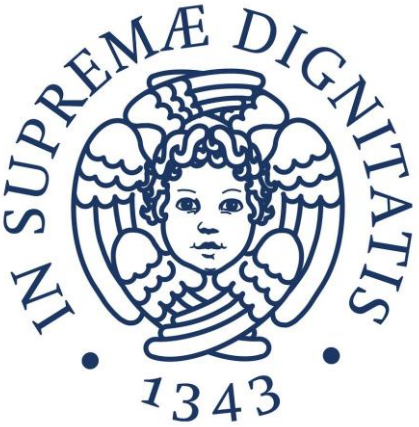


UNIVERSITY OF PISA

MSc IN DATA SCIENCE AND BUSINESS INFORMATICS

Optimization for Data Science

Neural Network with Momentum descent approach and Deflected Subgradient methods

Mattia Arancio Febbo - 561930

Michele Dicandia - 657494

ACADEMIC YEAR 2024-25

Index

Sommario

1 Introduction

Neural networks are fundamental models for tackling complex problems such as image classification, speech recognition, and natural language processing.
Their success depends on the ability to effectively optimize the model's parameters by minimizing a loss function that is often non-convex and highly complex.

This optimization process is particularly challenging since the convergence of the algorithms is influenced by the network topology, the loss function, and the initial parameter values.

In this project, we analyze the behavior of two optimization algorithms: the **Heavy Ball (momentum) method** and the **Deflected Subgradient method**.
These algorithms were chosen to explore different convergence strategies, including those applicable to non-differentiable functions.
The experiments are conducted on a neural network designed to solve a multiclass classification problem.

The report is structured as follows:

- **First section**: description of the adopted dataset, the neural network, and the loss function used.
- **Second section**: introduction of the two optimization algorithms and their theoretical properties.
- **Final section**: presentation and discussion of the experimental results.

1.1 Dataset

The dataset selected for this project is **Digits**, a commonly used dataset for evaluating classification algorithms. It is a copy of the test set from the optical recognition of handwritten digits dataset provided by the **UCI Machine Learning Repository**.

It consists of **1,797 grayscale images**, each with a resolution of **8 × 8 pixels (64 total pixels)**.
Each image represents a handwritten digit, divided into **10 classes** corresponding to the digits **0–9**.

The images, originally in raster format, are converted into numerical representations.
Each pixel represents a brightness level with a value between **0 (pure white)** and **16 (pure black)**.
This transformation allows each image to be represented as a **1 × 64 numerical vector**, where each element corresponds to the brightness intensity of a specific pixel.

An example of the pixel grid for one image is shown in **Figure 1**, where different brightness levels can be observed.

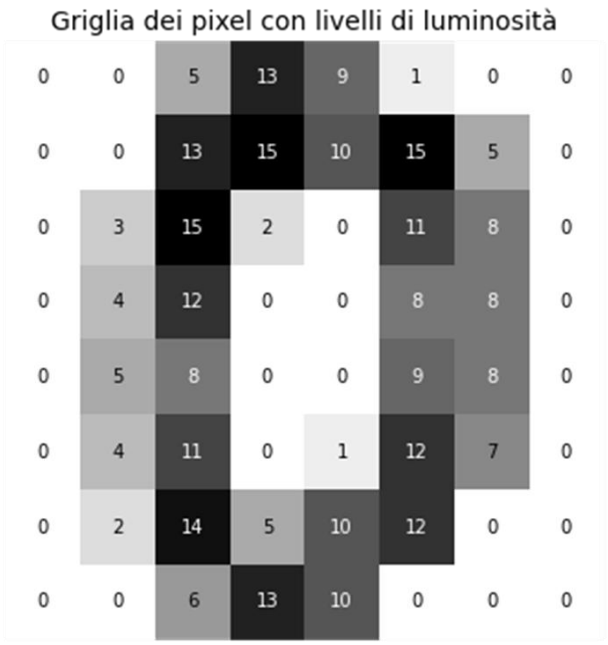


Figure 1: Example of a pixel grid for an image.

To standardize the data and improve the efficiency of optimization algorithms, pixel values are **normalized by dividing each value by 16**.
This process scales all pixel values to the range [0, 1], making the data more suitable for machine learning algorithms.

For model training, the Digits dataset (1,797 images) is divided as follows:

- **Training set**: 1,437 images
- **Test set**: 360 images

After preprocessing, the dataset is organized into matrices and vectors:

- **Training set (X_train)**: matrix of size 1437 × 64 containing the training images
- **Test set (X_test)**: matrix of size 360 × 64 containing the test images
- **Training labels (y_train)**: vector of size 1437 containing class labels for the training set
- **Test labels (y_test)**: vector of size 360 containing class labels for the test set

1.2 Neural Network Model

The model used to analyze this dataset is a **neural network** consisting of:

- an **input layer** with 64 neurons (corresponding to the 8×8 pixels),
- a **hidden layer** with 32 neurons, and
- an **output layer** with 10 neurons representing the output classes.

Since the goal is not to perform a deep machine learning architecture analysis, we do not focus on optimizing the hidden layer structure.

The **ReLU (Rectified Linear Unit)** function is used as the activation function for the hidden layer, while the **softmax** activation function is applied to the output layer.

The **cross-entropy loss function** is adopted due to its significant impact on the performance of neural networks.

Below is a detailed description of the learning process, along with the mathematical formulas supporting each step.

1.2.1 Parameters and Variables

- $\mathbf{x}$: input vector of size 64 × 1
- $\mathbf{W_1}$: weight matrix of the first layer, size 32 × 64
- $\mathbf{b_1}$: bias vector of the first layer, size 32 × 1
- $\mathbf{W_2}$: weight matrix of the second layer, size 10 × 32
- $\mathbf{b_2}$: bias vector of the second layer, size 10 × 1
- $\mathbf{a_1}$: activation of the first layer
- $\mathbf{z_1}$: linear output of the first layer
- $\mathbf{a_2}$, $\mathbf{z_2}$: activation and linear output of the second layer

1.2.2 Forward Pass

1. **First Layer (Hidden Layer):**

$$z_1 = W_1 x + b_1$$
$$a_1 = \text{ReLU}(z_1) = \max(0, z_1)$$

2. **Second Layer (Output Layer):**

$$z_2 = W_2 a_1 + b_2$$
$$a_2 = \text{Softmax}(z_2)$$

The **Softmax** function is defined as:

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

1.2.3 Loss Function

Cross-Entropy Loss for multiclass classification:

$$J(W_1, b_1, W_2, b_2) = -\sum_{i=1}^{m} \sum_{c=1}^{10} y_{i,c} \log(a_{i,c}^2)$$

where:

- $y_{i,c}$ is a binary indicator (1 if class c is the correct label for sample i),
- $a_{i,c}^2$ is the softmax output for sample i and class c,
- m is the number of samples.

The choice of **cross-entropy loss** combined with the **softmax** activation is advantageous compared to alternatives like **Mean Squared Error (MSE)** because cross-entropy penalizes misclassifications proportionally to their severity, leading to faster and more stable learning.

For instance, in the case of an incorrect prediction:

- True label $y = 1$, predicted probability $\hat{y} = 0.1$:
    - Cross-entropy loss: $L_{CE} = -\log(0.1) \approx 2.3$
    - MSE loss: $L_{MSE} = (1 - 0.1)^2 = 0.81$

Thus, cross-entropy imposes a **stronger penalty** on wrong predictions, producing larger gradients and faster learning compared to MSE, which may result in vanishing gradients near 0 or 1.

Additionally, **L1 regularization** is added to the loss function to penalize the absolute sum of the network's weights.
This tends to drive less relevant weights toward zero, effectively simplifying the model and reducing overfitting.

The total loss function is defined as:

$$J_{reg} = J + \lambda \sum | W_i |$$

where **λ** is the regularization parameter that controls the trade-off between accuracy and model simplicity.

The presence of the ReLU activation function makes the overall loss **non-convex**, introducing multiple local minima that affect the optimization process.
Moreover, due to L1 regularization, the loss becomes **non-differentiable** at points where weights are zero — making **subgradient-based methods** particularly useful for this kind of problem.

1.2.4 Optimization and Backpropagation

The goal of optimization is to determine the optimal parameters (weights) that minimize the loss function J_{reg}.
This is achieved through **backpropagation**, which efficiently computes the gradients of the loss with respect to all model parameters.
These gradients are then used to update the parameters using optimization algorithms such as **gradient descent**.

The optimization proceeds in two main steps:

1. **Forward Pass** — inputs are propagated through the network to produce the output $\hat{y}$.
2. **Backward Pass** — gradients of the loss with respect to each parameter are computed and propagated backward through the network.

For each weight w_{ij}, the partial derivative of the loss is computed as:

$$\frac{\partial J}{\partial w_{ij}} = \frac{\partial J}{\partial a_k} \cdot \frac{\partial a_k}{\partial z_k} \cdot \frac{\partial z_k}{\partial w_{ij}}$$

Finally, weights are updated according to the **gradient descent rule**:

$$w_i = w_{i-1} - \eta \cdot \nabla_w J$$

where **η** is the **learning rate**, controlling how quickly the model updates its parameters to minimize the loss.

2 Algorithms

The main goal of this project is to optimize a neural network by improving model accuracy through the iterative adjustment of its parameters, in particular the weights. These are crucial because they determine the network's ability to learn from input data and to model complex relationships so as to minimize the loss function.

Weights are initially assigned at random or drawn from specific probability distributions. This randomness is essential to break symmetry among neurons and to ensure the network can learn complex patterns during training. Poor initialization may lead to convergence issues or sub-optimal solutions.

During training, the weights are updated iteratively in a process known as **optimization**, whose purpose is to minimize the loss value. This process relies on techniques that use gradients or **subgradients** of the objective function—even when it is not everywhere differentiable.

To ensure effective optimization, we adopt two main algorithms: the **Heavy Ball method** and the **Deflected Subgradient method**.

2.1 Heavy Ball Method

The **Heavy Ball (HB)** method, also known as classic momentum, is an iterative optimization algorithm designed to improve the speed of convergence over standard gradient descent. The key idea is to introduce a **momentum term**, which stabilizes parameter updates and accelerates the approach to a minimum.

In HB, parameters are updated according to:

$$x_{i+1} \leftarrow x_i - \alpha_i \nabla f(x_i) + \beta_i (x_i - x_{i-1}),$$

where

- x_{i+1}: parameters at iteration $i + 1$;
- α_i: learning rate (step size);
- $\nabla f(x_i)$: gradient of the loss at x_i;
- $\beta_i(x_i - x_{i-1})$: momentum term, steering the next step using the previous displacement.

When $\beta_i = 0$, HB reduces to standard gradient descent. The momentum term is especially valuable for stability and efficiency on highly non-linear or rugged loss surfaces.

HB is particularly effective when the loss is **convex and smooth**, because a well-defined, continuous gradient lets momentum guide optimization stably and efficiently.
However, HB can be less effective when the loss is **non-differentiable**, as in our model where **L1 regularization** introduces kinks (especially when weights hit zero). In such cases HB may oscillate or stagnate near non-differentiable points. Since HB leverages gradient acceleration across iterations, undefined gradients hinder convergence, often making methods tailored to non-smooth objectives—such as **deflected subgradient**—more suitable.

