Sparse Logistic Regression with Majorization

Shouren Wang
sw5004@nyu.edu
M.S. Computer Engineering Program
Tandon School of Engineering, New York University

1 Introduction

Logistic regressing is one of the most classical and fundamental classification machine learning algorithms. In this report, I mainly discuss the sparse logistic regression with L1 norm. L1 norm is a penalty that can be applied to the objective function, and it can make the model sparse. Another classic application of L1 norm is lasso regression. But L1 norm is not everywhere differentiable, thus we cannot directly use its gradient to optimize the object function. In the following part, I propose the sub-gradient method to solve this problem.

Majorization provides an approximate approach to solve $\boldsymbol{\theta}^*$, If the cost function $\boldsymbol{\theta}^* = \arg min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ has no closed form solution. It uses a surrogate Q with closed form update to monotonically minimize the cost from an initial $\boldsymbol{\theta}_0$, which means solving an upper bound of cost function instead.

In this report, I explore the sparse logistic regression problems solved by majorization methods. In section 2, I do a brief review of logistic regression and apply L1 norm to the log-likelihood object function. In section 3, I discussed the majorization method and how to apply it to optimize sparse logistic regression problem. In section 4, the majorization method, the modified gradient descent (GD) method with L1 norm and GD with L2 norm are compared. In section 4, I propose the low-rank potential future directions.

2 Sparse Logistic Regression

2.1 Logistic Regression

A traditional hard linear classifier can be described by the following equations:

$$z = \theta^T x \tag{1}$$

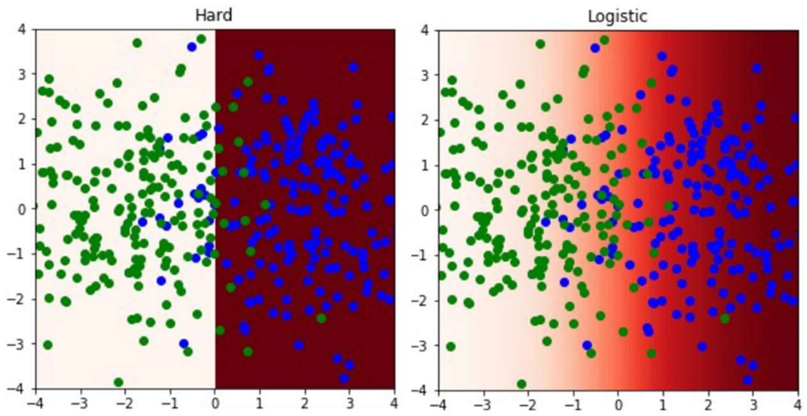
$$\hat{y} = \begin{cases} 1 & z > 0 \\ -1 & z < 0 \end{cases} \tag{2}$$

Logisitic decision classifier can be described by the following equations:

$$z = \theta^T x \tag{3}$$

$$P(y = 1|x) = \frac{1}{1 + e^{-z}} \tag{4}$$

The logistic classifier can generate a soft decision shown below:



Logistic regression can also be easily extended to multiple classes. In (4), the probability is predicted via the sigmoid function. For multi-class regression, the probabilities can be predicted via the softmax function:

$$P(y = k|x) = \frac{e^{z_k}}{\sum_{i=1}^{K} e^{z_i}} \tag{5}$$

Comparing with linear regression and some other machine learning methods, the key idea of fitting logistic regression is not optimizing a loss function, but the maximum likelihood principle. The likelihood function can be described as:

$$P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \textit{Probability of class labels given input } \boldsymbol{X} \textit{ and weights } \boldsymbol{\theta} \tag{6}$$

And the key idea of maximum likelihood is to make $P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$ higher, which means find the optimal $\boldsymbol{\theta}^*$ to maximize $P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$. Then the maximum likelihood estimation can be described as:

$$P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \prod_{i=1}^{N} P(y_i|\boldsymbol{x_i}, \boldsymbol{\theta}) \tag{7}$$

Define log likelihood:

$$J(\boldsymbol{\theta}) = \ln P(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \sum_{i=1}^{N} \ln P(y_i|\boldsymbol{x_i}, \boldsymbol{\theta}) \tag{8}$$

Find the optimal $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \arg max_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \tag{9}$$

And this method is also equivalent to minimize the logistic loss, for example, binary cross entropy loss in binary classification.

