of techniques, such as Taylor decomposition and conjugate directions, can significantly improve the convergence rate and efficiency of optimization algorithms. This is directly related to our topic, as our paper also discusses accelerated gradient descent methods and their modifications, analyzing their effectiveness for complex optimization problems. The methods considered in this paper, such as the Nesterov method, are relevant for large and complex optimization problems, as they provide faster convergence. This approach can be particularly useful in our study to compare gradient descent modifications used in complex machine learning systems.

4. Standard Gradient Descent

Standard gradient descent is one of the simplest and most common optimization methods. Its basic idea is to gradually update the model parameters based on the gradient of the loss function. The gradient indicates the direction in which the value of the loss function decreases the fastest. The goal is to find the minimum of the function by adjusting the model parameters in this direction.

The main stages of Standard Gradient Descent:

1. Gradient calculation. At each step of the method, the gradient of the loss function is calculated for all model parameters. The gradient is a vector of partial derivatives of the loss function for each parameter, which indicates the direction of the largest increase in the function.
2. Parameter update. After calculating the gradient, the model parameters are updated according to the formula [14]:

$$\theta_{new} = \theta_{old} - \alpha * \nabla J(\theta), \quad (1)$$

where θ – are the parameters of the model, α – is the learning rate, and $\nabla J(\theta)$ – is the gradient of the loss function $J(\theta)$.

3. Learning rate. This is a key parameter of the method. If the step is too small, the optimization process will be too slow, and if it is too large, it may “jump” over the minimum and cause instability in the process. Therefore, the correct choice of α is critical.
4. Iterations. The process of updating the parameters is repeated many times until a stop is reached (by convergence criteria or after a specified number of iterations).

Advantages of Standard Gradient Descent:

- Easy to implement. Standard Gradient Descent is easy to implement because for each iteration only need to calculate the gradient and update the model parameters.
- Efficiency for smooth functions. If the loss function is smooth and convex, the method can efficiently find the global minimum.

Disadvantages of Standard Gradient Descent:

- Problems with the choice of learning rate. An incorrect choice of learning rate can lead to very slow convergence or, conversely, to divergence.
- Delay due to computation. Standard Gradient Descent requires calculating the gradient on all data at each step, which can be slow when working with large datasets.
- Oscillations in areas of saddle points. In areas where the gradient is very small or varies unevenly, Standard Gradient Descent may get “stuck” or oscillate around the minimum without reaching it effectively.

In the following parts of the article, look at the modifications of the gradient descent, such as Momentum, Heavy Ball and Nesterov, which were developed to overcome some of the shortcomings of the Standard Gradient Descent.

5. Standard Gradient Descent

The Momentum gradient descent method is an improved version of the Standard Gradient Descent method that helps speed up convergence and avoid problems associated with oscillations in the direction of the minimum. The main idea behind this method is to add a “momentum” to the parameter update - the accumulated effect of previous gradients. This allows to maintain the direction of movement even if the gradients change slightly or oscillate.

In classical gradient descent, each update of the model parameters depends only on the current gradient. In the Momentum method, inertia is added, which is accumulated based on previous updates. This allows to accelerate the movement in the direction that is constantly “supported” by gradients and smooth out oscillations when parameters fluctuate around the minimum.

Momentum algorithm. At each step, the model parameters are updated using the following formulas [15]:

1. First, the “velocity” is calculated:

$$v_t = \beta * v_{t-1} + (1 - \beta) * \nabla J(\theta), \quad (2)$$

where v_t – is the velocity at iteration t , β – is the coefficient of inertia, $\nabla J(\theta)$ – is the gradient of the loss function at the current iteration.

2. After that, the model parameters are updated to reflect the velocity:

$$\theta_{t+1} = \theta_t - \alpha * v_t, \quad (3)$$

where θ_t – are the current parameters of the model, α – is the learning rate, v_t – is the velocity used to update the parameters.

Advantages of Momentum:

- Accelerated convergence. The Momentum method allows to quickly approach the minimum in convex problems, especially in gentle sections of the function. Momentum accumulation allows not to slow down the movement in the direction where the gradient remains unchanged.
- Reducing oscillations. One of the key problems with Standard Gradient Descent is the oscillation of parameters in directions where gradients often change sign. Momentum allows to smooth out these oscillations by accumulating inertia and avoid stopping at saddle points or surfaces with small gradients.
- Better performance on curved surfaces. On difficult surfaces where the minimum is surrounded by deep valleys or hills, Momentum allows to continue in the selected direction even when the current gradient is too small or changes too quickly.

