Applications of Adaptive Gradient Methods

1 Introduction

We consider the following stochastic optimization problem:

$$\min_{w \in C} \phi(w), \quad \phi(w) = f(w) + g(w), \qquad f(w) = \mathbf{E}_{\xi \sim D} f(\xi, w), \tag{1}$$

where ξ is a random variable, drawn from a distribution D.

In this lecture, we consider two applications of AdaGrad. One is FTRL for large scale logistic regression. Another is the Adam algorithm for deep learning optimization.

2 FTRL

FTRL is a modification of RDA and Adagrad, and it becomes popular in the industry for solving large scale L_1-L_2 regularized regression, due to its use at Google for large scale click through rate (CTR) estimation [4] for computational advertising. The original paper with fixed regularizer was presented in [3]. The reason for using RDA instead of proximal gradient is because better sparsity can be achieved with RDA.

In the last lecture, we introduced a version of AdaGrad with RDA, as in Algorithm 1, which is equivalent to AdaGrad when $g(w) = 0$.

However, the original version of AdaGrad-RDA of [1], following [5], has used a different version as stated in Algorithm 2. It was noticed in [3] that this version is not equivalent to AdaGrad even when $g(w) = 0$, and the performance of this algorithm is inferior to Proximal AdaGrad. The FTRL method tries to fix this problem by correcting the bias, and the resulting algorithm is Algorithm 3. It can be shown with this correction, this algorithm is the same as AdaGrad when $g(w) = 0$. That is:

$$-\Lambda_t \bar{g}_t = w^{(t-1)} - \Lambda_t g_t$$

when $g(w) = 0$.

In the proximal case with $g(w) \neq 0$, when the algorithm converges, $\mathbf{E}_{B_t} g_t = \nabla f(w^{(t-1)}) \neq 0$ but $\mathbf{E}_{B_t} g_t + \nabla g(w^{(t-1)}) \to 0$. The total amount of weighted gradients in $-\Lambda_t \bar{g}_t$ is $\Lambda_t t$. Therefore we need to give a matching proximal weighting of $t\Lambda_t$ in the algorithm. Even if Λ_t is small, $t\Lambda_t$ is large, which leads to stability and better sparsity with a sparse regularizer, when compared to proximal gradient method (which has one gradient $-\Lambda_t g_t$, with matching proximal mapping weighting of Λ_t.

In the CTR application of [4], one goes through data once only in an online fashion. One can replace Line 10 of Algorithm 3 by

$$w^{(t)} = \text{prox}_{\Lambda_t g}(-\Lambda_t \bar{g}_t),$$

which is equivalent to changing the regularizer to $t^{-1}g(w)$ at time step t.

Algorithm 1: AdaGrad-RDA

```
   Input: $f(\cdot)$, $g(\cdot)$, $w^{(0)}$, $\eta_0, \eta_1, \eta_2, \dots$
          $h(w)$ (default is $h(w) = 0.5\eta_0\|w\|_2^2$)
   Output: $w^{(T)}$
1  Let $\tilde{\alpha}_0 \in \partial h(w^{(0)})$
2  Let $\tilde{\Lambda}_0 = 0$
3  $\tilde{g}_0^2 = [\epsilon, \dots, \epsilon]$
4  for $t = 1, 2, \dots, T$ do
5      Randomly select a minibatch $B_t$ of $m$ independent samples from $D$
6      Let $g_t = \nabla_w f_{B_t}(w^{(t-1)})$
7      Let $\tilde{g}_t^2 = \tilde{g}_{t-1}^2 + g_t^2$
8      Let $\Lambda_t = \eta \mathrm{diag}(\tilde{g}_t^{-1})$
9      Let $\tilde{\alpha}_t = \tilde{\alpha}_{t-1} - \Lambda_t g_t$
10     Let $\tilde{\Lambda}_t = \tilde{\Lambda}_{t-1} + \Lambda_t$
11     Let $w^{(t)} = \mathrm{prox}_{\tilde{\Lambda}_t g}(\tilde{\alpha}_t)$
   Return: $w^{(T)}$
```

