Comparison between gradient descent and BCGD methods

Optimization for Data Science (2022/2023)

Betti Gianmarco (ID: 2097050)
Marinelli Andrea (ID: 2091700)
Rinaldi Giorgia (ID: 2092226)

May 15, 2023

Contents

1 Introduction

The aim of the report is to implement and compare three algorithms in order to solving a semi-supervised learning binary classification problem. The used algorithms are:

1. Gradient descent;
2. Block coordinate gradient descent with randomized rule;
3. Block coordinate gradient descent with Gauss-Southwell rule.

The algorithms have been tested on a synthetically generated 2D point dataset and then on three publicly available datasets.

2 Datasets

The datasets used to test the implemented algorithms are:

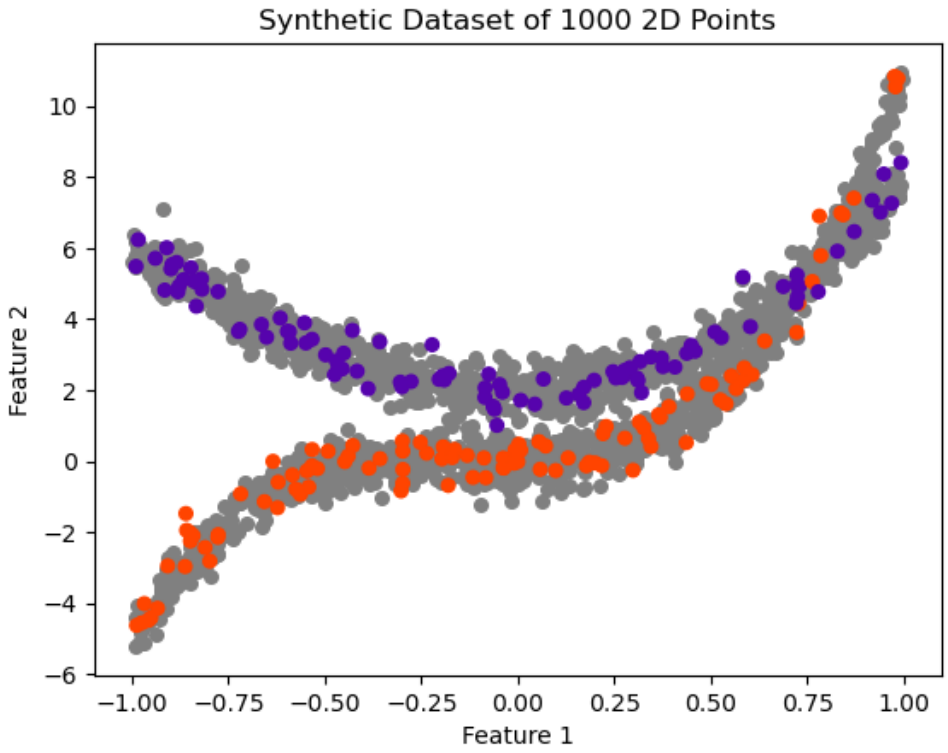
- **Breast Cancer**: this dataset contains information about the characteristics of some cells present in digital images of breast tissues. Each record represents a cell and contains data such as the size, shape, and homogeneity of the nucleus. The goal is to classify the cells as benign or malignant based on their characteristics;
- **Iris**: This dataset contains information about three species of iris (Iris setosa, Iris virginica, and Iris versicolor) and their botanical characteristics, such as the length and width of the sepals and petals. The goal is to correctly classify the species based on these characteristics; Here, only 2 species out of 3 have been considered in order to have a binary classification problem.
- **Diabetes**: This dataset contains information about some medical characteristics of patients with diabetes, such as age, BMI, blood pressure, glucose level, etc. The goal is to predict whether a patient will develop diabetes or not based on their medical characteristics.

2.1 Preprocessing

All the datasets have been loaded and preprocessed in order to simulate a semi-supervised learning problem with only two classes and where only 10% of the examples were labeled.

2.2 Points generation

The first dataset on which the algorithms were tested consists of 2000 synthetic 2D points. 1000 points were generated for each of the two clusters using the following functions:



$$x^{(i)} = \begin{cases} x_1^{(i)} & = u_i \\ x_2^{(i)} & = u_i^3 + 5u_i^2 + 2 + z_i \end{cases} \qquad \forall i \in \text{cluster}_1$$

$$x^{(j)} = \begin{cases} x_1^{(j)} & = u_j \\ x_2^{(j)} & = 8u_j^3 + 3u_j^2 + z_i \end{cases} \qquad \forall j \in \text{cluster}_2$$

Where $U \sim \text{Uniform}(-1, 1)$ and $Z \sim \mathcal{N}(0, 1)$ are random variables used respectively to generate a random value in the range (0,1) and to generate noise. Only 10% of those examples has been labeled with a value either +1 or -1, obtaining the situation shown in figure.

