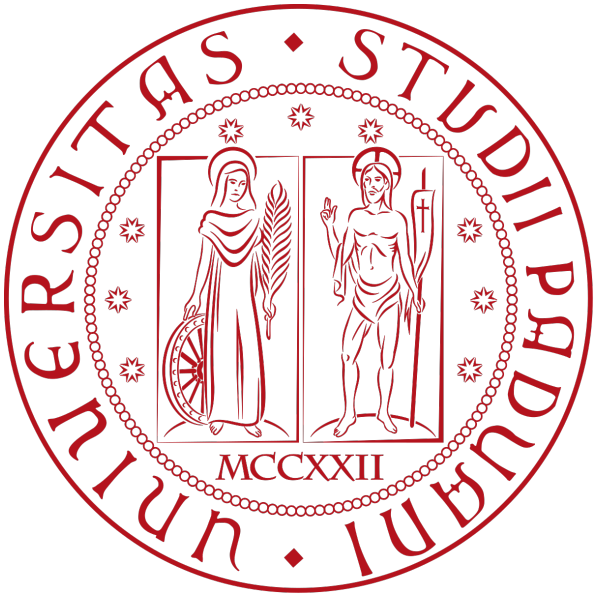


Analysis of Gradient Descent and BCGD methods

Bernal Cañez Francisco Alejandro ID: 2151939
Capasso Giovanni ID: 2096996

May 2025

Contents

1 Introduction

In this essay, we present a semi-supervised learning problem for binary classification, where only a small subset of the data is labeled, while the majority remains unlabeled. The objective is to find the missing labels by leveraging both labeled and unlabeled data. To achieve this, we define a similarity-based loss function that encourages consistent labeling among similar data points. We then minimize this loss using Gradient Based Methods and Block Coordinate Gradient Descent (BCGD) to iteratively improve the label assignments for the unlabeled data.

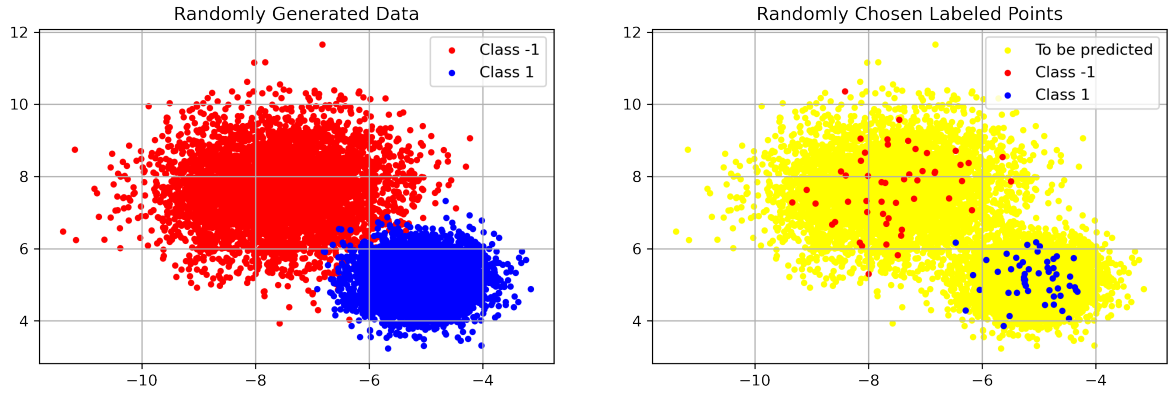
2 Dataset

2.1 Data

The analysis will be performed on the following datasets:

- A synthetic dataset, generated from two clusters in a three-dimensional space, is used to simulate a binary classification task. The two classes are constructed as Gaussian clusters, each with a randomly chosen center and an independently assigned standard deviation. The total number of samples is 10,000, with class labels set to -1 and +1. To mimic a semi-supervised learning setting, only 1% of the data from each class is randomly selected and retained as labeled. The remaining 99% of the dataset is treated as unlabeled, with the corresponding true labels retained exclusively for evaluation purposes.

