Università di Padova
Dipartmento di Matematica

Gradient and Coordinate Descent Methods: Theory, Implementation, and Experimental Insights

Optimization for Data Science

Contributors: Lazzari Tommaso
Ludergnani Brenno
Movila Dumitru
Safa Nasser

Date: May 2025

Table of Contents

1 Introduction

Optimization methods play a central role in modern machine learning and data science, forming the backbone of many algorithms used for classification, regression, and clustering tasks. Among these methods, Gradient Descent and its variants remain fundamental due to their simplicity, efficiency, and theoretical grounding. In this work, we investigate and compare the performance of several optimization algorithms applied to a binary classification problem, with the objective of understanding their convergence behavior and practical effectiveness. Specifically, we consider three main approaches: Gradient Descent (implemented with fixed step size, Armijo rule, and exact line search), Coordinate Minimization, and Block Coordinate Gradient Descent. Each method is first analyzed from a theoretical standpoint, highlighting its update mechanism, convergence guarantees, and computational trade-offs. Subsequently, we conduct a series of experiments on both synthetic datasets and the Breast Cancer dataset from the *scikit-learn* library to empirically evaluate their performance. Through this combined theoretical and empirical analysis, we aim to provide a deeper understanding of the strengths and limitations of these optimization algorithms within the context of supervised learning.
The theoretical background and convergence analyses presented in this work are primarily based on the lecture notes of the course "Optimization Methods for Data Science" at the University of Padova, authored by Professor Rinaldi Francesco [1]. The material has been adapted and reformulated for clarity and consistency with the experimental part of this study.

2 Gradient Descent

Gradient descent schemes can be traced back to Cauchy and represent the simplest way to minimize a differentiable function f on $mathbbR^n$. The main idea here is using a linear approximation of $f(x_k + d)$ to calculate the search direction at each iteration. In practice, we approximate $f(x_k + d)$ with the function $\eta_k(d)$ defined as follows:

$$\eta_k(d) := f(x_k) + \nabla f(x_k)^T d$$

Then we choose d_k as the direction that minimizes $\eta_k(d)$ over the unitary euclidean ball. In practice, we consider problem:

$$\min_{||d||=1} \eta_k(d)$$

which is equivalent to

$$\min_{||d||=1} \nabla f(x_k)^T d$$

Using the Cauchy-Schwarz inequality, we can prove that the optimal direction is $d_k^* = -\nabla f(x_k)/||\nabla f(x_k)||$.

Hence the classic *gradient method* calculate each iterate as follows:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

A general scheme for gradient descent algorithms is given below.

Algorithm 1 Gradient method

1: Choose a point $x_1 \in \mathbb{R}^n$
2: **for** $k = 1, \ldots$ **do**
3: **if** x_k satisfies some specific condition **then**
4: **stop**
5: **end if**
6: Set $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$, with $\alpha_k > 0$ a suitably chosen stepsize
7: **end for**

2.1 Gradient Descent Convergence Analysis

Theorem 1 *Let $f : \mathbb{R}^n - > \mathbb{R}$ be a function with Lipschitz continuous gradient having Lipschitz constant $L > 0$. Gradient method with fixed stepsize $\alpha_k = 1/L$, satisfies:*

$$f(x_{k+1}) - f(x^*) \leq \frac{2L||x_1 - x^*||^2}{k}$$

This result says that the gradient method has convergence rate $O(1/k)$. Taking into account this piece of information, we can calculate the number of iterations needed to get an optimality gap lower or equal than ϵ. In practice, we want

$$f(x_{k+1}) - f(x^*) \leq \frac{c}{k} \leq \epsilon$$

with $c > 0$ a specific constant that depends on the values reported in **Theorem 1**, we hence need a number of iterations of the order $O(1/\epsilon)$ If we uso Armijo line search to calculate α_k at each step, we obtain the same convergence rate with slighlty different constant c (which depends on the parameters of the line search).

