Algorithmic Methods in Optimization

Applications in semi supervised learning

Università degli studi di Padova

Abhishek Varma Dasaraju, Anna Badalyan, Brenda Eloísa Téllez Juárez,
Rebecca Di Francesco

May, 2022

Optimization

Introduction

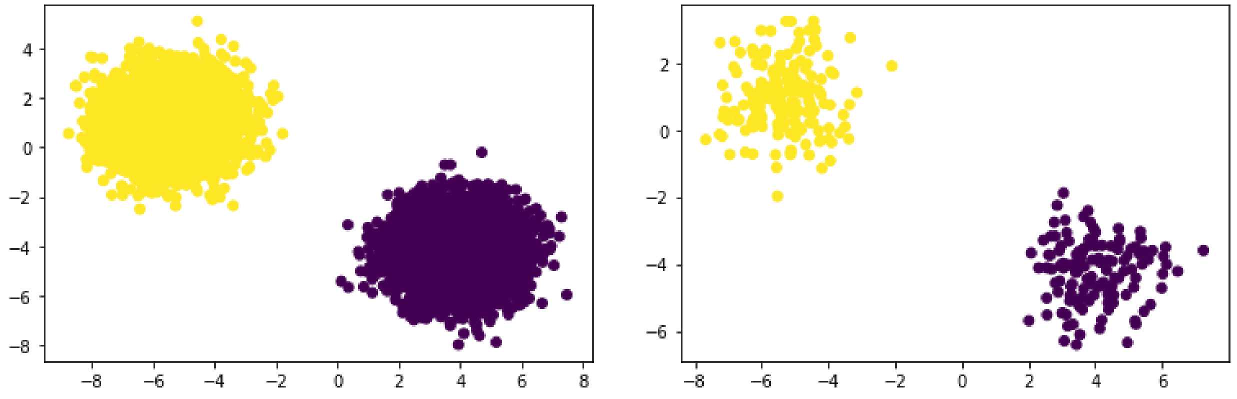
The purpose of this work is to develop optimization algorithms namely: Gradient Descent, Randomized BCGD, and Cyclic BCGD, and compare their performance with respect to the time and to the computational complexity in a context of semi-supervised learning.

1 Implementation

1.1 Dataset

At first, we generated the dataset by creating two blobs of points labelled -1, 1. Then, we randomly sampled a subset of the dataset that accounted for the 97% of the original using train_test_split. In this way we could have a semi-supervised learning instance to work on while having the ground truth of labels kept apart to compare the results at the end.

Table 1: Labeled dataset and data without labels.



1.2 Similarity function

In order to choose a proper similarity function, we need to make sure that the value of the objective function is minimized with the correct values of Y. The weights W_{ij} need to be small when the points belong to different classes (located far from each other). When the points Y_i and Y_j belong to the same class, the weights W_{ij} need to be large, as in the end,

these values will disappear from the objective function and the wrong prediction of the class will be penalized with the larger weight.
We used Euclidian distance as a distance metrics:

$$d(x, y) = \sqrt{\sum_{i}^{n} (X_i - Y_i)^2} \tag{1}$$

Firstly, we tried the following ways to define the similarity:

1. $W_{ij} = 1/((d(Y_i, Y_j) + 1)$
2. $W_{ij} = 1/((d(Y_i, Y_j) + 0.0001)$
3. $W_{ij} = exp(-d(Y_i, Y_j)^2 * 0.5)$–$RBF similarity\ with\ gamma = 0.5$

In the first case, the values were scaled from 0 to 1 and in the second case from 1 to 10000. However, the value of the objective function for the correct labels vector Y was large (8532640.85 and 9343856.86), while evaluating the objective function at $Y_i = 0$, for i from 1 to 9700, gave a smaller loss of 715414.81 and 1433381.5 respectively. In the third case, the value of the objective function with the correct y was equal to 0.03, while the value for the y set to 0s was 488749.
Thus, we observed that the first 2 similarity measures were not a proper choice in the case of our problem, while the third one was a good choice.
The reason the RBF similarity worked better than the linear transformation is that the exponential transformation brought far points much closer to 0, while the similar points were close to 1.

1.3 Choosing the right step size with fixed line search

Since this minimization problem is in a quadratic programming form, we exploited the structure of the problem by calculating a fixed step size that is $1/L$, where L is Lipschitz constant of the strictly convex function. So, to figure out the L we computed the Hessian of the function with respect to the unlabeled Y's and we extracted the maximum eigenvalue of the Hessian:

$$\frac{\partial^2 f}{\partial(Y_j)} = 2\sum_{i=1}^{l} W_{ij} + 2\sum_{i=1}^{u} \overline{W}_{ij} \tag{2}$$

$$\frac{\partial^2 f}{\partial(Y_j)\partial(Y_i)} = -2\overline{W}_{ij} \tag{3}$$

In practice we noticed that this step size worked well on the toy dataset with the gradient descent but it was slow in terms of reaching convergence for the BCGD algorithms. So, we experimented with a higher learning rate and this helped in reducing the number of iterations for the Cyclic and Randomized BCGD.