3 Formalization

3.1 Loss function

The given Loss function is:

$$\min_{y \in R^u} \sum_{i=1}^{l} \sum_{j=1}^{u} \omega_{ij} (y^j - \bar{y}^i)^2 + \frac{1}{2} \sum_{i=1}^{u} \sum_{j=1}^{u} \bar{\omega}_{ij} (y^j - y^i)^2$$

where:

- y^j represents the label assigned to the j-th element;

- $\bar{y}^i$: determined label ((?))
- w_{ij}: similarity measure matrix between labeled and unlabeled data;
- $\bar{w_{ij}}$: similarity measure matrix between unlabeled data;
- u : unlabeled data
- l: labeled data

Weights inversely proportional to the distance; algorithms should give label according to similarity measure in order to get y^i - $y^j = 0$

3.2 Similarity Measure function and weights

The given weight function is defined using the Euclidean distance between pairs of examples in the input space:

$$w(x, y) = e^{-10||x-y||_2^2}$$

This weight function is a similarity metric that assigns a weight to each pair of examples based on their distance in the input space. It gives higher weight to pairs of examples that are close to each other in the input space, and lower weight to pairs of examples that are far apart. By incorporating this weight function into the unsupervised loss term of the loss function, the model is encouraged to produce similar predictions for pairs of examples that are close to each other in the input space, and dissimilar predictions for pairs of examples that are far apart.

3.3 Gradient

The following equation represents the gradient of the loss function with respect to y^j:

$$\nabla_{y^j} f(y) = 2\sum_{i=0}^{l} \omega_{ij}(y^j - \bar{y}^i) + 2\sum_{i=0}^{u} \bar{\omega}_{ij}(y^j - y^i)$$

$$= 2(\sum_{i=0}^{l} \omega_{ij} + \sum_{i=0}^{u} \bar{\omega}_{ij})y^j - 2\sum_{i=0}^{l} \omega_{ij}\bar{y}^i - 2\sum_{i=0}^{u} \bar{\omega}_{ij}y^i$$

$$= 2\left[(\sum_{i=0}^{l} \omega_{ij} + \sum_{i=0}^{u} \bar{\omega}_{ij})y^j - \sum_{i=0}^{l} \omega_{ij}\bar{y}^i - \sum_{i=0}^{u} \bar{\omega}_{ij}y^i\right]$$

3.4 Accuracy

In order to assess the performance of the algorithms, an accuracy measure has been defined as follow. First of all we mapped each predicted labels $y^{(i)}$ to a value either +1 or -1, passing it through the function:

$$\text{sign}_0(y^{(i)}) = \begin{cases} +1 & \text{if } y^{(i)} > 0 \\ -1 & \text{otherwise} \end{cases}$$

Then the binary predictions have been compared to the true labels and the proportion of correct predictions has been computed. So, the accuracy of the prediction is computed using the following formula:

$$\textbf{Accuracy} \ = \sum_{i=1}^{u} \frac{(y_{\text{true}}^{(i)} - y_{\text{pred}}^{(i)})}{u}$$

3.5 Hessian Matrix

Those functions are used to calculate the Hessian matrix and the Lipschitz constat for the whole problem, and for the single variable. The Hessian is calculated as follows:
with:

$$k \neq j \rightarrow \nabla_{y^j y^k} f(j) = -2 \ \bar{\omega}_{kj}$$

$$k = j \rightarrow \nabla_{y^j y^j} f(j) = 2 \left[(\sum_{i=0}^{l} n \ \omega_{ij}) + (\sum_{i=0}^{u} \bar{\omega}_{ij}) - \bar{\omega}_{jj} \right]$$

$$H_{n\times n} = 2 \begin{bmatrix} (\sum_{i=0}^{l} \omega_{i1}) + (\sum_{i=0}^{u} \bar{\omega}_{i1}) - \bar{\omega}_{11} & -\bar{\omega}_{12} & \cdots & -\bar{\omega}_{1n} \\ -\bar{\omega}_{21} & (\sum_{i=0}^{l} \omega_{i2}) + (\sum_{i=0}^{u} \bar{\omega}_{i2}) - \bar{\omega}_{22} & \cdots & -\bar{\omega}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\bar{\omega}_{n1} & -\bar{\omega}_{n2} & \cdots & (\sum_{i=0}^{l} \omega_{in}) + (\sum_{i=0}^{u} \bar{\omega}_{in}) - \bar{\omega}_{nn} \end{bmatrix}$$

3.6 Step-size

In this study fixed step-size has been employed and its selection was based on the computation of the Lipschitz constant L, which is equivalent to the largest eigenvalue of the Hessian matrix.