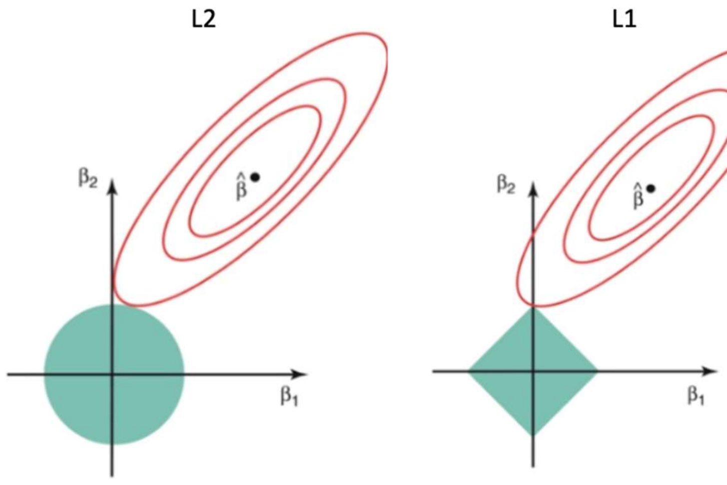
2.1 Sparse Logistic Regression with L1 Norm

The L1 norm is a penalty scaler applied to the object function. In many cases, there are too many data features, but only a small part of them contribute to the classification. The redundant features may cause some problems like overfitting. The L1 norm is an effective method to reduce the redundant features, thus leading to a sparse model. Now we apply the L1 norm to the objective function in (8), and it becomes:

$$J_{l1}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \ln P(y_i|\boldsymbol{x_i}, \boldsymbol{\theta}) - \lambda\|\boldsymbol{\theta}\| \tag{10}$$

Where λ is a scaler can be set before training, and the large λ is, more sparse the model should be.

Compared to L1 norm, L2 norm is also often applied to object functions. Because L2 is continuously differential, the object functions with L2 norm can be solved by some optimizers like gradient descent directly. L2 norm leads to a model that is small in magnitude but not sparse. The following figure shows the difference between L1 and L2 norm, and intuitively shows why L1 norm leads to a sparse model.



3 Majorization

3.1 Bounded Partition Function

The log-linear partition function $Z(\boldsymbol{\theta})$ is defined as:

$$P(y|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} h(y) e^{\boldsymbol{\theta}^T f(y)} \tag{11}$$

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{y}} h(y) e^{\boldsymbol{\theta}^T f(y)} \tag{12}$$

Where $y \in \Omega$, $|\Omega| = n$, $\boldsymbol{\theta} \in \mathbb{R}^d$, $Z(\boldsymbol{\theta}) \in \mathbb{R}^d$, $h(y)$ is a scaler function which denotes the mapping $h: \Omega \to \mathbb{R}^+$and $\boldsymbol{f}: \Omega \to \mathbb{R}^d$. We adopted the analogous quadratic upper-bound on the partition function that was proposed by Choromanska and Jebara in 2012, and then get a quadratic upper-bound of $Z(\boldsymbol{\theta})$ in (12):

$$Z(\boldsymbol{\theta}) \leq z e^{(\frac{1}{2}(\boldsymbol{\theta}-\widetilde{\boldsymbol{\theta}})^T \boldsymbol{\Sigma}(\boldsymbol{\theta}-\widetilde{\boldsymbol{\theta}})+(\boldsymbol{\theta}-\widetilde{\boldsymbol{\theta}})^T \boldsymbol{\mu})} \tag{13}$$

