Optimization for Data Science - Homework 1
Study of gradient descent and block coordinate gradient descent to approach a semi-supervised classification problem

Luis Marcos Lopez Casines
luismarcos.lopezcasines@studenti.unipd.it

Laia Porcar Guillamon
laia.porcarguillamon@studenti.unipd.it

Philippe Robert
philippe.robert@studenti.unipd.it

Stefanija Galevska
stefanija.galevska@studenti.unipd.it

1. Introduction and motivation

Nowadays it's relatively inexpensive to acquire data. The challenge arises when obtaining the labels of these data for a learning problem. A skilled human agent or a physical experiment are often required, which have a high associated cost. For this reason, having a fully labeled training set becomes almost unfeasible. Therefore, in many cases it's only possible to obtain a small set of labeled data and enough amounts of unlabeled one. Semi-supervised classification is used in these cases. It falls between supervised learning, when all data labels are known, and unsupervised learning, when no labels are present.

In this work, a semi-supervised learning problem is approached via the use of different optimization algorithms, such as standard gradient descent and block coordinate gradient descent (BCGD) with cyclic rule, randomized rule and random permutations. For this task, a set of random points and labels are generated, for most of which the label is later dropped. The accuracy at which the data points are classified with each of these algorithms is obtained, as well as the loss function. A subsequent analysis of the loss function against the number of iterations and the CPU time is performed. Finally, the algorithms are tested with a real world dataset.

2. Problem definition

We consider a binary classification problem with approximately the same number of points in each cluster. We have *l* labeled examples: $\left(\bar{x}^i, \bar{y}^i\right), \quad i = 1, ..., l$ and *u* unlabeled ones: $\bar{x}^j, \quad j = 1, ..., u$. The task is to find the labels $\bar{x}^j$ of the unlabeled points based on the precept that similar features will correspond to similar labels. In order to do this the inverse of the euclidean distance is defined as measure of similarity between two points, being $w_{i,j}$ the similarity between labeled examples *i* and unlabeled examples *j*, and $\bar{w}_{i,j}$ the similarity between unlabeled examples. We are thus left with the problem of the minimization of Eq. (1):

$$\min_{y \in \mathbb{R}^u} \sum_{i=1}^{l} \sum_{j=1}^{u} w_{ij} \left(y^j - \bar{y}^i\right)^2 + \frac{1}{2} \sum_{i=1}^{u} \sum_{j=1}^{u} \bar{w}_{ij} \left(y^i - y^j\right)^2. \tag{1}$$

The gradient of Eq. (1) with respect to y^j takes the form of Eq. (2):

$$\nabla y^j f(y) = 2 \sum_{i=1}^{l} \omega_{ij} \left(y^j - \bar{y}^i\right) + 2 \sum_{i=1}^{u} \bar{w}_{ij} \left(y^j - y^i\right). \tag{2}$$

The task is hence to classify the unlabeled examples with gradient descent and block gradient descent studied in the course *Optimization for Data Science*, making use of Eq. (2) to calculate the gradient.

3. Generated dataset

For the first part of this work, an own generated dataset of points will be employed.

3.1. Generation of the data

A total of 6000 [1] points were randomly generated following a standard normal distribution. They were all initially assigned a label 1 or -1 so that they are equally distributed between the two clusters. The relative distance between the two clusters has been carefully chosen to be the one shown in Fig. 1. In such manner, they are not too close to make the classification task a hard and inaccurate one, neither too far to make the problem too easy and naive.

A set of 1% of the points was randomly chosen to preserve the original labels, while the labels of the rest were

[1] The initial idea was to generate 10,000 points but this number turned out to be computationally very expensive.

removed and kept apart for subsequent analysis of the accuracy calculation. Furthermore, a set of null initialized labels was assigned to this unlabeled point set. An initialized label of 0 is half way between the two labels the problem uses and thus seems a reasonable starting point for the labels to be learned. Other option is to randomly initialize them to 1 and -1, but we found this option to be less effective because given a wrongly assigned label, it takes the gradient more steps to assign it to its correct label.

