Optimization for Data Science - Homework

Jacopo Guidolin
jacopo.guidolin@studenti.unipd.it

Mattia Miolato
mattia.miolato@studenti.unipd.it

Filippo Santin
filippo.santin@studenti.unipd.it

Giovanni Vedana
giovanni.vedana.1@studenti.unipd.it

1. Introduction

In this brief report, we will exhibit how we approached the task we were required to solve in the assigned homework.

Our task was to first implement three different optimization methods on a mock dataset in order to then utilize those methods on real data.

2. Dataset

The real dataset we decided to work with consist of 2000 samples. Each sample represents a mobile phone model with its features (*processor speed, battery storage capacity, internal memory available, etc.*). These samples are classified into four price ranges and each class consists of 500 samples.

Source: https://www.kaggle.com/iabhishekofficial/mobile-price-classification

3. Method

3.1. Building and testing the model

In the first part of the homework we were asked to first randomly generate a set of points in two dimensions to then label a small subset of them. In order to do this we chose two different normal distributions, $N_1((\frac{3}{2}, \frac{3}{2}), \mathbf{1})$ and $N_2((-\frac{3}{2}, -\frac{3}{2}), \mathbf{1})$. We sampled 100 points from each distribution and 40 of these 200 generated points got labeled according to the normal distribution they got generated from (Figure 1).

Then we were asked to define a proper similarity measure. Given two points $x, y \in \mathbb{R}^n$ we defined the similarity between x and y as

$$\sigma(x, y) = e^{-\frac{(x-y)\cdot(x-y)}{2}}.$$

We used this particular similarity measure, an exponential decreasing function, so that as the distance between x and y increases, the similarity approaches 0, and on the other hand, as the distance decreases, the similarity approaches 1. This measure has the desirable property of being bounded resulting in reasonable similarity values even between overlapping points.

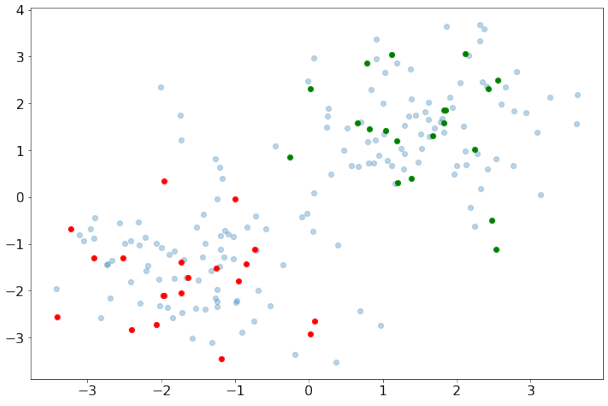

Figure 1. Randomly generated dataset.

We used this similarity measure to generate two matrices W and $\bar{W}$. These two matrices measure the pairwise similarity between labeled and unlabeled points and between couples of unlabeled points, respectively.

We then defined the loss function $\mathcal{L}$ using these matrices

$$\mathcal{L}(y) = \sum_{i=1}^{\ell}\sum_{j=1}^{u} w_{ij}(y^j - \bar{y}^i)^2 + \frac{1}{2}\sum_{i=1}^{u}\sum_{j=1}^{u} \bar{w}_{ij}(y^j - y^i)^2$$

where y is the vector of the predicted labels and $\bar{y}$ contains the already existing labels.

We proceeded with the implementation of our optimization framework on the simulated dataset.

As we were asked to do we utilized three different methods for the optimization: Gradient Descent, Randomized BCGD and Cyclic BCGD with blocks of dimension 1.

What we wanted to achieve was for our method to correctly classify unlabeled points according to the normal distribution they belonged to. Testing these methods on our

randomly generated dataset led to satisfactory results (cfr. Figure 2).

Therefore we proceeded to the analysis of our real dataset.