Disadvantages of Momentum:

- Setting up hyperparameters. For the method to work efficiently, it is necessary to properly configure the β parameter, which controls the level of inertia. An excessively large value of β can lead to excessive momentum accumulation and “jumping” the minimum, while a value that is too small may not provide a sufficient acceleration effect.
- Possibility of instability. If the learning rate α is chosen incorrectly, the method may become unstable, leading to divergence or oscillations around the minimum.

Gradient descent with momentum is widely used in neural networks and large machine learning models. It helps to cope more efficiently with large parameter spaces and complex loss functions, making it one of the most popular optimization methods. Momentum is also the basis for many modern modifications, such as Nesterov Accelerated Gradient, which further improve optimization performance.

The next step is to consider the Heavy Ball and Nesterov methods, which build on the ideas of Momentum, adding their own improvements for even greater optimization efficiency.

6. Heavy Ball method

The Heavy Ball method is one of the modifications of the gradient descent, which is based on similar principles as Momentum. The main idea is to add inertia to the process of updating parameters, which helps to speed up convergence and reduce oscillations. The name of the method comes from the physical analogy of moving a heavy ball on an inclined plane, where inertia helps to move in the direction of the minimum, overcoming obstacles such as local minima and plateaus.

In this method, each new step takes into account not only the current gradient, but also the previous direction of movement, which allows to “accumulate” speed and direct the movement to the minimum more efficiently. By analogy with physics, this is similar to how a heavy object continues to move under the influence of inertia even after the force (gradient) stops acting on it.

The parameters in the Heavy Ball method are updated using the following formula [16]:

$$w_{k+1} = w_k - \alpha_k * \nabla f(w_k) + \beta_k * (w_k - w_{k-1}), \quad (4)$$

where w_{k+1} - is the new value (updated parameter), w_k - is the current value (current parameter), w_{k-1} - is the previous value (previous parameter), α_k - is the step (learning rate), β_k - is the momentum parameter, $\nabla f(w_k)$ - is the gradient of the function $f(w)$ at point w_k .

Advantages of Heavy Ball:

- Speed up convergence. As in the Momentum method, inertia helps to move faster to the minimum, especially on flat parts of the function where Standard Gradient Descent can be

slow. The accumulation of speed helps to keep moving even when the gradient becomes small.

- Oscillation smoothing. The Heavy Ball method smoothes out oscillations that can occur in conventional gradient descent, especially in cases with high curvature or highly elongated minima.
- Stability in saddle points. Unlike classic gradient descents, which can get “stuck” in saddle points where the gradient is very small, Heavy Ball keeps moving forward due to inertia.

Disadvantages of Heavy Ball:

- The need for careful tuning. The method requires the correct choice of both the learning rate α and the inertia coefficient β . Incorrect choice of parameters can lead to instability or too slow convergence.
- Jumping over the minimum. If the inertia is too large, the method may “jump” over the minimum and start oscillating around it instead of achieving stable convergence.
- Delays are possible in complex landscapes. Although the method works well on smooth functions, in very complex landscapes with numerous local minima, inertia can prevent the fastest possible finding of the global minimum.

The Heavy Ball method is used in problems that require faster convergence than Standard Gradient Descent. It is suitable for problems with a large number of parameters, such as neural network optimization, especially in situations where the loss function has a complex shape with wide minima or plateaus. Heavy Ball is a good option for problems where the speed of convergence is important, but the stability of the optimization process cannot be sacrificed.

In the next part, consider the Nesterov Accelerated Gradient method, which is another advanced version of the gradient descent, based on the ideas of momentum and inertia, but adds its own features for even greater efficiency.

7. Heavy Ball method

The Nesterov Accelerated Gradient (NAG) method is an advanced modification of the gradient descent based on the idea of Momentum, but with additional acceleration. The main innovation of the method is that it updates the parameters not only based on the current gradient, but also taking into account the predicted future state. This allows the model to take into account where it will move in advance and adjust the steps more accurately.

Unlike the classical Momentum method, where the gradient is calculated based on current parameters, the Nesterov method calculates the gradient based on the future position. This allows the algorithm to be more “proactive” and better adapt to the landscape of the loss function.