To sum up, the data provided to our algorithms are a dictionary of points with their original labels, a dictionary of points with initialized labels and a dictionary of these last points with their original labels, to be used in further classification accuracy analysis.

3.2. Definition of the algorithms and parameters

As has been said before, this work implements four optimization algorithms: standard gradient descent and BCGD with cyclic rule, randomized rule and random permutations.

The standard gradient descent algorithm calculates the gradient for a point and updates the label of that point before calculating next point's gradient. It keeps doing so until a stopping condition is reached. The BCGD algorithm with cyclic rule calculates first the gradient for each point before updating all the labels. It is done for all non labeled points (a cycle), and if the stopping condition is not reached yet, it starts a new cycle.Finally the BCGD algorithm with randomized selection calculates the gradient for each randomly chosen point, updates its label, and continues with the next point until the stopping condition is met. Here, there are two options, random permutation and random sampling. In random permutations all the points are chosen randomly without replacement, while in random sampling a point at a time is randomly taken with replacement.

These algorithms can be modified by changing some parameters which affect their convergence. One is the step size α, which in this work is taken to be fixed. A value of α between 10^{-5} and 10^{-4} usually performs well. The other is the stopping condition, which is defined as a certain threshold for the value of the gradient. In order to have a good compromise between result and computing time, the stopping condition is usually taken to be $gradient < 10^{-3}$. Naturally, by playing with these parameters the loss function can be minimized to a certain goal. The final accuracy of the problem will, however, depend on an extra factor. After the stopping condition is reached, the labels are discretized in a way that, if the final label is negative, -1 is assigned to it and 1 otherwise. In the unlikely case that a label turns out to be zero, then a 1 or -1 is randomly assigned to it.

The position and form of the clusters, the number of generated points and the number of labeled points are also useful parameters, but for this work, we are keeping them fixed to what has been said in previous sections.

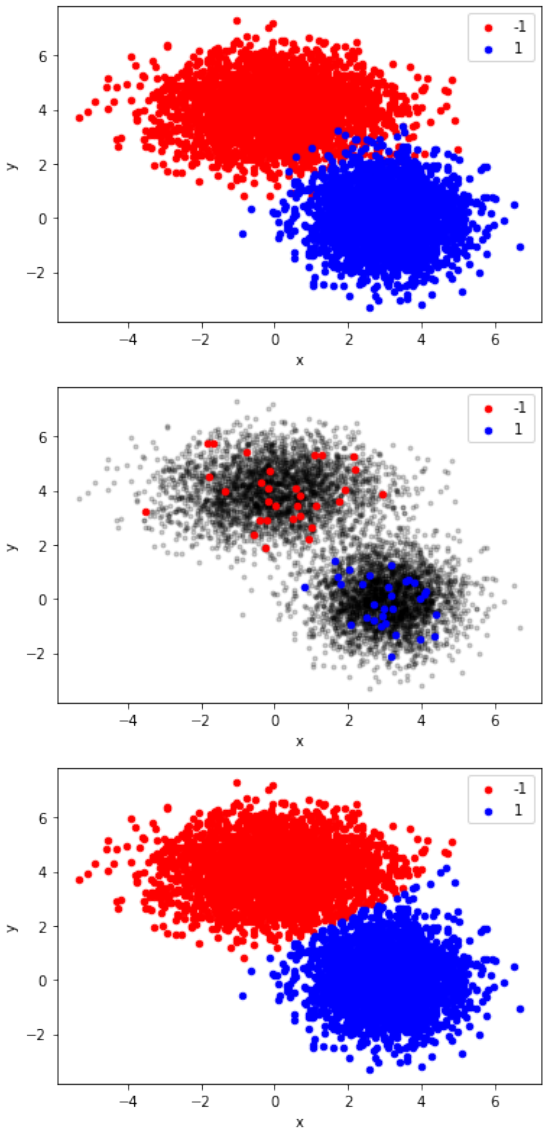

Figure 1: Example of generated clusters of points (top) and same clusters with only a set of the original labels left (bottom). The red cluster was conceived to have lightly more dispersion than the blue cluster.

