HomeWork 4

Puxin Xu

1. Compare the testing accuracy with that of the solutions obtain by l2 regularized regression and by SVM

l^2 regularized regression:

Lamda	01_loss	hinge_loss
0.01	0.012242627	0.036506195
0.1	0.010016694	0.025387141
1	0.013912076	0.023543015
10	0.076238175	0.116609761

SVM:

Mosek		
Linear		
C	01_loss	hinge_loss
0.01	0.010573178	0.020183827
0.1	0.014468559	0.045595318
1	0.021146355	0.10685606
10	0.020033389	0.143022965

First order methods for Sparse Logistic Regression:

Lambda	01_loss	hinge_loss
0.001	0.013912076	0.099192458
0.01	0.008903728	0.022192082
0.05	0.017250974	0.033864297
0.1	0.028380634	0.047709489
0.5	0.101279911	0.170908379

As it shows above, the result is reasonable, because the accuracy is similar with what I obtained using SVM and l2 regularized regression. Actually, with lambda = 0.01, the testing accuracy is even better than SVM and l2 regularized regression.

2. Try several values of lambda and compare efficiency of accelerated and regular methods.

Based on the additional condition that the iteration times are no more than 2000.

lambda	0.001	0.01	0.05	0.1	0.5
iteration of regular methods	2000	2000	2000	2000	2000
time of regular methods	55.8948134	54.9321175	54.4685548	52.7626235	62.9746017
iteration of accelerated methods	1508	1318	151	165	64
time of accelerated methods(s)	65.7354239	51.010462	7.9953247	7.84360528	2.34430118

In this case, accelerated method is more efficient because of less iteration and time.

3. Test the efficiency of both proximal gradient and accelerated algorithms for different step size strategies.

Regular methods-never increase μ between iteration					
lambda	0.001	0.01	0.05	0.1	0.5
number of function	2010	2010	2010	2010	2009
number of gradient	2000	2000	2000	2000	2000
Regular methods- increase μ by a factor of two at the beginning of each iteration					
lambda	0.001	0.01	0.05	0.1	0.5
number of function	4001	4003	4004	1819	203
number of gradient	2000	2000	2000	907	99
Accelerated gradient descent tested on digits					
lambda	0.001	0.01	0.05	0.1	0.5
number of function	1508	1318	151	165	64
number of gradient	1495	1308	145	158	56

Note : Thank Liyuan Cao for teaching me the sub-gradient when w = 0.

And in my coding, I used |sign(w)| as a switch to avoid a loop for calculating the gradient.