1.4 Defining the stopping criteria

As a stopping criteria, we used the norm of the gradient multiplied by the absolute value of the step size.

2 Results

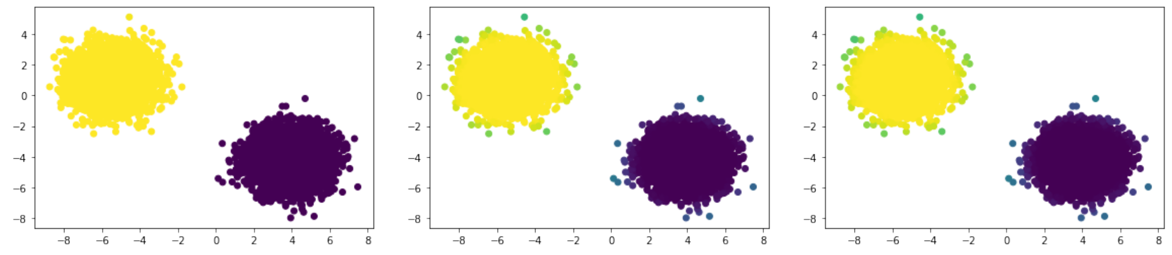


Figure 1: Optimization algorithms

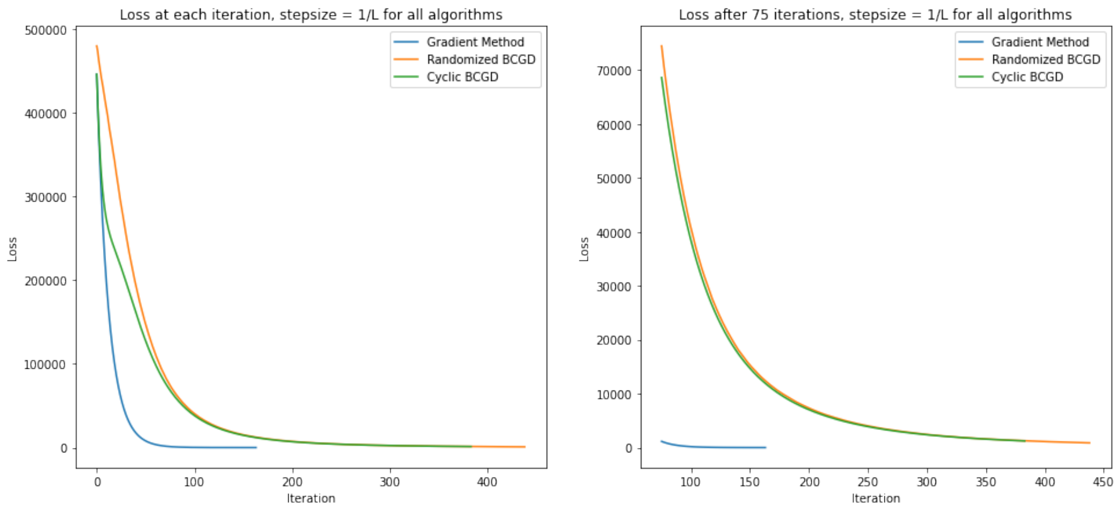
(a) Gradient Descent (b) Randomized BCGD (c) Cyclic BCGD

2.1 Comparing the efficiency of the algorithms

2.1.1 Comparing loss vs iterations with a step size of 1/L

Comparing the results of the three algorithms we could test in practice what we studied with the theory of optimization so far. The gradient descent was the algorithm that guaranteed the lower iteration complexity and a remarkable and consistent decrease in loss starting from the first iterations. While the Randomized BCGD and the Cyclic BCGD presented very similar behaviors in terms of loss convergence, with a higher number of iterations needed in order to converge.

Figure 2: Plots with the step size = 1/L in all algorithms



2.1.2 Comparing loss vs time with a step size 1/L

We can observe in figure 3 that the gradient method requires less iterations to converge, for this reason in the figure 4 we can notice that it is the fastest method in terms of total time required but compared to the others showed a significantly higher computational cost per iteration. On the other hand, the randomized BCGD requires the greatest quantity of iterations but with respect of CPU time per iteration on average is two milliseconds faster than the gradient. Finally, for the cyclic method, we can realize that it requires the lowest CPU time per iteration, also a slightly lower number of iterations than the randomized method.