- The Breast Cancer Wisconsin (Diagnostic) Dataset is designed to support the classification of tumors as either malignant or benign. Each feature represents a characteristic of the cell nuclei observed in fine needle aspirate (FNA) images. The dataset includes 30 numerical features and 569 instances. The dataset shows class imbalance, with 357 samples labeled as benign and 212 as malignant.
  The dataset is publicly available on Kaggle at the following link:
  https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data



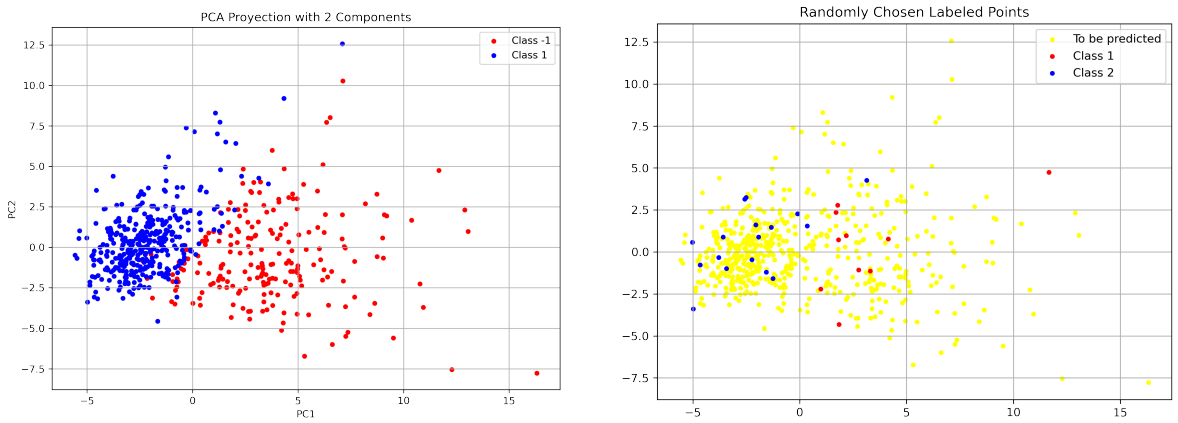
(a) Original Gaussian clusters

(b) Semi-supervised data setup

Figure 1: Visualization of the synthetic dataset. The first plot shows the ground-truth clusters, while the second distinguishes labeled and unlabeled data. Unlabeled points are shown in yellow; labeled samples are colored red and blue.

2.2 Preprocessing

As will be discussed later, the loss function depends on a set of weights that represent similarity between data points, typically based on their pairwise euclidean distances. Because of this, the model is highly sensitive to the curse of dimensionality, particularly, when dealing with high-dimensional data and a limited number of samples. To mitigate this issue, dimensionality reduction is applied using Principal Component Analysis (PCA). In particular, PCA is used to reduce the dimensionality of the breast cancer dataset, while it is not applied to the synthetic dataset, which is already low-dimensional by construction. This technique reduces the original 30 features to just 2 principal components, while still capturing the majority of the variability of the data. Since PCA is applied, the original features have different ranges, and a distance-based similarity function is used, it is fundamental to standardize the data to ensure that all variables contribute equally to the analysis.
For the purposes of our semi-supervised learning setup, only 5% of the data is used as labeled examples, while the remaining 95% is treated as unlabeled.



(a) (b)

Figure 2: Left: PCA projection of the dataset with true class labels. Right: Semi-supervised setup showing randomly selected labeled samples (red and blue) and unlabeled points (yellow).

3 Objective Function Formulation

3.1 Loss Function

The loss function used in this work is defined as follows:

$$f(y) = \sum_{i \in \mathcal{L}} \sum_{j \in \mathcal{U}} w_{ij}(y_j - \bar{y}_i)^2 + \frac{1}{2} \sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{U}} \bar{w}_{ij}(y_j - y_i)^2 \tag{1}$$

Where:

- y_j is the current label assigned to the j-th unlabeled sample.
- $\bar{y}_i$ is the known label of the i-th labeled sample.
- w_{ij} is the weight measuring similarity between labeled sample i and unlabeled sample j.
- $\bar{w}_{ij}$ is the weight measuring similarity between two unlabeled samples i and j.

The meaning of this loss function is governed by the weights. These are based on the similarity between data points in feature space. The closer they are, the larger is the weight. As a result, the optimization process encourages label values to be similar between samples that are close to each other. Specifically, it pushes terms like $(y_i - y_j)^2$ toward zero when the corresponding weight w_{ij} or $\bar{w}_{ij}$ is large, effectively promoting local label consistency among similar samples, with closer samples receiving higher weights.

