A semi-supervised learning problem via Gradient Methods and Block Coordinate Gradient Methods

Anthony Palmieri
anthony.palmieri@studenti.unipd.it

1. Introduction

In this homework we focused on a semi-supervised learning problem. Specifically, given a dataset composed of labeled observations, we discarded all but 3% of the total amount of labels and then tried to infer these discarded labels via optimization algorithms.
In our case we analyzed two datasets:

1. a dataset randomly generated via Sklearn library;
2. the Census dataset, which can be found here.

We'll come back to discuss these datasets later on.
The optimization algorithms we tested are respectively:

- Gradient Descent;
- Gradient Descent with Nesterov Momentum;
- Cyclic Block Coordinate Gradient Descent;
- Uniform Randomized Block Coordinate Gradient Descent (i.e. $P(i_k = i) = 1/b \quad \forall i \in \{1, \dots, b\}$);
- Nesterov Randomized Block Coordinate Gradient Descent (i.e. $P(i_k = i) = \frac{L_i}{\sum_{j=1}^{b} L_j} \quad \forall i \in \{1, \dots, b\}$);
- Gauss-Southwell Block Coordinate Gradient Descent.

For each one of these algorithms we used a fixed step-size strategy, in particular we used the optimal fixed step-sizes given by the Lipschitz constants of the gradient of the Loss function.
For the classic gradient methods we used the Lipschitz constant of the whole gradient, whereas for the block coordinate gradient methods we used the block-wise Lipschitz constants.
Before talking about how we derived these constants, let's first have a look at the Loss function we used to model the problem.

2. Loss Function

Recall that our dataset is composed of labeled and unlabeled observations, let's say we have l labeled observations and u unlabeled ones.
Let's also suppose to have defined a similarity measure to compute some sort of weights between labeled-unlabeled observations and unlabeled-unlabeled observations.
Let's call these weight matrices W and $\overline{W}$ respectively (we'll come back to discuss the nature of these matrices later on).
Let's also use $\overline{y}$ to indicate the vector of labels for labeled examples and y to indicate the vector of unknown labels we wish to predict, so to have $\overline{y} \in \mathbb{R}^l$ and $y \in \mathbb{R}^u$.
With this notation in mind we define the loss function

$$J(y) = \sum_{i=1}^{l} \sum_{j=1}^{u} w_{ij}(y_j - \overline{y}_i)^2 + \frac{1}{2} \sum_{i=1}^{u} \sum_{j=1}^{u} \overline{w}_{ij}(y_j - y_i)^2$$

and the problem we wish to solve is

$$\min_{y \in \mathbb{R}^u} J(y).$$

We're dealing with a convex unconstrained optimization problem, meaning that a point y^* is an optimal solution *if and only if* $\nabla J(y^*) = 0$, so it makes sense to use the norm of the gradient at each step as stopping criterion.
Let's also have a quick look at the two terms that appear in the loss function:

- Minimizing the first term means that for each unlabeled observation we want its label to be as close as possible to the labels of the "closest" labeled observations;
- On the other hand, minimizing the second term means that we want unlabeled observations "close" to each other to have similar labels.

The concept of closeness is addressed by the weight matrices W and $\overline{W}$, which we are now going to discuss.

3. Similarity Measures and Weight Matrices

To compute the weight matrices $W \in \mathbb{R}^{l\times u}$ and $\overline{W} \in \mathbb{R}^{u\times u}$ we used a *Gaussian-like similarity measure*, specifically, given two points $P_1,\ P_2 \in \mathbb{R}^n$ their similarity is given by

$$Similarity(P_1, P_2) = \exp\left(-\frac{\|P_1 - P_2\|}{2\sigma^2}\right).$$

There are two main reasons behind this choice:

- As we already said, we want points that are close to each other to share similar labels, in this sense, given a point P_j, points that are farther away from P_j shouldn't have much influence on determining its label y_j. This is perfectly captured by the Gaussian similarity measure, as points distant from one another will result in weights close to 0, and consequently a little influence on determining their respective labels;
- Gaussian similarity measure also comes really handy when dealing with numerical instability, in fact thanks to the parameter σ we can easily adjust weights that are too close to zero.

For the first dataset we used $\sigma = 1$, since being the points clustered in a small region there weren't any weights too close to zero.
For the second dataset on the other hand, some points resulted being quite far from each other, leading the respective weight to be too close to zero. In this case to avoid numerical instability we used $\sigma = 100$, in order to re-increase the weights.
After discussing the Weight matrices, we can finally move on and talk about how we derived the Lipschitz constants.