Figure 3: Plots with the step size = 1/L in all algorithms

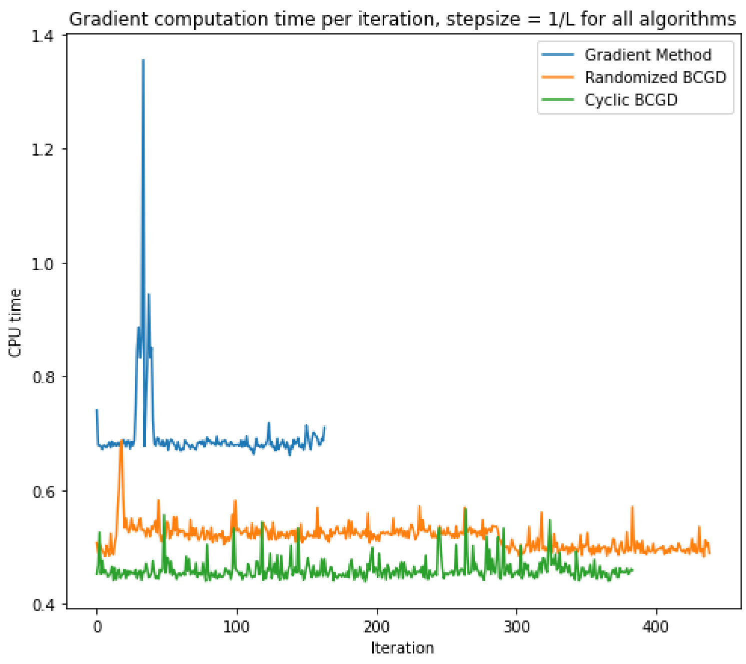
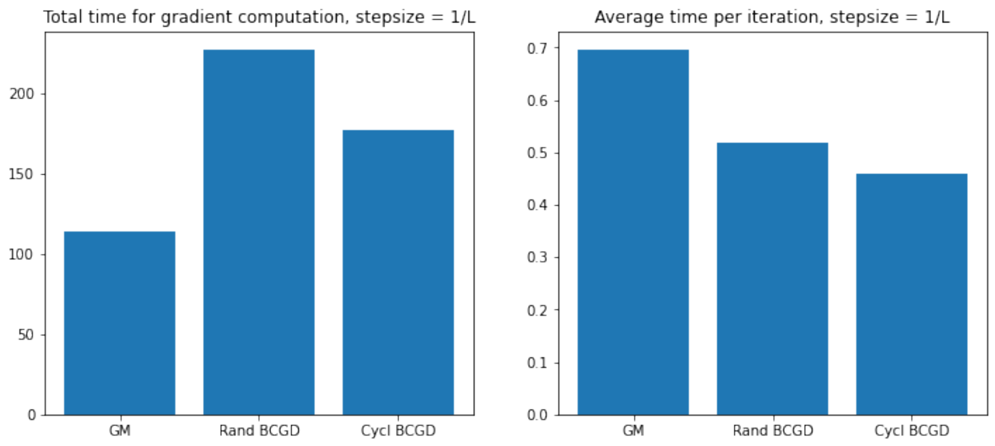


Figure 4: Plots with the step size = 1/L in all algorithms



2.2 Experiments

We decided to experiment with the stepsize to see if the iterations complexity could be reduced for the BCGD algorithms. What noticed that with an increased in the stepsize from 1/L (0.0004) to 0.001 the two BCGD algorithms could minimize the function faster. Moreover by further increasing the stepsize to 0.005 (figure 6) the randomized and the cyclic BCGD algorithms could converge even better than the gradient descent. In figure 7, we plotted the loss curves for the different step sizes tested for the Randomized BCGD and the Cyclic BCGD to highlight their improvement in iterations convergence when the learning rate was increased.
In terms of gradient computation time for each iteration, we can confirm from the plots in the figure 7 that choosing a different step size will not have an impact on this aspect. However, regarding the total time (i.e. the total time required to run an algorithm before a stopping criteria is met) we can see that for the gradient it has increased, while for the BCGD algorithms these larger step sizes due to the reduction in iteration complexity have decreased also the total time.

