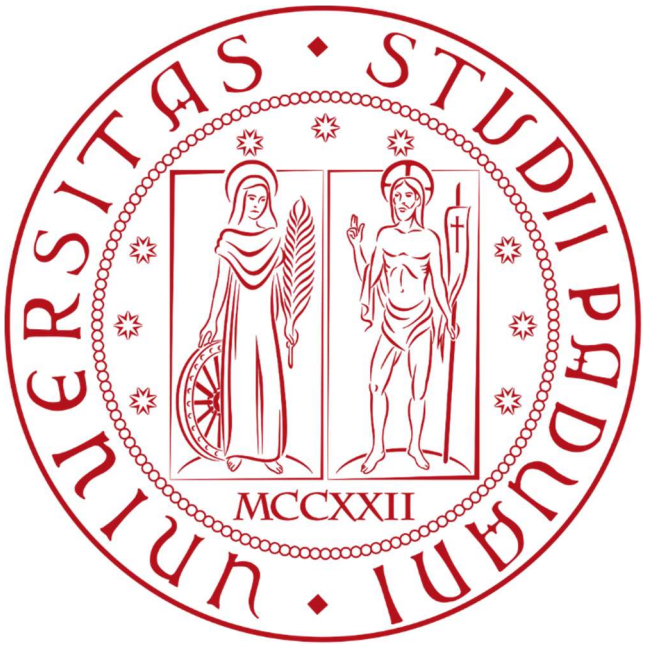

Semi-Supervised Learning using Gradient Descent Methods

Optimization for Data Science 2022/2023

Auriane Mahfouz, 2072042

Inês Caeiro Neves Silveira Jesus, 2073570

José Javier Chacón Mejía, 2071875

Table of Contents

1. Introduction

In this report, we describe how we performed semi-supervised learning tasks using gradient descent methods learnt in class to minimize the loss function. The methods that will be discussed are the classic Gradient Descent and both the Block Coordinate Gradient Descent with Randomized rule and with Gauss-Southwell rule. The goal is to compare the performances of the different methods in the optimization of our loss function. To do this, we will first study this function regarding its Lipschitz continuity of the gradient and strong convexity, which will allow us to optimize the algorithms. Then, these methods will be applied to a randomly generated two-dimensional binary dataset, a Breast Cancer Prediction dataset and a Water Quality Prediction dataset. In both cases, only a small sample of the data will be labeled. Lastly, the methods will be compared through their accuracy across the number of iterations and CPU time spent executing the code.

2. Problem Formulation and Settings

2.1. Minimization of the Loss Function

In order to perform the learning task, we need to minimize the loss function.

As such, the problem we will be solving is

$$\min_{y\in\{-1,1\}^u} f(y) = \sum_{i=1}^{l}\sum_{j=1}^{u} w_{ij}(y_j - \bar{y}_i)^2 + \sum_{i=1}^{u}\sum_{j=1}^{u} \bar{w}_{ij}(y_j - y_i)^2$$

where:

u – Number of unlabeled samples

l – Number of labeled samples

w_{ij} – Similarity between labeled sample *i* and unlabeled sample *j*

$\bar{w}_{ij}$ – Similarity between unlabeled samples *i* and *j*

$\bar{y}^i$ – Label of labeled sample *i*

y^j – Label of unlabeled sample *j*

2.2. Similarity Function

For the weights on the loss function, we need to define a similarity function that, given two points, attributes a score between 0 and 1. The closer it is to 1, the more similar these points are considered. For the purposes of this work, we chose a function of the type:

$$similarity(x, y) = \frac{1}{1 + C\|x - y\|_2^p}$$

Setting $C = 1000$ and $p = 3$, we get our similarity function.

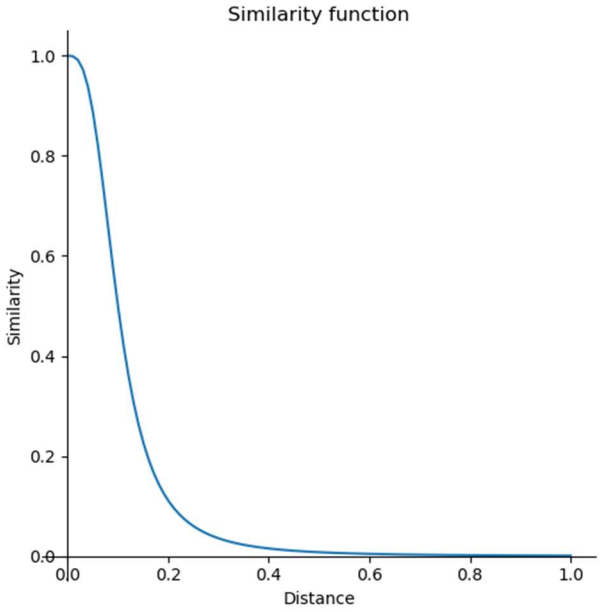

Figure 1 - Effect of the distance between two points on their similarity score

Looking at Figure 1, it is possible to observe that the closer the distance is to 0, the more significant is the similarity increase. Consequently, for distances greater than 0.2, we get a similarity that is practically null.

2.3. Accuracy

As with any learning task, it is useful to define a proper accuracy function that will express the percentage of data that is correctly classified and, thus, allow for the evaluation of the model performance. The accuracy score is given by:

$$accuracy = \frac{\#\ of\ correct\ predictions}{\#\ of\ total\ predictions}$$

In this case, a prediction of an iterate value is given by the sign function of this value.

3. Methods Definition

The three methods used on this project are the Gradient Descent, the Block Coordinate Gradient Descent (BCGD) with Gauss-Southwell rule and the Randomized BCGD. For each, we will use the contents studied in class to improve their convergence rates.

3.1.Line search

We want to determine if the fixed stepsize approach is suitable for this problem. Under certain conditions, such as Lipschitz continuity of the gradient and strong convexity of the function, it is possible to obtain an optimal fixed stepsize. To find out if this is the case, the Hessian matrix of the loss function is calculated.

3.1.1. Hessian matrix

To compute the Hessian matrix of the loss function, we first need the gradient of *f*.

$$\nabla_{y^j} f(y) = 2\left(\sum_{i=1}^{l} w_{ij}\left(y^j - \bar{y}^i\right) + \sum_{i=1}^{u} \overline{w}_{ij}\left(y^j - y^i\right)\right)$$

Now, we distinguish between two cases: the diagonal and the non-diagonal elements of the Hessian Matrix. For the diagonal, we calculate:

$$h_{jj} = \nabla_{y^j}\nabla_{y^j} f(y) = \nabla_{y^j} 2\left(\sum_{i=1}^{l} w_{ij}\left(y^j - \bar{y}^i\right) + \sum_{i=1}^{u} \overline{w}_{ij}\left(y^j - y^i\right)\right)$$
$$= 2\left(\sum_{i=1}^{l} w_{ij} + \sum_{\substack{i=1 \\ i\neq j}}^{u} \overline{w}_{ij}\right)$$

For every other entry, since $j \neq k$, we have instead:

$$h_{jk} = \nabla_{y^k}\nabla_{y^j} f(y) = \nabla_{y^k} 2\left(\sum_{i=1}^{l} w_{ij}\left(y^j - \bar{y}^i\right) + \sum_{i=1}^{u} \overline{w}_{ij}\left(y^j - y^i\right)\right) = -2\overline{w}_{kj}$$

Following Gershgorin's disc theorem, since the Hessian is a Hermitian strictly diagonally dominant matrix with real positive diagonal entries, then it is positive definite.