Theorem 2 *Let $f : \mathbb{R}^n - > \mathbb{R}$ be a σ-strongly convex function with Lipschitz continuous gradient having Lipschitz constant $L > 0$. Gradient method with fixed stepsize $\alpha_k = 1/L$ satisfies:*

$$f(x_{k+1}) - f(x^*) \leq (1 - \frac{\sigma}{L})^k (f(x_1) - f(x^*))$$

This result says that gradient method has convergence rate $O(c^k)$, with the constant $0 \leq c \leq 1$ that depends on the values reported in **Theorem 2**. In the literature, the value $\frac{L}{\sigma}$ is called the *condition number*. It is easy to see that, the higher the condition number, the slower will be the convergence rate of our algorithm.
Taking into account this piece of information, we can calculate the number of iterations needed to get an optimality gap lower or equal than ϵ. In practice, we want

$$f(x_{k+1}) - f(x^*) \leq c^k \leq \epsilon$$

We hence have

$$k \log(c) \leq \log(\epsilon)$$

keeping in mind that $\tilde{c} = \frac{1}{\log(c)} < 0$, we have

$$k \geq -\tilde{c} \log(\epsilon^{-1})$$

Thus we get a number of iterations of the order $O(log(1/\epsilon))$. Considering the result obtained before for f with Lipschitz continuous gradient (that is order $O(1/\epsilon)$, we can easily see that gradient is much faster when handling strongly convex functions. A slightly better rate can be obtained when using a different fixed stepsize.

Theorem 3 *Let $f : \mathbb{R}^n -> \mathbb{R}$ be a σ-strongly convex function with Lipschitz continuous gradient having Lipschitz constant $L > 0$. Gradient method with fixed stepsize $\alpha_k = 2/(\sigma + L)$ satisfies:*

$$f(x_{k+1}) - f(x^*) \leq \frac{L}{2} \left(\frac{\frac{L}{\sigma} - 1}{\frac{L}{\sigma} + 1} \right)^{2k} ||x_1 - x^*||^2$$

Concluding, the gradient method is based on a simple idea and is very easy to implement. Furthermore, each iteration is relatively cheap. The results described in this subsection show that the algorithm is very fast when dealing with well-conditioned and strongly convex problems.

3 Coordinate Minimization Method

When the number of variables n is large, it becomes computationally expensive to calculate full gradients, which means gradient descent methods might not be efficient. Taking into account optimality conditions for unconstrained optimization, we can say that x^* is an optimal solution if and only if $\nabla f(x^*) = 0$, namely $\nabla_i f(x^*), \forall i = 1, \ldots, n$.
In order to get the optimal solution, it makes sense to search along each coordinate; if at some point, the objective is not decreasing at every coordinate direction, then we have obtained the optimum. This motivates the so-called *Coordinate minimization algorithms* (also known as *Coordinate descent* algorithms). So, the idea is to split the optimization process into a sequence of simpler optimizations, with the motivation to exploit the special structure of the problem. A general scheme for this class of algorithms is given below.

Algorithm 2 Coordinate minimization method

1: Choose a point $x_1 \in \mathbb{R}^n$
2: **for** $k = 1, \ldots$ **do**
3: **if** x_k satisfies some specific condition **then**
4: **stop**
5: **end if**
6: Pick coordinate i from 1 to n and set

$$s_k^{(i)} = \arg \min_{x^{(i)} \in \mathbb{R}} f(x^{(i)}, \mathbf{x}_{-i})$$

7: Use $s_k^{(i)}$, with $i = 1, \ldots, n$, to build x_{k+1}
8: **end for**

The symbol $\boldsymbol{x}_{-i}$ indicates the set of all variables but $x^{(i)}$. The scheme is very general and embeds different strategies depending on the choice of $x^{(i)}$ and $\boldsymbol{x}_{-i}$. Obviously, the method we have just described makes practical sense if the minimizations to be done at Step 4 are fairly easy. Since $x^{(i)}$ is a scalar, but there are also other cases of interest, where the block of variables is multidimensional.