Figure 5: Loss vs iterations for 0.001

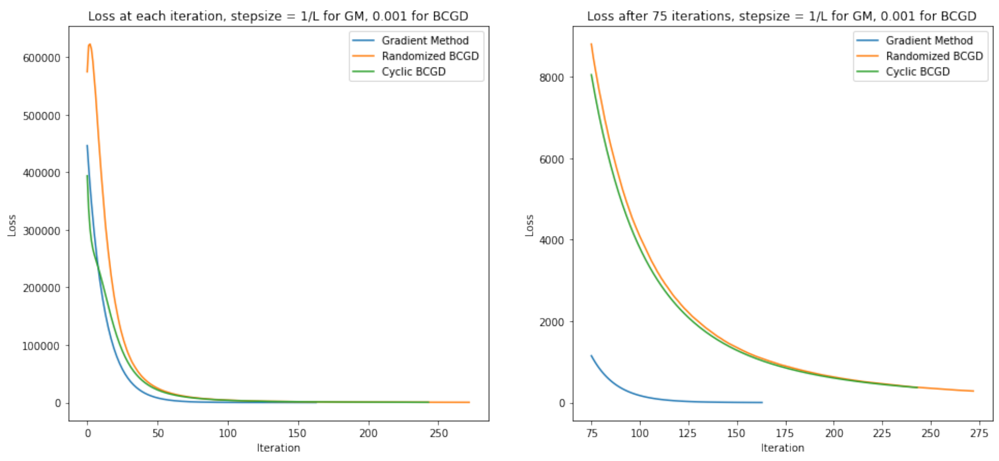


Figure 6: Loss vs iterations for 0.005

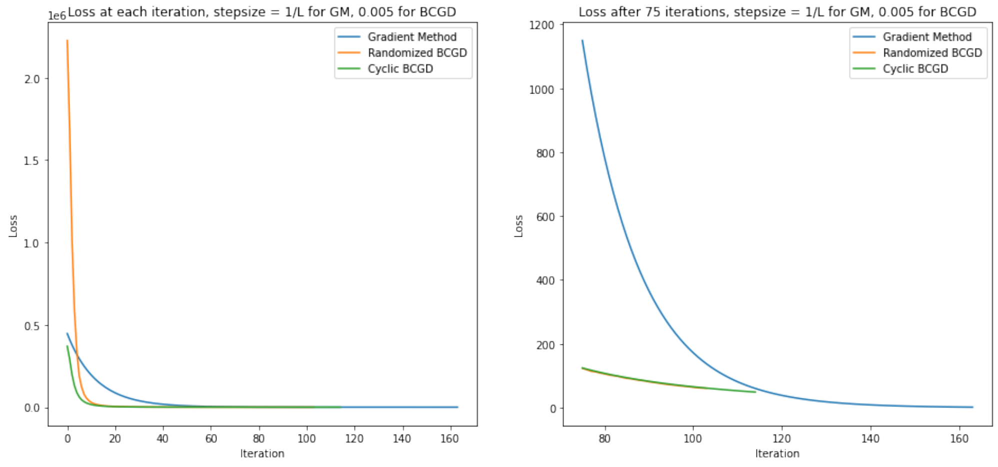


Figure 7: Step size comparison for BCGD methods