Algorithm 2: AdaGrad-RDA [1]

```
   Input: $f(\cdot)$, $g(\cdot)$, $w^{(0)}$, $\eta_0, \eta_1, \eta_2, \dots$
          $h(w)$ (default is $h(w) = 0.5\eta_0\|w\|_2^2$)
   Output: $w^{(T)}$
1  Let $\tilde{\alpha}_0 \in \partial h(w^{(0)})$
2  $\tilde{g}_0^2 = [\epsilon, \dots, \epsilon]$
3  $\bar{g}_0^2 = 0$
4  for $t = 1, 2, \dots, T$ do
5      Randomly select a minibatch $B_t$ of $m$ independent samples from $D$
6      Let $g_t = \nabla_w f_{B_t}(w^{(t-1)})$
7      Let $\tilde{g}_t^2 = \tilde{g}_{t-1}^2 + g_t^2$
8      Let $\Lambda_t = \eta \mathrm{diag}(\tilde{g}_t^{-1})$
9      Let $\bar{g}_t = \bar{g}_{t-1} + g_t$
10     Let $w^{(t)} = \mathrm{prox}_{t\Lambda_t g}(-\Lambda_t \bar{g}_t)$
   Return: $w^{(T)}$
```

```
Algorithm 3: FTRL (follow the regularized leader)
  Input: $f(\cdot)$, $g(\cdot)$, $w^{(0)}$, $\eta_0, \eta_1, \eta_2, \ldots$
      $h(w)$ (default is $h(w) = 0.5\eta_0\|w\|_2^2$)
  Output: $w^{(T)}$
1 Let $\tilde{\alpha}_0 \in \partial h(w^{(0)})$
2 $\tilde{g}_0^2 = [\epsilon, \ldots, \epsilon]$
3 $\bar{g}_0^2 = 0$
4 for $t = 1, 2, \ldots, T$ do
5     Randomly select a minibatch $B_t$ of $m$ independent samples from $D$
6     Let $g_t = \nabla_w f_{B_t}(w^{(t-1)})$
7     Let $\tilde{g}_t^2 = \tilde{g}_{t-1}^2 + g_t^2$
8     Let $\Lambda_t = \eta \mathrm{diag}(\tilde{g}_t^{-1})$
9     Let $\bar{g}_t = \bar{g}_{t-1} + g_t - (\Lambda_t^{-1} - \Lambda_{t-1}^{-1}) w^{(t-1)}$
10    Let $w^{(t)} = \mathrm{prox}_{t\Lambda_t g}(-\Lambda_t \bar{g}_t)$
  Return: $w^{(T)}$
```

3 Adam

The ADAM algorithm [2] has been widely used in deep learning optimization. It is a combination of the momentum (heavy-ball) formulation of acceleration, together with the RMSprop version of AdaGrad. The corresponding Nesterov's acceleration version is called Nadam.

The original algorithm, without the proximal term ($g(w) = 0$), is given in Algorithm 4. We can change it to proximal gradient method as in Algorithm 5, where we remove the normalization $1/(1-\rho)$ from Line of the original algorithm. Otherwise the two algorithms are equivalent without proximal terms. We do not use this normalization to be consistent with other methods.

With proximal term, it can be modified according to stochastic accelerated gradient method (heavy-ball version) in Lecturer 21, with coordinate-wise AdaGrad learning rate from RMSprop. It is presented in Algorithm 6. To see the difference of Algorithm 6 and Algorithm 5, we note that when $g = 0$, the HB version uses

$$p_t = \sum_{s \le t} \beta^{t-s} \Lambda_s g_s,$$

compared to Adam, which uses

$$\bar{g} = \sum_{s \le t} \beta^{t-s} g_s.$$

The Nesterov's proximal gradient version, with AdaGrad learning rate, is given in Algorithm 7. Note that the moment is $p = w^{(t)} - w^{(t-1)}$ in these representations. The RDA version is given in Algorithm 8. We behaves better with proximal terms. Without proximal terms (that is, $g(w) = 0$), Algorithm 7 and Algorithm 8 are equivalent. A version of acceleration, presented in Algorithm 9, is also equivalent when $g(w) = 0$.

For more complex regualrizers, it is also possible to apply AdaGrad with accelerated linearized ADMM. We will not list the resulting algorithm here.

Algorithm 4: ADAM (without proximal term)

```
   Input: $\phi(\cdot)$, learning rates $\eta$ , $\beta$ (default is 0.9), $\rho$ (default is 0.999), $\epsilon > 0$ (default is $10^{-8}$ ,
          $w^{(0)}$
   Output: $w^{(T)}$
1  Let $\tilde{g}_0^2 = 0$
2  Let $\bar{g}_0 = 0$
3  for $t = 1, 2, \ldots, T$ do
4      Randomly pick $B_t \sim D$
5      Let $g_t = \nabla_w f_{B_t}(w^{(t-1)})$
6      Let $\tilde{g}_t^2 = \rho\tilde{g}_{t-1}^2 + (1-\rho)g_t^2$
7      Let $\bar{g}_t = \beta\bar{g}_{t-1} + g_t$
8      Let $\Lambda_t = \eta\mathrm{diag}((\tilde{g}_t/(1-\rho) + \epsilon)^{-1})$
9      Let $w^{(t)} = w^{(t-1)} - \Lambda_t\bar{g}_t$
   Return: $w^{(T)}$
```