4 Algorithms

To solve the problem of label assignment for all unlabeled examples, three different algorithms have been implemented, each of which is described in the following paragraphs:

4.1 Gradient Descent

The gradient descent was implemented by setting a maximum number of iterations to 60 and using the stopping rule $||\nabla f(x_k)|| \leq \varepsilon$, where ε was fixed at 10^{-5}. The starting point was randomly selected, assigning a label of +1 or -1 to the unlabeled points. The updates were calculated using the formula:

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

4.2 Randomized BCGD

In the implementation of Randomized BCGD, blocks of size one were considered, so at each iteration, only a single coordinate was updated, specifically the label of the selected example. The label was randomly chosen at each iteration using a uniform distribution. The updates were calculated as follows:

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

4.3 Gauss-Southwell BCGD

In comparison to Randomized BCGD, the Gauss-Southwell BCGD algorithm selects the coordinate by considering the block with the highest gradient. The rest of the implementation remains the same as in Randomized BCGD.

5 Comparison

After testing the three algorithms on the synthetically generated dataset, similar performance can be observed, as shown in the graphs in Figure 1 and Figure 3. However, some differences between the algorithms become apparent when they were tested on the other datasets, as it can be seen in Figure 2 and Figure 4.

5.1 Analyse of accuracy as a function of CPU time

As can be observed from the graphs below, in the case of the synthetic dataset, the accuracy, calculated based on CPU time, has very similar values for each algorithm. Conversely, in the case of the Diabetes dataset, the accuracy values vary with different CPU time values. In fact, it is easily noticeable that the "Gauss-Southwell BCGD" method reaches a high level of accuracy much faster, while the "Randomized BCGD" method requires significantly more CPU time. Finally, it is evident that the "Gradient Descent" method does not achieve the same level of accuracy as the previous methods but rather a much lower one.