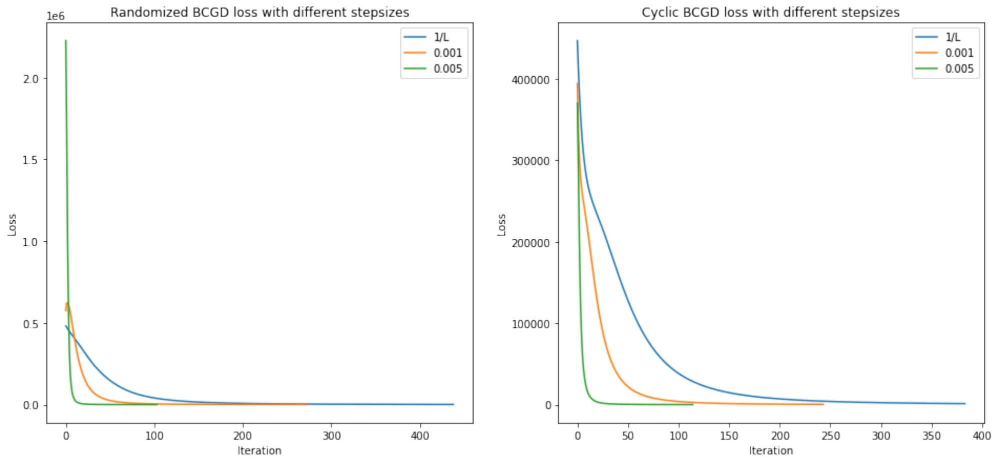


Figure 8: Computation time per iteration for 0.001

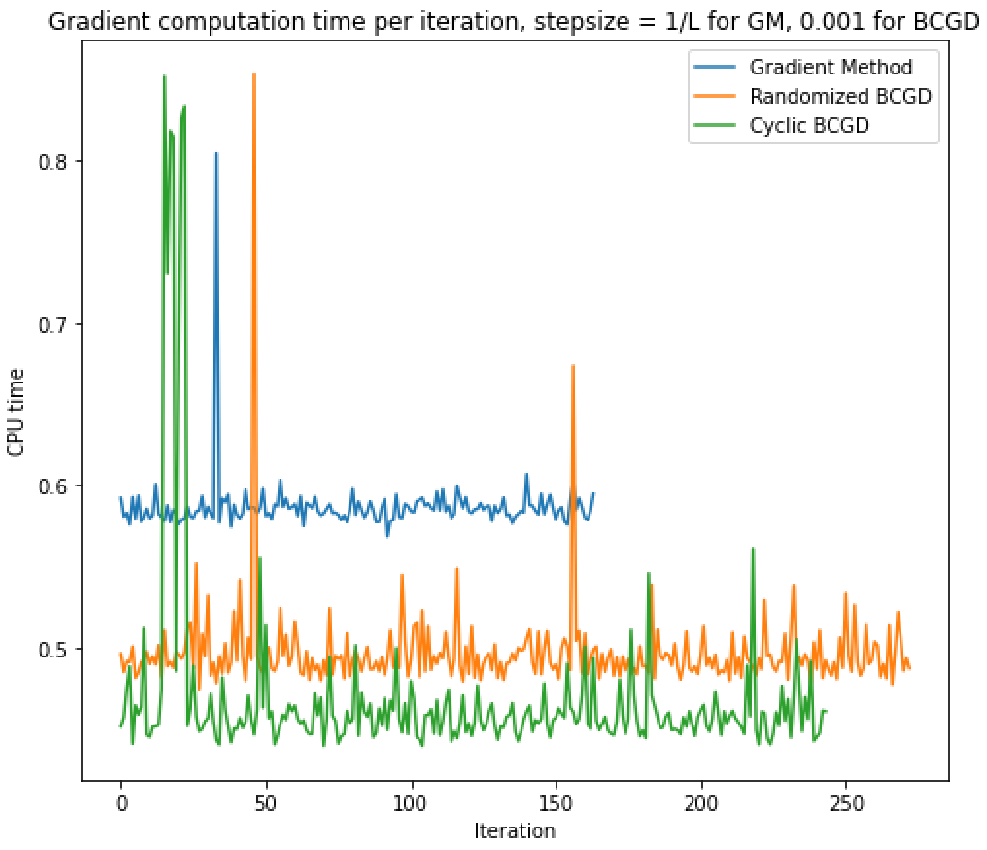


Figure 9: Computation time per iteration for 0.005