3.1 Coordinate Minimization Schemes

Now we describe two different schemes that fit into this framework:

- **Gauss-Seidel scheme:** when updating each coordinate, at Step 4, we set the coordinates $\boldsymbol{x}_{-i}$ to the most up-to-date value, that is

  $$\boldsymbol{x}_{-i} = (s_k^{(1)}, \ldots, s_k^{(i-1)}, s_k^{(i+1)}, \ldots, s_k^{(n)})$$

  At Step 5, we simply set

  $$x_{k+1}^{(i)} = s^{(i)}$$

  to get the new iterate. This has a potential advantage of converging faster, but minimizations need to be done in sequential order.

- **Jacobi scheme:** when updating each coordinate, at Step 4, we set the coordinates $\boldsymbol{x}_{-i}$ to be the solution obtained from previous cycle, that is

  $$\boldsymbol{x}_{-i} = (x_k^{(1)}, \ldots, x_k^{(i-1)}, x_k^{(i+1)}, \ldots, x_k^{(n)})$$

  The Jacobi update does not take into account intermediary iterates until we complete all coordinates. At Step 5, we need to gather the information coming from the different minimization steps in order to generate a new iterate that guarantees a good decrease of the objective function. This strategy enables us to run the update for each coordinate in parallel.

When f is strictly convex and smooth, the algorithm converges to a solution. If f is non-convex or non-smooth, the algorithm might not converge at all.

4 Block Coordinate Gradient Methods

We now consider the same unconstrained minimization problem as before and assume that the variables are partitioned in b blocks of dimension $n_i, i = 1, \dots, b$ such that $n = n_1 + \cdots + n_b$. We define matrices $U_i \in \mathbb{R}^{nxn_i}$ and indicate the identity matrix $I_n = |U_1| \dots |U_b|$. Thus we get the i-th block as follows

$$x^{(i)} = U_i^T x$$

We further indicate the i-th vector of partial derivatives as

$$\nabla_i f(x) = U_i^T \nabla f(x)$$

Then, assuming that
Assumption 1

- f has Lipschitz continuous gradient, with constant L;
- $f(., \boldsymbol{x}_{-i})$ has Lipschitz continuous gradient with constant L_i, that is

$$||\nabla f(x + U_i h_i) - \nabla f(x)|| \leq L_i ||h_i||, \quad \text{for all } h_i \in \mathbb{R}^{n_i} \text{ and } x \in \mathbb{R}^n$$

and denoting with $L_{max} = \max_i L_i$ and $L_{min} = \min_i L_i$.
we finally report the general scheme of a block coordinate gradient descent method.

Algorithm 3 Block coordinate gradient descent (BCGD) method

1: Choose a point $x_1 \in \mathbb{R}^n$
2: **for** $k = 1, \dots$ **do**
3: **if** x_k satisfies some specific condition **then**
4: **stop**
5: **end if**
6: Set $y_0 = x_k$, pick a set of blocks $S \subseteq \{1, \dots, b\}$, and set $l = |S|$
7: **for** $i = 1, \dots, l$ **do**
8: Select $j_i \in S$ and set

$$y_i = y_{i-1} - \alpha_i U_{j_i} \nabla_{j_i} f(y_{i-1})$$

9: **with** $\alpha_i > 0$ calculated using a suitable line search
10: **end for**
11: Set $x_{k+1} = y_l$
12: **end for**

As we can see, at eachiteration we pick some blocks accordinag to a given rule and run an update in a (possibly) sequential fashion using gradient information. We finally build up the new point x_{k+1} before ending the iteration.

4.1 BCGD schemes

There exist different strategies for selecting blocks at each iteration. Listed here are a few options:

- **Cyclic Order:** run all blocks in cyclic order, i.e. from 1 to b at each iteration;
- **Random Permutation:** run cyclic order on a set of permuted indices;
- **Almost Cyclic Order:** each block $i \in \{1, \dots, b\}$ picked at least once every $B < \infty$ succesive iterations
- **Random Sampling:** randomly select some block i to update;
- **Gauss-Southwell:** at each iteration, pick block i so that

$$i = \arg \max_{j \in \{1, \dots, \}} |\nabla_j f(x_k)|$$

Algorithm 4 Gauss–Southwell BCGD method

1: Choose a point $x_1 \in \mathbb{R}^n$
2: **for** $k = 1, \dots$ **do**
3: **if** x_k satisfies some specific condition **then**
4: **stop**
5: **end if**
6: Pick block i_k such that