2.1.1 Convergence properties of HB

HB performs best when the objective satisfies properties that ensure fast and stable convergence—most notably **τ-strong convexity** and **L-smoothness**.

Definitions (and their optimization impact):

1. **Convexity.** A function f is convex if, for all x, y and $\alpha \in [0,1]$,

$$f(\alpha x + (1 - \alpha)y) \leq \alpha f(x) + (1 - \alpha) f(y).$$

Convex functions have a single global minimum, simplifying optimization.

2. **τ-strong convexity.** f is strongly convex with parameter $\tau > 0$ if, for all x, y,

$$f(y) \geq f(x) + \nabla f(x)^{\top}(y - x) + \frac{\tau}{2} \| y - x \|^2.$$

This guarantees sufficient curvature and avoids flat minima, improving gradient-based convergence.

3. **Smoothness.** A function is smooth if its gradient exists everywhere and is continuous.
4. **L-smoothness.** f is L-smooth if its gradient is Lipschitz:

$$\| \nabla f(x) - \nabla f(y) \| \leq L \| x - y \|.$$

This bounds how fast the gradient can change, reducing oscillations.

For our objective (cross-entropy + L1):

- Cross-entropy is convex **in the logits**; L1 is convex ($| w |$ is convex); the sum of convex functions is convex.
  However, due to **ReLU activations**, the loss is **not convex in the weights** of the network (nonlinearities introduce local minima).
- The function is **not strongly convex**: cross-entropy generally does not satisfy τ-convexity, and L1 has no quadratic curvature.
- The function is **not smooth**: cross-entropy is differentiable, but L1 is **non-differentiable at** $w = 0$. Hence it is not L-smooth.

Since our objective is neither τ-strongly convex nor L-smooth, HB is not theoretically ideal for our setting.

2.1.2 Convergence rate of HB

Under **τ-strong convexity** and **L-smoothness**, HB attains an optimal linear rate. With condition number $\kappa = L/\tau$, the error decreases roughly like

$$\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{k},$$

i.e., fast (linear) contraction when κ is moderate.

Typical parameter choices that achieve the optimal rate are:

$$\alpha = \frac{4}{(\sqrt{L} + \sqrt{\tau})^2}, \beta = \left(\frac{\sqrt{L} - \sqrt{\tau}}{\sqrt{L} + \sqrt{\tau}}\right)^2.$$

In our case (cross-entropy + L1), the loss is **neither L-smooth nor τ-strongly convex**:

- Without strong convexity ($\tau \approx 0$), we cannot guarantee exponential (linear) contraction; rates become **sublinear**.
- Lack of smoothness leads to stronger oscillations; a fixed step can be too aggressive in some regions and too timid in others.

Practically, we must **tune α and β conservatively**:

- **Learning rate α:** smaller than in the smooth case; a common heuristic is $\alpha \approx 1/L$, where $L$is an upper estimate of curvature.
- **Momentum β:** large βcan amplify oscillations when τ-convexity and L-smoothness fail.

2.2 Deflected Subgradient Method

The **Deflected Subgradient (DSG)** method is an advanced variant of the standard subgradient method. It is designed for better efficiency and faster convergence on optimization problems with **non-differentiable** or highly irregular objectives. Its distinctive feature is the use of a **deflected direction**, combining the current subgradient with information from previous iterations to damp abrupt changes and reduce oscillations on non-smooth landscapes.

2.2.1 Standard Subgradient Method

The standard update is:

$$x_{i+1} = x_i - \alpha_i g_i,$$

where

- x_i: current point,
- g_i: a subgradient of $f$at x_i,
- α_i: step size.

Because the objective is not everywhere differentiable, g_i is not a classical gradient but belongs to the subdifferential. This method can suffer from significant oscillations and slow convergence on non-smooth or highly irregular problems.

2.2.2 Deflected Subgradient Method

In DSG the search direction is modified with a **deflection term**:

$$d_i = \gamma_i g_i + (1 - \gamma_i) d_{i-1},$$

where $\gamma_i$balances the influence of the current subgradient $g_i$and the previous direction d_{i-1}. The parameter update becomes:

$$x_{i+1} = x_i - \alpha_i d_i.$$

- If $\gamma_i = 1$, DSG reduces to the standard subgradient method.
- If $\gamma_i < 1$, the previous direction has more weight, typically yielding a more stable trajectory.

Step-size choices (crucial for both speed and stability) include:

- **Constant step size** $\alpha = c$when the problem is well-conditioned and $f$is τ-convex and L-smooth.
- **Polyak step size** (if a good estimate of $f^{\backslash *}$is available):

$$\alpha_i = \frac{f(x_i) - f^{\backslash *}}{\| g_i \|^2}.$$

- **Diminishing step size**, ensuring

$$\sum_{i=0}^{\infty} \alpha_i = \infty, \sum_{i=0}^{\infty} \alpha_i^2 < \infty,$$

e.g., $\alpha_i = \frac{c}{i+1}$.

Since our objective is **non-smooth** and not strongly convex, a **diminishing schedule** is the safest default. Polyak's rule is a valid alternative if $f^{\backslash *}$can be estimated accurately.

2.2.3 Convergence of DSG

DSG improves over the standard subgradient on non-smooth problems, but its guarantees depend on properties of $f$and on parameter choices.

Convergence holds under (among others):

- **Diminishing step size** as above;
- **Convex or pseudo-convex objective**:
    - If $f$is convex, DSG converges to the global optimum;
    - If $f$is pseudo-convex, DSG converges to a critical point (not necessarily global);
- **Bounded subgradients** $\| g_i \| \leq G$to avoid excessively large steps;
- Proper control of γ_i:
    - If $\gamma_i \to 0$, DSG asymptotically behaves like standard subgradient and inherits its guarantees;
    - If $\gamma_i$stays too large (close to 1), convergence can fail because the method overweights past directions.

In our setting (cross-entropy + L1), a diminishing step size can be enforced, but **global convergence is not guaranteed** because the overall objective is **not convex**. Moreover, the subgradient may be **unbounded at** $w = 0$for the L1 term (it jumps from $-\lambda$to $+\lambda$), which can cause instability. To increase stability, choose γ_i**moderate** or **decreasing**.

2.2.4 Convergence rate of DSG

Rates depend strongly on convexity and step size:

- If $f$is **τ-strongly convex**, DSG can achieve

$$f(x_k) - f(x^{\backslash *}) = \mathcal{O}(1/k)$$

(sublinear, but faster than the standard subgradient).

- If $f$is **convex** (not strongly), a typical rate is

$$f(x_k) - f(x^{\backslash *}) = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right).$$

- If $f$is **non-convex**, there is no general guarantee of reaching the global optimum; DSG may converge to a point with small changes in the objective (often a local minimum) and can stop at suboptimal solutions in highly irregular landscapes.

In our case (cross-entropy + L1), the function is **not strongly convex** (due to L1). Hence the most realistic asymptotic rate is:

$$\mathcal{O}\left(\frac{1}{\sqrt{k}}\right).$$

3 Experiments

This section reports the results of the Heavy Ball and Deflected Subgradient implementations, applied both to standard benchmark functions and to the neural-network model built on the dataset under study.

To validate the implementations, the algorithms were tested on two functions with different characteristics:

- **Convex function: Matyas function**, defined as

$$f(x, y) = 0.26\,(x^2 + y^2) - 0.48\,x\,y.$$

It has a well-defined global minimum at $(0,0)$ with $f(0,0) = 0$. Its shape is symmetric—a paraboloid with concentric elliptical level curves. The Matyas function is commonly used to assess the convergence and stability of optimization algorithms.

- **Non-convex function: Himmelblau's function**, defined as

$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2.$$

It has four global minima located at $(3,2)$, $(-2.805,3.131)$, $(-3.779,-3.283)$, and $(3.584,-1.848)$, all with $f(x, y) = 0$.

The main metrics used to evaluate the algorithms are:

- **Function value.** A direct measure of optimization effectiveness—the final value of the objective after the iterations. In a neural network this corresponds to the loss and indicates the quality of the solution. Note the final value may not equal the absolute minimum reached at any time during the process, especially under instability.

- **Instability.** The degree of algorithmic instability, measured via relative variations of the objective value across iterations. The instability is computed as:

$$\mathrm{unst}(x) = \frac{10^5}{|x|} \sum_{i=2}^{|x|} \max\left(0, \frac{x[i] - x[i-1]}{x[i-1]}\right)$$

where $x[i]$is the function value at iteration $i$and $|\ x\ |$is the total number of iterations. This normalizes relative increments by the number of iterations, indicating how prone the algorithm is to undesirable fluctuations. It is a heuristic with limitations (e.g., it weights early and late increments equally, though their meanings differ in convergence).

- **Convergence speed.** The number of iterations required for the algorithm to **first** enter a specified relative band around the final optimal value obtained. The score ranges from 0 to 100, where 100 means the value was already inside the band at the first iteration. This metric must be interpreted together with the final value reached. Formally:

$$cs(x) = 100 - \frac{\arg\max_i \left(x[i] \le \min(x) \cdot \left(1 + \frac{\rho}{100}\right)\right) \cdot 100}{|x|}$$

where $\rho \geq 0$is the tolerance relative to the final optimal value. A score of 100 indicates the range was reached at the very first iteration.

Execution stopped under any of the following **stopping criteria**:

1. **Gradient/subgradient norm.** Stop when the norm drops below a threshold ε, indicating an optimal solution has been found.
2. **Maximum iterations.** Stop when the preset iteration limit is reached ("stopped solution").
3. **No significant decrease.** Stop if, for 10 consecutive iterations, the difference between the function value at iteration $i$and at $i-1$is below ε, indicating further improvements are negligible.
4. **Prolonged instability.** If for 20 consecutive iterations the algorithm's instability exceeds a preset critical threshold (conservative criterion, e.g., value > 1000), terminate to avoid wasting compute on an ineffective configuration.

A high instability does **not** necessarily mean the algorithm cannot converge, but it indicates an irregular optimization trajectory. With the current parameters, the configuration may not be efficient, so it is preferable to stop and discard it rather than spend time and compute on a potentially inferior option.

- **Section 3.1** runs convergence tests on Matyas and Himmelblau to assess convergence capabilities.
- **Section 3.2** applies the algorithms to a neural network, examining how parameter choices affect optimization and overall performance.