3.2 Weights

The weights encode the degree of similarity between data points and determine how much influence one point should exert on another during the learning process.The higher the weight, the more similar the two points are expected to be in the feature space. To compute these similarities, a Gaussian kernel is employed, defined as:

$$\text{sim}(\vec{a}, \vec{b}) = \exp\left(-\frac{\|\vec{a} - \vec{b}\|^2}{2\sigma^2}\right) \tag{2}$$

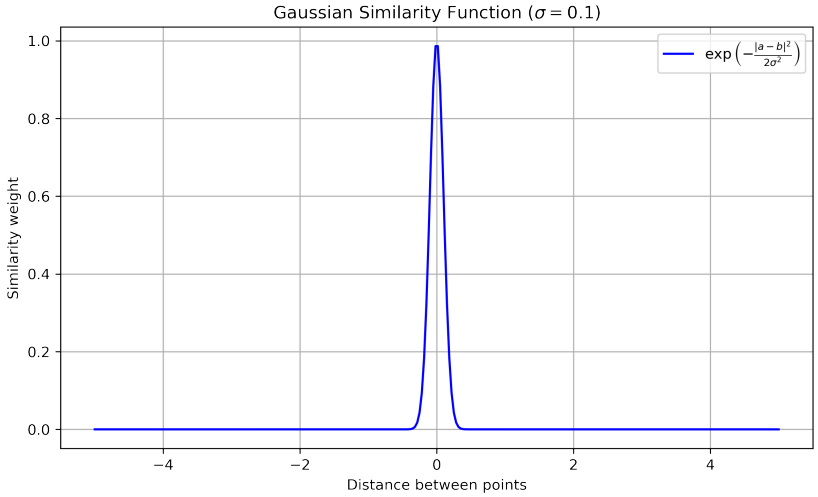


Figure 3: Similarity function using a Gaussian kernel with $\sigma = 0.1$.

3.3 Gradient of the Loss Function

The gradient of the loss function with respect to a generic unlabeled variable y_j is given by:

$$\nabla_{y_j} f(y) = 2\left[\sum_{i \in \mathcal{L}} w_{ij}(y_j - \bar{y}_i) + \sum_{i \in \mathcal{U}} \bar{w}_{ij}(y_j - y_i)\right] \tag{3}$$

3.4 Hessian Matrix

Computing the Hessian matrix of the loss function is easy in this case and it really pays off, as we can get the best of the following methods by establishing parameters in terms of L and σ, which correspond to the largest and smallest eigenvalues of the Hessian, respectively. The Hessian matrix H is of the form:

$$H_{ij} = \begin{cases} 2\Big(\sum_{i\in\mathcal{L}} w_{ij} + \sum_{j\in\mathcal{U}} \bar{w}_{ij}\Big) & \text{if} \quad i = j, \\ -2\bar{w}_{ij} & \text{if} \quad i \neq j \end{cases}$$

Since we have labeled samples and treat them as fixed during optimization, H becomes positive definite, ensuring the loss function is strongly convex. Optimization can benefit from customized step sizes and momentum coefficients depending on the condition number $\kappa = \frac{L}{\sigma}$.

4 Algorithms

The following methods will be explored in the analysis:

- Gradient Descent;
- Gradient Descent with improved rate;
- Heavy Ball;
- Accelerated Gradient;
- BCGM Gauss-Southwell.

To obtain a more comprehensive perspective, we considered all the gradient based methods covered in class, as they are straightforward to implement once gradient descent is in place. Each method was run for up to 100 iterations. For BCGD, one iteration is defined as a full pass over all unlabeled variables, i.e., |unlabeled points| individual coordinate updates. An early stopping criterion is applied when the norm of the full gradient falls below 10^{-6}, that is:

$$\|\nabla f(x)\| < 10^{-6}$$

4.1 Gradient Descent

Gradient descent has been implemented in its most basic form to serve as a benchmark for all subsequent algorithms

$$y_k = y_{k-1} - \frac{1}{L}\nabla f(y_{k-1}) \tag{4}$$

4.2 Gradient Descent with improved rate

In the case of strong convexity, convergence is guaranteed when using a step size

$$\alpha_k = \frac{2}{\sigma + L}$$

instead of the standard

$$\alpha_k = \frac{1}{L}$$

which, depending on the value of σ may be up to twice as large as the previous step size.
The corresponding gradient descent update becomes:

$$y_k = y_{k-1} - \frac{2}{\sigma + L}\nabla f(y_{k-1}) \tag{5}$$

4.3 Heavy Ball

The Heavy Ball method improves upon standard gradient descent by introducing a momentum term that leverages information from previous iterations. This addition is specifically designed to address issues of slow convergence and oscillatory behavior, which often occur in narrow or curved regions of the objective function. By incorporating the difference between the last two iterates, the method advances in a more stable and directed manner—an approach commonly known as an extrapolation step.

where the step size and momentum coefficient are chosen as

$$\alpha_k = \frac{4}{(\sqrt{L} + \sqrt{\sigma})^2}, \qquad \beta_k = \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^2,$$

and $\kappa = \frac{L}{\sigma}$ is the condition number (as previously mentioned).