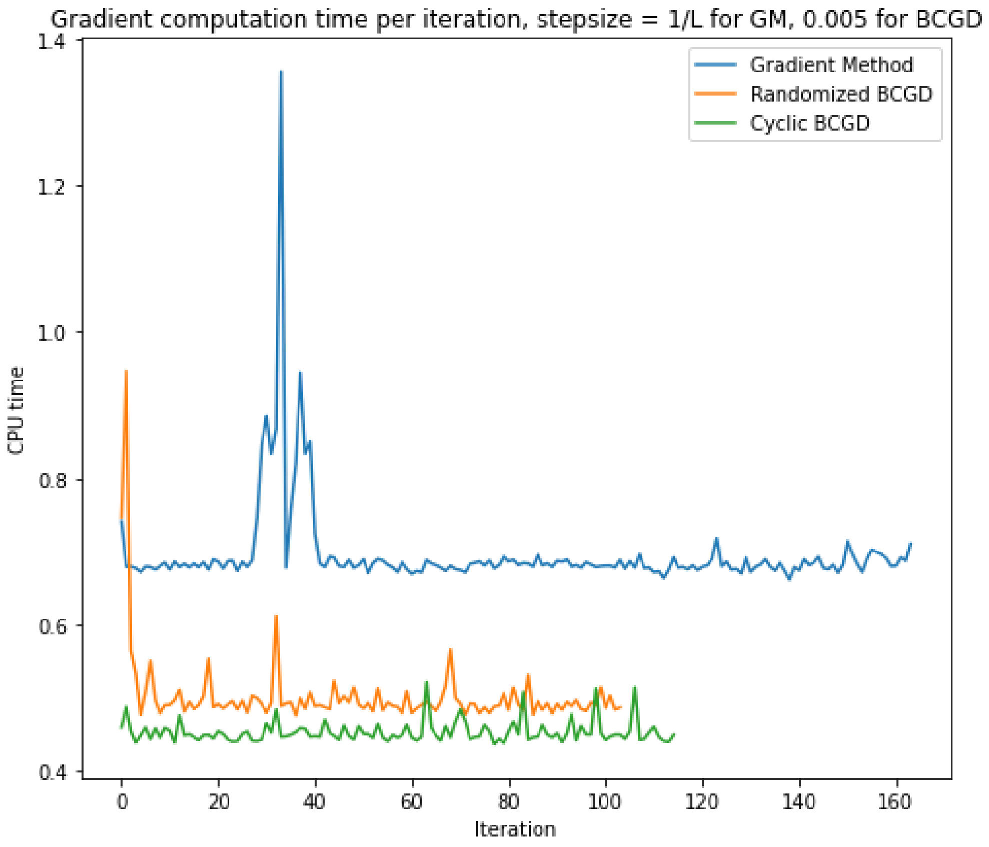


Figure 10: Total and average gradient computation time for 0.001

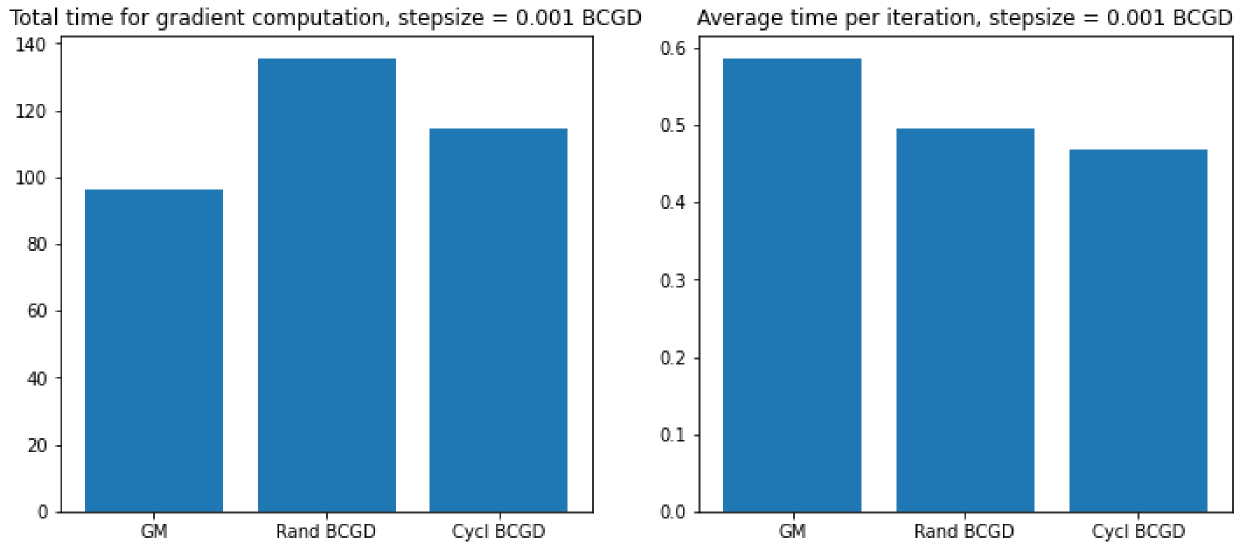
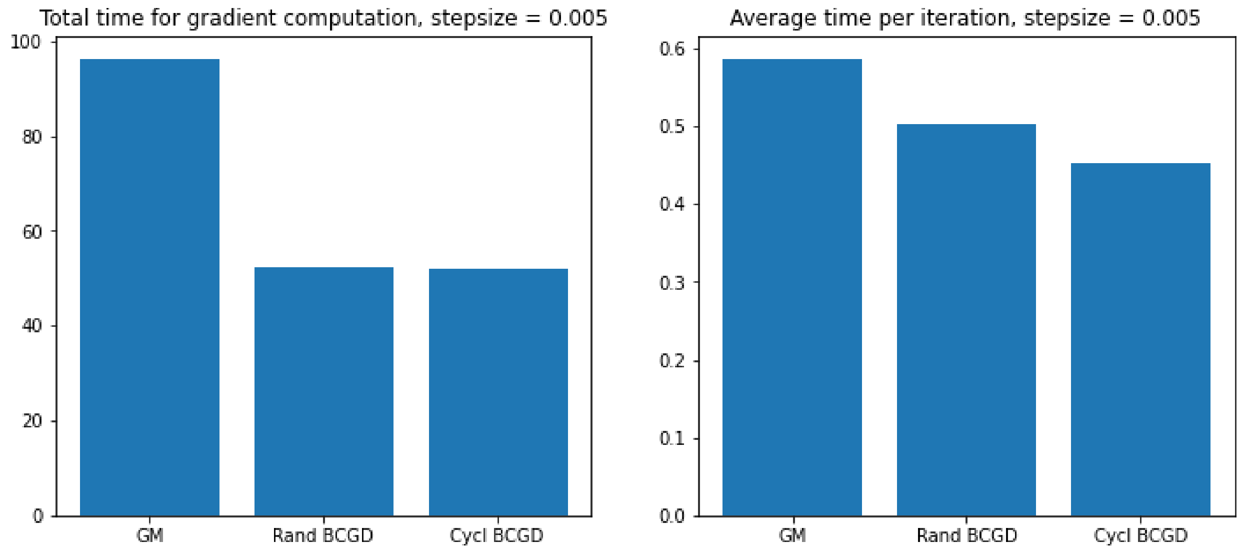