3.1 Convergence tests

We analyze Heavy Ball and Deflected Subgradient on Matyas (convex) and Himmelblau (non-convex).
Experiments start from the initial point $(4,4)$and include a preliminary grid search to tune the main parameters. The explored grids were:

- **Alpha (α)**: [0.01,0.02,0.05,0.07,0.1,0.12,0.15,0.2,0.25,0.3,0.5,0.7,1.0]
- **Momentum (Heavy Ball)**: [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]
- **Gamma (γ, Deflected Subgradient)**: [0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]

Pseudocode used in the study:

Algorithm 1 — Heavy Ball

```
w0 ← initialize weights
m0 ← 0                  // initial momentum
set α (lr), μ (momentum), ε (tol), max_iter
i ← 0
while i < max_iter do
   g_i ← ∇f(w_i)        // gradient
   compute ‖g_i‖        // gradient norm
   compute instability over loss history
   compute convergence speed
   m_i ← μ · m_{i-1} − α · g_i // update momentum
   w_{i+1} ← w_i + m_i  // update weights
   if ‖g_i‖ < ε then break
```

13: if avg change of last 10 losses < ε then break
14: if avg instability of last losses > 1000 then break
15: i ← i + 1
16: end

Algorithm 2 — Deflected Subgradient

1: initialize weights w0, previous direction d0 ← 0
2: set α (initial), γ (deflection), ε, max_iter
3: i ← 1
4: while i ≤ max_iter do
5: compute subgradient g_i ∈ ∂f(w_i)
6: compute ‖g_i‖
7: compute instability over loss history
8: compute convergence speed
9: d_i ← γ · g_i + (1 − γ) · d_{i−1} // deflection
10: α_i ← α / i // diminishing step size
11: w_{i+1} ← w_i − α_i · d_i // update
12: if ‖g_i‖ < ε then break
13: if avg change of last 10 losses < ε then break
14: if avg instability of last losses > 1000 then break
15: i ← i + 1
16: end

Many configurations were evaluated via grid search. Given the low computational cost, we inspected all combinations to study parameter effects. To avoid premature stopping after a single iteration, we set $\varepsilon = 1e-12$.

For **Deflected Subgradient**, a **constant step** was preferred over DSS (Diminishing Step Size) or Polyak for these reasons:

- Matyas is τ-convex and L-smooth, so a constant step suffices for good rates.
- Although Himmelblau is not τ-convex, constant steps worked better empirically: DSS often required >1000 iterations due to the progressive decrease of αeach iteration (the "Scylla problem"), significantly slowing optimization.

Key observations:

- **Matyas.** Larger $\alpha$is generally more efficient—iterations to optimum decrease as αincreases—though too large a step causes instability (oscillations/divergence).
- **Himmelblau.** Unlike Matyas, smaller αyields easier convergence; the method quickly finds one of the global minima, typically (3,2).
- **HB instability.** On both functions (especially Matyas), many HB configurations reached the global minimum, but numerous combos were discarded due to high instability, reducing reliability.

Best configurations (Tables 1 and 2):

Algorithm	Optimal Parameters	Main Results

Heavy Ball	$\alpha = 1, \beta = 0.6$	Iterations: 104 Instabilty: 0 Convergence speed: 89.7 Final Loss: 0 Run time: 0.06
Deflected Subgradient Method	$\alpha = 1, \gamma = 0.2$	Iterations: 11 Instability: 0 Convergence speed: 99.0 Final Loss: 0 Run time: 0.00

Table 1: Optimal parameters applied to the Matyas function

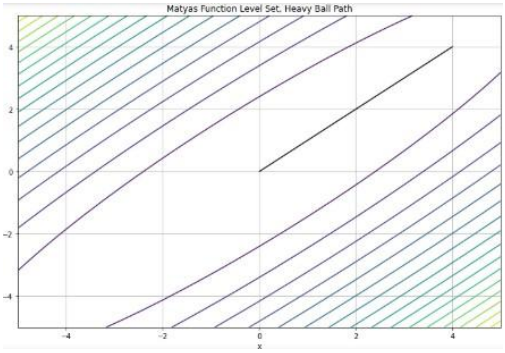
Algorithm	**Optimal Parameters**	**Main Results**
Heavy Ball	$\alpha = 0.01, \beta = 0.1$	Iterations: 46 Instability: 0 Convergence speed: 95.5 Final loss: 0 Run time: 0.02
Deflected Subgradient Method	$\alpha = 0.02, \gamma = 0.9$	Iterations: 11 Instability: 0 Convergence speed: 99.0 Final loss: 0 Run time: 0.00

Table 2: Optimal parameters applied to the Matyas function

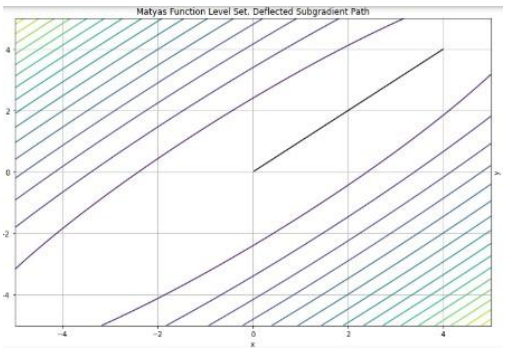
From the tables we see that, with optimal parameter choices, the deflected subgradient method (with a constant step size) is significantly more efficient than the Heavy Ball method. This advantage is evident in runtime, convergence speed, and the number of iterations needed to reach the optimum, all of which are markedly lower than for Heavy Ball.

As for convergence on the Matyas function, both algorithms reach the optimum point $(0,0)$along smooth trajectories. However, Heavy Ball attains the minimum with precision $\varepsilon = 1\text{e} - 12\text{in } 104$ iterations, whereas the DSG (Deflected Subgradient) method requires only 11 iterations to reach the same point, demonstrating greater efficiency in terms of convergence speed. Figure 2 shows both algorithms converging to the minimum on the Matyas function.

Regarding convergence on the Himmelblau function, the Heavy Ball (HB) method follows a more direct path, slightly damped due to the momentum effect—reaching convergence in 46 iterations



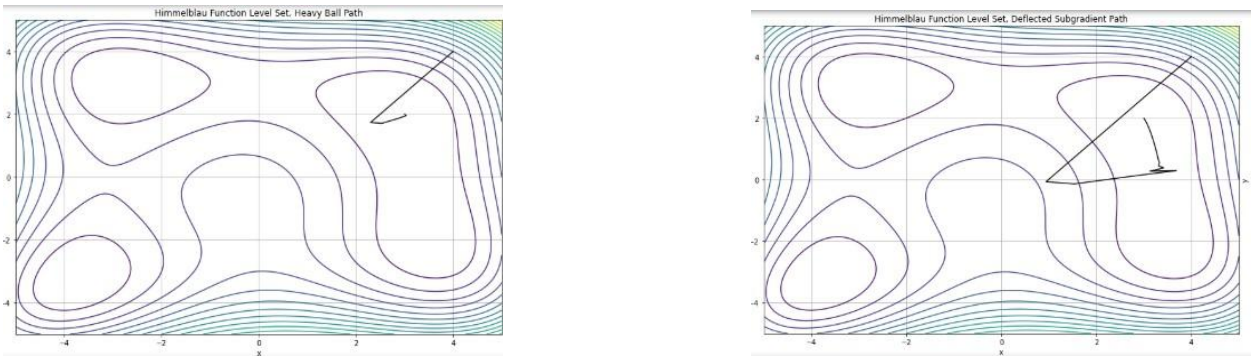
(a) Heavy Ball



(b) Deflected Subgradient

Figura 2: Matyas functions

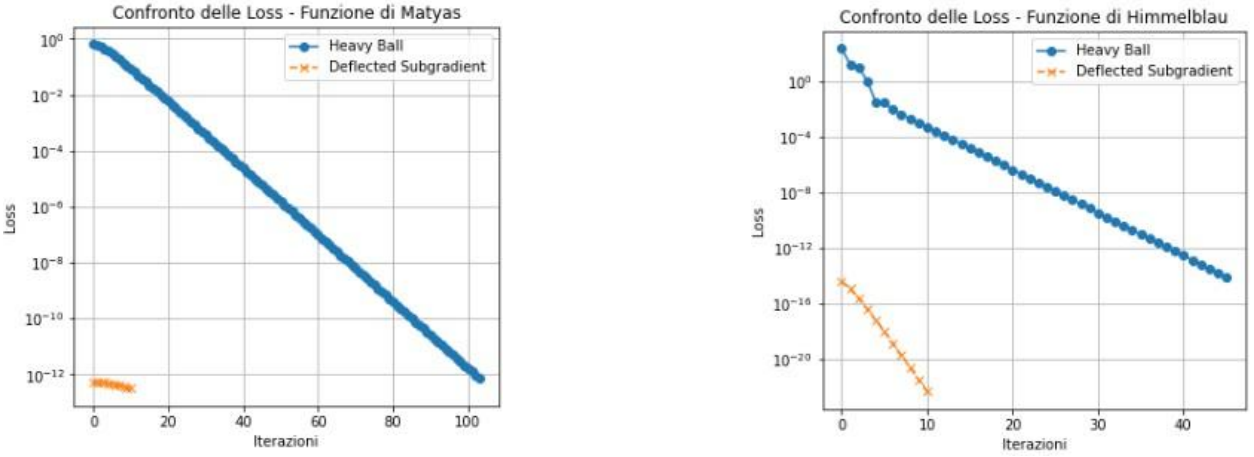
The Deflected Subgradient (DSG) method, on the other hand, follows a more irregular trajectory, characterized by sharper turns due to the high deflection parameter. Although its path may appear broader, DSG manages to converge in just 11 iterations, once again demonstrating greater efficiency in terms of convergence speed. Figure 3 shows the convergence of both algorithms to the minimum point (3,2)in the Himmelblau function.



(a) Heavy Ball *(b) Deflected Subgradient*

Figure 3: Himmelblau functions

Finally, Figure 4 shows a comparison between the convergence speeds of the Heavy Ball and Deflected Subgradient methods applied to the Matyas and Himmelblau functions.
The analysis is performed by evaluating the function value on a logarithmic scale with respect to the number of iterations.



(a) Matyas Function *(b) Himmelblau Function*

Figure 4: convergence speed

3.2 Neural Network

After implementing and testing the optimization algorithms on standard mathematical functions, we applied the **Heavy Ball (HB)** and **Deflected Subgradient (DSG)** methods to a neural network developed in Python. The dataset used and the structure of the neural network are described in Sections 1.1 and 1.1.1, respectively.

To ensure proper gradient propagation and stabilize the training process, the neural network was initialized using the **He Initialization** method (also known as **Kaiming Initialization**), specifically designed for **ReLU** activation functions.

The He initialization formula used is:

$$W \sim \mathcal{N}(0, \frac{2}{\text{fan\_in}})$$

where:

- **fan_in** is the number of input neurons in the layer,
- the weights are initialized with a normal distribution with mean 0 and variance 2/fan_in,
- the biases are initialized to 0.

This approach was chosen to prevent the **vanishing gradient** problem and improve optimization stability.

For the **Deflected Subgradient Method (DSG)**, a **Diminishing Step-Size (DSS)** was used instead of a constant step size. As discussed in Section 2.2.2, this choice is motivated by the fact that neural networks are not necessarily τ-convex, making a constant step potentially unstable. The DSS strategy allows for dynamically adapting the learning rate, avoiding unwanted oscillations.

The analysis was divided into **three sections**, each corresponding to a different value of the regularization parameter **λ** (0.01, 0.1, 1.0). This choice aimed to study the effect of **L1 regularization** on the performance of both algorithms and to identify optimal configurations for each λ.

For each algorithm, a **grid search** was performed while keeping λ fixed. All possible combinations of the parameters **α** and **β (or γ)** were tested to determine the configuration providing the best performance.