4.4 Accelerated Gradient

The Accelerated Gradient Method, proposed by Nesterov (1983), shares structural similarities with the Heavy Ball method but introduces a key difference in how the momentum is applied. Each iteration consists of two steps: first, an extrapolation step computes an intermediate point

$$\eta_k = y_k + \beta_k(y_k - y_{k-1}),$$

where β_k is chosen based on the properties of the objective function. In particular, we used

$$\beta_k = \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}$$

Then, a gradient step is performed at η_k , updating the iterate as

$$y_{k+1} = \eta_k - \alpha_k \nabla f(\eta_k) \tag{6}$$

typically with $\alpha_k = \frac{1}{L}$. This approach improves convergence rates compared to standard gradient descent, particularly in smooth convex settings.

4.5 BCGM Gauss-Southwell

In the Block Coordinate Gradient Descent (BCGD) algorithm, we update one coordinate y_j at a time, selecting the index corresponding to the largest absolute gradient component, that is:

$$j = \arg\max_i |\nabla_{y_i} f(y)|$$

The update rule is:

$$y_k^j = y_{k-1}^j - \frac{1}{L} \nabla_{y^j} f\left(y_{k-1}^j\right) \tag{7}$$

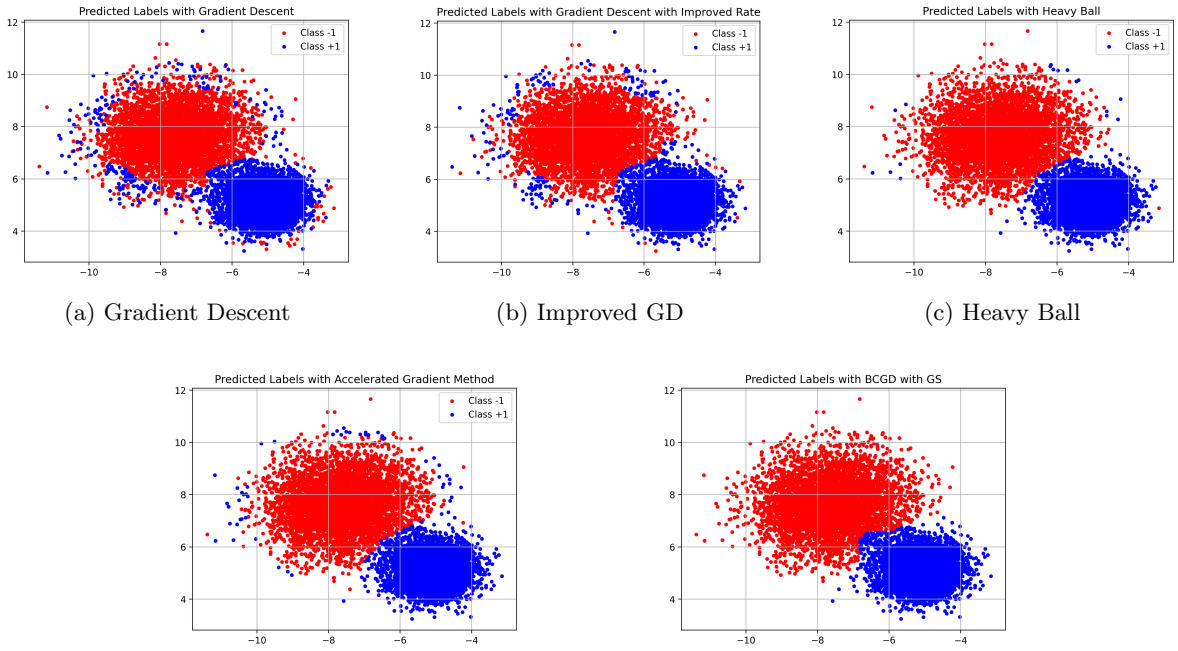
This simple yet effective strategy ensures that the algorithm greedily descends along the most influential direction at each iteration while respecting the smoothness of the objective function.