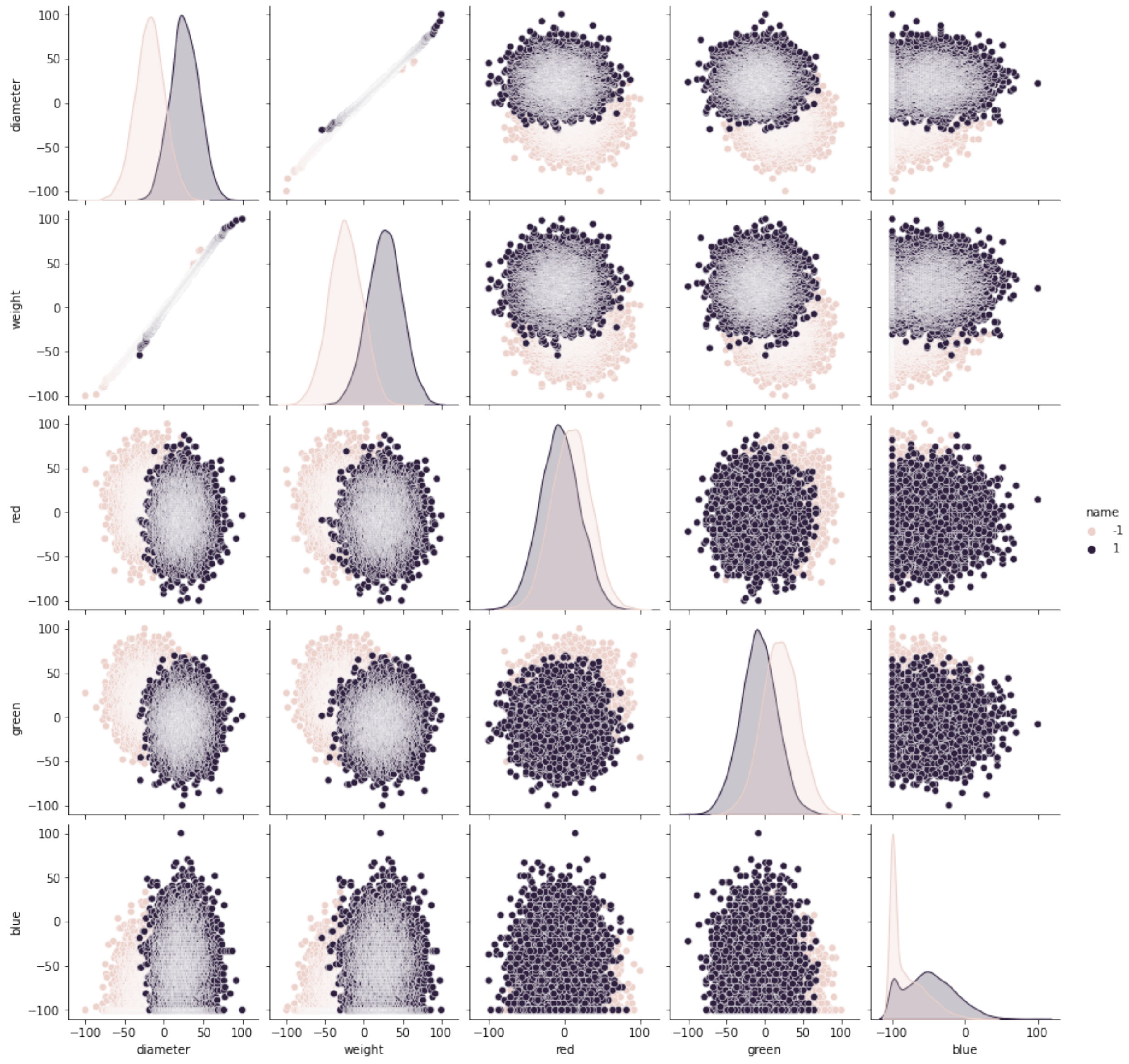
Figure 11: Total and average gradient computation time for 0.005



