Homework5 Report for Algorithms for Big-Data Analysis

Haiwen Huang 1500010657

April 25, 2018

1 Introduction

In this homework, I implemented three algorithms to solve the problem. The first two algorithms are AdaGrad and Adam, and the last is proposed by Chang Wang and me. We name our algorithm (temporarily) as StabAda, and test it here. More analysis and experiments of it will be shown in our final project. (Runyu Zhang is a member of our group, with whom we have productive discussion).

The problem of our homework is

$$\min_{w \in R^d} \frac{1}{n} \sum_{i=1}^{n} f_i(w) + \lambda ||w||_1, \tag{1}$$

where $f_i(w) = log(1 + exp(-y^i w^T x^i))$ and $\lambda > 0$.

I use two data sets: MNIST and Covertype. And the labels of MNIST dataset has been modified into binary label to classify digits into even and odd. We will briefly review the information of the data sets again in next section. We test MNIST in `main1.m` and Covertype in `main2.m`. For usage, you can directly run them, the code will download the data set for you automatically.

2 Data set description

The data in MNIST are 28*28 images and labels are integers from 0 to 9. We modify the label to be -1 and 1 so that even numbers become 1 and odd become -1. The data set has already divided training set and test set.

The data in Covertype has 581012 observations and 54 columns of measures. The data file is 581012*55, with the last column being the label (cover type of forests). We divide the data set into two parts: the first 400000 observations become the training set (about 70% as in the paper) and the remaining the test set.

3 Algorithms

3.1 Proximal Gradient Method

Since $||x||_1$ is non-differentiable, we use proximal gradient method to formulate our optimization.

Since our problem is formulated as 1, we can easily get its proximal update:

$$x_{k+1} = prox_{\tau\lambda||x||_1}(x_k - \tau\nabla f(x_k)). \tag{2}$$

Here τ is the step size, and can be calculated by different algorithms in the next subsection. And since the proximal function of $||x||_1$ is just soft-thresholding, the update is just

$$x_{k+1} = shrinkage(x_k - \tau\nabla f(x_k), \tau\lambda). \tag{3}$$

3.2 Algorithms for step size

We implemented three algorithms: AdaGrad, Adam and StabAda and only talked about their usual form here, the proximal form can be seen in corresponding MATLAB codes. Their main differences lie in two parts: step direction and step size (here the step direction estimate will replace∇f in 2 and step size estimate will replace τ in 2). In their essence, AdaGrad uses $H_k = \frac{1}{\alpha}diag(\sum_{i=1}^{k} g_i. * g_i)^{\frac{1}{2}}$ to (in a way) approximate Hessian matrix in Newton method, and this defines a new variable metric, which leads to the update rule

$$x_{k+1} = \underset{X\in C}{\operatorname{argmin}} \{< g_k, x >, \frac{1}{2} < x - x_k, H_k(x - x_k) >\},$$

which is equivalent to

$$x_{k+1} = x_k - \alpha \cdot diag(\sum_{i=1}^{k} g_i. * g_i)^{-\frac{1}{2}} g_k.$$

Adam uses another method called exponential moving average to adaptively changing the step size. It uses exponential decay rather than adding all of them to utilize the past gradients, thus "possibly" avoids the gradients decreasing too quickly. Adam also uses momentum to estimate the step direction. Its update rule can be summarized as

$$x_{k+1} = x_k - \frac{\alpha}{v_t} \cdot m_t,$$

where $m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t$ and $v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$. Here g_t^2 means element-wise multiplication.

As for our method StabAda, we only introduce the update rule here. The details (analysis, convergence proof and more experiments) can be seen in our final project report. The update rule is the same as Adam, except that β_2 changes with t. So the difference is $v_t = \beta_{2,t} v_{t-1} + (1 - \beta_{2,t})g_t^2$, and $\beta_{2,t} = \frac{\sum_{i=1}^{t-1} i^{-\gamma}}{\sum_{i=1}^{t} i^{-\gamma}}$. Here γ is another hyperparameter.

Algorithm	Iteration times	$\|\|f_k - f_{k-1}\|\|$
AdaGrad	398	1e-5
Adam	3387	1e-5
StabAda	2017	1e-5

Table 1: MNIST, $\lambda = 0.001$

4 Numerical Results

We compare the performance of different optimization algorithms using iteration times when achieving the same error. We use $||f_k - f_{k-1}|| < \epsilon$ as our stopping rule. Therefore we can compare the convergence speed of different methods in different λ settings. Another interesting thing is how the oscillation of different methods can differ. In practice, we set ϵ as $1e-5$ for MNIST ,and $1e-4$ for Covertype data set.

We test different λ's on different data set. The detail of the results can be seen in tables and figures.

Algorithm	Iteration times	$\|\|f_k - f_{k-1}\|\|$
AdaGrad	8233	1e-5
Adam	6336	1e-5
StabAda	3507	1e-5

Table 2: MNIST, $\lambda = 1$

5 Analysis of Numerical Results

First we have two general remarks:

1. Algorithms perform very differently in different problems. And data set can also influence the performance of optimization algorithm. The Covertype data set can be quite unstable during training.