$$i_k = \arg \max_{j \in \{1, \dots, b\}} \|\nabla_j f(x_k)\|.$$

7: Set

$$x_{k+1} = x_k - \frac{1}{L} U_{i_k} \nabla_{i_k} f(x_k)$$

8: **end for**

4.2 BCGD Convergence Analysis

Theorem 4 *Let $f : \mathbb{R}^n - > \mathbb{R}$ be a function satisfying **Assumption 1**, then Gauss-Southwell BCGD method, satisfies*

$$f(x_{k+1}) - f(x^*) \leq \frac{2Lb||x_1 - x^*||^2}{k}$$

If we further have that $f : \mathbb{R}^n - > \mathbb{R}$ is a σ-strongly convex function, then Gauss-Southwell BCGD method satisfies:

$$f(x_{k+1}) - f(x^*) \leq \left(1 - \frac{\sigma}{bL}\right)^k (f(x_1) - f(x^*))$$

5 Synthetic Data Experiment

The synthetic data experiment was designed to evaluate the performance of the implemented optimization algorithms in a controlled binary classification setting. A total of 400 data points were uniformly sampled from the unit square $[0,1]^2$, among which 120 (L) were treated as labeled $\{\hat{x}^i, \hat{y}^i\}\forall i = 1,\ldots,L$ and 280 (U) as unlabeled $\{x^j\}\forall i = 1,\ldots,U$. Each labeled point was assigned a random binary label (0 or 1), forming the labeled dataset used for supervised learning. To model the relationships between points, similarity matrices were constructed based on pairwise Euclidean distances: W captures the similarities between labeled and unlabeled points, while $\hat{W}$ captures similarities among unlabeled points. Both matrices were transformed into similarity scores by taking the reciprocal of the distances (with a small ϵ added for numerical stability), such that closer points correspond to higher similarity values. This setup provides a simple yet effective synthetic environment for testing the algorithms' convergence behavior and classification performance before applying them to real-world data.

5.1 Loss Function

$$\min_{y\in\mathbb{R}^y} \sum_{i=1}^{L}\sum_{j=1}^{U} w_{ij}(y^j - \hat{y}^i)^2 + \frac{1}{2}\sum_{i=1}^{U}\sum_{j=1}^{U} \hat{w}_{ij}(y^i - y^j)^2$$

The loss function used in the synthetic data experiment quantifies how well the predicted labels of the unlabeled data points align with both the labeled data and the internal structure of the unlabeled set. It consists of two main components. The first term measures the weighted squared differences between the predicted labels of unlabeled points and the true labels of labeled points, where the weights W_{ij} encode the similarity between labeled point i and unlabeled point j. This term encourages unlabeled points to adopt labels similar to nearby labeled examples. The second term penalizes discrepancies between the predicted labels of pairs of unlabeled points, weighted by their similarity $\hat{W}_{ij}$, and scaled by a factor of $\frac{1}{2}$. This promotes smoothness among unlabeled predictions by enforcing consistency across closely related points. The overall objective thus balances label fidelity and structural smoothness, forming a typical semi-supervised learning loss that the optimization algorithms aim to minimize.

5.2 Results

Table 1 reports the comparative performance of all implemented algorithms on the synthetic dataset. All methods successfully minimized the loss function to approximately the same objective value (around 2.4792×10^4), indicating convergence to a common stationary point. Among the Gradient Descent (GD) variants, the exact line search method achieved convergence in the fewest iterations (50) and with the shortest runtime (14.47 seconds), confirming its efficiency in selecting optimal step sizes. The fixed step size variant also performed effectively, reaching a similar solution with moderate computational cost (38.25 seconds over 247 iterations). In contrast, the Lipschitz-based fixed step size method exhibited significantly slower convergence, requiring 5000 iterations and resulting in the longest runtime (778.38 seconds), likely due to an overly conservative step size choice. The GD variant employing the Armijo rule demonstrated stable convergence with a balanced trade-off between precision and efficiency.
The Block Coordinate Gradient Descent (BCGD) and Coordinate Minimization algorithms both reached the optimal objective value, with the latter being the most computationally efficient overall, converging in only 30 iterations and 5.49 seconds. BCGD, while accurate, required a considerably higher number of iterations (5000) and runtime (764.05 seconds), reflecting the increased computational overhead of block-wise updates. Overall, these results confirm that adaptive step-size strategies such as line search or coordinate-based updates can substantially enhance convergence speed without compromising accuracy.