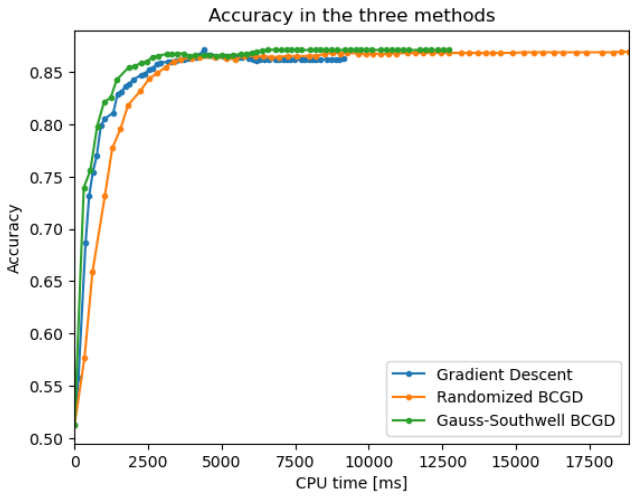


Figure 1: Synthetic dataset

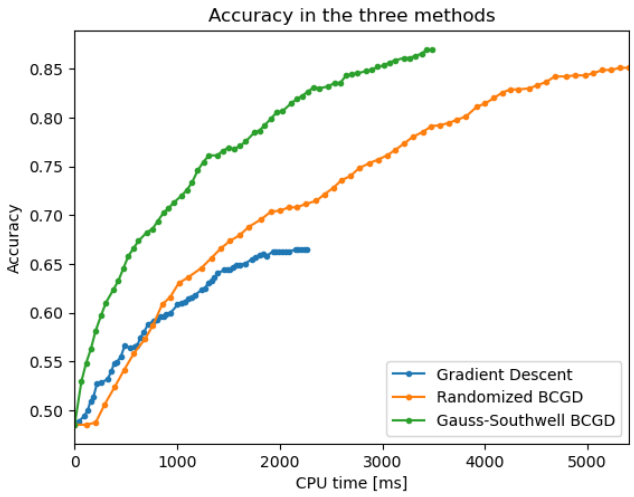


Figure 2: Diabetes dataset

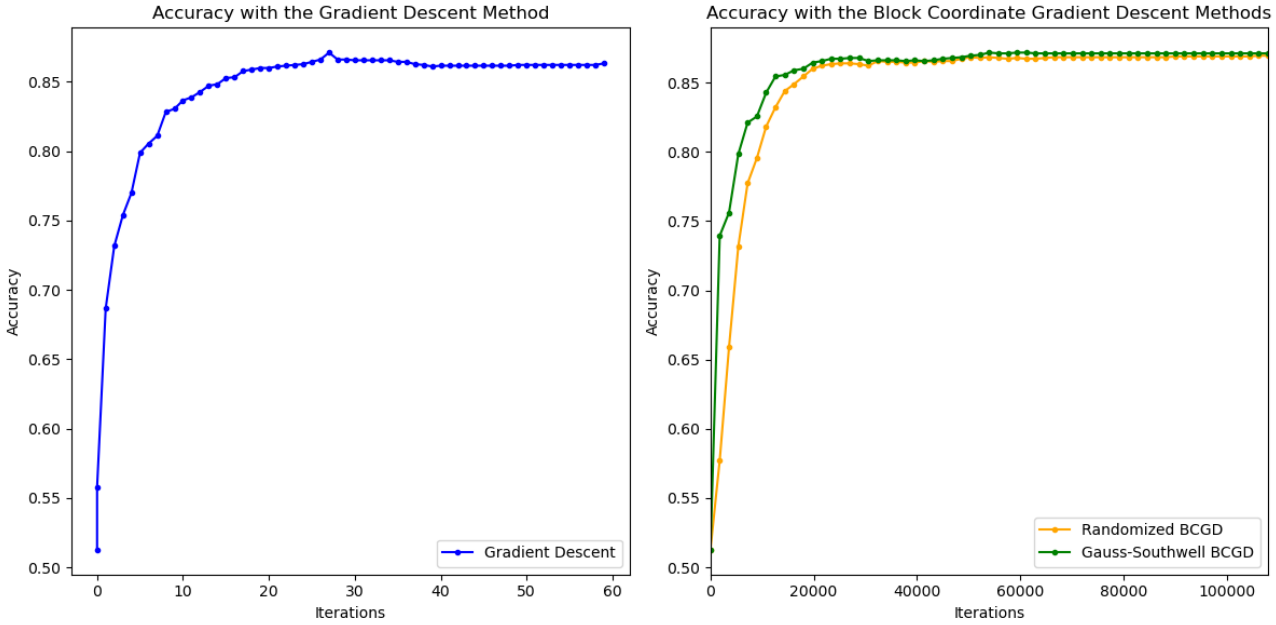


Figure 3: Synthetic dataset

5.2 Analyse of accuracy as a function of the iteration numbers

Furthermore, the accuracy value of the results can be analyzed, calculated based on the number of algorithm iterations. In both datasets, it can be observed that the accuracy values coincide for both "BCGD methods" and "Gradient Descent". The only difference lies in the fact that in "BCGD methods," a significantly larger number of iterations is required compared to "Gradient Descent."

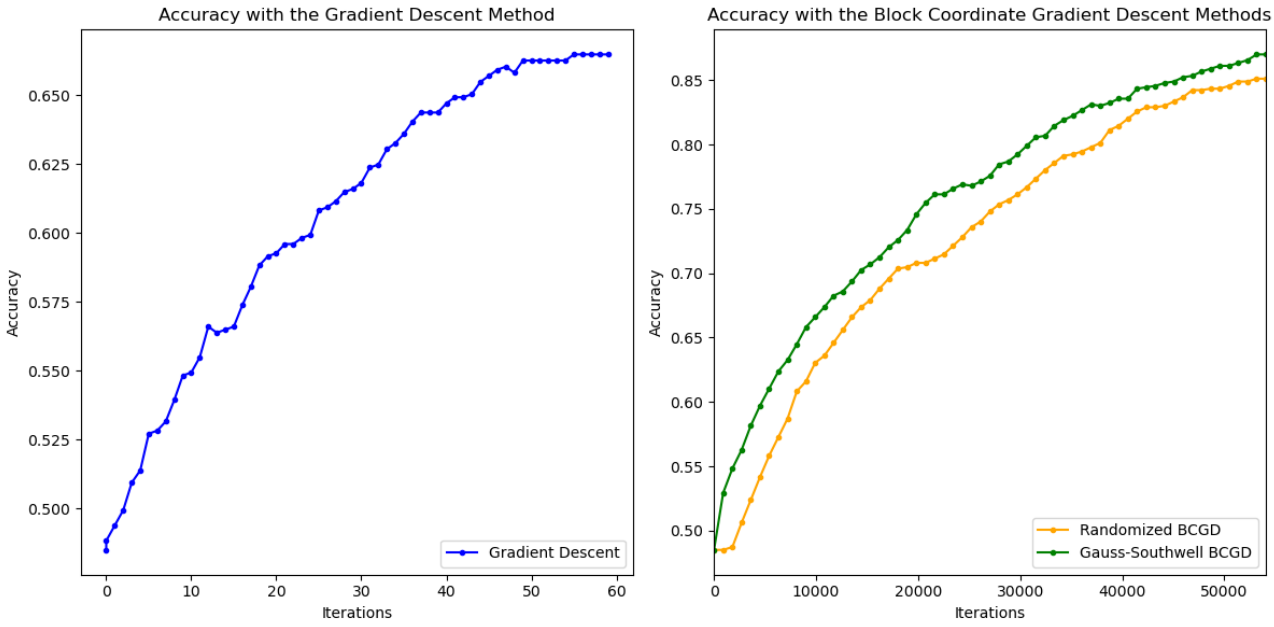


Figure 4: Diabetes dataset