In a physical analogy, this is similar to how a heavy ball (which moves due to inertia) would not only roll down an inclined plane, but also “predict” where the next slope will be to more effectively adjust its movement.

Model parameters in Nesterov method can be updated in the following steps [17]:

$$v_t = \beta * v_{t-1} + (1 - \beta) * \nabla f(\theta_{fp}), \quad (5)$$

$$\theta_{t+1} = \theta_t - \alpha * v_t, \quad (6)$$

where θ_t — are the parameters at iteration t , v_t — is the velocity at iteration t , α — is the learning rate, β — is the momentum term, $\nabla f(\theta_{fp})$ — is the gradient at the future position.

Advantages of the Nesterov method:

- Faster convergence. Since the method uses the predicted position to calculate the gradient, it makes better use of the information about the direction of movement, which contributes to faster convergence compared to the classic Momentum.

- Better adaptation to the function landscape. Nesterov method is more sensitive to changes in the curvature of the function, as it “looks ahead” and adjusts the direction of movement to match the predicted position. This allows it to better cope with complex landscapes with numerous local minima.
- Oscillation reduction. Similar to Momentum, Nesterov method helps reduce oscillations, especially in problems with large curvature or saddle points. However, due to the “prediction” of the future position of the parameters, this method does it even more effectively.
- Better stability in complex problems. Nesterov method is less prone to situations where parameters “jump” over the minimum or get stuck in local minima, which makes it more stable in complex optimization problems.

Disadvantages of the Nesterov method:

- Complicated computation. Although the method provides better convergence, it requires additional computations to estimate the predicted position of the parameters. This can increase computational complexity, especially when working with large models.
- Adjusting hyperparameters. As with Momentum, Nesterov method requires careful tuning of the α (learning rate) and β (inertia coefficient) parameters. Incorrect settings can lead to discrepancies or slow convergence.

The Nesterov Accelerated Gradient method is widely used in neural networks and complex machine learning models where loss functions have a rough or complex landscape. It can significantly speed up training, especially in problems where classical gradient descent methods face difficulties in stability and convergence speed.

Nesterov method is one of the most popular optimization algorithms due to its ability to accelerate learning and efficient use of gradient direction information. It is often used in combination with other optimization methods to provide even more efficient and faster model training.

8. Experimental research

In the practical part of the paper, an experimental comparison of the mentioned optimization methods, such as Standard Gradient Descent, Momentum method, Heavy Ball method, and Nesterov method, was conducted on various mathematical functions. The main goal of the study was to investigate the effectiveness of these methods on complex functions using the Analytic Hierarchy Process. The evaluation of effectiveness was conducted based on several key criteria: the number of iterations, average time per iteration, total execution time, and the function value at the final iteration. The following functions were used for this purpose:

- Himmelblau's function is a multimodal nonlinear function known for its four global minima [18].
- The Rosenbrock function is a standard test function for optimization, with a hard-to-find global minimum [19].
- The Rastrigin function is a strongly oscillating function with many local minima, which is used to test the stability of optimization methods [20].
- The Ackley function is a nonlinear function with a large number of local minima, which poses difficulties for gradient methods [21].
- The Beale function is a three-dimensional function that has one global minimum and several local ones, representing a problem with high nonlinearity [22].

Each of the methods was applied to solving optimization problems. For each combination of method and function, the main criteria values were measured, and then the values were normalized using the minimax method. If the minimum and maximum values of the criteria coincided (which

could happen, for example, if the methods did not reach the minimum), the normalized values were set to 1.

The criteria are as follows:

- I – the number of iterations,
- T_{avg} – the average time per iteration,
- T_{total} – the total execution time,
- $f(x_{final})$ – the value of the function at the last iteration.

Perform normalization of values:

$$\hat{I} = (I - I_{min}) / (I_{max} - I_{min}), \quad (7)$$

$$\widehat{T_{avg}} = (T_{avg} - T_{avg_{min}}) / (T_{avg_{max}} - T_{avg_{min}}), \quad (8)$$

$$\widehat{T_{total}} = (T_{total} - T_{total_{min}}) / (T_{total_{max}} - T_{total_{min}}), \quad (9)$$

$$\widehat{f(x_{final})} = (f_{final} - f_{final_{min}}) / (f_{final_{max}} - f_{final_{min}}). \quad (10)$$

In the practical part of the study, the Analytic Hierarchy Process was applied to evaluate and compare the effectiveness of different optimization methods: Standard Gradient Descent, Momentum method, Heavy Ball method and Nesterov method. This approach has allowed to systematically consider optimization methods in terms of several criteria, which contributed to a more informed choice of the best method for specific problems. Stages of application of the analytic hierarchy process are presented below.