5 Results on Synthetic Dataset

In this section, we test the mentioned methods on our synthetic dataset.

5.1 Predicted Class Labels Using Various Optimization Algorithms

All methods demonstrate clear separation between the two classes.
The graphs below show the predicted class labels generated by each optimization algorithm when applied to the synthetic data set.



(a) Gradient Descent (b) Improved GD (c) Heavy Ball

(d) Accelerated Gradient (e) BCGD with GS

Figure 4: Predicted class labels on the synthetic dataset using various optimization methods. Red and blue represent the two predicted classes.

5.2 Accuracy vs. Iterations

As shown in Figure 5, all methods exhibit rapid initial improvement, with BCGD in particular reaching high accuracy earlier than the others. Gradient Descent, on the other hand, converges more slowly and achieves a slightly lower final accuracy.

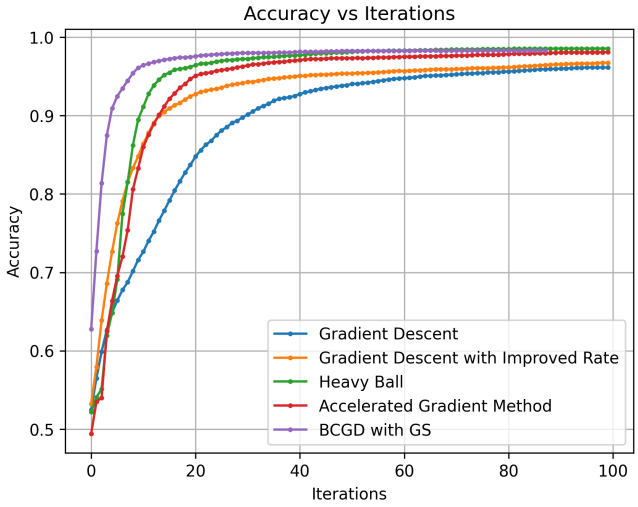


Figure 5: Evolution of classification accuracy over iterations for various optimization methods.

5.3 Accuracy vs. CPU Time

As shown in Figure 6, all methods reach high accuracy within a reasonable time frame, but BCGD with GS rule achieves this result in the shortest amount of time. Momentum-based approaches, such as Heavy Ball and Accelerated Gradient Method, also perform efficiently and exhibit similar behavior. In contrast, Gradient Descent requires significantly more time to reach comparable accuracy.

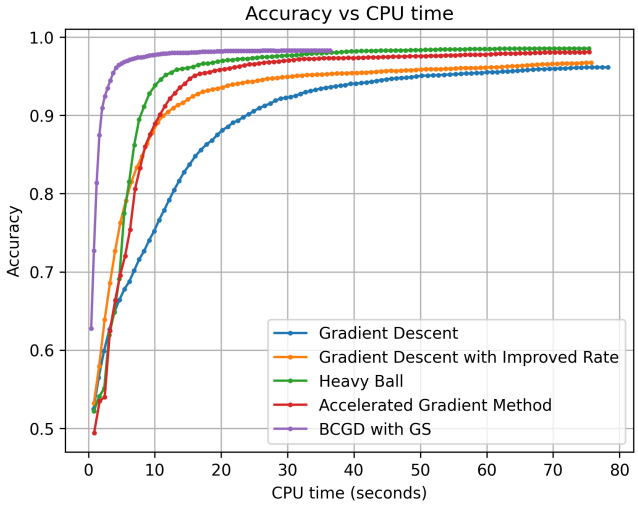


Figure 6: Accuracy achieved over CPU time for various optimization methods.

5.4 Loss vs. Iterations

As shown in Figure 7, all the methods show a rapid decrease in loss during the early iterations. Gradient Descent requires a large number of iterations, around 20, to reduce the loss significantly toward zero. BCGD with GS exhibits slower convergence but maintains consistent progress throughout.