Where z, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can be computed by the following algorithm:

```
Algorithm 1 ComputeBound
Input Parameters $\tilde{\boldsymbol{\theta}}, \mathbf{f}(y), h(y)\ \forall y \in \Omega$
Init $z \to 0^+, \boldsymbol{\mu} = \mathbf{0}, \boldsymbol{\Sigma} = z\mathbf{I}$
For each $y \in \Omega$ {
        $\alpha = h(y)\exp(\tilde{\boldsymbol{\theta}}^\top \mathbf{f}(y))$
        $\mathbf{l} = \mathbf{f}(y) - \boldsymbol{\mu}$
        $\boldsymbol{\Sigma} += \frac{\tanh(\frac{1}{2}\log(\alpha/z))}{2\log(\alpha/z)}\mathbf{l}\mathbf{l}^\top$
        $\boldsymbol{\mu} += \frac{\alpha}{z+\alpha}\mathbf{l}$
        $z += \alpha$   }
Output $z, \boldsymbol{\mu}, \boldsymbol{\Sigma}$
```

Algorithm 1 yields the second-order Taylor approximation (the Hessian) of the log-partition function. From (13) we can find that the upper bound of $Z(\boldsymbol{\theta})$ is an exponential quadratic function. Then iteratively at a fixed $\widetilde{\boldsymbol{\theta}}$, if z, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are computed, we don't have to do $\arg min_{\boldsymbol{\theta}} Z(\boldsymbol{\theta})$, instead we can update $\widetilde{\boldsymbol{\theta}}$ as:

$$\widetilde{\boldsymbol{\theta}} \leftarrow \widetilde{\boldsymbol{\theta}} - \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} \tag{14}$$

3.2 Bounded Log-likelihood with L1 Norm

Now we apply the majorization method to sparse logistic regression. The distribution over all y can be defined as:

$$P(y|x_j, \boldsymbol{\theta}) = \frac{1}{Z_{x_j}(\boldsymbol{\theta})} e^{\boldsymbol{\theta}^T \boldsymbol{f}_{x_j}(y)} \tag{15}$$

$$Z(\boldsymbol{\theta}) = \sum\nolimits_{y \in \Omega} e^{\boldsymbol{\theta}^T \boldsymbol{f}_{x_j}(y)} \tag{16}$$

Where $y_j \in \Omega = -1, 1$, and $\boldsymbol{f}_{x_j}(y) = y x_j$.

By applying the method in (8) we can get a log-likelihood function, then by applying the L1 norm, we can get the following sparse objective function:

$$J(\boldsymbol{\theta}) = \sum\nolimits_{j=1}^{N} [-\ln Z_{x_j}(\boldsymbol{\theta}) + \boldsymbol{\theta}^T \boldsymbol{f}_{x_j}(y)] - \lambda \|\boldsymbol{\theta}\| \tag{17}$$

According to (13), we can get the lower bound of $J(\boldsymbol{\theta})$ in (17):

$$J(\boldsymbol{\theta}) \geq J_{bounded}(\boldsymbol{\theta})$$
$$= \sum_{j=1}^{N} [-\ln z_j - \frac{1}{2}(\boldsymbol{\theta}-\widetilde{\boldsymbol{\theta}})^T \boldsymbol{\Sigma}_j (\boldsymbol{\theta}-\widetilde{\boldsymbol{\theta}}) - (\boldsymbol{\theta}-\widetilde{\boldsymbol{\theta}})^T \boldsymbol{\mu}_j + \boldsymbol{\theta}^T \boldsymbol{f}_{x_j}(y)] - \lambda\|\boldsymbol{\theta}\| \qquad (18)$$

Where $\boldsymbol{\mu} = \sum_{j=1}^{N} \boldsymbol{\mu}_j$, $\boldsymbol{\Sigma} = \sum_{j=1}^{N} \boldsymbol{\Sigma}_j$ and $\boldsymbol{f} = \sum_{j=1}^{N} \boldsymbol{f}_{x_j}$. Now, instead of maximizing $J(\boldsymbol{\theta})$ in (17), we can directly maximize $J_{bounded}(\boldsymbol{\theta})$ in (18), which is the lower bound of $J(\boldsymbol{\theta})$.