3.1.2. Stepsize

To follow the fixed stepsize approach, it is important to wisely select the stepsize, because if it is too large it might never converge, overshooting the optimal solution, and if it is too small, it might take too much time to do it.

Because the Hessian will be positive definite, we get that the loss function *f* is σ-strongly convex and has Lipschitz continuous gradient with constant *L*, where:

1. *L* is the largest eigenvalue of the Hessian, and
2. σ is the smallest eigenvalue of the Hessian.

On the BCGD methods, it is convenient to also determine L_i, the Lipschitz constant with respect to each block *i*, which, since the block size used is one, is equal to the Hessian diagonal element h_{ii}.

As such, we can determine the best fixed stepsizes for the gradient descent methods.

- $\alpha_k = \frac{2}{\sigma + L}$ for the classic Gradient Descent,
- $\alpha_k = \frac{2}{\sigma + L_i}$ for the Block Coordinate Gradient Descents.

With these stepsize improvements, we expect to get a linear convergence rate.

3.2. Gradient Descent

The first step is to randomly initialize a vector of size equal to the number of unlabeled points and values 1 or -1.

Then, the process consists of getting the gradient, $\nabla f(y)$, of the current iterate vector, y_k, and performing the update

$$y_{k+1} = y_k + \frac{2}{\sigma + L} \nabla_{\mathrm{y}} f(y_k) .$$

The gradient stops after it has performed the maximum number of iterations allowed or if it has reached the stopping condition defined,

$$\sqrt{2\sigma\epsilon} \geq \left\| \nabla_{\mathrm{y}} f(y_k) \right\|_2 .$$

3.3. Block Coordinate Gradient Descents

For these methods and with a block size of 1, the core idea is to pick a coordinate *i* and perform the update on this coordinate.

$$y_{k+1} = y_k + \alpha_{\mathrm{k}} \nabla_i f(y_k).$$

The difference lies mainly in how this coordinate is picked.

Since we are optimizing one coordinate at a time, we will perform more iterations, specifically the new maximum number of iterations will be defined by the product between the previously defined maximum number of iterations and the number of coordinates (unlabeled samples).

3.3.1. Block Coordinate Gradient Descent with Randomized Rule

In this approach, the block coordinate to be updated is picked randomly through all the blocks and the gradient is the respective vector of partial derivatives of *f*, $\nabla_i f(y_k)$.

The following two strategies were used to improve the convergence rate of this algorithm:

1. Using $\alpha_k = \frac{1}{L_i}$ as a stepsize.
2. Instead of using a uniform probability function to choose the coordinate *i* in each iteration, setting this probability to $\frac{L_i}{\sum_{i=1}^{b} L_i}$, where *b* is the number of blocks (number of unlabeled samples, in our case).

3.3.2. Block Coordinate Gradient Descent with Gauss-Southwell Scheme

Opposite to the Randomized BCGD, the Gauss-Southwell technique consists of finding and picking the coordinate *i* that maximizes $\nabla_{\mathrm{i}} f(y_k)$, so the whole gradient needs to be evaluated. For this, we will also use $\alpha_k = \frac{1}{L_i}$ as the fixed stepsized.

4. Methods Application

Our goal is to perform semi supervised learning using the methods mentioned above.

The analysis will be held on three datasets with binary target; one randomly generated, and two publicly available on Kaggle (click here for Breast Cancer Prediction dataset and here for Water Quality Prediction dataset).

4.1. Generated Dataset

For this dataset, two clusters with labels -1 and 1 are randomly generated in a two-dimensional plane, each with 1000 points.

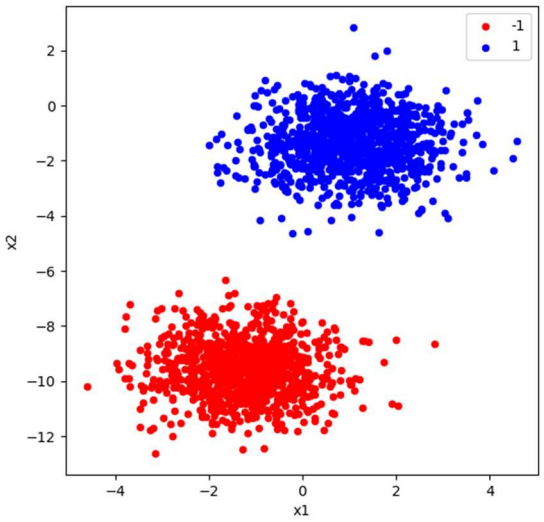

Figure 2 - Plot of the two randomly generated clusters

4.1.1. Dataset preparation

Out of the 2000 points generated, only 10% of each cluster will be selected randomly to be left labeled. The remaining points will be considered unlabeled.

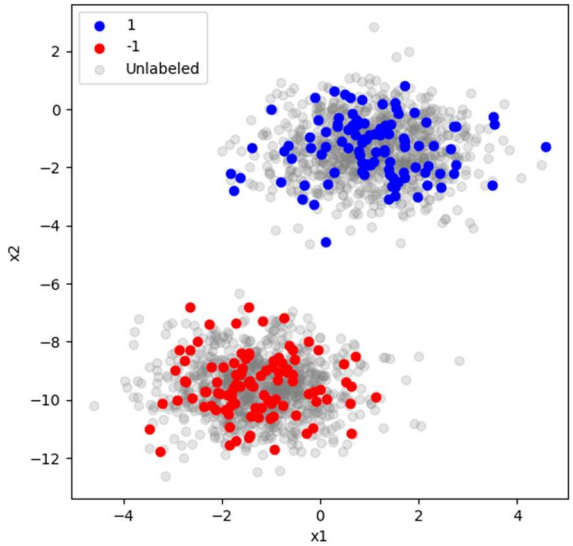

Figure 3 - Plot of labeled samples of classes 1 and -1 vs unlabeled samples

4.1.2. Gradient Descent

After performing the Gradient Descent, these were the results obtained:

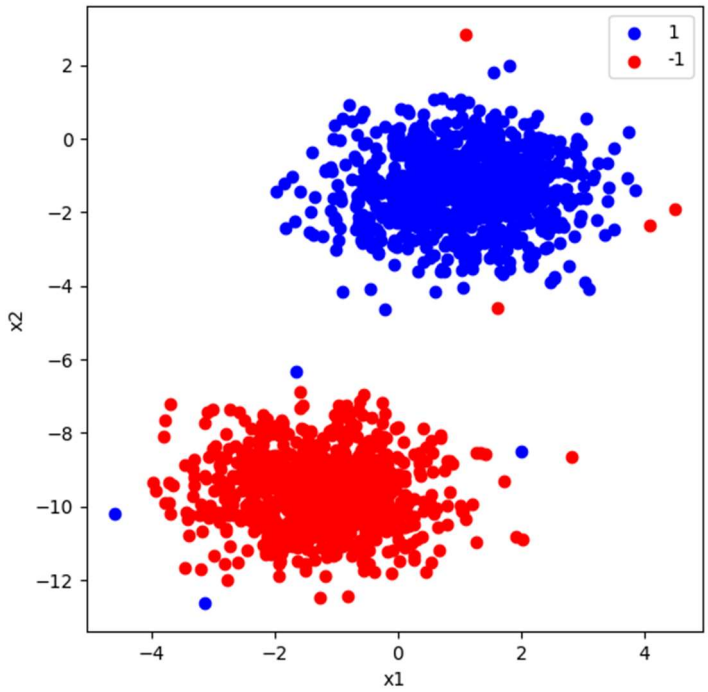

Figure 4 - Gradient Descent labeling outcome on the generated dataset

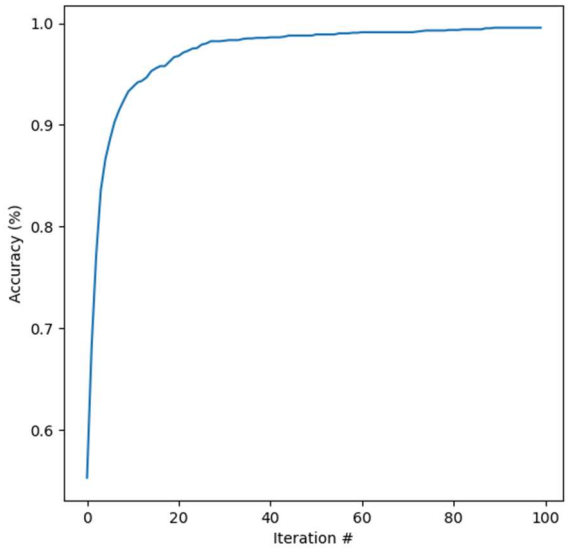

Figure 5 - Accuracy vs Iterations for Gradient Descent

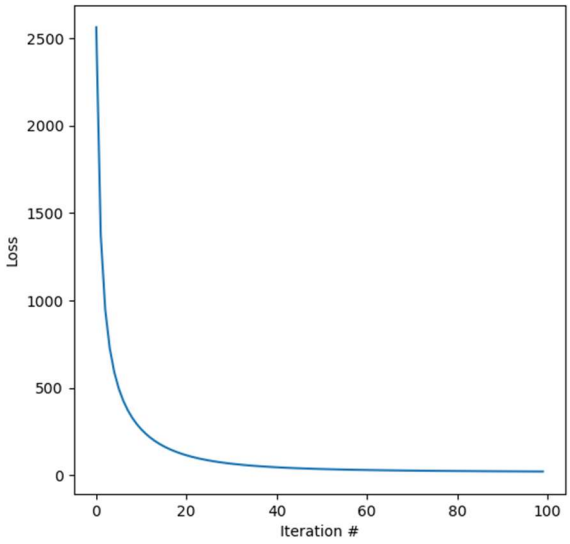

Figure 6 - Loss vs Iterations for Gradient Descent

We can see that after 40 iterations the accuracy and loss values start to stabilize. By the end of this method, we get a loss value of approximately 21,65. Only a few of the unlabeled points were misclassified and so we got almost 100% accuracy.

4.1.3. Randomized BCGD

When we preformed BCGD with Randomized Rule, we got these results instead:

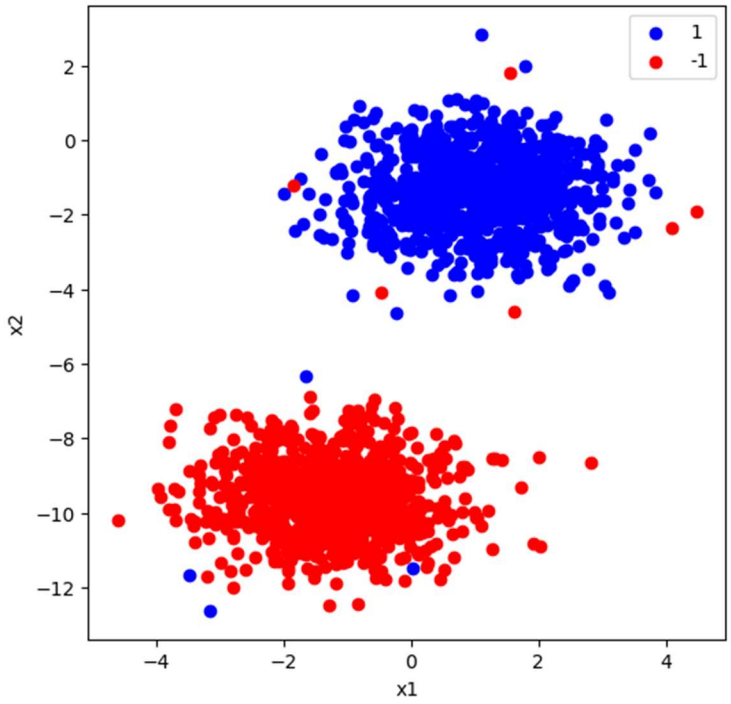

Figure 7 - Randomized BCGD labeling outcome in generated dataset

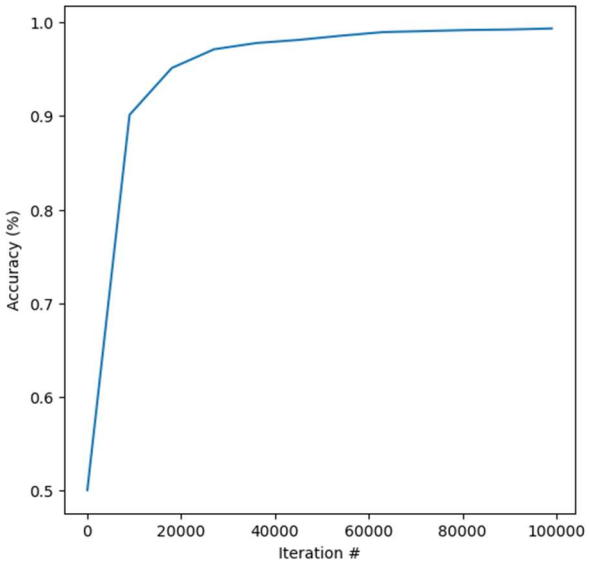

Figure 8 – Accuracy vs Iterations for Randomized BCGD

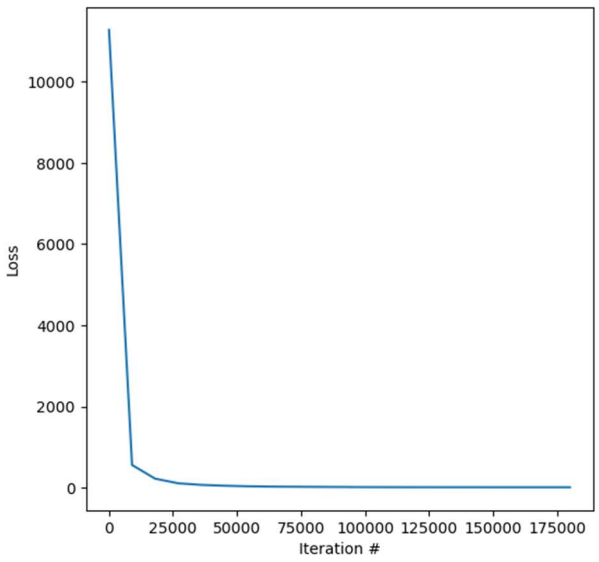

Figure 9 – Loss vs Iterations for Randomized BCGD

Here, we can see the loss converged abruptly to around 12, and we got just ten points that were misclassified, so again almost 100% accuracy.