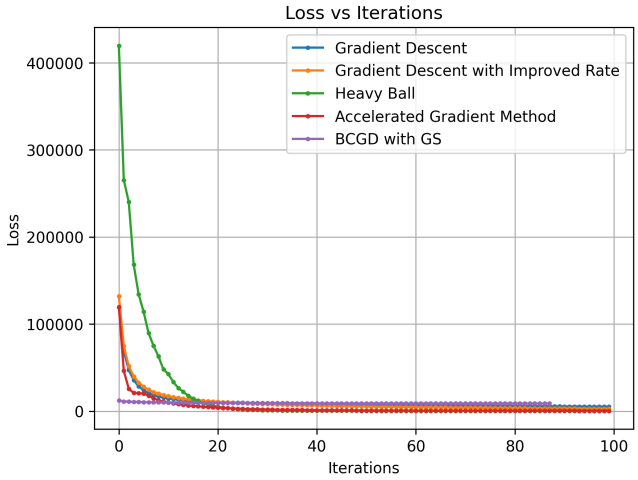


Figure 7: Loss values over 100 iterations for various optimization algorithms.

5.5 Loss vs. CPU Time

As shown in Figure 8 the Heavy Ball method stands out with the fastest initial decrease, achieving a low loss in minimal time. While Gradient Descent requires significantly more time to reach similar loss levels.

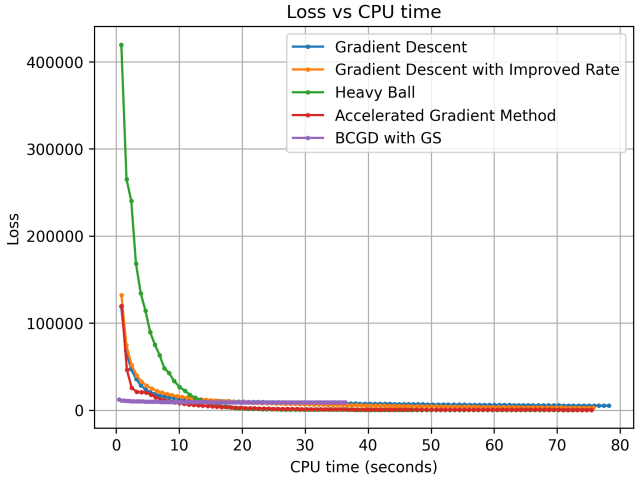


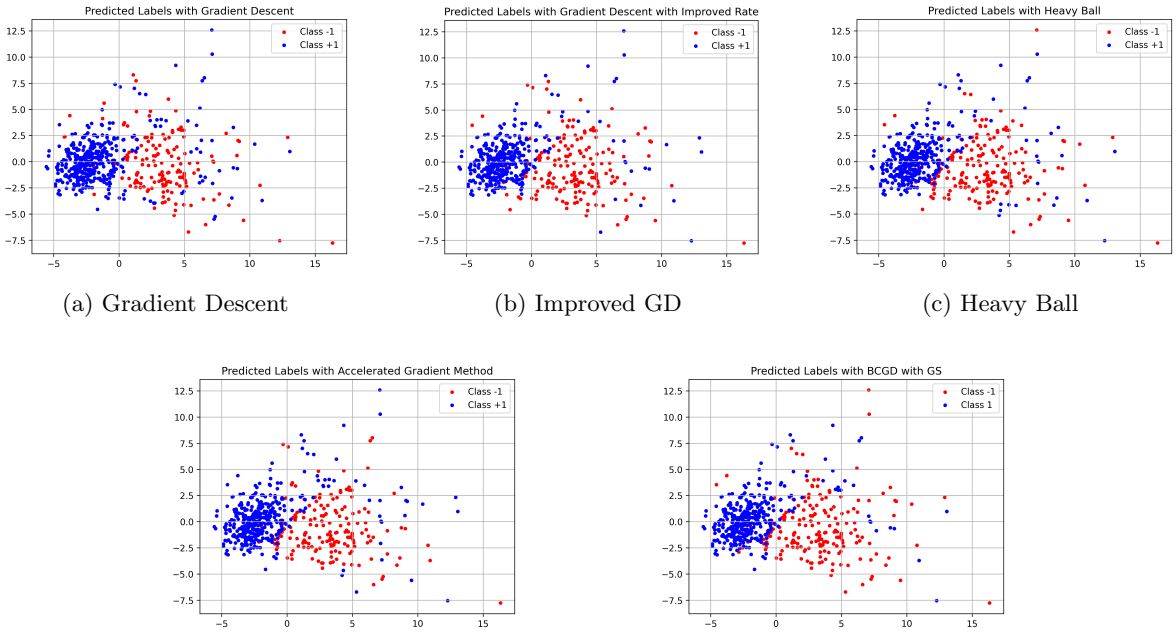
Figure 8: Loss reduction over CPU time for different optimization methods.