4. Fixed Step-sizes and Lipschitz Constants

We now try to derive the Lipschitz constant for the whole gradient, as well as the Lipschitz constants for the various blocks.
Notice that in this case we are dealing with one-dimensional blocks, so the gradient w.r.t. the j-th block is just the partial derivative of J with respect to y_j. Let's start by looking at these partial derivatives. We have

$$\begin{aligned}\nabla_j J(y) = \frac{\partial J}{\partial y_j} &= 2\left(\sum_{i=1}^{l} w_{ij}(y_j - \overline{y}_i) + \sum_{i=1}^{u} \overline{w}_{ij}(y_j - y_i)\right)\\ &= 2\left(\sum_{i=1}^{l}(w_{ij}y_j - w_{ij}\overline{y}_i) + \sum_{i=1}^{u}(\overline{w}_{ij}y_j - \overline{w}_{ij}y_i)\right)\\ &= 2\left(y_j\sum_{i=1}^{l} w_{ij} - \sum_{i=1}^{l} w_{ij}\overline{y}_i\right) +\\ &+ 2\left(y_j\sum_{i=1}^{u}\overline{w}_{ij} - \sum_{i=1}^{u}\overline{w}_{ij}y_i\right).\end{aligned}$$

At this point $\forall j \in \{1,\dots,u\}$ let's define:

- W_j j-th column of W, $\ W \in \mathbb{R}^{l\times u}$
- $S_j = \sum_{i=1}^{l} w_{ij}$ the sum of the elements of W_j
- $\overline{W}_j$ j-th column of $\overline{W}$, $\ \overline{W} \in \mathbb{R}^{u\times u}$
- $\overline{S}_j = \sum_{i=1}^{u} \overline{w}_{ij}$ the sum of the elements of $\overline{W}_j$

With this notation in mind we can rewrite

$$\begin{aligned}\nabla_j J(y) &= 2\left(y_jS_j - \overline{y}^TW_j + y_j\overline{S}_j - y^T\overline{W}_j\right)\\ &= 2\left(y_j(S_j + \overline{S}_j) - \overline{y}^TW_j - y^T\overline{W}_j\right)\\ &= 2\left(y^T diag(S+\overline{S})_j - \overline{y}^TW_j - y^T\overline{W}_j\right)\\ &= 2\left(y^T(diag(S+\overline{S}) - \overline{W})_j - \overline{y}^TW_j\right)\end{aligned}$$

where $diag(S+\overline{S})$ is the diagonal matrix having the vector $S+\overline{S}$ as diagonal, and the j subscript indicates the j-th column of the related matrix.
Let's consider now $h \in \mathbb{R}$ and e_j the j-th vector of the canonical basis of $\mathbb{R}^u$. We have

$$|\nabla_j J(y + he_j) - \nabla_j J(y)| = \dots = 2|(S_j + \overline{S}_j - \overline{w}_{jj})| \cdot |h|$$

from which we deduce the block-wise Lipschitz constants

$$L_j = 2|(S_j + \overline{S}_j - \overline{w}_{jj})| \quad \forall j \in \{1,\dots,u\}.$$

Moreover we can also rewrite the whole gradient as

$$\nabla J(y) = 2\left(y^T(diag(S+\overline{S}) - \overline{W}) - \overline{y}^TW\right).$$

Let's consider now $h \in \mathbb{R}^u$, we have

$$\|\nabla J(y+h) - \nabla J(y)\|_2 = \dots = 2\|h^T(diag(S+\overline{S}) - \overline{W})\|_2.$$

Recall that a matrix norm is said to be *consistent* if $\forall A \in \mathbb{R}^{m\times n}$ and $\forall x \in \mathbb{R}^n$ we have $\|Ax\| \leq |||A||| \cdot \|x\|$.
It can be shown that every *induced matrix norm* is consistent and being $|||\cdot|||_2$ induced matrix norm we have

$$\|\nabla J(y+h) - \nabla J(y)\|_2 \leq 2|||diag(S+\overline{S}) - \overline{W}|||_2 \cdot \|h\|_2,$$

from which we can finally deduce the Lipschitz constant

$$L = 2|||diag(S+\overline{S}) - \overline{W}|||_2.$$

From the theoretical analysis we saw in class, we know that using the Lipschitz constants for the step-sizes yields the best results in terms of convergence rate. We then define the step-sizes as:

- $\alpha = 1/L$ for classic Gradient Descent methods;
- $\alpha_j = 1/L_j \quad \forall j \in \{1,\dots,u\}$ for BCGD methods.