The optimization parameters explored were:

- **α (learning rate):**
  - [1, 2, 5, 10, 15, 20, 40] (for DSG with λ = 0.01 and λ = 0.1)
  - [0.01, 0.05, 0.1, 0.2, 0.5, 0.75, 1] (for all other cases)
- **μ (momentum factor, only for HB):** [0.1, 0.3, 0.5, 0.7, 0.9]
- **γ (deflection factor, only for DSG):** [0.1, 0.3, 0.5, 0.7, 0.9]

For the **Deflected Subgradient Method (DSG)** with λ = 0.01 and λ = 0.1, a range of **α ≥ 1** was used, applying the **Diminishing Step-Size (DSS)** rule.
This decision was based on empirical evidence showing that smaller α values (α < 1) led to inefficient improvements and excessively slow convergence.

The parameter exploration was performed through **Grid Search**, evaluating algorithm performance in terms of:

- Objective function value
- Convergence speed
- Instability

The **maximum number of iterations** was set to **1000**, and the **stopping criterion** was fixed at $\epsilon = 10^{-4}$, ensuring a reasonable trade-off between execution time and accuracy for both algorithms.

3.2.1 Analysis with λ = 0.01

Initially, the Heavy Ball and Deflected Subgradient algorithms were tested using a very low regularization value (λ = 0.01). This setup aimed to minimize the effect of regularization and evaluate the pure optimization capabilities of both methods.

From the results obtained, the **Heavy Ball** algorithm exhibited relatively stable behavior. Learning rates (**α**) between **0.1 and 0.2** favored faster convergence, while higher momentum values (**μ** between 0.7 and 0.9) further improved convergence speed. However, larger α values (e.g., α = 0.5) introduced significant instability, with instability values sometimes exceeding 500.

The configuration that produced the best results for the **Heavy Ball algorithm** was:

- **α = 0.2**
- **μ = 0.9**
- **Objective function = 1.047**
- **Convergence speed = 29.3**
- **Instability = 19.67**

These results suggest that, with a low level of regularization, the algorithm can achieve satisfactory convergence performance, even though the objective function value never drops below 1.0.

Table 3 below lists the ten best configurations found for the Heavy Ball algorithm, sorted by loss, convergence speed, and instability, with λ = 0.01.

α	μ	**Objective function**	**Convergence speed**	**Instability**
0.20	0.9	1.047	29.3	19.67
0.10	0.9	1.054	13.9	6.10
0.20	0.7	1.066	17.6	9.19
0.05	0.9	1.097	27.7	0.39
0.50	0.7	1.098	34	508.25
0.50	0.5	1.102	21	209.51
0.20	0.5	1.139	6.6	4.45
0.50	0.9	1.140	36.9	158.01
0.75	0.5	1.142	15.6	966.10
0.50	0.3	1.51	25.7	114.744

Table 3: Heavy Ball with λ = 0.01 and 1000 iterazions

The **Deflected Subgradient (DSG)** method achieved a higher final objective function value compared to the **Heavy Ball (HB)** method, with the best configuration reaching a value of **1.2359**. This suggests that although DSG provides greater stability, it may stop prematurely without reaching a deeper minimum.

From the perspective of **convergence speed**, DSG exhibited more variable behavior than HB. Some configurations satisfied the stopping criterion before reaching the maximum iteration limit — with the best one stopping at **638 iterations** — while others reached the full **1000 iterations**.

Regarding **instability**, the method proved to be extremely stable in most configurations, with values equal to **0** in almost all cases.
The only exception was the configuration with **α = 5** and **γ = 0.5**, which recorded an instability value of **2.7236**, still lower than the most unstable HB configurations.
Unlike HB, DSG showed a strong dependence on the **α** parameter, while its sensitivity to **γ** was weaker. In particular:

- Higher **α** values (10–20) favored faster convergence and lower objective function values.
- Lower **α** values (2–5) led to poorer results, with objective function values above **2.8**.
- The **γ** parameter (deflection factor) did not have a significant impact on overall performance, yielding similar results across different values. However, lower **γ** values (0.1–0.3) showed a slight tendency to improve convergence speed compared to higher ones.

The configuration that produced the **best results** for the Deflected Subgradient algorithm with **λ = 0.01** was the following:

- **α = 20**
- **γ = 0.1**
- **Objective function = 1.235**
- **Convergence speed = 47.3**
- **Instability = 0.0**

Table 4 reports the **ten best configurations** found for the Deflected Subgradient algorithm, sorted by objective function value, convergence speed, and instability, with **λ = 0.01**.

α	γ	**Objective function**	**Convergence speed**	**Instability**
20	0.1	1.235	47.3	0.0
15	0.1	1.373	40.8	0.0
10	0.1	1.497	13.7	0.0
5	0.3	1.823	10.1	0.0
5	0.5	1.828	8.4	2.72
5	0.1	1.978	9.4	0.0
2	0.9	2.801	11.8	0.0
2	0.7	2.803	12.0	0.0
2	0.5	2.811	12.0	0.0
2	0.3	2.874	11.7	0.0

Table 4: Deflected Subgradient with λ = 0.01 and 1000 iterazions

A crucial aspect to evaluate is whether the two algorithms converge to the same minimum or if one of them stops at a shallower local minimum.
Although **HB** achieves a lower objective function value, **DSG** uses a **Diminishing Step-Size (DSS)**, which could allow it, in the long run, to approach the same minimum. However, the limit of **1000 iterations** may have prevented DSG from exploring the parameter space more thoroughly, thus restricting its optimization capability.

The absence of instability in DSG suggests that the algorithm may have **prematurely stabilized in a local minimum**, while HB, despite showing more irregular behavior, continues improving the objective function until the end of the iterations. This may indicate that **HB has a greater ability to escape shallow local minima**, finding better solutions in the long term.

After completing the first **grid search phase with λ = 0.01**, the best configurations for both the Heavy Ball and Deflected Subgradient algorithms showed promising performance. However, the maximum limit of **1000 iterations** may have prevented the algorithms from reaching a more precise level of optimization.

To deepen the analysis, the results were refined by selecting the **best-performing parameters** identified in the first phase. In this second phase, the **maximum number of iterations** was increased to **10,000**, and the **ϵ value** was reduced to **1e−6**, aiming for greater precision and improved convergence quality.

The configurations selected for this second phase of analysis were as follows:

- **For the Heavy Ball algorithm:**
  - α (learning rate): [0.1, 0.2]
  - μ (momentum): [0.7, 0.9]
- **For the Deflected Subgradient algorithm:**
  - α (learning rate): [10, 15, 20]
  - γ (deflection factor): [0.1, 0.3]

After performing the grid search using these optimized parameters, the best results obtained by the **Heavy Ball algorithm** are reported in **Table 5** below:

α	γ	**Objective function**	**Convergence speed**	**Instability**
0.2	0.7	0.9794	14.16	21.611
0.1	0.9	0.9812	43.79	14.001
0.2	0.9	0.9858	65.64	29.172
0.1	0.7	0.9937	33.22	8.089

Table 5: Heavy Ball with λ = 0.01 and 10000 iterazions

The results show a clear improvement compared to the previous grid search performed with 1,000 iterations. The objective function value dropped below **1.00**, demonstrating that increasing the maximum number of iterations allowed the algorithm to continue optimizing and reach a more accurate solution.

A comparison among the different tested configurations shows the following:

- The **best configuration** in terms of objective function minimization was achieved with **α = 0.2** and **μ = 0.7**, resulting in a **loss of 0.9794**.
- The configuration with **α = 0.1** and **μ = 0.9** showed good stability, achieving an objective function value of **0.9812** and a higher convergence speed (**43.79** vs **14.16**).
- With **α = 0.2** and **μ = 0.9**, the algorithm exhibited an even faster convergence (**65.64**), but at the cost of stability, with a significant increase in instability.

These results suggest that, to achieve an optimal balance between objective minimization, stability, and convergence speed, the ideal configuration is likely:

- **α = 0.1**, **μ = 0.9** — this combination ensures low instability and good convergence speed.

The **best results obtained with the Deflected Subgradient algorithm** are presented below in **Table 6**.

α	γ	**Objective function**	**Convergence speed**	**Instability**

20	0.1	1.131	39.78	0.000020
15	0.1	1.234	28.64	0.0
10	0.1	1.357	27.50	0.0
10	0.3	3.721	99.80	463.89
15	0.3	5.345	99.80	1964.58
20	0.3	7.568	100	4301.579

Table 6: Deflected Subgradient with λ = 0.01 and 10000 iterazions

The best configuration identified for the **Deflected Subgradient (DSG)** method has the following parameters:

- **α = 20**, **γ = 0.1**
- **Objective function = 1.131**
- **Convergence speed = 39.78**
- **Instability = 0.000020**

The DSG algorithm showed more heterogeneous results compared to the **Heavy Ball (HB)** method, with a final objective function value of **1.131**, indicating an improvement over the results obtained in the first phase of the analysis.

In terms of **convergence speed**, DSG demonstrated a good optimization capability, with the best configuration reaching a value of **39.78**. However, some configurations exhibited extremely high values, suggesting that the algorithm may behave unstably under certain conditions.

Regarding **instability**, the method maintained its characteristic high stability for configurations with **γ = 0.1**, where the maximum recorded value was **0.000020**. However, for **γ = 0.3**, the algorithm displayed highly unstable behavior, with values exceeding **4300**.

The use of a **Diminishing Step-Size (DSS)** in DSG may allow the algorithm to progressively approach the same minimum reached by HB, although it does so significantly more slowly.

3.2.1.1 Convergence Rate Analysis with λ = 0.01

After analyzing the results obtained with the regularization parameter **λ = 0.01**, we proceed to compare the **theoretical** and **empirical** convergence rates for both algorithms: **Heavy Ball (HB)** and **Deflected Subgradient (DSG)**.

The comparison is based on the **relative gap**, defined as:

$$\text{Relative gap} = f(x_k) - f^*$$

where:

- $f(x_k)$represents the value of the objective function at the k-th iteration;
- $f^*$is the estimated value of the global minimum, obtained by running the algorithm (with the same parameters) for **100,000 iterations** and taking the minimum objective function value reached.

This comparison allows us to evaluate how quickly the algorithms approach the global minimum and to compare **empirical efficiency** with **theoretical convergence predictions**.

Using the **relative gap** enables a fairer comparison between the two algorithms by eliminating scale differences in the objective function values. Moreover, representing the results on a **log-log scale** helps verify whether the empirical behavior of the algorithms aligns with the expected theoretical convergence rate.

The **theoretical convergence rates** predicted for the two algorithms are as follows:

- **Heavy Ball:** Theoretically, this method should exhibit a convergence rate of **O(1/k)**. However, due to the presence of **L1 regularization** and the **non-smooth** nature of the loss function (which combines cross-entropy and regularization), the actual convergence may be slower than the theoretical estimate.

- **Deflected Subgradient:** For this algorithm, the expected theoretical convergence rate is $\mathbf{O(1/\sqrt{k})}$. In general, subgradient-based methods tend to show a slower decay of the loss compared to methods that exploit gradient acceleration.