6 Results on Public Datasets

As previously mentioned, the model has also been evaluated on publicly available datasets. All the analyses presented in this section have been conducted using only 5% of the data as labeled, with the remaining 95% treated as unlabeled, following a semi-supervised learning setup.
The following presents the results obtained from these experiments.

6.1 Predicted Class Labels by various Optimization Methods

This subsection presents the predicted class labels for the unlabeled data obtained using each of the optimization methods. These visualizations provide a qualitative assessment of the decision boundaries formed by each method and highlight differences in classification behavior across different optimization methods.



(a) Gradient Descent (b) Improved GD (c) Heavy Ball

(d) Accelerated Gradient (e) BCGD with GS

Figure 9: Predicted class labels by each optimization method.

6.2 Accuracy vs. Iterations

As shown in Figure 10, all methods exhibit a rapid initial improvement, particularly within the first 10 iterations. Momentum-based approaches, such as the Heavy Ball and Accelerated Gradient Method, converge significantly faster than standard Gradient Descent. Notably, the Block Coordinate Gradient Descent with the GS rule achieves high accuracy early on and maintains strong performance throughout, highlighting the effectiveness of coordinate-wise updates in this setting.

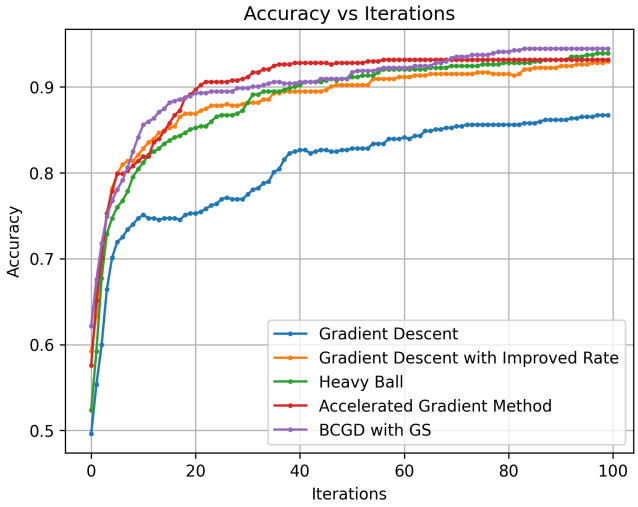


Figure 10: Evolution of classification accuracy over iterations for various optimization methods.

6.3 Accuracy vs. CPU Time

As shown in Figure 11, Momentum-based approaches, particularly the Accelerated Gradient Method and Heavy Ball, demonstrate faster convergence in terms of computation time compared to standard Gradient Descent. Although BCGD with GS rule converges more slowly in the early stages, it makes steady progress and ultimately achieves high accuracy.

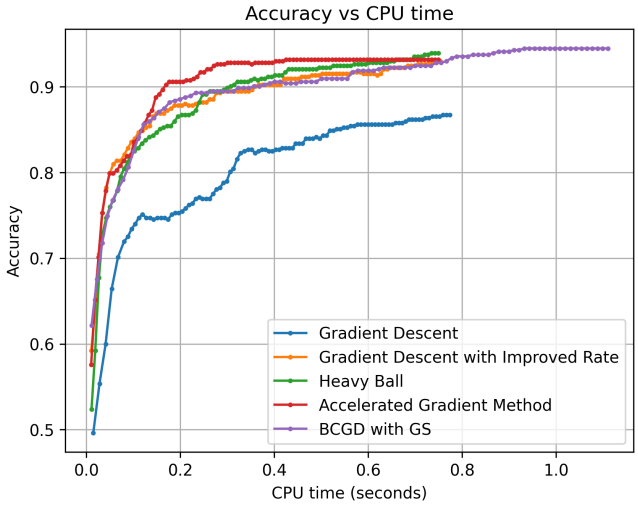


Figure 11: Accuracy achieved over CPU time for various optimization methods.

6.4 Loss vs. Iterations

A shown in Figure 12, all methods show a steep initial reduction in loss during the early iterations. Momentum-based approaches, in particular the Heavy Ball and Accelerated Gradient Method, converge more quickly and exhibit smoother trajectories.