5. Starting Points

As a last thing, before commenting the results we obtained on the two datasets, let's spend some words about the starting points we used.
We experimented with two different starting points:

- a randomly generated vector $y_0 \in \mathbb{R}^u$, with components uniformly distributed in $[-1, +1]$;
- a vector $y_{0,\,kNN} \in \mathbb{R}^u$ generated using a k-nearest neighbor strategy, with the j-th component being the most frequent label among the labeled k-nearest neighbors of the unlabeled observation x_j.

For the deterministic methods both of these starting points led to similar final accuracy scores, whereas for the randomized methods the kNN was able to achieve greatly better performances, moving from 93.8% to 100% on the first dataset and from 65.6% to 75.8% on the second dataset, an increase of about 10%.
We also noticed that in general, the kNN strategy was able to achieve a greater reduction of the norm of the gradient compared to the randomly generated starting point, a situation again particularly evident on the randomized methods.
Moreover on the first dataset, starting with $y_{0,kNN}$ led to an accuracy score constantly equal to 1 at each iteration, for all the methods. This could be due to the fact that being the classes already well separated, $y_{0,kNN}$ corresponds exactly to the vector of labels we're trying to predict.

6. First Dataset Analysis

The first dataset we used to test our optimization algorithms is a dataset randomly generated using the make_blobs method from sklearn library.
The dataset consists of 10000 points in $\mathbb{R}^2$, already clustered into two clusters, with every point of a cluster having the same label in $\{-1, +1\}$. We then discarded all but 3% of the total amount of labels using a randomly generated vector of indices. The situation is the following

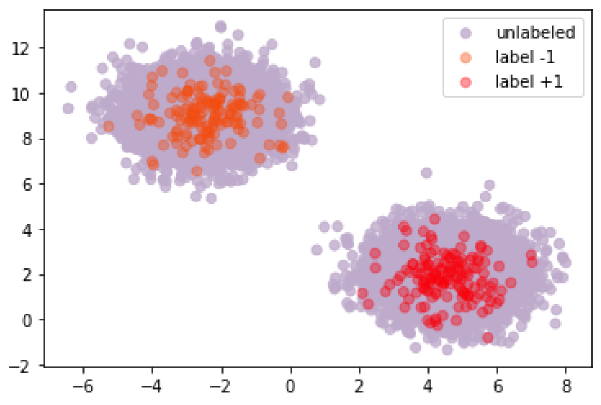

Figure 1: Dataset generated via Sklearn

On this dataset, using y_0 as starting point, Gradient Descent, Gradient Descent with Nesterov Momentum and Cyclic Block Coordinate Gradient Descent performed relatively well, achieving an accuracy score of 1 in a very little time. Even the Gauss-Southwell method performed relatively well, even though it required longer execution times.
On the contrary, randomized methods took much more iterations to reach comparable accuracy levels, namely 20000.
In the following a plot depicting the accuracy trend for the various algorithms, using the randomly generated starting point.

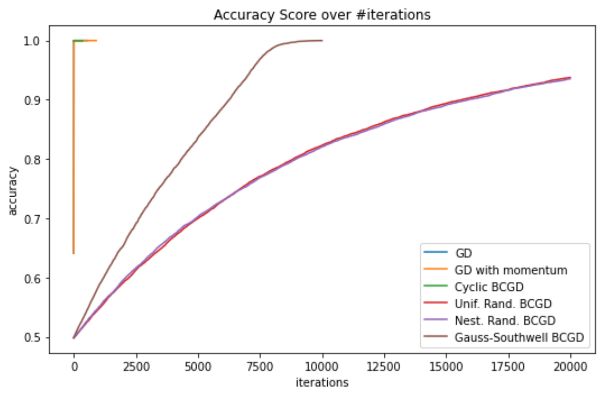

Figure 2: Accuracy vs Iterations using y_0

Using $y_{0,\,kNN}$ instead, as we said before the accuracy score remained stuck on 1 at each iteration for all the methods, as shown in the picture below.

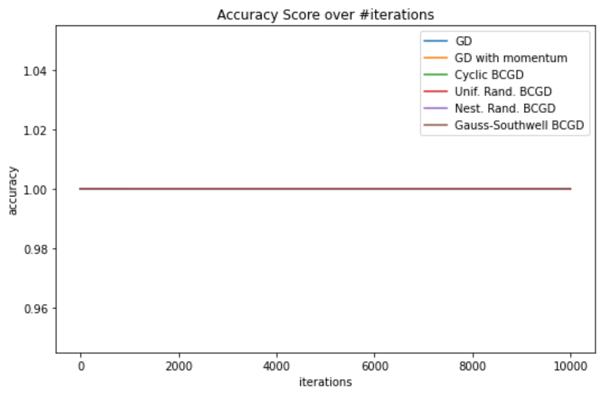

Figure 3: Accuracy vs Iterations using $y_{0,\,kNN}$

7. Census Dataset Analysis

The second dataset we used is a real dataset, the Census dataset.
Briefly, this dataset consists of 32561 instances (we imported the training set only), each one of them having 14 features, and we wish to predict whether the income exceeds 50K/yr based on census data. As before, we're dealing with a binary classification problem where the features