2. With different λ, performances of algorithms change a lot. λ is an indicator of the sparsity of w we want. The larger the λ, the sparser we want w to be, and the faster the training of all the algorithms in general.

Algorithm	Iteration times	$\|\|f_k - f_{k-1}\|\|$
AdaGrad	3093	1e-5
Adam	385	1e-5
StabAda	485	1e-5

Table 3: MNIST, $\lambda = 10$

Algorithm	Iteration times	$\|\|f_k - f_{k-1}\|\|$
AdaGrad	87	1e-4
Adam	500	0.0126
StabAda	47	1e-4

Table 4: Covertype, $\lambda = 0.001$,$maxiter = 500$

Algorithm	Iteration times	$\|\|f_k - f_{k-1}\|\|$
AdaGrad	269	1e-4
Adam	301	1e-4
StabAda	380	1e-4

Table 5: Covertype, $\lambda = 1$,$maxiter = 500$

In specific, in the problem of MNIST, the performance of StabAda is stably better than the other two algorithms in terms of efficiency and accuracy. The performance of Adam is also comparable to StabAda, just slightly better. Although AdaGrad can sometimes yield quite good results, it's very unstable in terms of efficiency.

In the problem of Covertype, the performance of Adam is the most stable and usually has the best convergence speed. As for the other two, AdaGrad is slightly worse than StabAda and still the robustness of StabAda is quite obvious.

Algorithm	Iteration times	$\|\|f_k - f_{k-1}\|\|$
AdaGrad	135	1e-4
Adam	84	1e-4
StabAda	234	1e-4

Table 6: Covertype, $\lambda = 10$,$maxiter = 500$

Algorithm	Iteration times	$\|\|f_k - f_{k-1}\|\|$
AdaGrad	188	1e-4
Adam	52	1e-4
StabAda	46	1e-4