But there is a problem in maximizing $J_{bounded}(\boldsymbol{\theta})$ in (18) which is that L1 norm is not continuously differentiable everywhere, thus $J_{bounded}(\boldsymbol{\theta})$ is also not continuously differentiable everywhere. To solve this problem, we can use subgradient instead of gradient. Define:

$$\widetilde{\nabla} J_{bounded}(\boldsymbol{\theta}) = \begin{cases} \nabla j(\boldsymbol{\theta}) + \lambda, when\ \boldsymbol{\theta} > 0\ or\ \boldsymbol{\theta} = 0^+ \\ \nabla j(\boldsymbol{\theta}) - \lambda, when\ \boldsymbol{\theta} > 0\ or\ \boldsymbol{\theta} = 0^- \end{cases} \qquad (19)$$

Where,

$$j(\boldsymbol{\theta}) = -\frac{1}{2}(\boldsymbol{\theta}-\widetilde{\boldsymbol{\theta}})^T \boldsymbol{\Sigma} (\boldsymbol{\theta}-\widetilde{\boldsymbol{\theta}}) - (\boldsymbol{\theta}-\widetilde{\boldsymbol{\theta}})^T \boldsymbol{\mu} + \boldsymbol{\theta}^T \boldsymbol{f}(y) \qquad (20)$$

For the terms $\boldsymbol{\theta}^T \boldsymbol{f}_{x_j}(y)$ and $\lambda\|\boldsymbol{\theta}\|$, their second ordered partial derivatives, the Hessian are O matrices. Then we can get the Hessian matrix $\boldsymbol{\Sigma}$ of $J_{bounded}(\boldsymbol{\theta})$, then we can apply Newton method to update $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - \eta \boldsymbol{\Sigma}^{-1} \widetilde{\nabla} J_{bounded_k} \qquad (21)$$

Where η is the learning rate. We can also apply gradient descent methods like:

$$\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - \eta \widetilde{\nabla} J_{bounded_k} \qquad (22)$$

But comparing with (21) method in (22) is nor preferred in this study. The reason is that the performance is relatively bad, specifically, it is slow.

The algorithm can be described as:

```
Algorithm 2
Input: $x_j, y_j, \boldsymbol{f}_{x_j}(y), \lambda$
Output: $\theta = \tilde{\theta}$
Init: $\tilde{\theta}$
 while not converged do
    for $j = 1, ..., t$ do
       $\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j \leftarrow$ Algorithm1
    end
    Compute $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ for $k = 1, ...,$
       get $\boldsymbol{\Sigma}, \widetilde{\nabla} J_{bounded_k}$ from (18) and (19), then $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k - \eta \boldsymbol{\Sigma}^{-1} \widetilde{\nabla} J_{bounded_k}$
    end
    $\tilde{\theta} \leftarrow \theta^{k+1}$
end
```

4 Experiments

4.1 Parameters

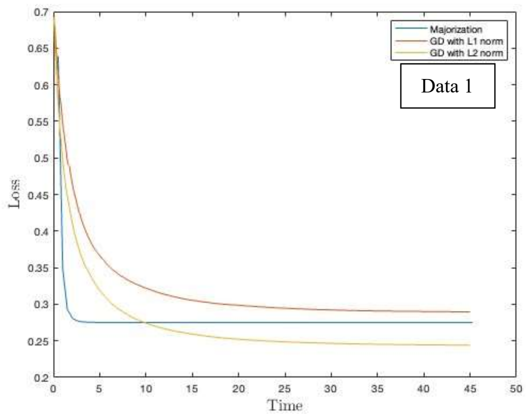
The information of hyperparameter and dataset are listed as following:

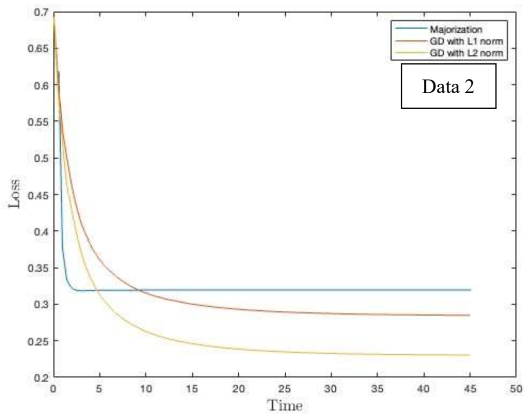
	Dataset 1&2	Dataset 3	Dataset 4
λ	0.01	0.1	0.01
leanring rate	0.01	0.001	0.01
threshold	0.001	0.001	0.001
samples	1000	4600	1500
features	200	57	784

For comparison, I also ran the modified gradient descent (GD) method (with sub-gradient for L1 norm) with L1 norm and GD method with L2 norm, and BFGS with L1 norm. I also ran the majorization optimized by GD in (22) on dataset 4, but it was only for comparison. All the methods above used the same set of hyperparameters.

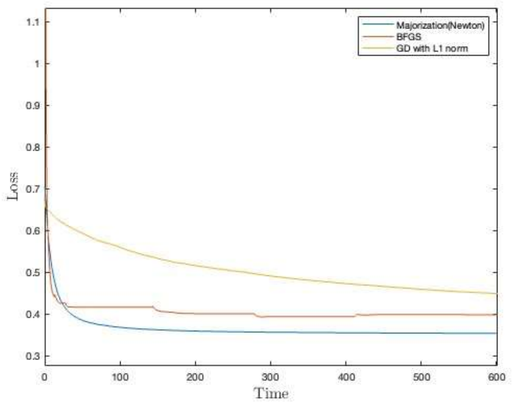
4.2 Loss and Predict Performance

The following figures show the training loss (logistic loss, which is the opposite of log-likelihood) of the majorization method, the modified GD method with L1 norm and GD method with L2 norm on dataset1 and dataset2. We can find that the majorization method not only converges faster, but also on some datasets even shows better convergence than the modified GD method. We can conclude that optimizing the bounds does not significantly affect the optimization process causing it to lose precision.

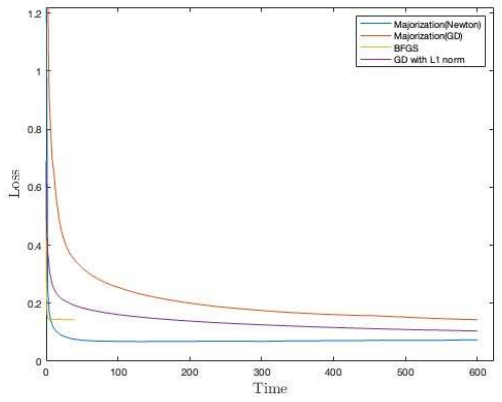




The following figures show the training loss on dataset 3. We can find that although majorization method converges slower than BFGS, but still faster than GD and performs better.



The following figures show the training loss on dataset 4. We can find that majorization with GD update method (orange line) converges very slow. Presumably the reason is that at each iteration the algorithm does both bounds calculation and gradient descent. This is the reason why I do not prefer method in (22).