3.3. Results

The four algorithms classify the points with an accuracy of 98%. To get an idea of the difference in performance between them the loss function has been plotted against the number of iterations and the CPU time (Fig. 2). The BCGD with cyclic rule minimizes the loss function faster than the other methods, closely followed by the BCGD with randomized rule. This happens for both the number of iterations and the CPU time.

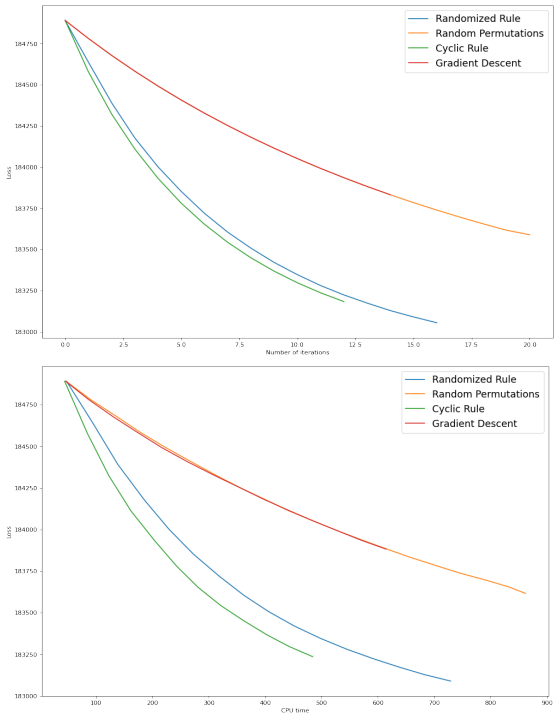

Figure 2: Loss function vs number of iterations (top) and Loss functions vs CPU time (bottom) for the Breast Cancer dataset.

4. Publicly available dataset

Once our algorithms prove to work with our generated data, the next step is to try their effectiveness with a real-world dataset. We will use the Breast Cancer dataset available in Kaggle.

4.1. dataset description

The Breast Cancer dataset consists on 32 features that describe 569 tumors according to their size, texture, compactness, concavity, symmetry, etc. The target column is "diagnosis", which classifies tumors in malign (M) and benign (B). There are no NaN or missing values.

For the purposes of this work, two features are needed to represent the points. The columns "radius_mean" and "symmetry_mean" have been chosen since they show a good separation of the clusters for the benign and malign cases. Nevertheless, the clusters are much closer and spread than the ones used in Section 3. The rest of the columns are dropped. The labels have been changed from [B,M] to the ones the algorithms work with: [1,-1].

Finally, a subset of 20% of the points is randomly chosen to preserve the original labels. The labels of the other 80% are set apart for the final calculation of the accuracy, while a new set of null initialized labels is assigned to the non labeled points (Fig. 3).