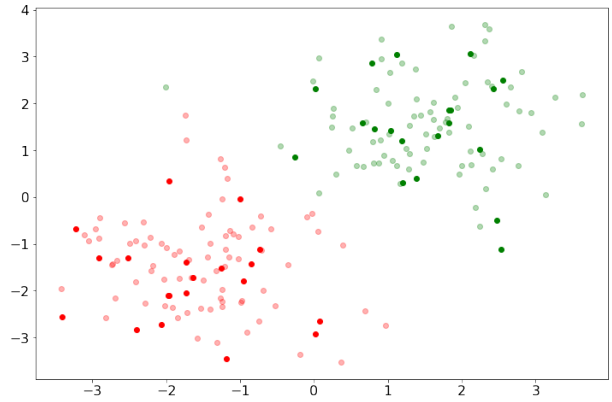

Figure 2. Classification on the random dataset. Accuracy: 98.75%

3.2. Application on real data

In order to test our model on the selected dataset we decided to consider only two of the four existing price classes. To proceed we stripped unwanted samples from the dataset and kept only the ones from the two price classes we selected.

After having rescaled the features of our data in order to normalize them we were ready to observe our method in action.

We proceeded considering that the convexity of the loss function assured the convergence of the traditional Gradient Descent method.

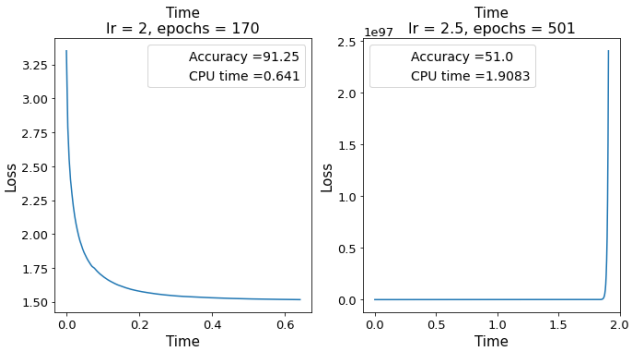

Figure 3. Time\Loss graphics with Gradient Descent when using different learning rates. In the first case the method converges rapidly while an excessive learning rate led to divergence in the latter one.

To optimize our training we decided beforehand to introduce some devices. We added a tolerance threshold on the absolute difference of the values of the loss function between two consecutive iterations. In this way the execution will stop if no substantial improvements on the value of the loss function occurred. We did not implement this type of early stopping when utilizing the Random BCGD method since for this particular method we could have no improvements over several iterations due to the randomness of the selection. Therefore we bounded the maximum number of epochs during execution to further prevent unnecessary iterations or stopping divergent behaviours.

For each method we fine tuned the learning rate in order to assure both convergence and optimal performance (cfr. Figure 3).

4. Conclusion

As we can see represented in Figure 4 and Figure 5 the Gradient Descent method obtains better accuracies compared with the others while maintaining lower CPU times. We expected these results due to the convexity of the analyzed function and the moderate dimension of the space we are working with.

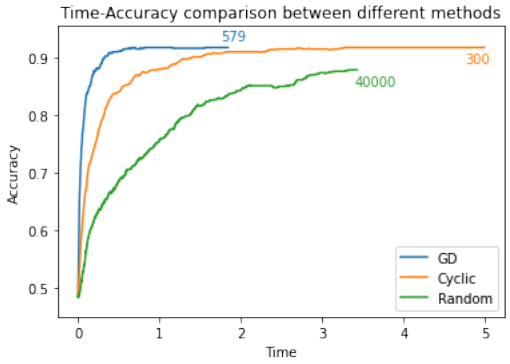

Figure 4. The Time\Accuracy lines shown are labeled with the number of corresponding epochs.

The Randomized BCGD Method while needing a large number of epochs to reach acceptable accuracies does it in a reasonable amount of time thanks to its inexpensive iterations. Conversely the Cyclic BCGD method despite its expensive iterations manages to reach high accuracies in feasible amounts of time.

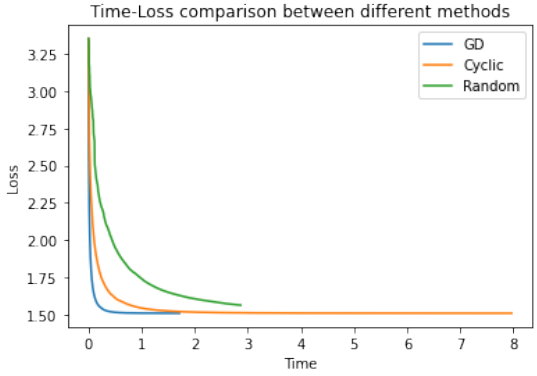

Figure 5.