Building a tree of alternatives: In the context of this stage, a hierarchical structure was formed, where at the top level was the overall objective of the study - to evaluate the effectiveness of optimization methods. Below it were the key criteria: number of iterations, average time per iteration, total execution time and function value at the last iteration. At the lowest level of the tree were the alternatives, which are optimization methods.

Constructing a matrix of pairwise comparisons of criteria: A matrix of pairwise comparisons of criteria was created for further analysis. Each criterion was ranked relative to the others in terms of its importance for achieving the overall objective. This allowed the priorities of the criteria to be fixed and their weight to be taken into account in subsequent calculations.

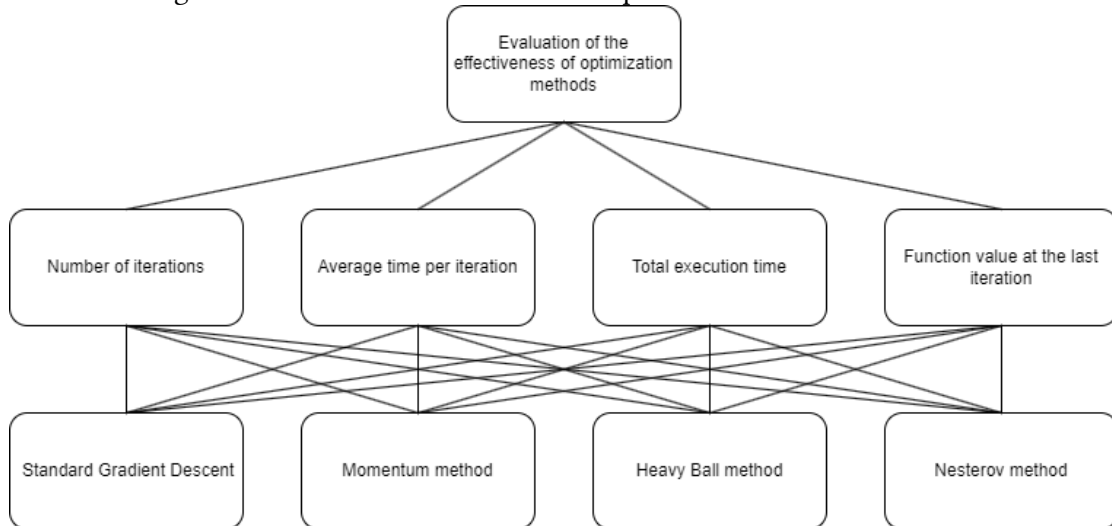


Figure 1: Example of an alternatives tree

Construction of matrices of pairwise comparisons of alternatives: Next, a matrix of pairwise comparisons of alternatives was constructed for each criterion. In this matrix, the optimization methods were evaluated using normalized values of the criteria. This approach allowed each method to be compared in the context of all criteria.

Table 1

Table of pairwise comparisons of criteria

	Iterations	Avg Iteration Time (s)	Total Time (s)	Function Value
Iterations	1	3	0.166667	0.125
Avg Iteration Time (s)	0.333333	1	0.142857	0.111111
Total Time (s)	6	7	1	0.5
Function Value	8	9	2	1

Matrix Analysis: In this step, the matrices obtained were analyzed, resulting in a vector of criteria weights and vectors of alternative weights for each criterion.

Determination of weights of alternatives: Based on the weights obtained in the previous step, the final weights of the alternatives in terms of achieving the objective were determined. For this purpose, a calculation was made using the following formula [23]:

$$W_k = \sum_i^n (w_i * p_{ik}), \quad (11)$$

where W_k – total weight of method k, w_i – weight of the i-th criterion, p_{ik} – priority of method k by criterion i, n - total number of criteria.

Thus, the final weight W_k was calculated for each method, where a higher value indicates a better result. After completing the calculations, visualization of the trajectories along which the methods moved to the minimum of each function was performed, which allowed to get a visual representation of the behavior of different optimization methods in each case. This information is presented below in the form of tables and figures.

