ML Assignment 1

Yash Saxena 160823
Md. Masood Alam 16817394
Ashish Kumar Gupta 160161
Sachin Garg 16807598
Hitesh Gupta 160296

Preprint. Under review.

Part 1

1.1 Primal Proximal Gradient Descent with Acceleration

Proximal function for the LASSO
The proximal gradient method is fairly simple: at each iteration t=0,1,2....

$$x_{t+1} = prox_{\gamma_t g}(x_t - \gamma_t \nabla f(x_t)) = \underset{x}{\operatorname{argmin}} \{\frac{1}{2}\gamma_t \|x - [x_t - \gamma_t \nabla f(x_t)]\|_2^2 + g(x)\} \tag{1.1}$$

Note that :

- $f(x) = 0 : PG \Rightarrow proximal\ point\ algorithm$
- $g(x) = 0 : PG \Rightarrow gradient\ descent$
- $g(x) = \delta_x(x) : PG \Rightarrow projected\ gradient\ descent$

LASSO example of Proximal Gradient Method We provide an example to help demonstrate how this algorithm works on the LASSO problem:

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 + \lambda \|x\|_1 \tag{1.2}$$

First, we wish to calculate $\nabla f(x)$ and $prox_{\mu\|.\|_1}(x)$, where $f(x) = \frac{1}{2}\|Ax - b\|_2^2$. By inspection, we have:

$$\nabla f(x) = A^T(Ax - b) \tag{1.3}$$

$$prox_{\mu\|.\|_1}(x) = \underset{y}{\operatorname{argmin}} \{\frac{1}{2\mu}\sum_i (x_i - y_i)^2 + \sum_i \mid y_i \mid\} \tag{1.4}$$

We can separate this into each of the ith coordinates

$$[prox_{\mu\|.\|_1}(x)]_i = \underset{y_i}{\operatorname{argmin}} \{\frac{1}{2\mu}\sum_i (x_i - y_i)^2 + \sum_i \mid y_i \mid\} \tag{1.5}$$

The solution to this type of problem is well known as the soft thresholding operator.

$$S_\mu(x_i) = \begin{cases} x_i - \mu & \text{if } x_i > \mu \\ 0 & \text{if } \mid x_i \mid \leq \mu \\ x_i + \mu & \text{if } x_i < -\mu \end{cases} \tag{1.6}$$

For each coordinates, we have the solution in the form of this soft thresholding operator, so we can rewrite our LASSO problem in Proximal Gradient form as

$$x_{t+1} = S_{\lambda\gamma_t}(x_t - \gamma_t A^T(Ax_t - b)) \tag{1.7}$$

This is known as the Iterative Soft Thresholding Algorithm (ISTA).

1.2 Acceleration

Proximal Gradient with Backtracking Line-Search
Often times during our analysis, we set $\gamma_t = 1/L$, but in practice, we do not know L a priori, or it is difficult to solve for. For example, for the LASSO problem, $L = \lambda_{max}(A^T A)$,, but A may be large and difficult to work with. In practice we use backtracking line-search to find the local Lipschitz constant.
Backtracking Line Search for Lipschitz constant Here is how the line search works

- We initialise $L_0 = 1$ and some $\alpha > 1$.
- At each iteration t, we we find the smallest integer i such that $L = \alpha_i L_{t-1}$ satisfies the Lipschitz condition, specifically:

$$f(x^+) \leq f(x_t) + \nabla f(x_t)(x^+ - x_t) + \frac{L}{2}\left\|x^+ - x_t\right\|_2^2 \tag{1.8}$$

  where $x^+ = prox_{\frac{1}{L}}(x_t - \frac{1}{L}\nabla f(x_t))$.
  Then update $L_t = L$ and $x_{t+1} = x^+$

Optimizing LASSO Objective one Co-ordinate at a time

Here, $L(w)$ represents the **LASSO** cost function

$$L(w) = RSS(w) + \lambda\|w\|_1 = \sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{D} w_j h_j(x_i)\right)^2 + \lambda\sum_{j=0}^{D}\|w_j\| \tag{1.9}$$