Table 7: Covertype, $\lambda = 30$,$maxiter = 500$

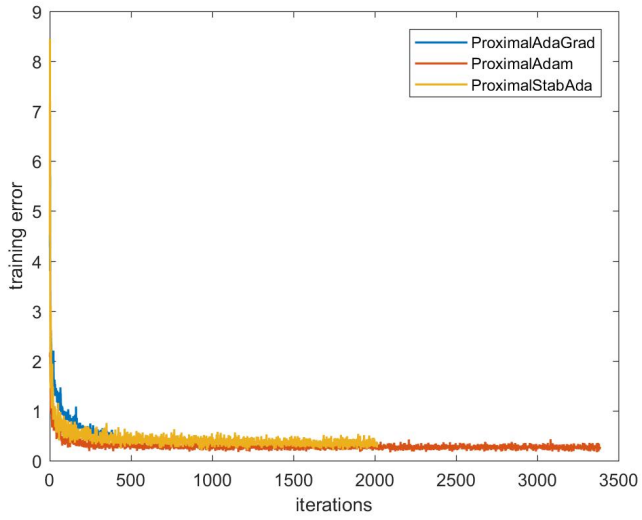

Figure 1: MNIST $lambda = 0.001$

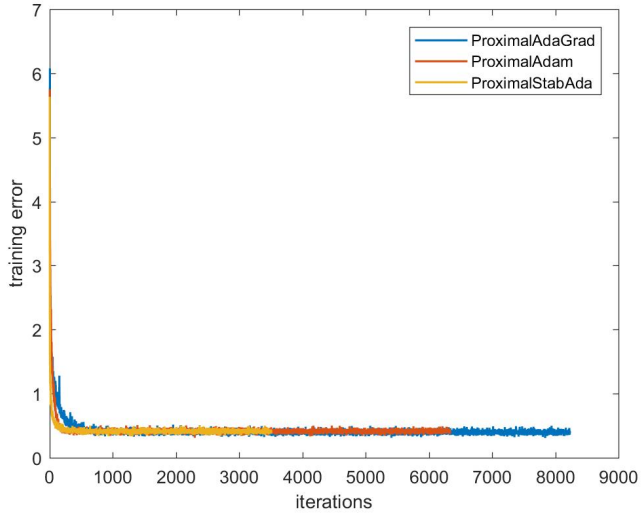

Figure 2: MNIST $lambda = 1$

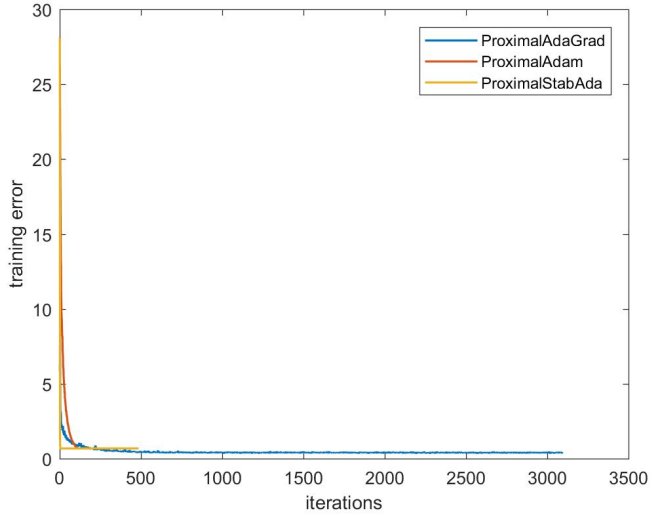

Figure 3: MNIST $lambda = 10$

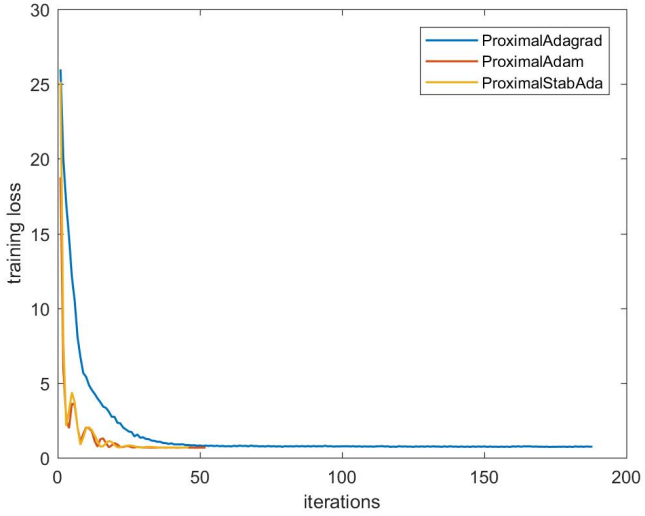

Figure 4: Covertype $\lambda = 0.001$

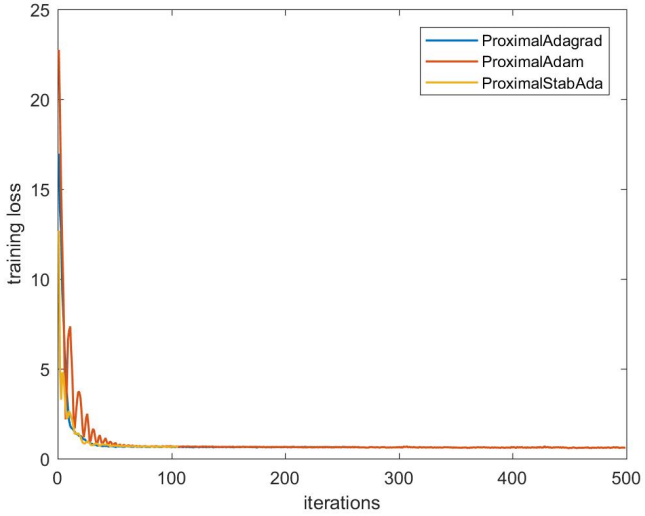

Figure 5: Covertype $\lambda = 1$

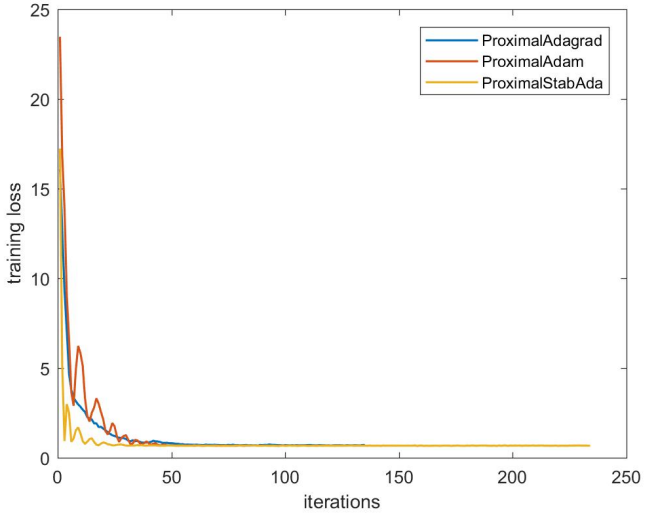

Figure 6: Covertype $\lambda = 10$

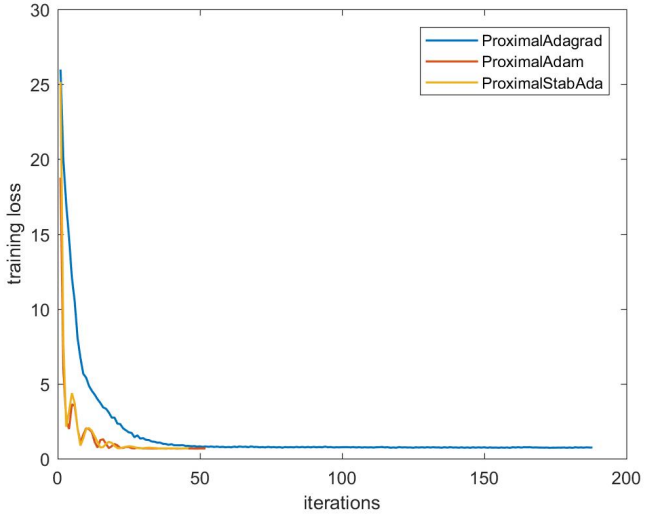

Figure 7: Covertype $\lambda = 30$