Table 2

Comparison of optimization methods on the Himmelblau function

	Iterations	Avg Iteration Time (s)	Total Time (s)	Function Value	Weight
Standard Gradient Descent	1	0	1	1	0.065604
Momentum	0.443787	0.121821	0.488829	0.16422	0.264477
Heavy Ball	0	1	0	0	0.528656
Nesterov	0.579882	0.885641	0.813413	0.28423	0.141263

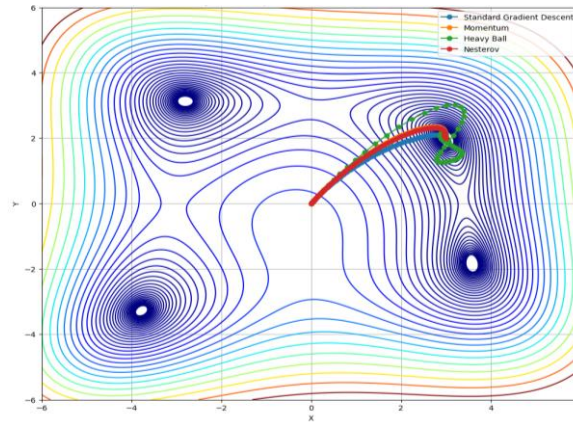
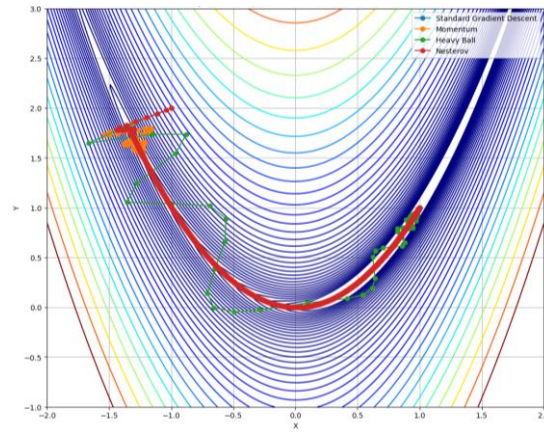
**Figure 2:** Paths taken by different optimization methods on the Himmelblau function

Table 3

Comparison of optimization methods on the Rosenbrock function

	Iterations	Avg Iteration Time (s)	Total Time (s)	Function Value	Weight
Standard Gradient Descent	1	0	0.669161	1	0.138973
Momentum	0.987661	0.578796	0.935468	0.99283	0.088688
Heavy Ball	0	0.657975	0	0	0.699239
Nesterov	0.994011	1	1	0.99358	0.073100

**Figure 3:** Paths taken by different optimization methods on the Rosenbrock function**Table 4**

Comparison of optimization methods on the Rastrigin function

	Iterations	Avg Iteration Time (s)	Total Time (s)	Function Value	Weight
Standard Gradient Descent	0	1	0	0	0.498951
Momentum	0.912088	0	0.765187	1	0.078360
Heavy Ball	1	0.082426	1	0.018886	0.244642
Nesterov	0.648352	0.124508	0.521598	0.341416	0.178047

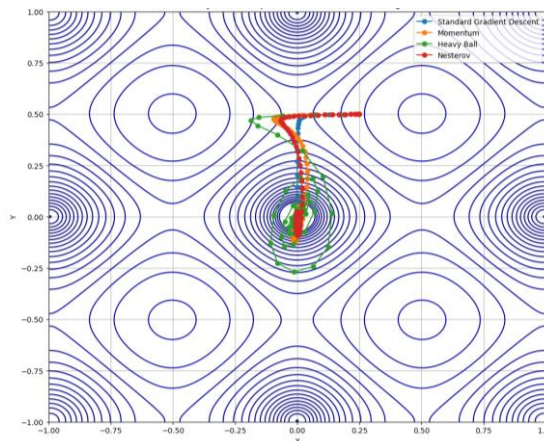
**Figure 4:** Paths taken by different optimization methods on the Rastrigin function

Table 5
Comparison of optimization methods on the Ackley function

	Iterations	Avg Iteration Time (s)	Total Time (s)	Function Value	Weight
Standard Gradient Descent	1	0	0	0.142444	0.357109
Momentum	1	0.359053	0.356137	0.03951	0.292598
Heavy Ball	1	0.319435	0.395656	1	0.112995
Nesterov	1	1	1	0	0.237298