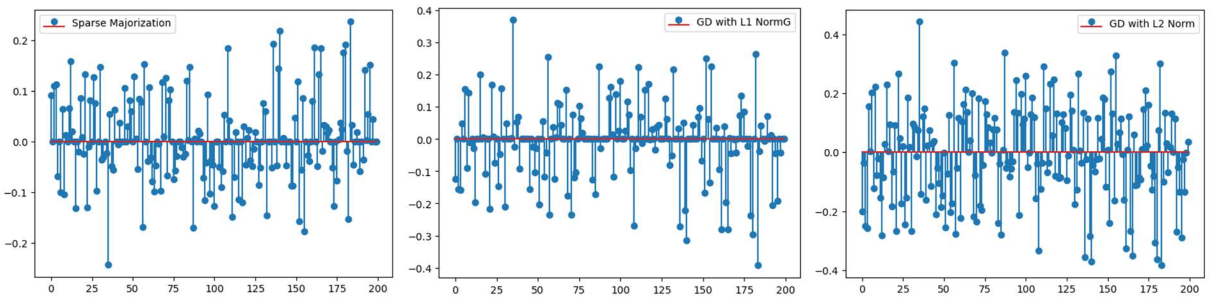


The following table shows the performance of different methods. We can find that although majorization optimizes the bounded object function, which is an approximated approach, the accuracy is still almost as good as non-approximated methods. Besides, the majorization method provides better convergence. Thus, we can conclude that the majorization method has achieved effective performance to optimize sparse logistic regression problem.

Dataset 1		Dataset 2	
Method	Accuracy	Method	Accuracy
Majorization	87.00%	Majorization	85.70%
GD, L1 norm	86.80%	GD, L1 norm	87.80%
GD, L2 norm	88.33%	GD, L2 norm	88.40%
Dataset 3		Dataset 4	
Method	Accuracy	Method	Accuracy
Majorization	80.70%	Majorization (Newton)	82.00%
GD, L1 norm	77.92%	Majorization (GD)	77.60%
GD, L2 norm	77.96%	GD, L1 norm	81.20%
BFGS	79.28%	GD, L2 norm	82.00%
		BFGS	76.80%

4.3 Sparsity Result

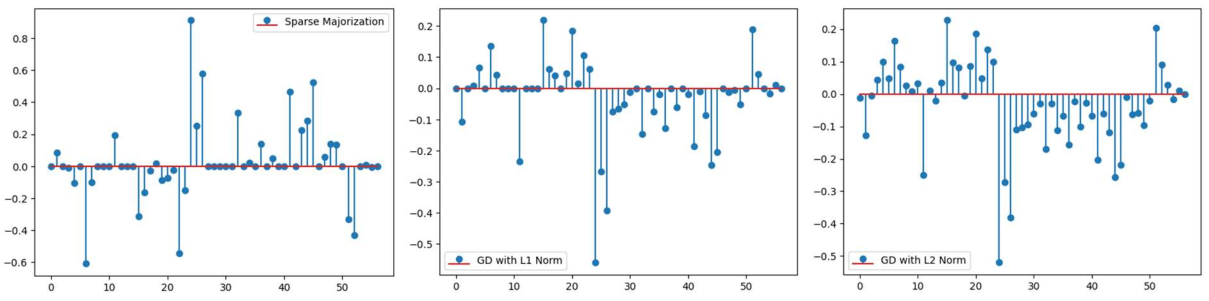
The following figures show the sparsity result on dataset 1. The stem plots show the trained $\boldsymbol{\theta}$ of the proposed sparse logistic regression optimized by majorization method, sparse logistic regression optimized by GD method and ordinary logistic regression with L2 norm optimized by GD respectively. We can find that the $\boldsymbol{\theta}$ of the sparse logistic regression model with L1 norm is sparser than the $\boldsymbol{\theta}$ of the logistic regression with L2 norm.



The following table shows the number of 0 elements of $\boldsymbol{\theta}$s mentioned above on dataset 1 (the larger the sparser):

Number of 0 of 200 elements in $\boldsymbol{\theta}$		
Sparse Logistic Regression (L1 norm, Majorization)	Sparse Logistic Regression (L1 norm, GD)	Logistic Regression (L2 norm, GD)
46	61	2

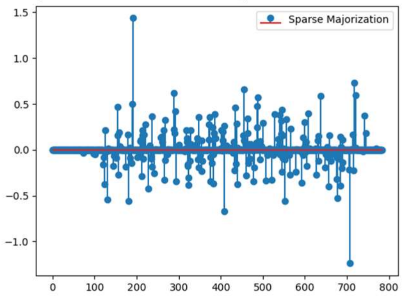
The following figures show the sparsity result on dataset 3.