In **Figure 5**, a comparison is presented between the **empirical convergence rate** of the Heavy Ball algorithm (with $\boldsymbol{\alpha = 0.1}$ and $\boldsymbol{\mu = 0.9}$) and the **theoretical rate $\mathbf{O(1/k)}$**. This comparison allows us to assess how closely the algorithm's actual behavior aligns with theoretical expectations.

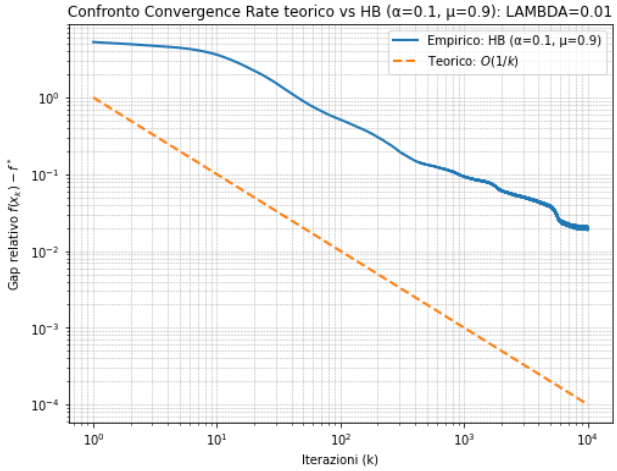


Figure 5: Comparison between HB convergence rate and theoretical convergence rate.

Initially, the algorithm follows a trend consistent with theoretical expectations, showing a gradual decrease in the objective function. However, in the later stages of optimization, a slight deviation from the theoretical curve can be observed.
This behavior is attributed to the effect of **L1 regularization**, which imposes a penalty on the neural network weights, and to the fact that the loss function is **not L-smooth**.

The **log-log scale** representation confirms that the decay of the objective function does not perfectly follow the theoretical **$\mathbf{O(1/k)}$** behavior, instead appearing slightly slower.

Overall, the **Heavy Ball algorithm with $\boldsymbol{\lambda = 0.01}$** proves to be effective, maintaining a good convergence speed and showing a stable decrease in loss.

In **Figure 6**, a similar comparison is performed for the **Deflected Subgradient algorithm** with $\boldsymbol{\alpha = 20}$ and $\boldsymbol{\gamma = 0.1}$.
In this case as well, the empirical behavior is compared with the expected theoretical rate of **$\mathbf{O(1/\sqrt{k})}$**, allowing an evaluation of the consistency between the observed and theoretical convergence trends.

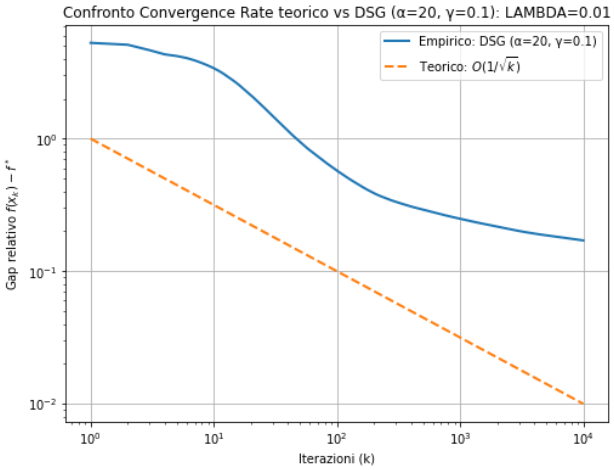


Figure 6: Comparison between DSG convergence rate and theoretical convergence rate.

The second plot compares the **empirical** and **theoretical convergence rates** for the **Deflected Subgradient (DSG)** method.
As expected, the empirical curve follows a slower trajectory than **Heavy Ball (HB)**, consistent with the theoretical prediction of $\mathbf{O(1/\sqrt{k})}$.

The behavior of the empirical curve shows a gradual stabilization in later iterations, suggesting that the algorithm is approaching a region of the objective function where variations between iterations become increasingly small.
However, compared to theoretical predictions, a slight deviation can be observed in the actual behavior, with a **slower-than-expected convergence rate**. This may be due to the algorithm's sensitivity to the parameter $\boldsymbol{\gamma}$, which might not have been optimally tuned in this configuration.

The absence of significant oscillations confirms that the algorithm maintains stable behavior.
Nevertheless, the **rate of decrease of the objective function** is lower than that of momentum-based methods such as **Heavy Ball**.

In **Figure 7**, a direct comparison between the two algorithms — **Heavy Ball (HB)** with $\boldsymbol{\alpha = 0.1, \mu = 0.9}$ and **Deflected Subgradient (DSG)** with $\boldsymbol{\alpha = 20, \gamma = 0.1}$ — is presented.
The comparison is based on the **relative gap**, allowing an assessment of the differences in convergence speed and stability between the two methods.

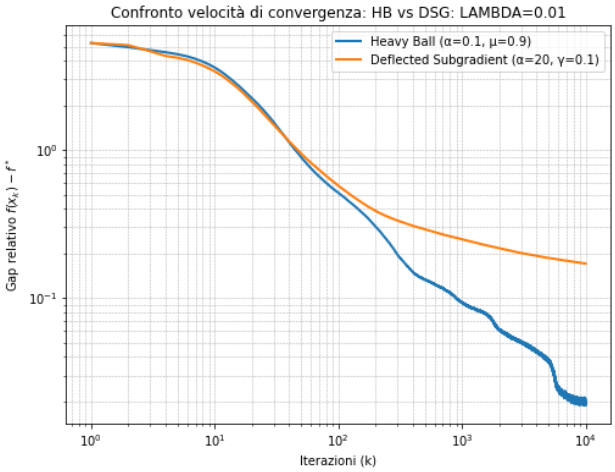


Figure 7: Comparison HB vs DSG .

The third plot presents a direct comparison between **Heavy Ball (HB)** and **Deflected Subgradient (DSG)** using the same **relative gap** criterion.

From the analysis of the curve, it can be observed that **Heavy Ball** converges faster than **Deflected Subgradient**.
In the initial iterations, both methods exhibit similar behavior, but after a certain number of iterations, HB continues to decrease more rapidly, while DSG progressively slows down.

The smoother trajectory of DSG confirms its higher stability, while HB—although showing some oscillations—manages to reach a lower objective function value in less time.

This difference reflects the fundamental characteristics of the two algorithms:

- **HB** exploits *momentum* to accelerate gradient descent.
- **DSG** adopts a more conservative update rule, limiting the descent speed.

The oscillations observed in HB's curve can be attributed to the **momentum effect**, which amplifies sensitivity to gradient variations.
However, despite this variability, HB achieves a **lower final objective function value** compared to DSG.

The analysis of the relative gap confirms that **Heavy Ball** reaches the minimum of the objective function more quickly than **Deflected Subgradient**, as predicted by optimization theory.
Nevertheless, the presence of **L1 regularization** and the **structure of the loss function** introduce some discrepancies compared to theoretical expectations.

DSG exhibited a slower convergence rate—consistent with the theoretical $\mathbf{O(1/\sqrt{k})}$ prediction—but maintained a smoother and more stable trajectory, without the oscillations observed in HB.
This suggests that while **HB** is generally more effective for minimizing the objective function, **DSG** may be preferable in scenarios where **optimization stability** is a critical requirement.

The use of a **logarithmic scale** in the graph clearly highlights how both algorithms follow trends consistent with their theoretical convergence rates, albeit with some deviations caused by the specific characteristics of the optimization problem.

3.2.1 Analisys with $\lambda = 0.1$

After analyzing the behavior of the optimization algorithms with a regularization value of **λ = 0.01**, a higher value of **λ = 0.1** was tested to evaluate how a stronger penalty on the weights affects the performance of the **Heavy Ball (HB)** and **Deflected Subgradient (DSG)** algorithms.

The goal of this phase is to understand whether a higher level of regularization improves the model's generalization ability and to assess its impact on the convergence rate and stability of the algorithms. As in the previous stage, the analysis was conducted in two steps:

- An initial **grid search** with a maximum of **1,000 iterations** and **ϵ = 1e−4**, aimed at quickly identifying the best configurations.
- A subsequent **refined analysis**, where the best parameters from the first phase were tested with **10,000 iterations** and **ϵ = 1e−6**, to more accurately evaluate the long-term behavior of the algorithms.

With **λ = 0.1**, the **Heavy Ball (HB)** algorithm exhibited behavior significantly different from what was observed with **λ = 0.01**.
The main results from the grid search are as follows:

- The **objective function value** is higher than in the λ = 0.01 case, ranging between **2.41 and 3.42**.
- Lower values of **α (e.g., 0.01)** provide greater stability, keeping instability low up to **μ = 0.5**.
- As **μ** increases, the **convergence speed** improves, but so does instability. For **μ = 0.9**, more pronounced oscillations in the loss values are observed.
- The **convergence speed** is considerably higher than that observed with λ = 0.01. In some cases, with a step size of **0.05**, convergence speed exceeded **90**.

The **best configuration** for the Heavy Ball algorithm with a maximum of **1,000 iterations** is the following:

- **α = 0.01**
- **μ = 0.1**
- **Objective function = 2.413**
- **Convergence speed = 43.0**
- **Instability = 15.42**

Table 7 reports the **10 best configurations** for the Heavy Ball algorithm with a maximum of **1,000 iterations** and **ϵ = 1e−4**.

α	μ	Objective Function	Convergence speed	Instability
0.01	0.1	2.413	43.0	15.42
0.01	0.3	2.414	55.5	4.91
0.01	0.5	2.421	68.1	15.51
0.01	0.7	2.471	80.7	28.35
0.01	0.9	2.587	85.9	63.70
0.05	0.1	2.858	88.7	38.86
0.05	0.3	2.863	91.0	247.89
0.05	0.5	2.899	93.1	144.09
0.05	0.7	3.134	95.7	246.91
0.10	0.1	3.425	94.3	346.94

Table 7: Heavy Ball with λ = 0.1 and 1000 iterations

With **λ = 0.1**, the **Deflected Subgradient (DSG)** method achieved a lower final objective function value compared to **Heavy Ball (HB)**, with results ranging between **2.30 and 2.32**.
This suggests that, under stronger regularization, DSG is able to obtain a more stable solution and is less sensitive to gradient fluctuations than HB.
From the standpoint of **convergence speed**, DSG's performance varies significantly depending on the chosen parameters.
The best configurations achieved values exceeding **80.0**, while others reached convergence in fewer than **600 iterations**.
Regarding **instability**, most configurations showed relatively low values, with some exceptions.
In particular, configurations with **α = 5** and **γ = 0.3** recorded high instability levels (**54.20**), suggesting that excessively large deflection parameters (**γ**) can compromise the algorithm's stability.
The **α parameter** has a strong impact on convergence speed — higher values such as **α = 5** tend to reduce the number of iterations required to converge but simultaneously increase the risk of instability.
The **γ parameter** (deflection factor) primarily affects algorithm stability.
Lower values of **γ (0.1 – 0.5)** provided the best performance, whereas higher values showed a tendency toward greater instability.
The **optimal configuration** for the Deflected Subgradient algorithm with a maximum of **1,000 iterations** is as follows:

- **α = 1.0**
- **γ = 0.5**
- **Objective function = 2.308**
- **Convergence speed = 50.6**
- **Instability = 6.67**

Table 8 reports the **10 best configurations** of the Deflected Subgradient method with a maximum of **1,000 iterations** and **$\epsilon = 1 \times 10^{-4}$**.