4.1.4. Gauss-Southwell BCGD

Finally, when applying the Gauss-Southwell approach, we get:

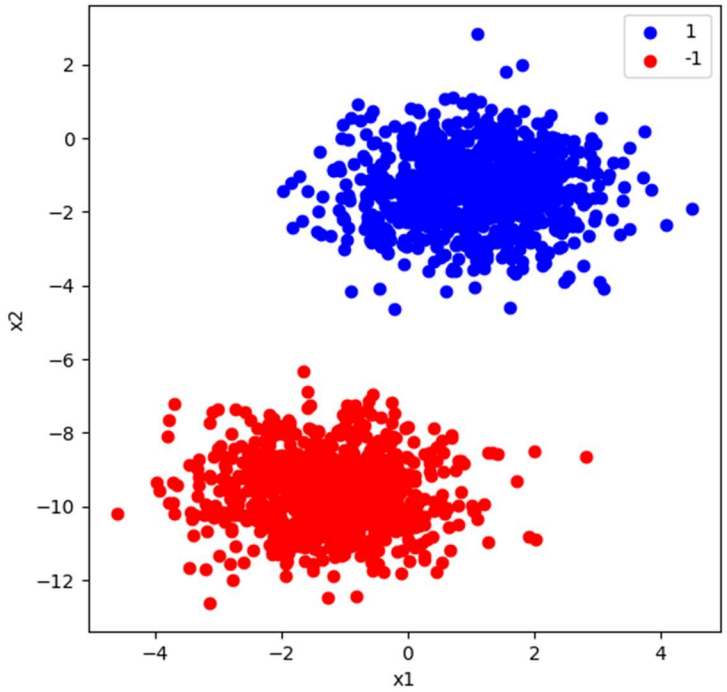

Figure 10 - Gauss-Southwell BCGD labeling outcome

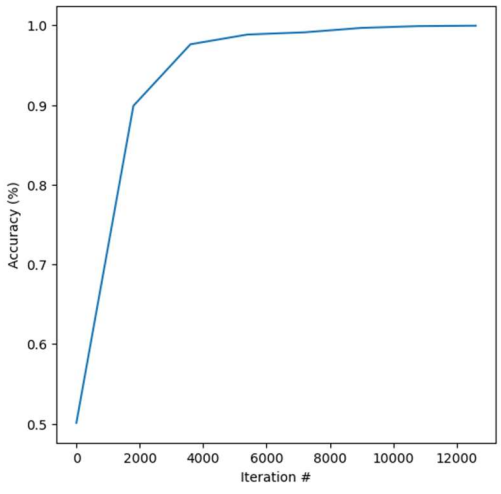

Figure 11 - Accuracy vs Iterations for Gauss-Southwell BCGD

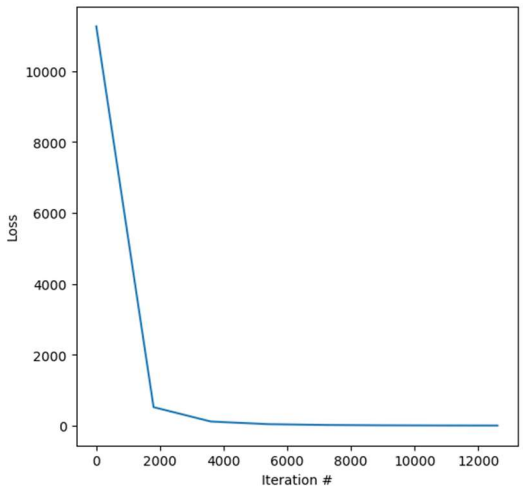

Figure 12 - Loss vs Iterations for Gauss-Southwell BCGD

This method converged to 100% accuracy rapidly, and the loss value update stopped being significant after 50 000 iterations, so the algorithm stopped, and we got a perfect classification of the unlabeled points with a loss value of 8.

4.1.5. Comparison

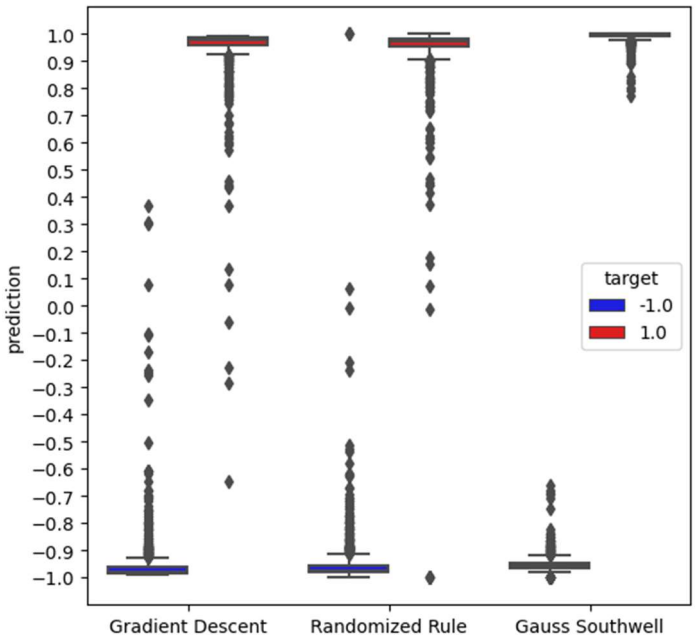

Figure 13 - Comparison of labeling outcome between the three methods

As was expected by looking at Figure 4, Figure 7 and Figure 10, we have a much more accurate prediction of labels when using the Gauss-Southwell BCGD. With this method, the predictions are very close to the respective target values and very far from 0.

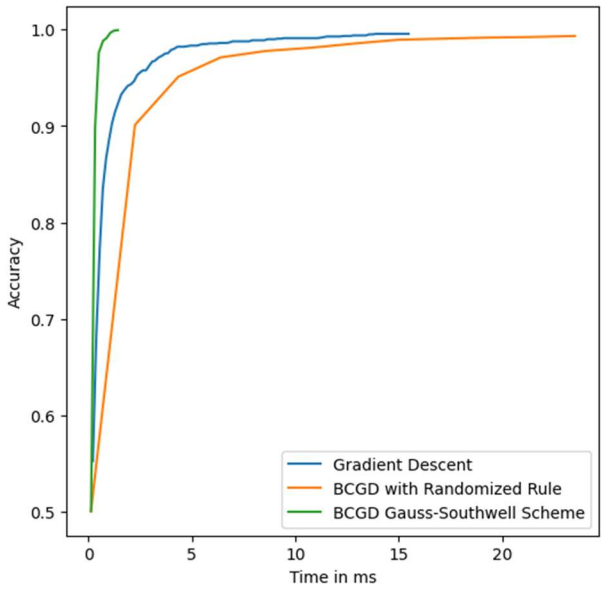

Figure 14 - Accuracy vs CPU time for the three methods

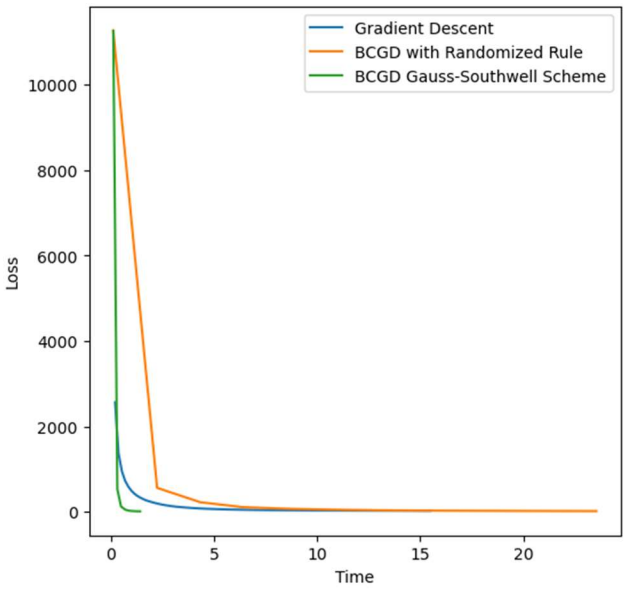

Figure 15 - Loss vs CPU time for the three methods

From these plots, the fastest method to converge and that overall presents the best results is the Block Coordinate Gradient Descent with Gauss-Southwell Scheme.