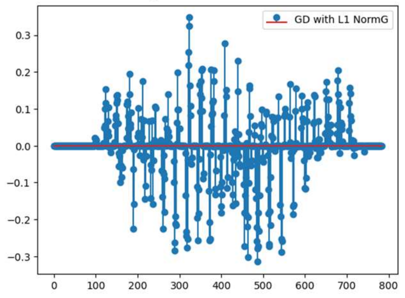


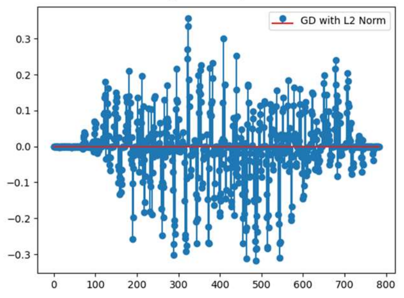
The following table shows the number of 0 elements of $\boldsymbol{\theta}$s mentioned above on dataset 3 (the larger the sparser):

Number of 0 of 57 elements in $\boldsymbol{\theta}$		
Sparse Logistic Regression (L1 norm, Majorization)	Sparse Logistic Regression (L1 norm, GD)	Logistic Regression (L2 norm, GD)
23	18	1

The following figures show the sparsity result on dataset 4, but the dataset has too many features may lead the stem plot hard to read but can provide an intuitive view of the sparsity.







The following table shows the number of 0 elements of $\boldsymbol{\theta}$s mentioned above on dataset 4 (the more 0 elements, the sparser the model is):

Number of 0 of 784 elements in $\boldsymbol{\theta}$		
Sparse Logistic Regression (L1 norm, Majorization)	Sparse Logistic Regression (L1 norm, GD)	Logistic Regression (L2 norm, GD)
428	537	186

From the above data, we can find that the model applied L1 norm can be much sparser than the model applied L2 norm.

5 Possible Future Improvements

5.1 Low Rank Bounds for large size $\boldsymbol{\Sigma}$

If the number of features of the dataset is very large, it may take both very large memory to store the Hessian and take a very long time to compute the inverse Hessian ($O(d^3)$). To solve this problem, the method of L-BFGS and Woodbury Formula can be adopted. The method projects each rank 1 update of $\boldsymbol{\Sigma}$ to a $\boldsymbol{V}$ in the form of $\boldsymbol{\Sigma} = \boldsymbol{V}^T\boldsymbol{S}\boldsymbol{V} + \boldsymbol{D}$, and the time complexity can be reduce to $O(tnd + k^3)$, where $\boldsymbol{S} \in \mathbb{R}^{k\times k}$, $\boldsymbol{V} \in \mathbb{R}^{k\times d}$, and k is much smaller than d.

5.2 Explore more methods to optimize the bound

Although the GD method does not work well for optimizing the bound, some other methods can be further explored. My assumption is that since the majorization method contains the procedure to compute the bound in each iteration, it should be better to apply methods with a quadratic convergence rate to optimize the bound, otherwise it may be too slow to converge, like GD.

References

[1] T.JebaraandA. Choromanska,“Majorizationforcrfsandlatentlikelihoods”,AdvancesinNeural Information Processing Systems, vol. 1, pp. 565b–574, 01 2012.

[2] P. Gong and J. Ye, “A modified orthant-wise limited memory quasi-newton method with convergence analysis”, ICML, 2015.

[3] Sundeep Rangan, “Introduction to Machine Learning”. 2020

[4] Su-In Lee, Honglak Lee, Pieter Abbeel and Andrew Y. Ng, “Efficient L1 Regularized Logistic Regression”, Stanford, CA 94305