1.2.1 Partial Derivative of $RSS(w)$

$$\begin{aligned}\frac{\partial RSS(w)}{\partial w_j} &= -2\sum_{i=1}^{N} h_j(x_i)\left(y_i - \sum_{j=0}^{D} w_j h_j(x_i)\right)\\ &= -2\sum_{i=1}^{N} h_j(x_i)\left(y_i - \sum_{k\neq j} w_k h_k(x_i) - w_j h_j(x_i)\right)\\ &= -2\sum_{i=1}^{N} h_j(x_i)\left(y_i - \sum_{k\neq j} w_k h_k(x_i)\right) + 2w_j\sum_{i=1}^{N} h_j(x_i)^2\end{aligned} \tag{1.10}$$

$$\begin{aligned}Let\ \rho_j = \sum h_j(x_i)\left(y_i - \sum_{k\neq j} w_k h_k(x_i)\right)\\ z_j = \sum_{i=1}^{N} h_j(x_i)^2\\ \implies \frac{\partial RSS(w)}{w_j} = -2\rho_j + 2w_j z_j\end{aligned} \tag{1.11}$$

1.2.2 Partial Derivative of $L1$ Penalty

Instead of taking gradient, we can think about the sub gradient (as the function is not differentiable. We know that $g(b) \geq g(a) + \nabla g(a)(b-a)$ for gradients. Hence, equivalently, we have $V \in \partial g(x)$ if $g(b) \geq g(a) + V(b-a)$. Here, $\partial g(x)$ represents the sub gradient of g at x.

Subgradient of L1 Term

$$\lambda \partial_{w_j} |w_j| = \begin{cases} -\lambda & \text{if } w_j < 0 \\ [-\lambda, \lambda] & \text{if } w_j = 0 \\ \lambda & \text{if } w_j > 0 \end{cases} \tag{1.12}$$

Summing Up

$$\partial_{w_j}[L(w)] = 2z_j w_j - 2\rho_j + \begin{cases} -\lambda & \text{if } w_j < 0 \\ [-\lambda, \lambda] & \text{if } w_j = 0 \\ \lambda & \text{if } w_j > 0 \end{cases} \tag{1.13}$$

Finally, we have

$$\partial_{w_j}[L(w)] = \begin{cases} 2z_j w_j - 2\rho_j - \lambda & \text{if } w_j < 0 \\ [-2\rho_j - \lambda, -2\rho_j + \lambda] & \text{if } w_j = 0 \\ 2z_j w_j - 2\rho_j + \lambda & \text{if } w_j > 0 \end{cases} \tag{1.14}$$

Remark : For optimal Solution, set subgradient to zero

- **Case I** $w_j < 0$

$$2z_j w_j - 2\rho_j - \lambda = 0 \tag{1.15}$$

$$\hat{w_j} = \frac{2\rho_j + \lambda}{2z_j} = \frac{\rho_j + \lambda/2}{z_j} \tag{1.16}$$

  Hence, for this case we have $\rho_j < \frac{\lambda}{2}$

- **Case II** $w_j = 0$

$$\text{For } \hat{w_j} = 0, [-2\rho_j - \lambda, -2\rho_j + \lambda] \text{ contains } 0 \tag{1.17}$$

$$-2\rho_j - \lambda \geq 0 \implies \rho_j \geq -\lambda/2 \tag{1.18}$$

$$-2\rho_j + \lambda \geq 0 \implies \rho_j \leq \lambda/2 \tag{1.19}$$

$$\implies -\lambda/2 \leq \rho_j \leq \lambda/2 \tag{1.20}$$

- **Case III** $w_j > 0$

$$2z_j \hat{w_j} - 2\rho_j + \lambda = 0 \tag{1.21}$$

$$\hat{w_j} = \frac{\rho_j - \lambda/2}{z_j} \tag{1.22}$$

  Hence, for this case we have $\rho_j > \frac{\lambda}{2}$

1.3 Optimal Solution

$$\hat{w_j} = \begin{cases} \frac{(\rho_j + \lambda/2)}{z_j} & \text{if } \rho_j < -\lambda/2 \\ 0 & \text{if } \rho_j \in [-\lambda/2, \lambda/2] \\ \frac{(\rho_j - \lambda/2)}{z_j} & \text{if } \rho_j > -\lambda/2 \end{cases} \tag{1.23}$$

Part 2

2.1 Primal Stochastic Coordinate Descent

1. **Hyperparameter tuning:** No hyperparameters involved.
2. **Validation:** No validation since no hyperparameters involved