α	γ	**Objective Function**	**Convergence speed**	**Instability**
1	0.5	2.308	50.6	6.67
2	0.3	2.312	71.2	9.93
2	0.5	2.313	68.6	40.50
1	0.9	2.314	63.5	18.10
5	0.1	2.315	65.7	6.65
2	0.1	2.317	84.2	2.03
2	0.7	2.319	58.1	11.62
1	0.7	2.320	66.3	1.50
5	0.3	2.325	50.8	54.21
2	0.9	2.326	50.3	28.01

Table 8: Deflected Subgradient with λ = 0.1 and 1000 iterations

After identifying the best parameters from the initial grid search, a more detailed analysis was conducted by increasing the **maximum number of iterations** to **10,000** and reducing **ϵ to 1e−6**, in order to obtain a more accurate estimate of the algorithms' convergence.
The configurations tested for each algorithm were as follows:

- **For Heavy Ball (HB):**
  - α = 0.01
  - μ = [0.1, 0.3, 0.5, 0.7, 0.9]

- **For Deflected Subgradient (DSG):**
  - α = [1, 2, 5]
  - γ = [0.1, 0.3, 0.5, 0.7, 0.9]

Compared to the results obtained with 1,000 iterations, the extended optimization produced a slight — though not significant — improvement in the objective function:

- The **minimum objective function value** obtained was **2.412**, very close to the value reached with fewer iterations.
- The **convergence speed** improved considerably, increasing from **43 to 94**, suggesting that the algorithm approached the optimum more rapidly than in the **λ = 0.01** case.
- The **instability** remained high but within acceptable limits.

The **optimal configuration** for the **Heavy Ball algorithm** with **10,000 iterations** is as follows:

- **α = 0.01**
- **μ = 0.5**
- **Objective function = 2.419**
- **Convergence speed = 96.76**
- **Instability = 25.67**

Below is **Table 9**, which reports the best configurations for the **Heavy Ball algorithm** with **10,000 iterations** and **ϵ = 1e−6**.

α	μ	**Objective Function**	**Convergence speed**	**Instability**
0.01	0.1	2.412	94.28	37
0.01	0.3	2.417	95.50	53.54
0.01	0.5	2.419	96.76	25.67
0.01	0.7	2.469	98.04	36.76
0.01	0.9	2.568	98.03	67.82

Table 9: Heavy Ball with λ = 0.01 and 10000 iterations

The **Deflected Subgradient (DSG)** method achieved a slightly lower final objective function value compared to **Heavy Ball (HB)**, with a minimum value of **2.3026**, showing an improvement over the **2.3082** recorded in the 1,000-iteration phase.

From the standpoint of **convergence speed**, DSG reached high values, with most configurations ranging between **90.0 and 97.76**.

This suggests that the algorithm maintained a stable trajectory and continued improving the objective function as the number of iterations increased.

Instability remained extremely low in most configurations, with values below **2.0** in the majority of cases. Only for **γ = 0.5** was a maximum value of **7.17** observed — still much lower than the oscillations seen in HB.

Differences between the **γ configurations** became more apparent in this phase. Lower values of **γ (0.1 – 0.3)** produced more stable results, while higher values led to greater instability, although still within manageable limits.

The **best configuration** for the **Deflected Subgradient algorithm** with **10,000 iterations** is as follows:

- **α = 2.0**
- **γ = 0.1**
- **Objective function = 2.302**
- **Convergence speed = 97.76**
- **Instability = 0.53**

Below is **Table 10**, which lists the **best configurations** for the **Deflected Subgradient algorithm** with **10,000 iterations** and **ϵ = 1e−6**.

α	γ	**Objective Function**	**Convergence speed**	**Instability**
2	0.1	2.302	97.76	0.53
1	0.1	2.3028	70.61	0.05
1	0.3	2.3030	90.83	0.35
2	0.3	2.3030	95.99	2.01
1	0.5	2.3032	94.54	2.02
5	0.1	2.3032	94.95	1.56
2	0.5	2.3033	95.56	7.17
1	0.7	2.3035	95.49	0.65
1	0.9	2.3039	95.25	3.91
2	0.7	2.3041	93.01	2.34

Table 10: Deflected Subgradient with $\lambda = 0.1$ *and 10000 iterations*

The analysis of the **quality of the minima** reached by the **Heavy Ball (HB)** and **Deflected Subgradient (DSG)** algorithms for a regularization value of **λ = 0.1** revealed significant differences compared to the **λ = 0.01** case.
The increased penalization on the weights affected both the **stability** and the **ability** of the algorithms to locate deep minima of the objective function.

From the results obtained after **10,000 iterations**, it emerges that **HB** reaches a **higher objective function value** than **DSG**, with a minimum of **2.413** versus **2.302**.
However, **HB** exhibits greater instability, with oscillations that increase as the **momentum parameter (μ)** grows.
In contrast, **DSG** maintains a smoother behavior, with low instability even under more aggressive configurations.
These results suggest that, with a limited number of iterations, **HB** descends quickly toward a local minimum, while **DSG** proceeds more gradually but more stably.

The analysis with a higher number of iterations (up to **10,000**) allowed for a more accurate assessment of the convergence behavior and any substantial differences between the two algorithms.
In this extended phase, **DSG** managed to reduce the objective function value to **2.302**, outperforming the minimum found by **HB**, which stabilized at **2.412**.
This result indicates that, unlike the **λ = 0.01** case, in this scenario **DSG** does **not get stuck in a suboptimal local minimum**, but instead continues to improve the solution, surpassing **HB** in the long run.
Moreover, **DSG's stability** remains a key strength, showing negligible instability compared to **HB**.

The objective function value of the **HB algorithm** oscillates between **2.41** and **2.42** from iteration **400** onward, up to iteration **10,000**.
This behavior indicates that the algorithm becomes trapped in a **local minimum**, which is less favorable than the one achieved by the **Deflected Subgradient method**.

The higher stability of **DSG**, combined with its ability to reach a **lower objective function value**, suggests that this algorithm is **more robust** and less prone to getting stuck in **poor-quality local minima**.
While **HB** shows a higher initial convergence speed, it suffers from increasing instability, which can compromise its ability to find optimal solutions in the long term.

In summary, increasing the **regularization parameter (λ)** allowed **DSG** to achieve **better performance** than **HB**, reversing the behavior observed in the **λ = 0.01** case.

3.2.2.1 Convergence rate analysis with λ = 0.1

As in the analysis conducted with **λ = 0.01**, a comparison will be made between the **theoretical** and **empirical convergence rates** of the algorithms under consideration, followed by a brief comparison between the two methods.
The procedures used are the same as those described in **Section 3.3.2**.

In the following **Figure 8**, the **empirical convergence** of the **Heavy Ball (HB)** algorithm is analyzed using the parameters **α = 0.01** and **μ = 0.5**, and compared with its **theoretical convergence rate**, expected to be **O(1/k)**.

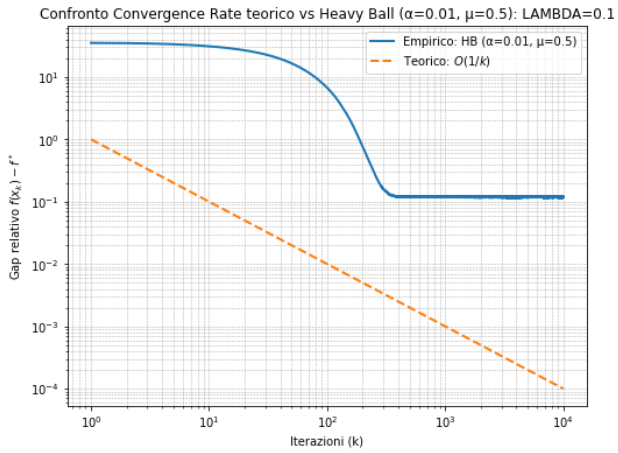


Figure 8: Comparison between theoretical and HB .

In the first graph, a comparison is shown between the **theoretical convergence rate O(1/k)** and the **empirical behavior** of the **Heavy Ball (HB)** algorithm with parameters **α = 0.01** and **μ = 0.5**.

During the initial iterations, the algorithm follows a trajectory consistent with theoretical predictions, showing a progressive reduction of the relative gap.
However, after a certain number of iterations, **HB enters a plateau phase**: the objective function stops decreasing significantly and begins to oscillate around a value between **2.41 and 2.42**.
This behavior suggests that the algorithm has stabilized in a **local minimum**.

The observed convergence rate becomes lower than the theoretical one in later iterations, indicating **suboptimal performance** compared to the expected **O(1/k)** behavior.
Moreover, no significant improvements are observed in the subsequent iterations, suggesting that **HB may have become trapped in a non-optimal local minimum**.

In the next **Figure 9**, a similar comparison is made for the **Deflected Subgradient (DSG)** algorithm, using the parameters **α = 2.0** and **γ = 0.1**, and comparing it with its **expected theoretical convergence rate**, **$O(1/\sqrt{k})$**.

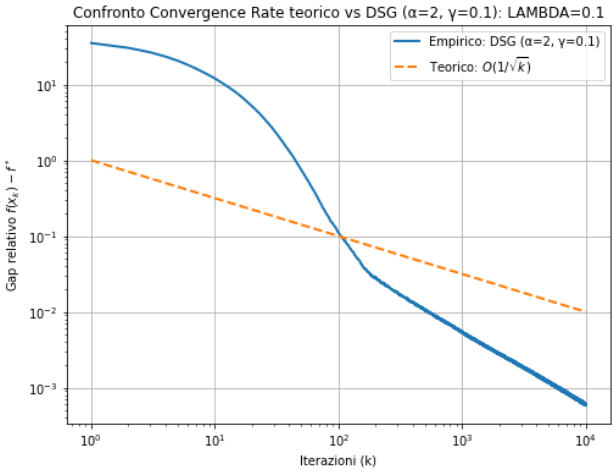


Figure 9: Comparison between theoretical and DSG .

The **Deflected Subgradient (DSG)** algorithm, although it initially appears to perform slightly below theoretical expectations, manages to **decrease consistently**, eventually even **surpassing the expected theoretical convergence rate**.
This result highlights the **effectiveness of the method even in the presence of regularization**, demonstrating that it is an excellent choice for ensuring **stability during the optimization process**.

The **L1 regularization** with **λ = 0.1** has a **significant impact on the convergence speed** compared to the case with **λ = 0.01**.
This finding suggests that the method is capable of maintaining a **stable and efficient behavior** even with stronger penalization terms.

In **Figure 10**, a **direct comparison** is made between the two algorithms:
Heavy Ball (HB) with parameters **α = 0.01** and **μ = 0.5**, and **Deflected Subgradient (DSG)** with parameters **α = 2.0** and **γ = 0.1**, using the **relative gap** as the evaluation metric.