6 Breast Cancer Dataset Experiment

The second experiment was conducted on a real-world dataset to further assess the robustness and practical effectiveness of the proposed optimization algorithms. Specifically, the Breast Cancer Wisconsin (Diagnostic) dataset [2] from the *scikit-learn* library was employed. This dataset contains 569 samples, each described by 30 continuous features that summarize characteristics of cell nuclei extracted from digitized images of breast tissue. The task is a binary classification problem, where each observation is labeled as either malignant (cancerous) or benign (non-cancerous). 20 % of the data was randomly selected and treated as labeled, while the remaining 80 % was considered unlabeled. All features were standardized to have zero mean and unit variance to ensure numerical stability and comparability across dimensions. As in the synthetic experiment, similarity matrices W and $\hat{W}$ were computed based on pairwise Euclidean distances between labeled and unlabeled points, and among unlabeled points respectively. These distances were converted into similarity scores by taking their reciprocal (with a small ϵ added for stability), thereby assigning higher weights to closer points. This experimental setup enables the evaluation of algorithmic performance in a realistic biomedical classification context, where data dimensionality and structure are considerably more complex than in the synthetic case.

6.1 Results

Overall, the experiments conducted on the Breast Cancer dataset confirm the findings observed in the synthetic case while highlighting the impact of data complexity on convergence behavior. All Gradient Descent variants successfully minimized the objective function to nearly identical values, demonstrating their robustness on real-world data. Among them, the Line Search method again achieved the best balance between precision and efficiency, converging in just 65 iterations with a runtime of 43.79 seconds. The Armijo rule also exhibited fast and stable convergence, requiring slightly more iterations but maintaining a low optimality gap. In contrast, both fixed and Lipschitz-based step size schemes required over 1400 iterations and runtimes exceeding 500 seconds, reflecting slower progress due to suboptimal step size selection. The Block Coordinate Gradient Descent (BCGD) method achieved a comparable final objective but with a significantly higher final gap and the longest runtime, suggesting limited efficiency in this high-dimensional setting. Finally, the Coordinate Minimization algorithm emerged as the most computationally efficient overall, reaching the optimal solution in only 41 iterations and under 16 seconds. These results collectively indicate that adaptive step-size strategies and coordinate-based updates offer substantial performance advantages when tackling large-scale, real-world optimization problems.

References

[1] Lecture notes of the course Optimization Methods for Data Science, Department of Mathematics, University of Padova. Prepared by Professor Rinaldi Francesco (unpublished material, 2025).

[2] Dua, D. and Graff, C. (2019). *Breast Cancer Wisconsin (Diagnostic) Data Set*. UCI Machine Learning Repository. Available at: `https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic)`