now are: *age, workclass, fnlwgt, education, education_num, marital_status, occupation, relationship, race, sex, capital-gain, capital-loss, hours-per-week, native-country*.
For a detailed explanation please read here.
Even though the original dataset consists of more than 30000 observations, for computational reasons we selected just the first 10000 of them, since running the algorithms on the entire dataset required very long execution times.
The dataset contains some missing values as well as some categorical variables, so before start working on it we had to deal with these two problems.
In our case we used the pandas library to easily remove the missing values (encoded with " ? ") and also to one-hot encode all the categorical variables.
In the end we moved from a dataset with dimensions (10000, 15) to a dataset with dimensions (9244, 104), containing no missing values and only numerical features.
On this dataset the maximum accuracy we were able to achieve is 75.8%. Again, using the randomly generated starting point y_0, the deterministic methods performed better then the randomized ones, as shown in Figure 4. Figure 5 instead shows the accuracy trend obtained using $y_{0,\,kNN}$ as starting point.

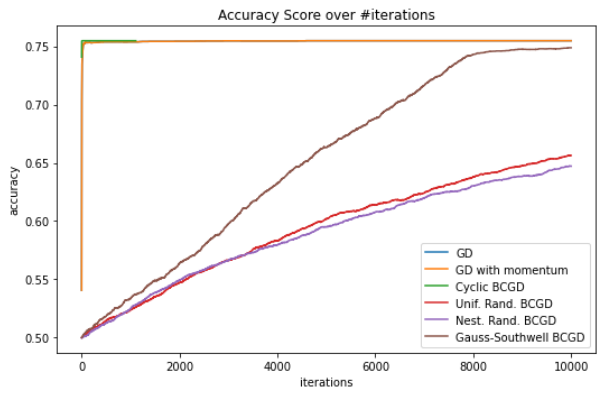

Figure 4: Accuracy vs Iterations using y_0

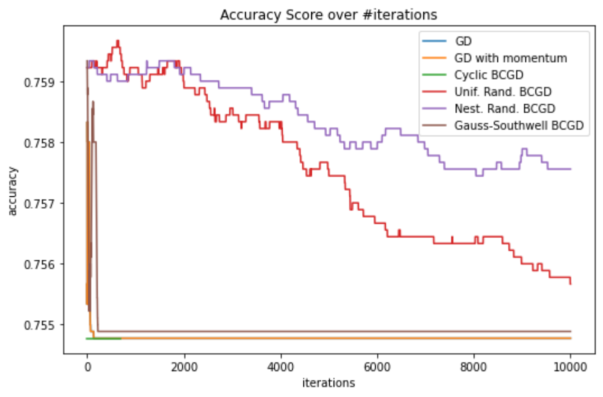

Figure 5: Accuracy vs Iterations using $y_{0,\,kNN}$

As we already discussed above, using the $y_{0,\,kNN}$ as starting point greatly improved the performances of the randomized methods, increasing the accuracy of about 10%.

8. Conclusions

In this homework we tested some of the main algorithms we saw in class on two datasets, both of them containing approximately 10000 observations.
What we observed is that the deterministic methods performed relatively better then the randomized ones, achieving greater accuracy scores in much fewer iterations as well as greater reduction in the norm of the gradient.
In particular, on the first dataset, using the randomly generated starting point the deterministic methods quickly achieved an accuracy score of 1, whereas the randomized methods required about double the number of iterations to reach the same accuracy levels.
On the second dataset on the other hand the situation was much more difficult and both deterministic methods as well as the randomized ones weren't able to surpass 75.8% of accuracy. Even in this case though, using the randomly generated starting point, deterministic methods performed better.
Interestingly enough, on both datasets, using the kNN-generated starting point greatly improved the performance of the randomized methods in terms of accuracy score, leading to the same accuracy levels of the deterministic approaches.
To conclude, we were able to achieve an accuracy of

- 100% on the first dataset;
- 75.8%. on the Census dataset.

For the second dataset in particular, it would be really interesting to see how the situation changes varying the dimensions of the blocks for BCGD methods, as well as applying some features selection techniques, however this is something we didn't try and consequently we won't discuss.