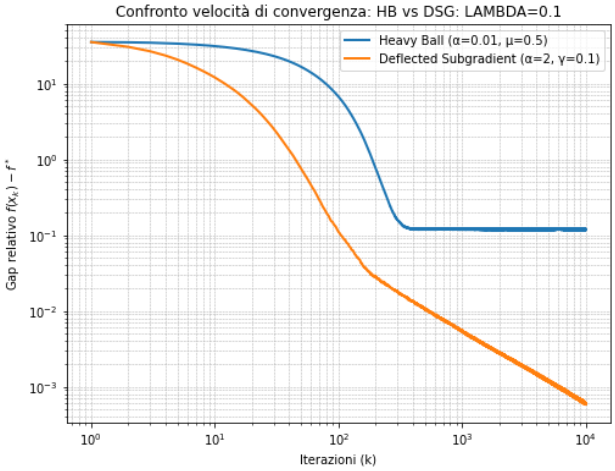


Figure 10: Comparison HB vs DSG .

The third graph presents a **direct comparison** between **Heavy Ball (HB)** and **Deflected Subgradient (DSG)** in terms of **convergence speed**.
It is clearly observed that, although **DSG** theoretically has a slower convergence rate than **HB**, it ultimately achieves a **better solution in the long run**.

While **HB** quickly stabilizes at an objective function value between **2.41 and 2.42**, **DSG** continues to improve, reaching a **lower value of 2.30**.
This behavior suggests that, although **HB** exhibits faster initial convergence, it tends to become **trapped in a shallower local minimum** compared to **DSG**.

In particular, **HB** reaches a solution rapidly but stabilizes prematurely at a higher objective value, indicating the presence of a **suboptimal local minimum**.
Conversely, **DSG** maintains a **more gradual improvement**, continuing to decrease the objective function even in the later iterations and achieving a **lower final value** than **HB**.

3.3 Analysis with $\lambda = 1$

After analyzing the behavior of the optimization algorithms with $\lambda = 0.01$ and $\lambda = 0.1$, a higher regularization value, $\lambda = 1.0$, was tested to evaluate how a significantly stronger penalty on the weights affects the performance of the **Heavy Ball (HB)** and **Deflected Subgradient (DSG)** algorithms.

The goal of this analysis is to determine whether such a high level of regularization leads to **greater model generalization and stability**, and how it impacts the **convergence rate** and **stability** of the two algorithms.

As in the previous tests, the evaluation was carried out in **two distinct phases**:

- **First phase:** A **grid search** was performed with a maximum of **1,000 iterations** and $\epsilon = 1e{-}4$, to identify the best initial configurations.
- **Second phase:** The best parameters identified in the first phase were then tested with **10,000 iterations** and $\epsilon = 1e{-}6$, to more accurately assess the long-term behavior of the algorithms.

With $\lambda = 1.0$, the **Heavy Ball (HB)** algorithm exhibited **extremely unstable behavior**, characterized by large oscillations in the objective function and very high values.

The main findings from the analysis are as follows:

- The **objective function value** is drastically higher than in the tests with $\lambda = 0.01$ and $\lambda = 0.1$, reaching **values above 30–70** in the most unstable configurations.
- Many configurations proved **unstable**, especially for $\mu \geq 0.7$, with instability levels exceeding **1000**.
- For $\mu = 0.1$ and $\mu = 0.3$, stability improved slightly, with an objective function value between **13.24** and **13.43**, although these values remain significantly higher than in the previous tests.
- Increasing μ leads to a **higher convergence speed** (above **95**), but at the expense of stability.
- Configurations with $\alpha = 0.05$ and $\alpha = 0.1$ were found to be **highly unstable**, particularly for $\mu \geq 0.5$.

The **best configuration** for the **Heavy Ball** algorithm with **1,000 iterations** (representing the best trade-off between stability and convergence speed) is as follows:

- **α = 0.01**
- **μ = 0.1**
- **Objective function = 13.43**
- **Convergence speed = 94.0**
- **Instability = 445.17**

Below, **Table 11** reports the best configurations for the **Heavy Ball** algorithm with **1,000 iterations** and **ϵ = 1e−4**.

α	μ	**Objective function**	**Convergence speed**	**Instability**
0.01	0.3	13.24	95.2	1138.70
0.01	0.1	13.43	94.0	445.17
0.01	0.5	14.53	93.8	795.79
0.01	0.7	19.19	97.4	1284.66
0.01	0.9	30.89	92.3	1004.09
0.05	0.1	58.31	98.6	671.05
0.05	0.3	59.07	99	1128.49
0.05	0.5	64.17	98.3	2343.45
0.05	0.7	73.99	98.7	7433.80
0.10	0.5	111.54	98.7	5985.84

Table 11: Heavy Ball with λ = 1 and 1000 iterations

For the **Deflected Subgradient (DSG)** algorithm, the behavior observed with **λ = 1.0** is markedly different from that of both the **Heavy Ball (HB)** algorithm and the tests conducted with lower regularization values.

The main findings from the analysis are as follows:

- The **objective function value** is **significantly lower** than that of HB, ranging between **2.35 and 2.57**, indicating that the method is **more effective at finding better solutions** even under strong regularization.
- **Instability remains low**, with generally very small values (below **20**) except for a few isolated configurations.
- An increase in **α** tends to improve the **convergence speed**, reaching values up to **66.3** in the best-performing configurations.

The **optimal configuration** for the **DSG algorithm** with **1,000 iterations** is as follows:

- **α = 0.20**
- **γ = 0.1**
- **Objective function = 2.35**
- **Convergence speed = 34.9**

- **Instability = 18.97**

Below, **Table 12** presents the best configurations for the **Deflected Subgradient (DSG)** algorithm with **1,000 iterations** and **ϵ = 1e−4**.

α	γ	**Objective function**	**Convergence speed**	**Instability**
0.20	0.1	2.35	34.9	18.97
0.20	0.3	2.40	20.6	124.22
0.50	0.1	2.42	16.9	82.61
0.10	0.7	2.43	12.4	32.80
0.10	0.5	2.43	9.4	73.05
0.10	0.9	2.44	11.1	562.50
0.20	0.7	2.46	12.6	86.78
0.75	0.1	2.49	12.9	158.40
1.00	0.1	2.54	8.1	239.36
0.10	0.3	2.57	5.3	5.00

Tabella 12: Deflected Subgradient con λ = 1 e 1000 iterazioni

After identifying the optimal parameters from the initial grid search, a more in-depth analysis was conducted by **increasing the number of iterations to 10,000** and **reducing ϵ to 1e−6**, in order to obtain a more precise estimate of the algorithms' convergence behavior.
The configurations tested for each algorithm were as follows:

- **For Heavy Ball (HB):**
  – **α = 0.01**
  – **μ = [0.1, 0.3, 0.5, 0.7, 0.9]**
- **For Deflected Subgradient (DSG):**
  – **α = [0.1, 0.2]**
  – **γ = [0.1, 0.3, 0.5, 0.7, 0.9]**

For the **Heavy Ball algorithm**, increasing the number of iterations led to a **slight improvement in the objective function**, though not a significant one compared to the previous results:

- The **minimum objective function** reached was **13.57**, a value very close to that obtained with fewer iterations.
- The **convergence speed** improved considerably, increasing from **43 to 73**, suggesting that the algorithm can approach the optimum more quickly.
- However, **instability remains high**, with values exceeding **900** even in the best configuration.

The **optimal configuration** for the **Heavy Ball algorithm** with **10,000 iterations** was as follows:

- **α = 0.01**
- **μ = 0.5**
- **Objective function = 14.07**
- **Convergence speed = 60.2**
- **Instability = 791.20**

Table 13 reports the best configurations of the **Heavy Ball (HB)** algorithm obtained with **10,000 iterations** and **ϵ = 1e−6**.

α	μ	**Objective function**	**Convergence speed**	**Instability**
0.01	0.3	13.24	99.52	1138.70

0.01	0.1	13.57	73.05	912.98
0.01	0.5	14.07	60.20	791.20
0.01	0.7	19.19	99.74	1284.66
0.01	0.9	30.89	99.23	1004.09
0.05	0.1	57.92	98.42	571.85
0.05	0.3	59.07	99.90	1128.49
0.05	0.5	64.17	98.83	2243.45
0.05	0.7	73.99	98.87	7433.80
0.05	0.9	120.94	99.88	3624.98

Table 13: Heavy Ball with $\lambda = 1$ and 10000 iterations

The **Deflected Subgradient (DSG)** algorithm shows a **clear improvement** over the **Heavy Ball (HB)** algorithm, achieving **significantly more stable results** and a **much lower objective function value**:

- The **minimum objective function** dropped to **2.307**, getting much closer to the optimum compared to what was observed with HB.
- The **convergence speed** increased substantially, exceeding values between **70 and 80** in the best configurations.
- **Instability** is practically zero, confirming that the method remains **extremely stable** even with a very high number of iterations.

The **optimal configuration** for the **DSG algorithm** with **10,000 iterations** is as follows:

- **$\alpha = 0.2$**
- **$\gamma = 0.1$**
- **Objective function = 2.307**
- **Convergence speed = 82.70**
- **Instability = 5.11**

Table 14 presents the best configurations obtained for the **Deflected Subgradient (DSG)** algorithm with **10,000 iterations** and **$\epsilon = 1e{-}6$**.

α	γ	**Objective function**	**Convergence speed**	**Instability**
0.1	0.3	2.307	65.60	3.86
0.1	0.5	2.307	77.80	23.12
0.2	0.1	2.307	82.70	5.11
0.1	0.7	2.310	73.57	9.68
0.2	0.3	2.312	71.08	22.34
0.1	0.9	2.314	68.63	123.45
0.2	0.7	2.319	58.50	22.23
0.1	0.1	2.323	20.14	0.50

Table 14: Deflected Subgradient with $\lambda = 1$ and 10000 iterations

The analysis of the convergence quality of the **Heavy Ball (HB)** and **Deflected Subgradient (DSG)** algorithms, under strong regularization (**$\lambda = 1.0$**), revealed even more pronounced differences compared to

tests with lower λ values.
The increased regularization penalty had a significant impact on performance, particularly affecting **stability** and the ability to identify **high-quality minima**.

The **HB algorithm** exhibited extremely **unstable behavior**. Configurations with higher **μ** values produced substantial oscillations in the objective function, in some cases exceeding **70**, which is markedly worse than the results obtained with lower regularization values.
Even in relatively more stable settings, such as **μ = 0.1** or **μ = 0.3**, the objective function settled between **13.24 and 13.57**, values significantly higher than those obtained with **λ = 0.1** or **λ = 0.01**. These results suggest that HB fails to adapt effectively to such a high level of regularization, resulting in a **severe degradation in solution quality**.

Increasing the number of iterations to **10,000** did not yield any meaningful improvement for HB. The minimum objective function value remained at **13.57**, still far from the results achieved by DSG. Furthermore, **instability** remained a critical issue: even in the best configuration, instability levels exceeded **900**, making the algorithm highly unpredictable and unreliable under strong regularization.