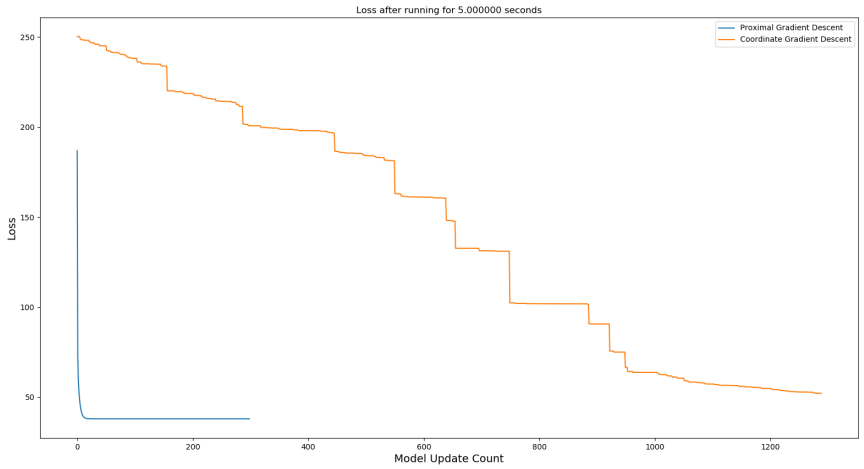
2.2 Primal Proximal Gradient Descent

1. **Hyperparameter tuning:** α is the hyperparameter. It should be greater than 0. We varied our hyperparameter and found divergence at $\alpha = 4$. Best results were found at $\alpha = 3$.
2. **Validation:** We performed **Held Out Validation** using 80-20 split and concluded that $\alpha = 3$ works best.

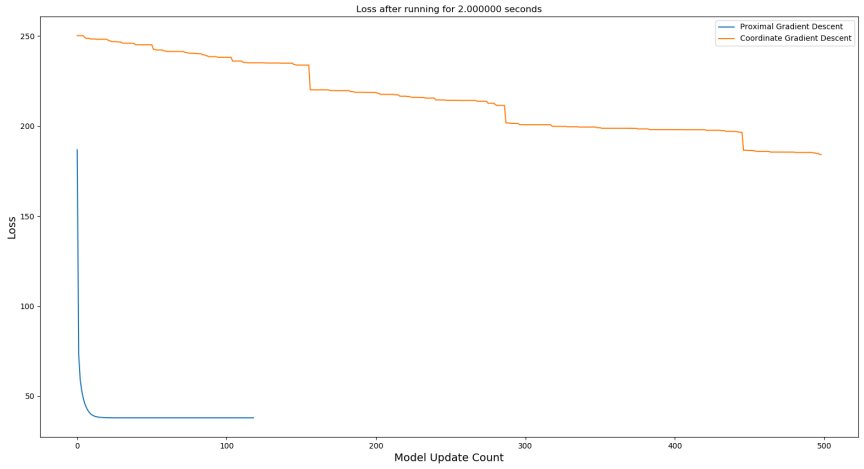
Part 3

Convergence Curves:

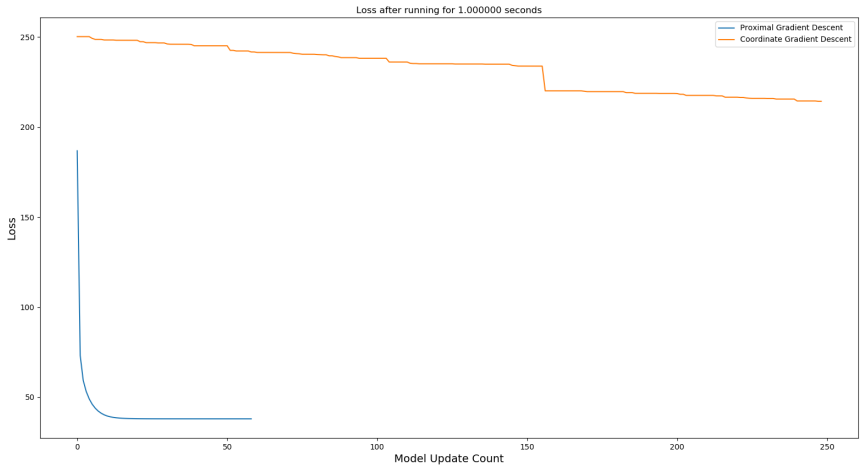
3.1 Loss after running for 5.0 seconds



3.2 Loss after running for 2.0 seconds



3.3 Loss after running for 1.0 seconds



Clearly, Primal Proximal Gradient Descent performs the best as it converges faster, as observed in the above graphs.

Part 4

Best method in terms of reducing the objective value as well as recovering w^* is Primal Proximal Gradient Descent.

References and Sources

- IE 598: Big Data Optimization, Lecture 19: Proximal Gradient Method and Its Acceleration, Lecturer: Niao He
- Deriving the lasso coordinate descent update