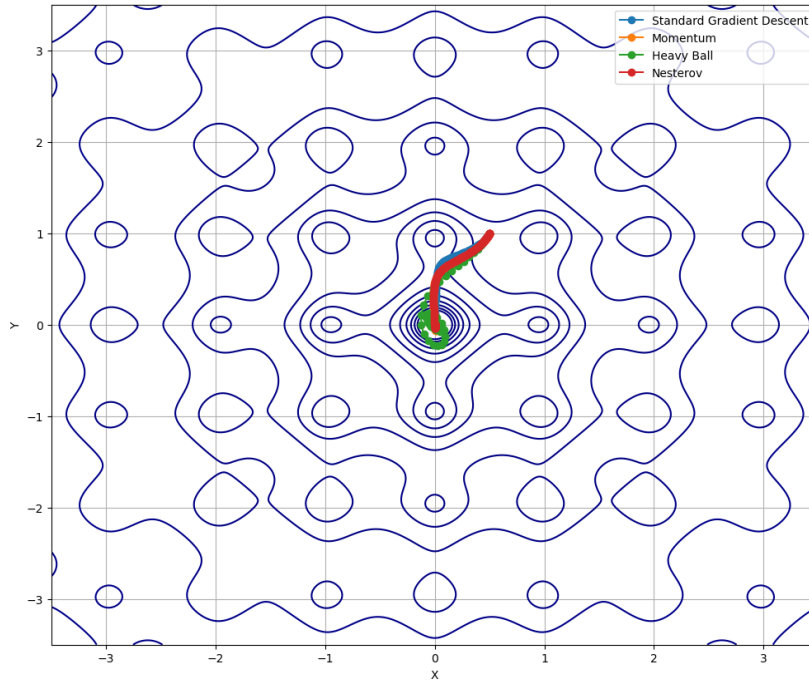


Figure 5: The paths taken by different optimization methods on the Ackley function

Table 6
Comparison of optimization methods on the Beale function

	Iterations	Avg Iteration Time (s)	Total Time (s)	Function Value	Weight
Standard Gradient Descent	1	0	0.676134	1	0.132482
Momentum	0.997308	0.778712	0.876355	0.99611	0.086562
Heavy Ball	0	0.679081	0	0	0.707523
Nesterov	0.997667	1	1	0.996426	0.073433

Based on these tables, several conclusions can be made about the results of the optimization methods applied to the Himmelblau, Rosenbrock, Rastrigin, Ackley, and Beale functions. Analyze each method on different functions.

- Himmelblau function: The Heavy Ball method has the highest weight of 0.528656, indicating its superiority in terms of performance on this function. The Standard Gradient

Descent has the lowest weight of 0.065604, showing weaker results compared to the other methods.