7 Appendix

Figure 1: Graphical Convergence Analysis on Synthetic Data

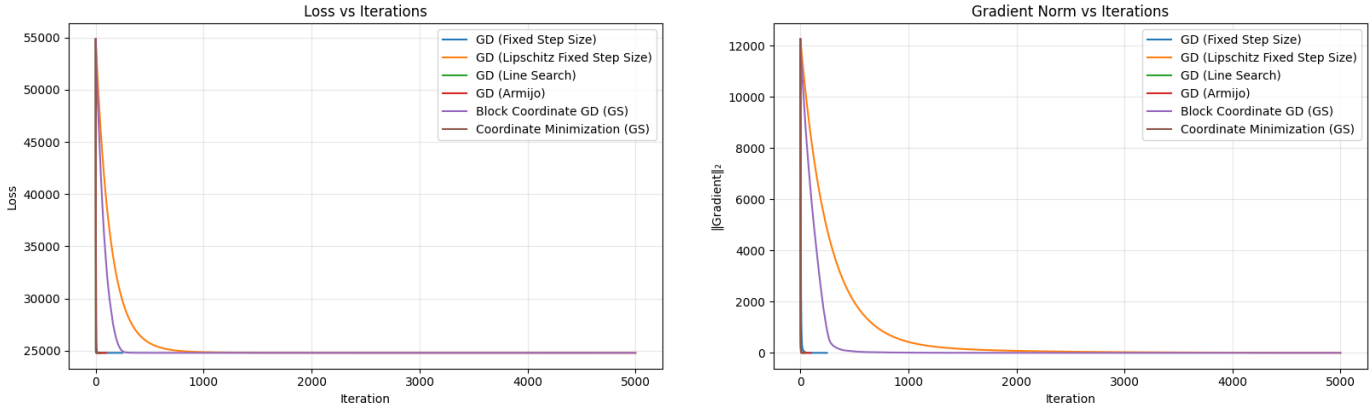

Table 1: Algorithm's Performances on Synthetic Data

	Final Objective	**Final Gap**	**Runtime (s)**	**Iterations**
GD - Fixed Step Size (0.0001)	24792.265144	0.000009	38.247224	247
GD - Lipschitz Fixed Step Size	24792.272316	3.183664	778.377508	5000
GD - Armijo Rule	24792.265144	0.000551	236.864154	97
GD - Line Search	24792.265144	0.000007	14.471834	50
BCGD	24792.265144	0.000640	764.050574	5000
Coordinate Minimization	24792.265144	0.000005	5.486262	30

Figure 2: Graphical Convergence Analysis on Synthetic Data

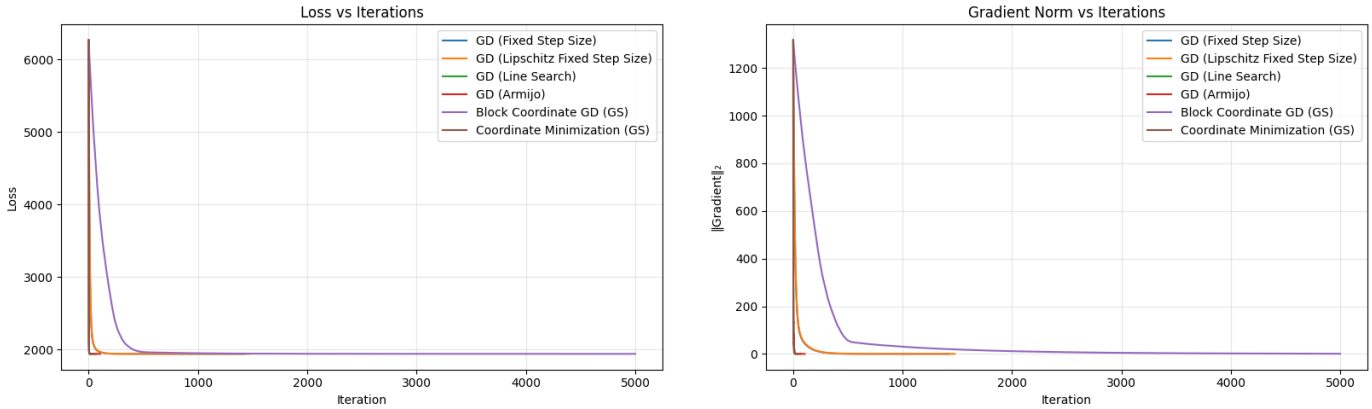

Table 2: Algorithm's Performances on Breast Cancer Dataset

	Final Objective	**Final Gap**	**Runtime (s)**	**Iterations**
GD - Fixed Step Size (0.0001)	1940.224215	0.000011	540.678052	1425
GD - Lipschitz Fixed Step Size	1940.224215	0.000013	546.359674	1477
GD - Armijo Rule	1940.224215	0.000027	364.460934	106
GD - Line Search	1940.224215	0.000009	43.785857	65
BCGD	1940.228774	0.676360	1843.232213	5000
Coordinate Minimization	1940.224215	0.000008	15.764525	41