In contrast, the **DSG method** demonstrated **significantly better performance**. With **λ = 1.0**, the algorithm reached an objective function value of **2.307**, much lower than that of HB. Moreover, DSG maintained **excellent stability**, with instability levels close to zero in most tested configurations.
These results indicate that DSG is **much better suited for high-regularization scenarios**, as it maintains a smoother optimization trajectory and converges toward higher-quality solutions.

The comparative analysis between the two algorithms shows that, with **λ = 1.0**, **DSG is clearly more effective** at finding a deeper and more stable minimum.
In contrast, **HB not only converges to poorer local minima** but also suffers from **severe oscillations** that hinder convergence.
The deterioration of HB's performance as regularization increases suggests that this algorithm is **not suitable** for scenarios with strong weight penalization.

In conclusion, with **λ = 1.0**, **DSG emerges as the superior algorithm**, providing **better-quality final solutions** and **significantly higher stability**.
HB, on the other hand, **fails to improve** the objective function value and becomes **highly unstable**, confirming that it is **not an optimal choice** in high-regularization contexts.

3.2.3.1 Convergence Rate Analysis with λ = 1

After analyzing the convergence behavior of the **Heavy Ball (HB)** and **Deflected Subgradient (DSG)** algorithms with **λ = 0.01** and **λ = 0.1**, we now consider the case where the **L1 regularization** reaches its highest level (**λ = 1**).
The goal of this analysis is to examine how such a strong penalization affects the **convergence speed** and **stability** of both algorithms.

As in the previous cases, the comparison is conducted using the **relative gap** metric.
Figure 11 shows the empirical convergence of the **Heavy Ball (HB)** algorithm, with parameters **α = 0.01** and **μ = 0.5**, compared to its theoretical convergence rate of **O(1/k)**.

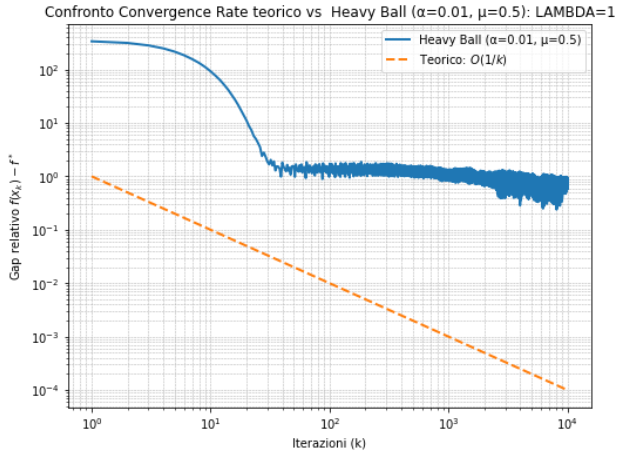


Figure 11: Comparison between theoretical and HB.

In the initial stages, the algorithm follows a behavior consistent with the expected theoretical convergence rate, showing a gradual reduction in the relative gap.
However, in the later iterations, **clear instability** emerges, with significant oscillations. This behavior is similar to what was observed with **λ = 0.1**, but in this case, the fluctuations are even more pronounced.

The instability can be attributed to **excessive acceleration** in the **Heavy Ball (HB)** method, which causes oscillations around the minimum rather than achieving smooth and stable convergence.
Compared to cases with lower λ values, the algorithm appears to be **more affected by the strong L1 regularization penalty**, which compromises its ability to reach a stable solution.

Figure 12 analyzes the **empirical behavior** of the **Deflected Subgradient (DSG)** algorithm, using parameters **α = 0.2** and **γ = 0.1**, and compares it with the **expected theoretical convergence rate**, given by **O(1/ k)**.

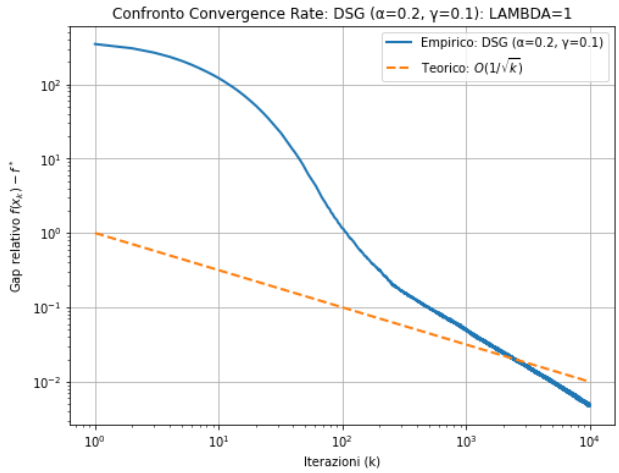


Figure 12: Comparison between theoretical and DSG .

Unlike the **Heavy Ball (HB)** algorithm, the **Deflected Subgradient (DSG)** method maintains a **consistent reduction** in the relative gap without exhibiting oscillations.
Although convergence appears slightly slower than the theoretical rate in the initial iterations, the algorithm

improves its behavior in later stages, eventually **aligning with or even surpassing** the expected theoretical convergence rate.

The **absence of oscillations** is a clear advantage of DSG over HB, demonstrating **greater robustness and stability**.
The presence of strong **L1 regularization** does not significantly affect the method's efficiency, confirming that **DSG is a stable and reliable choice** even under heavy penalization.

Similarly, **Figure 13** presents a **direct comparison** between the two algorithms: **Heavy Ball (HB)**, with parameters **α = 0.01** and **μ = 0.5**, and **Deflected Subgradient (DSG)**, with parameters **α = 0.2** and **γ = 0.1**, based on the **relative gap**.

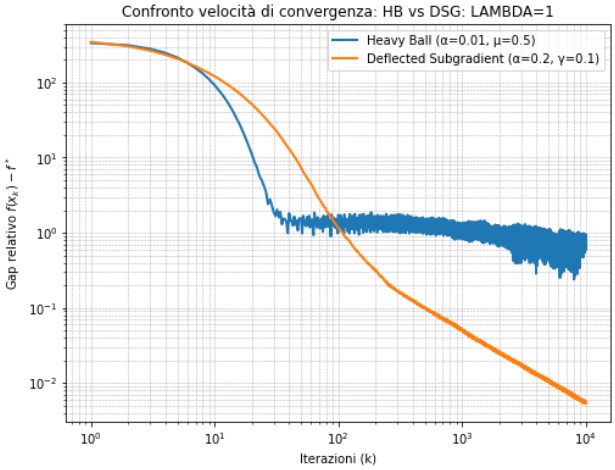


Figure 13: Comparison HB vs DSG .

The **Heavy Ball (HB)** algorithm exhibits a faster initial phase, showing a quicker reduction in loss compared to the **Deflected Subgradient (DSG)** method.
However, in the later stages, **HB becomes unstable**, and the resulting oscillations prevent smooth convergence toward the minimum.

In contrast, **DSG** starts off more slowly but ensures a **steady and fluctuation-free decrease**. Its **final convergence rate** surpasses that of HB, as the **relative gap continues to shrink**, while HB oscillates around unstable values without significant improvement.

The strong regularization with **λ = 1** further accentuates the limitations of HB, which becomes increasingly unstable as the number of iterations grows. Conversely, **DSG maintains a stable and consistent behavior** throughout the entire optimization process.

In the presence of strong regularization (**λ = 1**), the **Deflected Subgradient algorithm** proves to be **preferable to Heavy Ball** — not only does it offer greater **stability**, but it also achieves a **better final convergence rate**. Moreover, DSG maintains **consistent behavior** even in high-penalization scenarios, demonstrating **greater robustness** compared to HB.

4 Conclusions

5 4. Conclusions

In this work, an in-depth analysis was conducted on the performance of the **Heavy Ball (HB)** and **Deflected Subgradient (DSG)** optimization algorithms, applied to a neural network with a **cross-entropy loss function** and **L1 regularization**.
The main objective was to compare the **convergence speed**, **stability**, and **effectiveness** of the two methods under different levels of weight penalization, by varying the regularization parameter **λ** among **0.01**, **0.1**, and **1.0**.

The analysis was structured into several phases:

1. **Initial Grid Search** — A parameter search was conducted with a maximum of **1,000 iterations** and **ϵ = 1e−4** to identify the best configurations for each algorithm.
2. **Advanced Optimization** — Using the best parameters from the first phase, the algorithms were tested with up to **10,000 iterations** and **ϵ = 1e−6** to evaluate their long-term behavior.
3. **Convergence Rate Analysis** — The empirical performance of HB and DSG was compared with their theoretical convergence rates:
    - **$O(1/k)$** for HB, with possible deviations due to the non–L-smooth nature of the loss function.
    - **$O(1/\sqrt{k})$** for DSG, typical of subgradient-based algorithms.

From the results, several key insights emerged regarding the performance of the two algorithms:

- The **Heavy Ball (HB)** algorithm exhibits **faster initial convergence** compared to the **Deflected Subgradient (DSG)**, particularly for low regularization values (**λ = 0.01**). However, while DSG starts slower, it maintains a **more consistent and stable** convergence trajectory over time.
- As the number of iterations increases, **HB tends to become unstable**, especially for higher λ values. This instability manifests as significant oscillations in the objective function, likely caused by excessive acceleration from the momentum term.
  In contrast, **DSG maintains much higher stability** across all cases, following a smooth and fluctuation-free convergence path even under strong regularization.

In terms of the **effect of regularization**:

- For **λ = 0.01**, both algorithms show competitive results. HB performs slightly better in convergence speed but suffers from instability in later iterations.
- For **λ = 0.1**, DSG becomes more competitive, offering smoother and more stable convergence than HB, which continues to exhibit strong oscillations.
- For **λ = 1**, DSG clearly outperforms HB: the Heavy Ball method becomes completely unstable and ineffective, while DSG continues to deliver **reliable convergence** and achieves a **lower objective function value**.

Bibliography

[1] Vassilis Apidopoulos, Nicol`o Ginatta e Silvia Villa. «Convergence rates for the heavy-ball continuous dynamics for non-convex optimization, under Polyak–L ojasiewicz condition». *Journal of Global Optimization* 84.3 (2022), pp. 563–589.

[2] Jean-Franc¸ois Aujol, Charles Dossal e Aude Rondepierre. «Convergence rates of the Heavy-Ball method with Lojasiewicz property». Tesi di dott. IMB - Institut de Math´ematiques de Bordeaux; INSA Toulouse; UPS Toulouse, 2020, pp. 6–15.

[3] Stephen Boyd e Jonghyuk Park. «Subgradient Methods: Notes for EE364b, Stanford University, Spring 2013–14». Tesi di dott. Stanford University, 2014, pp. 4–12.

[4] G. D'Antonio. «Porting ed estensione di codice C++ per l'ottimizzazione non differenziabile». Tesi di dott. Universit`a di Pisa, 2006. Cap. 2.1, pp. 24–36. url: https://commalab.di.unipi.it/files/Theses/ dAntonio.pdf.

[5] A. Frangioni. *Nonsmooth Unconstrained Optimization*. Universit`a di Pisa. 2024-25.

[6] A. Frangioni. *Smooth Unconstrained Optimization*. Universit`a di Pisa. 2024-25.