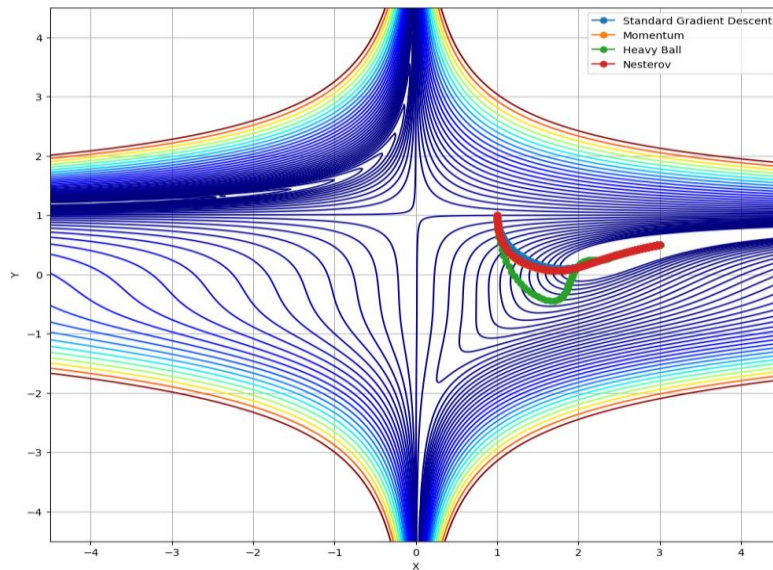


Figure 6: Paths taken by different optimization methods on the Beale function

- Rosenbrock function: Heavy Ball again shows the best result with the highest weight of 0.699239, demonstrating its stability and efficiency in achieving optimal values. Other methods, such as Momentum 0.085858 and Nesterov 0.073100, show lower weights, indicating their less efficient performance on this function.
- Rastrigin function: Standard Gradient Descent has the highest weight of 0.498951, indicating its performance on this function, while Momentum has the lowest weight of 0.078360, showing weaker performance compared to other methods.
- Ackley function: On this function, Standard Gradient Descent has the highest weight of 0.357109, indicating its effectiveness, Momentum and Nesterov also show good results, while the Heavy Ball method has the lowest weight of 0.112995, showing relatively weak results on this function.
- Beale function: On this function, the Heavy Ball method showed the highest weight of 0.707523, demonstrating high performance. The other methods performed less efficiently compared to Heavy Ball.

The analysis showed that the considered optimization methods demonstrate different efficiency on various test functions.

The Heavy Ball method consistently performs well on most functions. Its high accuracy and low execution time results in high weight values, which makes it one of the most efficient methods for most of the tasks considered.

Standard gradient descent shows a significant variation in results. On some functions (Rastrigin and Ackley functions), this method works very effectively, while on others (e.g., Himmelblau), its results are significantly inferior to other methods.

The Momentum and Nesterov methods show both bad and average results depending on the function. In some cases (Rosenbrock, Rastrigin and Beale functions), these methods show low weight values, but in other cases (Himmelblau and Ackley functions) they can be more effective.

The weights obtained from the analytic hierarchy process clear evaluation of methods' efficiency, taking into account several important aspects at the same time: the number of iterations, the average time of the iteration, the total time of execution of the method, and the accuracy of finding the minimum. This made it possible to clearly assess the advantages and disadvantages of each method in different conditions and compare them.

In general, the results of the study demonstrate that the choice of optimization method depends on the specific task and function. The Heavy Ball method showed the best overall results, while other methods, such as Standard Gradient Descent, can be effective in certain conditions.

9. Conclusions

In this article, the topic of optimization was analyzed, in particular, first-order methods, which are widely used to minimize loss functions in various problems. Optimization is an important aspect in both business and research, especially in machine learning, where it plays a key role in training models and improving their accuracy. One of the main approaches - gradient descent - was considered, as well as its main modifications, including Momentum, Heavy Ball, and Nesterov methods.

First-order methods, such as Standard Gradient Descent, have a number of advantages due to their simplicity and efficiency, but often face problems related to, for example, oscillations. Advanced modifications, such as Momentum, Heavy Ball, and Nesterov, may be better suited for specific tasks.

In addition, a review of the recent publications on the application of these methods in machine learning and other fields was conducted. The review confirmed the importance of using first-order optimization methods to solve applied problems requiring high accuracy and speed of model training.

In general, the comparison of different modifications of gradient descent showed that each method has its own strengths and weaknesses, and the choice of the optimal approach depends on the specific conditions of the problem. The use of more sophisticated methods, such as Nesterov Accelerated Gradient, can significantly improve results, reduce learning time, and increase the stability of optimization in real-world projects.

In the practical part of the work, several optimization methods were compared on classical test functions, such as Himmelblau, Rosenbrock, Rastrigin, Ackley, and Beale. The methods of Standard Gradient Descent, Momentum, Heavy Ball, and Nesterov were implemented. Using the Analytic Hierarchy Process allowed a complex evaluation of each method by a number of significant criteria: number of iterations, average time per iteration, total execution time, and function value at the last iteration. This approach provided a structured comparison, which facilitated a more accurate selection of the best method for different optimization problems. According to the evaluation results, the Heavy Ball method showed the best performance on most functions, while the Standard Gradient Descent and other methods had mixed results depending on the specifics of the functions.

Declaration on Generative AI

The authors did not use Generative AI tools in preparing the content, analyzing the data or creating the figures presented in the paper. All ideas, conclusions and figures are based on standard research and analysis methods.

References

- [1] Daoud, M.S., Shehab, M., Al-Mimi, H.M. et al. Gradient-Based Optimizer (GBO): A Review, Theory, Variants, and Applications. *Arch Computat Methods Eng* 30, 2431–2449 (2023). <https://doi.org/10.1007/s11831-022-09872-y>
- [2] Wang, Danjing, Huifang Li, Youwei Zhang and Baihai Zhang. “Gradient-Based Optimizer for Scheduling Deadline-Constrained Workflows in the Cloud.” (2021).
- [3] Mustapha, Aatila & Lachgar, Mohamed & Ali, Kartit. (2020). An Overview of Gradient Descent Algorithm Optimization in Machine Learning: Application in the Ophthalmology Field. *10.1007/978-3-030-45183-7_27*.