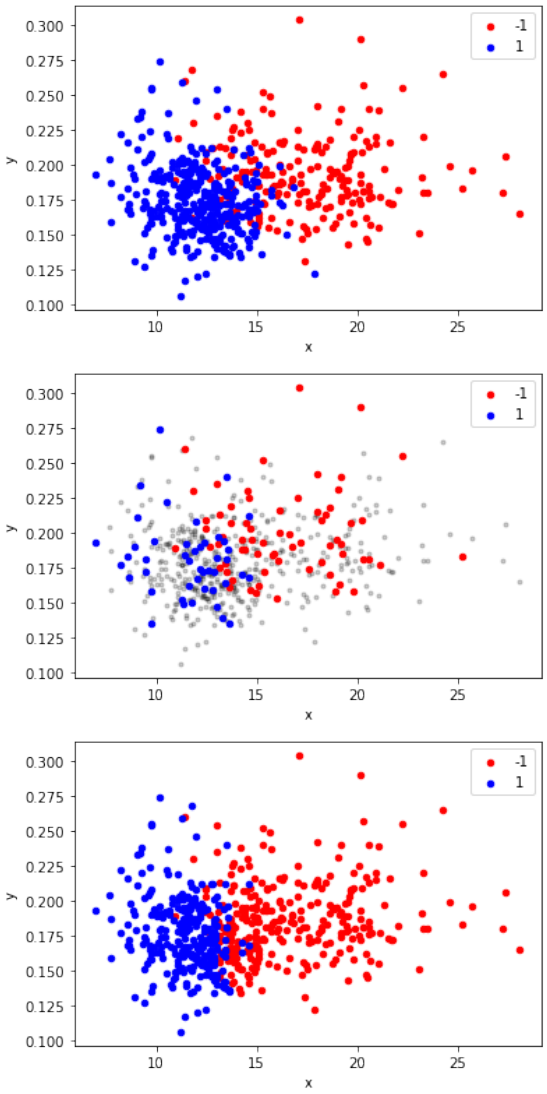

Figure 3: Clusters of the Breast Cancer dataset (top), a random selection of a set of points which preserve their original labels (middle) and final classification (bottom).

4.2. Results

Having the set of points with their original labels and the rest with initialized one, the previously used gradient descent and BCGD algorithms can be employed again with this new data. Since the amount of points in this case is considerably lower than that of the generated data, the running time is much less. An accuracy of 80% is obtained and an example of a classification result is shown in Fig. 3 (bottom). The loss function has been plotted vs. the number of iterations and CPU time for each of the algorithms, obtaining the plots shown in Fig. 4.

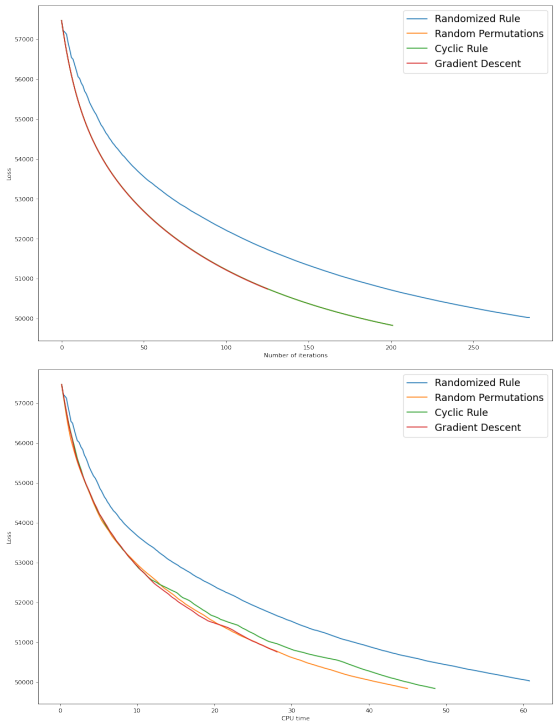

Figure 4: Loss function vs number of iterations (top) and Loss functions vs CPU time (bottom) for the Breast Cancer dataset.

5. Conclusions

All approaches considered to classify the generated data resulted on a satisfactory final accuracy. The analysis of the loss function with respect to the number of iterations and the CPU time reflects a clear victory of the BCGD with cyclic rule in both aspects. These results depend somewhat on the variability of the problem, in the sense that different runs can give slightly different results, as we have observed with experience. This is due to the amount of generated data points, 6000, which is still not such a large number as to always observe a superiority of the BCGD methods.

In the case of the Breast Cancer dataset we need to consider two important points. In the first place the number of points is relatively small. Second, the two clusters are very close to each other and not even a human eye could correctly classify some of the points that lie in the merging region. For this reason, the percentage of labeled points was chosen to be 20%. Nevertheless, the achieved accuracy of 80% is a good result given the previously mentioned drawbacks. As for a direct comparison among the algorithms, there is no clear winner with regard to the number of iterations needed to minimize the function. Except for BCGD with randomized rule, all algorithms exhibit a similar performance. As for the the CPU time spent minimizing the loss function, the standard gradient descent does a better, faster job, while the BCGD with randomized rule is the slowest.

In order to observe a better performance of the BCGD methods and a more accentuated difference among all algorithms a greater number of points is needed, as well as more computational power.