After that, the Classic Gradient Descent with improved stepsize performed better, with respect to CPU time, than the Randomized Block Coordinate Gradient Descent, which took the longest to converge to a solution; however, in the end, Randomized BCGD got a better result than the classic Gradient Descent.

4.2. Breast Cancer Prediction Dataset

The second dataset used to assess the performance of our three methods is a breast cancer prediction dataset which consists of 569 samples and 30 features describing the conditions of each patient to classify tumors into malignant (cancerous) or benign (non-cancerous).

4.2.1. Dataset preparation

Since the tumor is binary classified, we assign 1 for M (malignant) and –1 for B (benign).

The dataset is loaded and checked for any potential missing values. Having found no missing values, and in order not to fall in the curse of dimensionality problem, Principal Component Analysis (PCA) is adopted to reduce the dimension of the set of variables. For that reason, the data is split into training (labeled) and testing (unlabeled) sets with a ratio of 20% and 80% respectively. They are split as such to preserve our initial hypothesis which states to find the missing labels given very few samples of labeled data. Then, considering that PCA is a variance maximizing problem, the data is standardized so features with high variance will not cause any bias towards the result. After that, we fit PCA and plot the explained variance as shown in Figure 16. We notice that the first 4 principal components explain approximately 80% of the data variability. Therefore, the standardized sets are transformed by PCA using only those components. For instance, the data dimension has been reduced from 30 to 4 features. We can now proceed with the analysis.

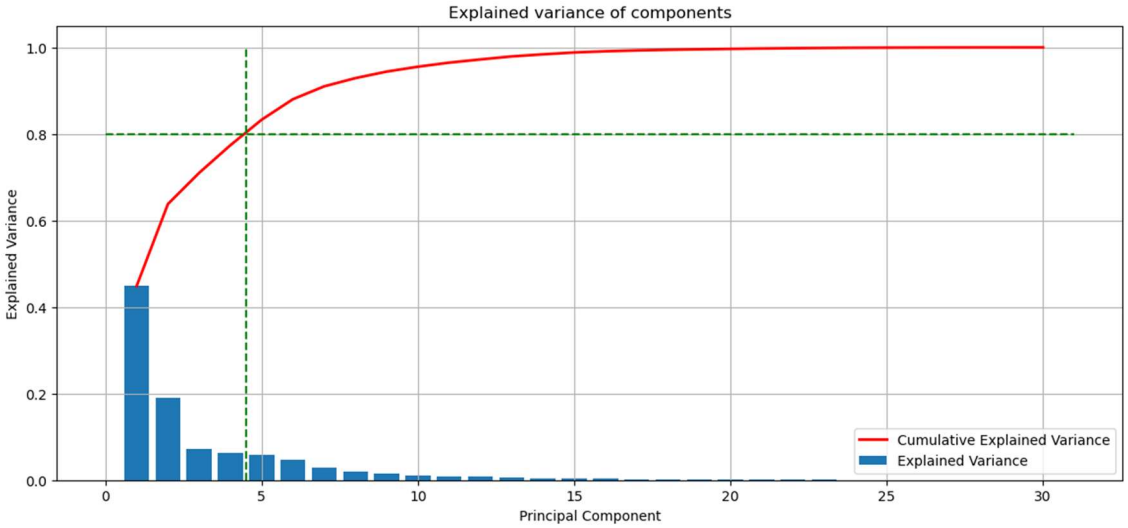

Figure 16 – Percentage of variance explained by each principal component

4.2.2. Methods comparison

Gradient Descent

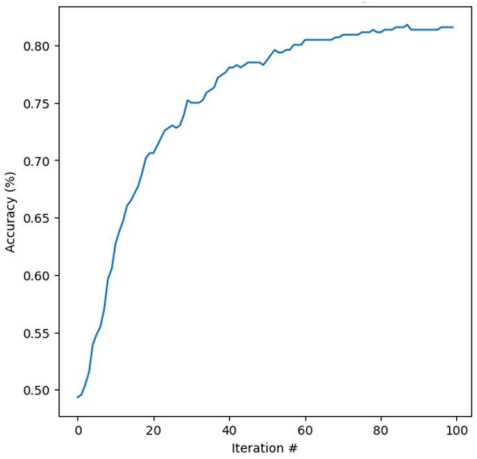

Figure 17 – Accuracy vs Iterations for Gradient Descent in Breast Cancer Prediction dataset

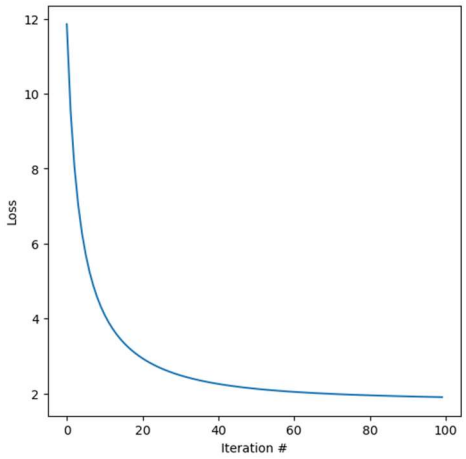

Figure 18 - Loss vs Iterations for Gradient Descent in Breast Cancer Prediction dataset

