Programming Project 2: Coordinate Descent

Aman Raj, PID: A53247556

February 6, 2019

1 Coordinate Descent

In this project we propose variants of Coordinate Descent algorithms and use it to solve a logistic regression problem. We took UCI *wine dataset* and take samples belonging to class label 1 and 2 only, thus there are total 170 samples in our training dataset each having 13 features. Our coordinate descent is based on standard gradient descent but it optimizes the loss function by updating only one coordinate of parameter at each iteration. In particular, we minimize the following logistic regression loss function.

$$L(w) = \sum_{i=1}^{n} ln(1 + e^{-y^i(wx^i+b)}) \tag{1}$$

where (x^i, y^i) is data and label pair, w is parameter, b is bias. In order to get a baseline, we first trained our implementation of Gradient Descent on the data using above loss function and following parameter update rule:

$$w_{t+1} = w_t + \eta_t \sum_{i=1}^{n} y^i x^i Pr_{w_t}(-y^i|x^i) \tag{2}$$

where $Pr_{w_t}(-y^i|x^i)$ is probability of doubt, $y^i \epsilon \{1, 2\}$, and $x^i \epsilon\ \mathbb{R}^d$, and parameters $w_t \epsilon\ \mathbb{R}^d$, η_t is step size. We chose $\eta_t = 0.01$ for our experiments.

1.1 Which coordinate to choose?

We experimented with three different approaches to pick a coordinate of parameter and update it during an iteration. We obtained the best result with an exhaustive search based method.

1. **Random**: With uniformly random we pick a coordinate w_i to update where $i = \{1, 2, .., d\}$.
2. **Round**: We start with coordinate w_1 update it in current iteration, in next iteration we pick w_2 and so on. Once we pass w_d, we repeat and thus go round in circle.
3. **Exhaustive Search**: Here we randomly sample p number of coordinates out of d and pick the coordinate among samples which minimizes the loss function the most if updated. Since wine dataset has small number of features, so we set *p=d.*

We train all three methods with total *iterations=200*, along with Gradient Descent.

1.2 How to set new value of coordinate?

In our coordinate descent methods, the chosen coordinate of parameter is updated using the following update rule:

$$(w_{t+1})_j = (w_t)_j + \eta_t \sum_{i=1}^{n} y^i (x^i)_j Pr_{w_t}(-y^i|x^i) \tag{3}$$

Here we are updating $w_{j^{th}}$ coordinate.

2 Pseudocode For Exhaustive Coordinate Descent

Our best Coordinate Descent algorithm *CD Exhaustive* is explained as follows:

Step 1 Randomly sample p number of coordinates out of d coordinates of parameter w.

Step 2 Now for each coordinate in p do as follows independently: compute gradient, update the coordinate in parameter w to get new value of w, use the new value of w and calculate loss function. Now we have p number of losses at this step.

Step 3 Select the coordinate in p that gives lowest loss after its update. Hence, at each iteration we are adaptivly checking for which coordinate will give lowest loss and pick that.

Step 4 Repeat Step1 - Step3 till we reach reach convergence or complete the total number of iterations.

3 Experimental Results

We present a comparison of Gradient Descent algorithm along with three variant of proposed Coordinate Descent in Figure 1. As, we can observe from the graph the proposed Exhaustive Coordinate Descent performs best among all the methods and finally converges to loss of Gradient Descent.