3 Applications to a real dataset

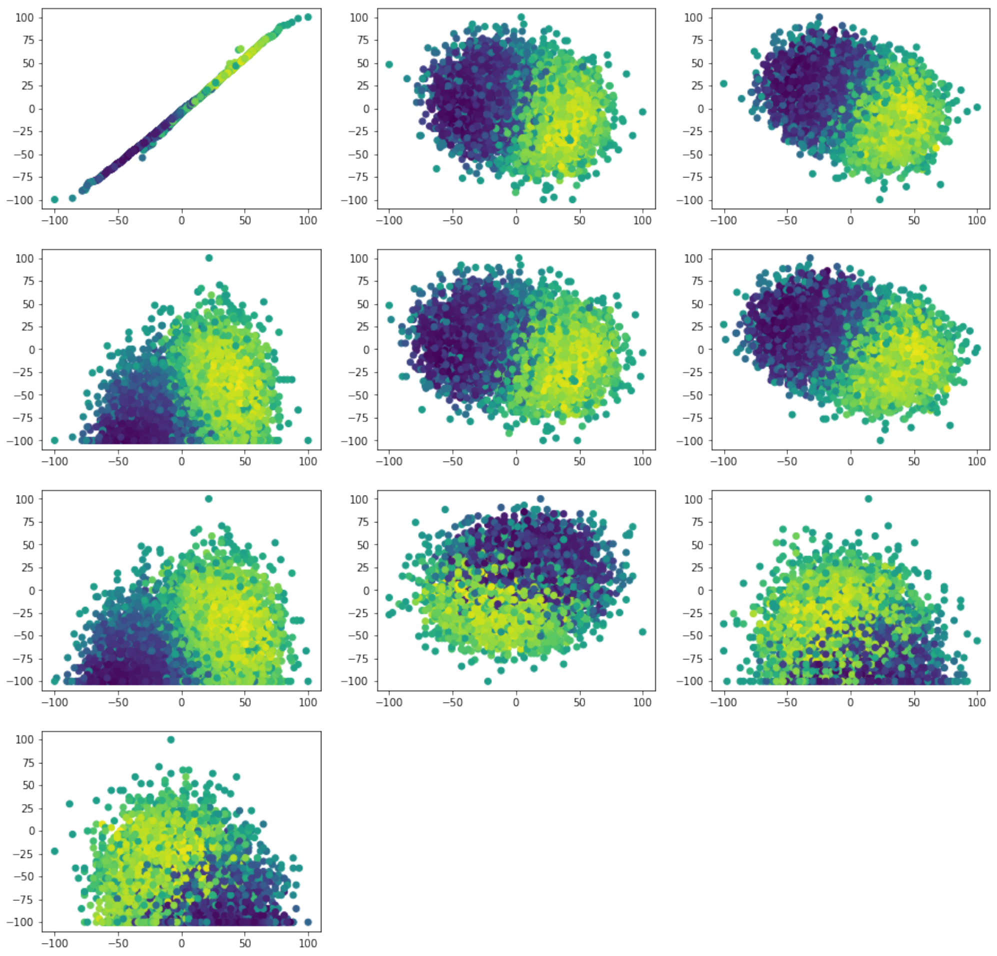
We have chosen the citrus dataset which contained 10000 observations on oranges and grapefruits. The dataset contained 5 features: diameter, weight, red, green, blue. The task was to predict the fruit type based on these features.

Figure 12: Exploring the dataset



Firstly, we normalized the data and scaled in the range between -100 and 100 in order to make the points more sparse. As the points in the dataset were not clearly separated, it was of paramount importance to choose the right similarity function. The experiments were carried out to choose the right value of gamma. We tested gamma = 1, 1/5, 1/50, /100, 1/200 and 1/400. The rbf kernel with gamma = 1/100 showed better results.
The gradient method algorithm converged to the minimum and predicted the correct signs for 8936 points (92% accuracy). The randomized and cyclic BCGD algorithms predicted only 8710 points (90% accuracy). We can observe that the GM algorithms performed better on this dataset. The randomized and cyclic BCGD did not converge as the loss was increasing with each iteration. This can be explained with the fact that the points were not clearly separable.

Figure 13: Predictions by Gradient Descent Algorithm



4 Bibliography

- Boyd, S., Vandenberghe, L. (2009). Convex Optimization. United Kingdom: Cambridge University Press.

- J. Kochenderfer, M., A. Wheeler, T. (2019). Algorithms for Optimization. The MIT Press.

- P. Bertsekas , D. (2015). Convex Optimization Algorithms. Athena Scientific.