Randomized BCGD

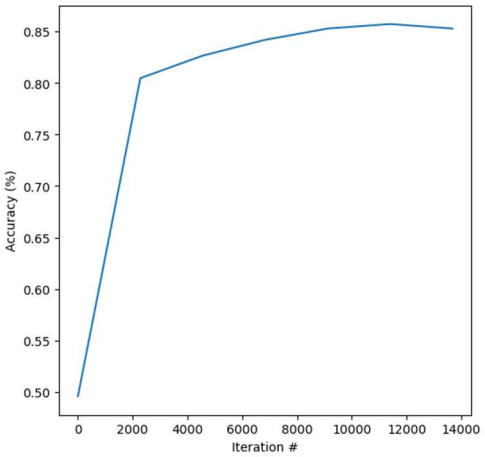

Figure 19 - Accuracy vs Iterations for Randomized BCGD

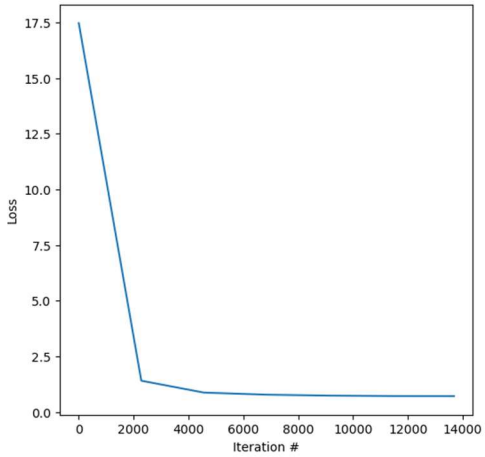

Figure 20 - Loss vs Iterations for Randomized BCGD

BCGD with Gauss-Southwell Scheme

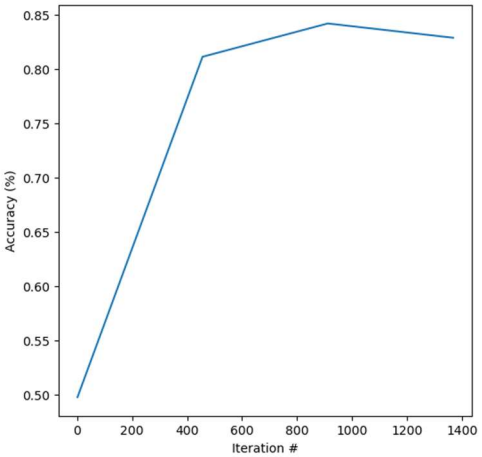

Figure 21 - Accuracy vs Iterations for GS BCGD

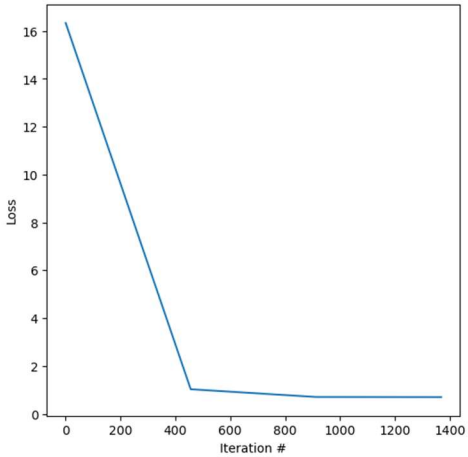

Figure 22 - Loss vs Iterations for GS BCGD

As seen in Figures 17-22, the accuracy of the three methods in the Breast Cancer Prediction dataset was lower than on the generated dataset, reaching a little over 80% accuracy. Nevertheless, the loss values reached were significantly better, converging to values close to zero. This is due to the fact that this dataset has a small number of samples (569 rows).

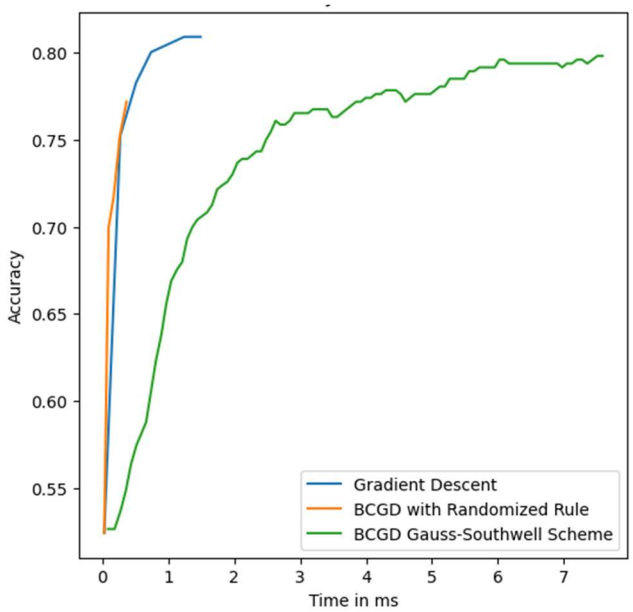

Figure 23 - Accuracy vs CPU time comparison for the three methods

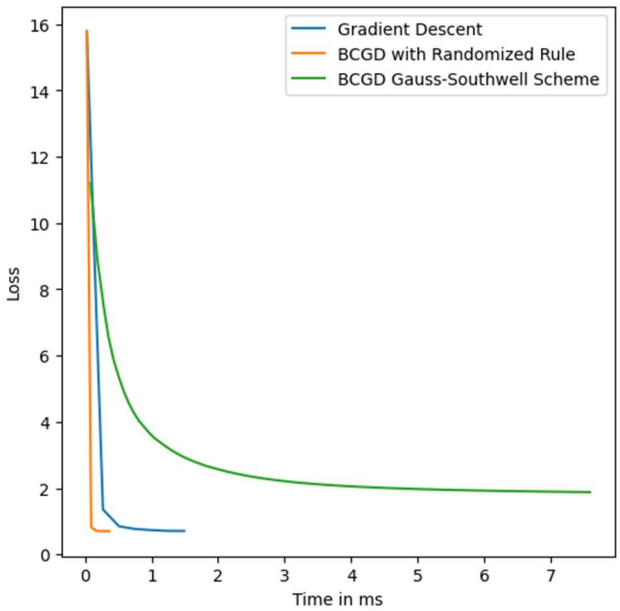

Figure 24 - Loss vs CPU time comparison for the three methods

Again, the low sample number in this dataset explains why it converges so fast and so close to zero for every method.

For this reason, we wanted to implement our algorithms on a public dataset with more samples, to observe the contrast in loss values for that case.

4.3. Water Quality Dataset

The third dataset used consists of 7999 samples and 20 features describing the conditions of water to determine if it has quality (1) or not (-1).

The urge to use this dataset as well comes from the fact that the previous dataset did not have sufficient samples to allow us to evaluate the models clearly.

4.3.1. Dataset preparation

The dataset is loaded and checked for any potential missing values. Three rows containing each two missing values were removed from the dataset leaving us with 7996 samples. We divide the dataset into 20% training and 80% testing. PCA is then fitted and plotted as displayed in Figure 25.

We can see that the variance of the principal components is roughly distributed. In this case we are going to choose 11 components that explain 80% of the data variability. Therefore, we are left with 11 attributes.