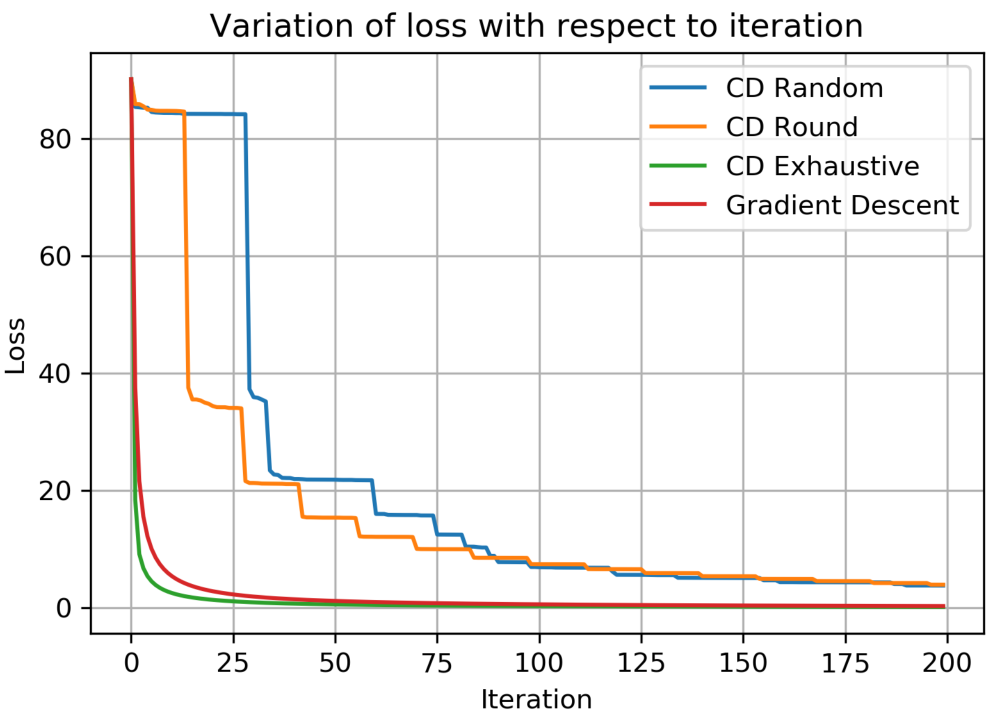

Figure 1: Comparison of Coordinate Descent methods with plain Gradient Descent without any regularizer.

4 Sparse Coordinate Descent

In order to get *k-sparse* solution for parameter w, we introduce an *L1* regularizer term $\lambda||w||$ in loss function of Equation 1 and accordingly modified the parameter update rule given in Equation 3 by introducing $\lambda\,\text{sign}(w_j)$. We found the best value of $\lambda = 2$. The value of k is taken as input and the algorithm chooses top k max parameters based on absolute values and rest is set to zero. In order to find which value of k gives lowest loss, we also plotted a graph. As Figure 2 shows k=*7* is best for wine dataset which means not all features are important. Also this method doesn't always find the best *k-sparse* solution as the loss start increasing with increase in k after some value.

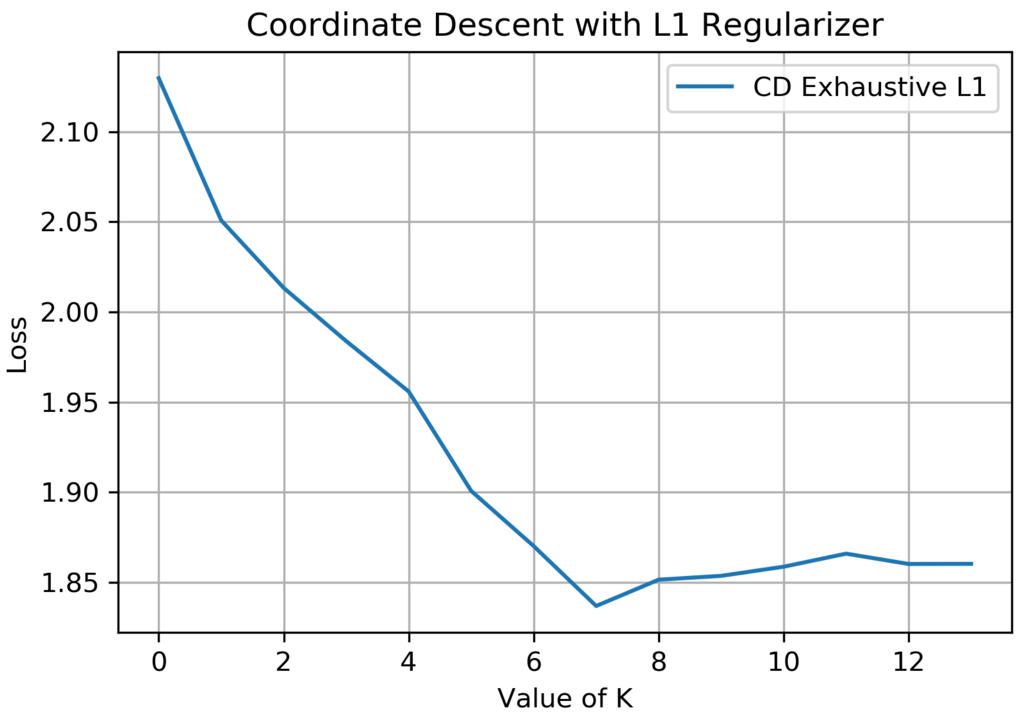

Figure 2: Varying K to get sparse solution for parameter w with L1 Regularizer introduced in Exhaustive Coordinate Descent.

5 Critical Evaluation

The Figure 1 shows, Exhaustive Coordinate Descent Method is adaptive in nature and very close to the performance of Gradient Descent and even gives faster convergence. Looking at graph there seems to have less scope for further improvement for the given wine dataset. This algorithm can be even applied in case when loss function is not differential everywhere. This algorithm in general works best for case when our loss function is convex.