- [4] Morozov, V., Kalnichenko, O. (2020). The method of interaction modeling on basis of deep learning of neural networks in complex IT-projects. *International Journal of Computing* 19(1), 88–96.
- [5] Morozov, V., Deineha, V., Khlevnyi, A., (2023), Research on the Use of Machine Learning Methods for Forecasting Time Series when Making Management Decisions in IT Projects Under Martial Law, *CEUR Workshop Proceedings*, 3624, pp. 192–204.
- [6] How Machines Learn: The Power of Gradient Descent – URL: <https://towardsai.net/p/artificial-intelligence/how-machines-learn-the-power-of-gradient-descent>
- [7] S. Ruder, “An overview of gradient descent optimization algorithms,” Sep. 2016. [Online]. Available: <http://arxiv.org/pdf/1609.04747v2>
- [8] Nakerst, G., Brennan, J., & Haque, M. (2020). Gradient descent with momentum to accelerate or to super-accelerate? <https://doi.org/10.48550/ARXIV.2001.06472>
- [9] Gradient Descent and Momentum: The Heavy Ball Method – URL: <https://boostedml.com/2020/07/gradient-descent-and-momentum-the-heavy-ball-method.html>
- [10] Revati Gunjal, Syed Shadab Nayyer, S.R. Wagh, N.M. Singh, Controlled gradient descent: A control theoretical perspective for optimization, *Results in Control and Optimization*, Volume 15, 2024, 100417, ISSN 2666-7207, <https://doi.org/10.1016/j.rico.2024.100417>.
- [11] Sutskever, I. & Martens, J. & Dahl, G. & Hinton, G.. (2013). On the importance of initialization and momentum in deep learning. 30th International Conference on Machine Learning, ICML 2013.
- [12] Nermeen Abou Baker, Paul Szabo-Müller, Uwe Handmann, Year: 2021, Transfer learning-based method for automated e-waste recycling in smart cities, SC, EAI, DOI: 10.4108/eai.16-4-2021.169337
- [13] Hiroaki ARATA, Masayuki KISHIDA, Takahiko KURAHASHI, Texture shape optimization analysis using a new acceleration gradient method based on the Taylor expansion and conjugate direction, *Journal of Fluid Science and Technology*, 2022, Volume 17, Issue 4, Pages JFST0011, Released on J-STAGE November 15, 2022, Online ISSN 1880-5558, <https://doi.org/10.1299/jfst.2022jfst0011>, https://www.jstage.jst.go.jp/article/jfst/17/4/17_2022jfst0011/_article/-char/en
- [14] A Simple Guide to Gradient Descent Algorithm – URL: <https://medium.com/@datasciencewizards/a-simple-guide-to-gradient-descent-algorithm-60cbb66a0df9>
- [15] Gradient Descent With Momentum – URL: <https://towardsdatascience.com/gradient-descent-with-momentum-59420f626c8f>
- [16] Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization Methods for Large-Scale Machine Learning. In *SIAM Review* (Vol. 60, Issue 2, pp. 223–311). Society for Industrial & Applied Mathematics (SIAM). <https://doi.org/10.1137/16m1080173>
- [17] Neural Networks Optimization Algorithms – URL: <https://www.restack.io/p/neural-networks-answer-optimization-algorithms-cat-ai>
- [18] Himmelblau Function – URL: <https://www.indusmic.com/post/himmelblau-function>
- [19] Rosenbrock Function – URL: <https://www.sfu.ca/~ssurjano/rosen.html>
- [20] Rastrigin Function – URL: <https://www.sfu.ca/~ssurjano/rastr.html>
- [21] Ackley Function – URL: <https://www.sfu.ca/~ssurjano/ackley.html>
- [22] Beale Function – URL: <https://www.sfu.ca/~ssurjano/beale.html>
- [23] Stofkova, J.; Krejnos, M.; Stofkova, K.R.; Malega, P.; Binasova, V. Use of the Analytic Hierarchy Process and Selected Methods in the Managerial Decision-Making Process in the Context of Sustainable Development. *Sustainability* 2022, 14, 11546. <https://doi.org/10.3390/su141811546>