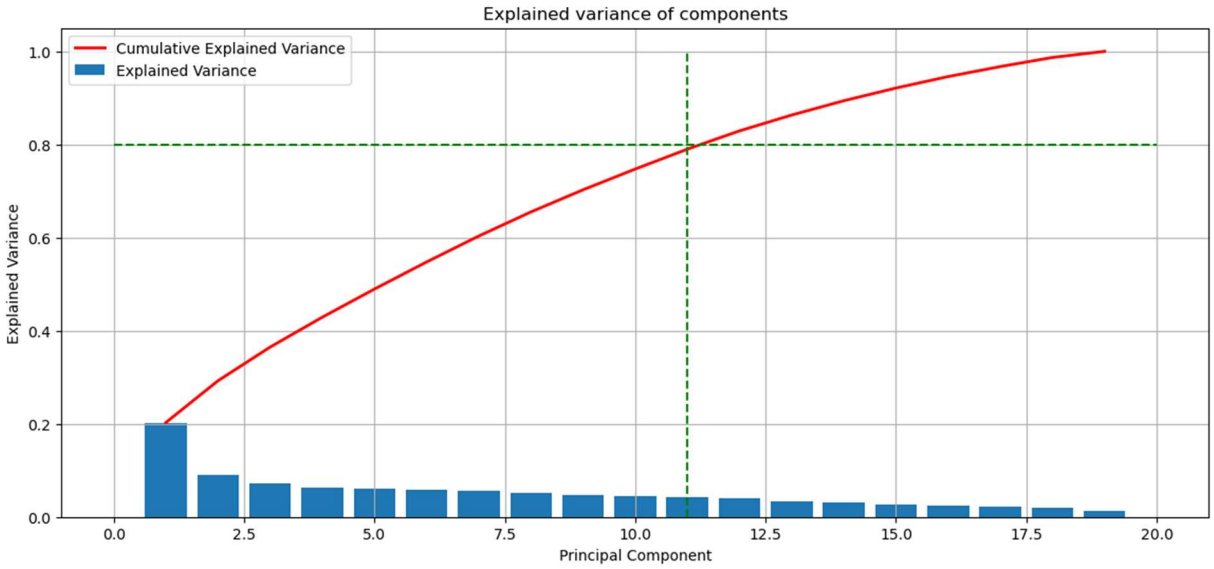

Figure 25 - Percentage of variance explained by each principal component

4.3.2. Methods comparison

Gradient Descent

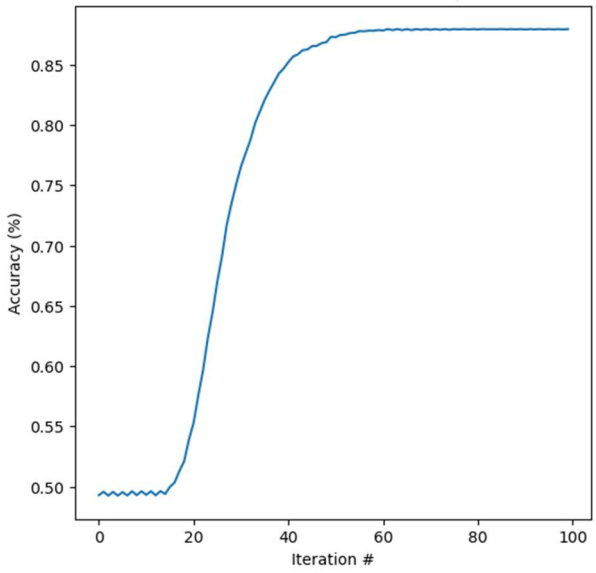

Figure 26 – Accuracy vs Iterations for Gradient Descent in Breast Cancer Prediction dataset

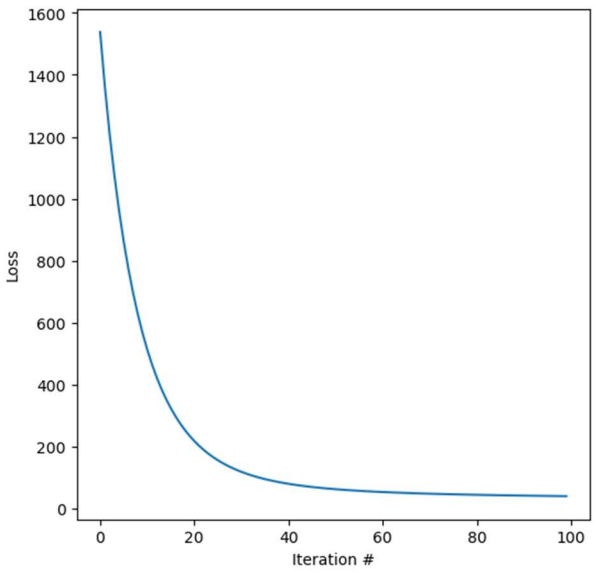

Figure 27 - Loss vs Iterations for Gradient Descent in Breast Cancer Prediction dataset