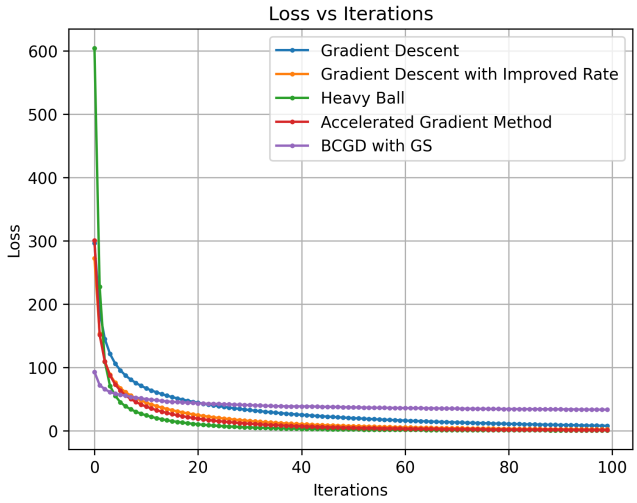


Figure 12: Loss values over 100 iterations for various optimization algorithms.

6.5 Loss vs. CPU Time

A shown in Figure 13, Momentum-based approaches, especially the Heavy Ball and Accelerated Gradient Method, achieve rapid and efficient loss reduction. In contrast, BCGD with the Gauss-Southwell rule reduces loss more slowly but maintains consistent progress. In addition, BCGD with GS rule starts with a noticeably lower initial loss compared to the other methods, which may be due to its coordinate-wise update mechanism.

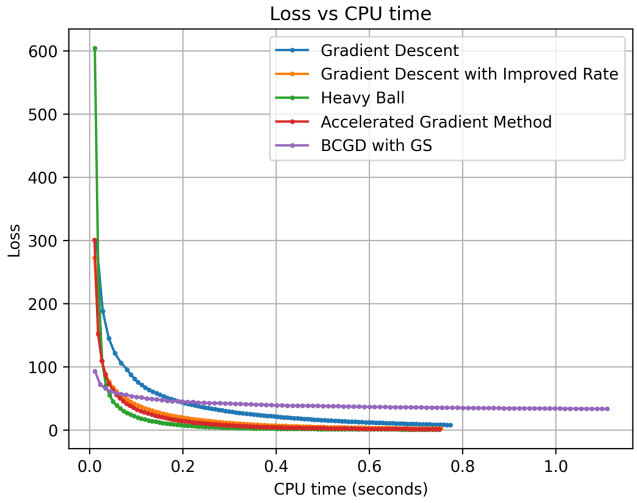


Figure 13: Loss reduction over CPU time for different optimization methods.

6.6 Accuracy with Different Percentages of Unlabeled Data

To investigate how varying the proportion of labeled data affects classification performance, we evaluated each optimization algorithm under different labeling scenarios.
As shown in Table 1, all optimization methods demonstrate a consistent improvement in classification accuracy as the percentage of labeled data increases, which aligns with expectations in semi-supervised

learning.
Among the methods, the Heavy Ball and BCGD with Gauss-Southwell rule show strong performance, particularly at higher label proportions. Notably, even with only 1.0% labeled data, the Accelerated Gradient Method and Improved Gradient Descent outperform standard Gradient Descent by a significant margin, highlighting the effectiveness of incorporating momentum and adaptive strategies.

Method	**1.0%**	**2.5%**	**5%**	**7.5%**	**10.0%**
Gradient Descent	0.589	0.817	0.867	0.886	0.903
Gradient Descent (Improved Rate)	0.853	0.862	0.929	0.915	0.936
Heavy Ball	0.840	0.944	0.939	0.920	0.942
Accelerated Gradient Method	0.874	0.908	0.931	0.938	0.930
BCGD with (Gauss-Southwell)	0.681	0.878	0.944	0.917	0.934

Table 1: Classification accuracy for different optimization methods across varying percentages of labeled data.

7 Conclusion

In this report, we evaluated the performance of various optimization algorithms on both synthetic and real-world datasets under a semi-supervised setting. The results demonstrated that momentum-based methods, such as Heavy Ball and Accelerated Gradient Method, consistently achieve faster convergence and better accuracy compared to Standard Gradient Descent. Among all the methods evaluated, BCGD with the Gauss-Southwell rule emerged as the most effective method, offering a balanced trade-off between high accuracy and low computational cost.
Moreover, the final experiments reported in Table 1 clearly showed that the proportion of labeled data has a significant impact on accuracy, with greater label availability consistently leading to improved performance across all methods.