Randomized BCGD

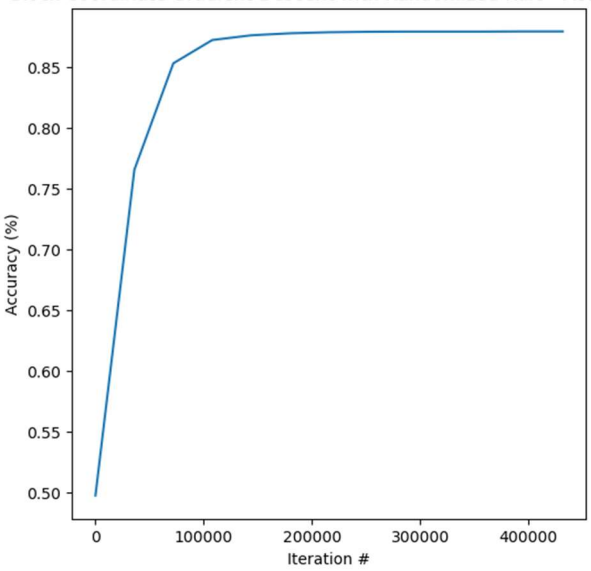

Figure 28 - Accuracy vs Iterations for Randomized BCGD

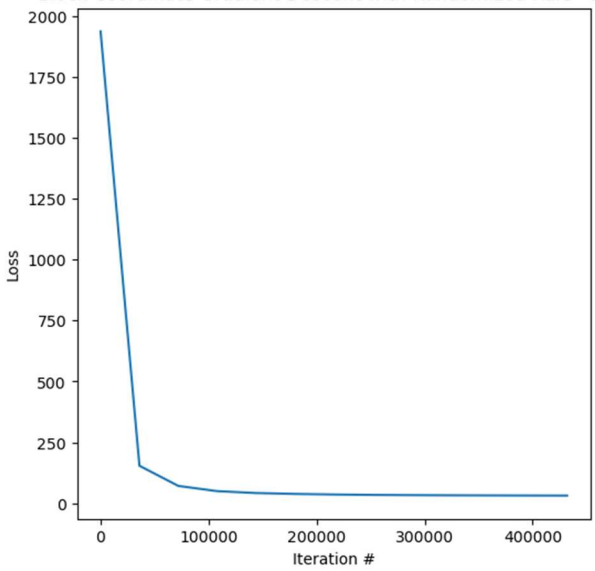

Figure 29 - Loss vs Iterations for Randomized BCGD

BCGD with Gauss-Southwell Scheme

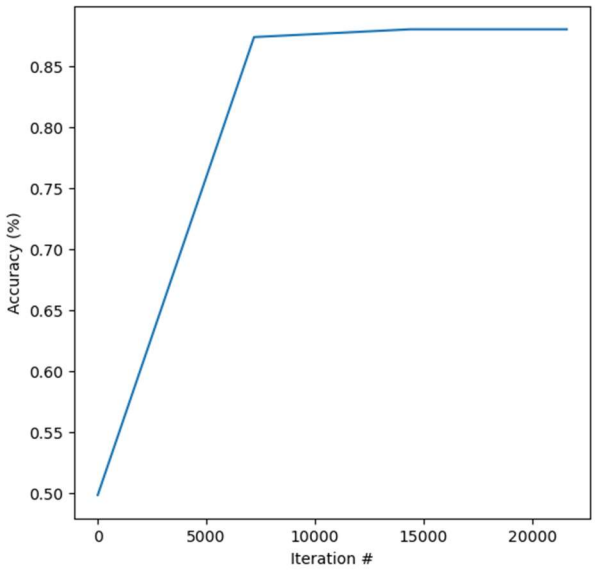

Figure 30 - Accuracy vs Iterations for GS BCGD

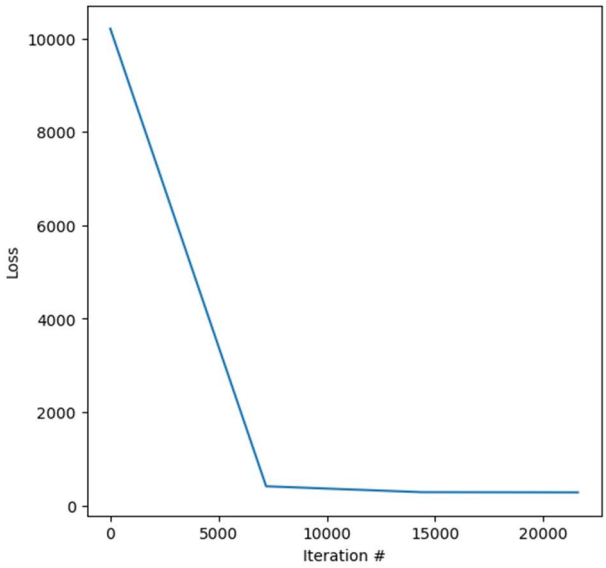

Figure 31 - Loss vs Iterations for GS BCGD

In this case, the high number of samples in this dataset causes a very high loss value for the first iteration and makes the algorithms take a longer time to finish than on the other datasets. Even so, in the end, we get 88% accuracy with a significant loss decrease for the three methods, which converge to 32 for the Block Coordinate Gradient Descent methods and 40 for the classic Gradient Descent.

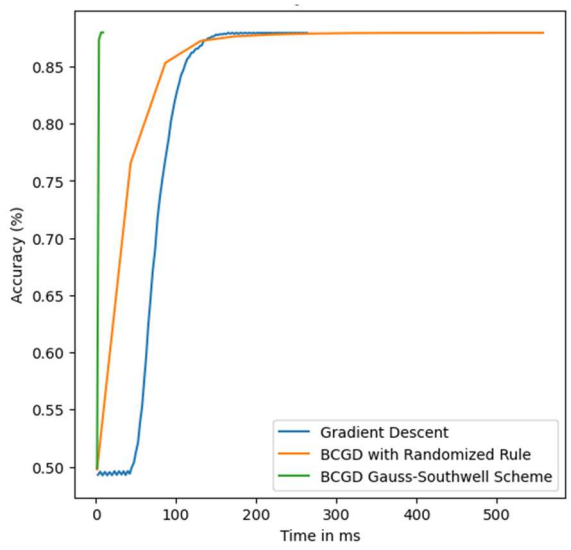

Figure 32 - Accuracy vs CPU time comparison between the three methods

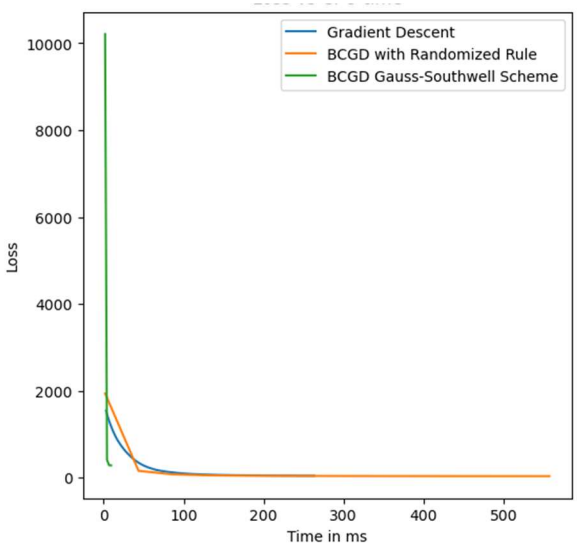

Figure 33 - Loss vs CPU time comparison between the three methods

Once again, BCGD with Gauss-Southwell approach reveals to be the method with the best performance among the three executed, converging extremely fast and to good values, reaching 88% accuracy and a loss value of 32 in just 37 seconds, before satisfying the stopping condition.

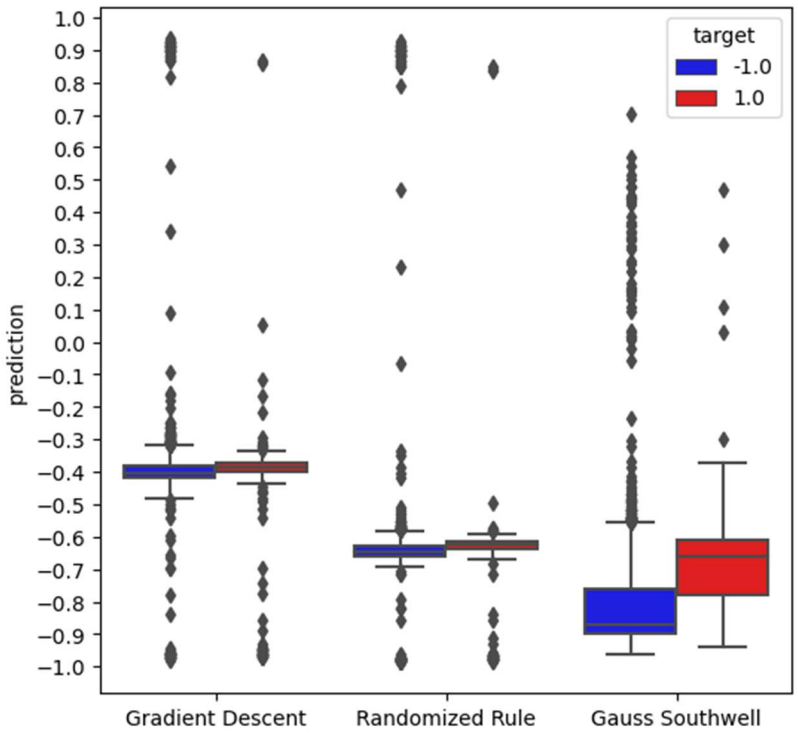

Figure 34 - Boxplot representation of the distribution of the predictions vs target on the three methods

From the boxplots it's clear that most of the samples were assigned a -1 label. This is because this dataset is unbalanced from the beginning. Oversampling should be applied to this case in order to get a balance between the two classes and thus better results. However, we can still see how the descent methods optimize the loss function with efficacy.

5. Conclusion

This project allows us to understand the differences between the variants of the classic gradient descent explored. We can see how each method adapts and behaves in different conditions, such as number of samples, number of features and unbalanced data.

Even though the difference is not very notorious because we are not working with big datasets, we can still observe the methods differ in how they converge, how long they take to do it and to which value they converge.

The classic Gradient Descent does not need many iterations, because with each iteration the update in the loss is very significant. However, it does not give us the best solution and more iterations (and consequently more time) are needed for this method if we want to achieve better results.

The Randomized BCGD, even with improvements on the stepsize and selection of coordinate, seems to take a long time to converge to a good solution. This is because only one random coordinate is updated at a time.

Even though from our results the Gauss-Southwell BCGD method always converged really quickly, it is also very computationally expensive and under other circumstances should not be the method to choose. Besides that, we can see that facing an optimization problem, there is not a method that is the best for all conditions, so the problem needs to be studied with caution and different methods need to be experimented, so that the optimal solution and best method can be selected among these.

Bibliography

Gallier, J. H. (2022). *Eigenvectors and Eigenvalues.* Retrieved from https://www.cis.upenn.edu/~cis5150/cis515-20-sl10.pdf

Rinalid, F. (2022). *Optimization for Data Science.* Retrieved from Moodle: https://stem.elearning.unipd.it/course/view.php?id=5040

Universiteit Utrecht. (n.d.). *Optimization strategies for supervised learning.* Retrieved from https://www.uu.nl/sites/default/files/presentation_slides.pdf