Algorithm 5: Prox-ADAM

```
   Input: $\phi(\cdot)$, learning rates $\eta$ , $\beta$ (default is 0.9), $\rho$ (default is 0.999), $\epsilon > 0$ (default is $10^{-8}$ ,
          $w^{(0)}$
   Output: $w^{(T)}$
1  Let $\tilde{g}_0^2 = 0$
2  Let $\bar{g}_0 = 0$
3  for $t = 1, 2, \ldots, T$ do
4      Randomly pick $B_t \sim D$
5      Let $g_t = \nabla_w f_{B_t}(w^{(t-1)})$
6      Let $\tilde{g}_t^2 = \rho\tilde{g}_{t-1}^2 + (1-\rho)g_t^2$
7      Let $\bar{g}_t = \beta\bar{g}_{t-1} + g_t$
8      Let $\Lambda_t = \eta\mathrm{diag}((\tilde{g}_t + \epsilon)^{-1})$
9      Let $w^{(t)} = \mathrm{prox}_{\Lambda_t g}(w^{(t-1)} - \Lambda_t\bar{g}_t)$
10     $\bar{g}_t = \Lambda_t^{-1}(w^{(t-1)} - w^{(t)})$
   Return: $w^{(T)}$
```

Algorithm 6: Prox-AdaHB

```
  Input: $\phi(\cdot)$, learning rates $\eta$ , $\beta$ (default is 0.9), $\rho$ (default is 0.999), $\epsilon > 0$ (default is $10^{-8}$ ,
         $w^{(0)}$
  Output: $w^{(T)}$
1 Let $\tilde{g}_0^2 = 0$
2 Let $p_0 = 0$
3 for $t = 1, 2, \ldots, T$ do
4     Randomly pick $B_t \sim D$
5     Let $g_t = \nabla_w f_{B_t}(w^{(t-1)})$
6     Let $\tilde{g}_t^2 = \rho\tilde{g}_{t-1}^2 + (1-\rho)g_t^2$
7     Let $\Lambda_t = \eta \mathrm{diag}((\tilde{g}_t + \epsilon)^{-1})$
8     Let $w^{(t)} = \mathrm{prox}_{\Lambda_t g}(w^{(t-1)} + \beta p_{t-1} - \Lambda_t g_t)$
9     Let $p_t = w^{(t)} - w^{(t-1)}$
  Return: $w^{(T)}$
```

Algorithm 7: Prox-AdaACCL

```
  Input: $\phi(\cdot)$, learning rates $\eta$ , $\beta$ (default is 0.9), $\rho$ (default is 0.999), $\epsilon > 0$ (default is $10^{-8}$ ,
         $w^{(0)}$
  Output: $w^{(T)}$
1 $\tilde{g}_0^2 = 0$
2 $p_0 = 0$
3 for $t = 1, 2, \ldots, T$ do
4     Randomly pick $B_t \sim D$
5     Let $v^{(t)} = w^{(t-1)} + \beta(w^{(t-1)} - w^{(t-2)})$
6     Let $g_t = \nabla_w f_{B_t}(v^{(t-1)})$
7     Let $\tilde{g}_t^2 = \rho\tilde{g}_{t-1}^2 + (1-\rho)g_t^2$
8     Let $\Lambda_t = \eta \mathrm{diag}((\tilde{g}_t + \epsilon)^{-1})$
9     Let $w^{(t)} = \mathrm{prox}_{\Lambda_t g}(v^{(t)} - \Lambda_t g_t)$
  Return: $w^{(T)}$
```

Algorithm 8: Prox-AdaACCL-RDA

```
Input: $\phi(\cdot)$, learning rates $\eta$ , $\beta$ (default is 0.9), $\rho$ (default is 0.999), $\epsilon > 0$ (default is $10^{-8}$ ,
       $w^{(0)}$
Output: $w^{(T)}$
1  Let $\tilde{g}_0^2 = 0$
2  Let $u^{(0)} = w^{(0)}$
3  Let $\tilde{\alpha}_0 = 0$
4  Let $\tilde{\Lambda}_0 = 0$
5  for $t = 1, 2, \ldots, T$ do
6      Randomly pick $B_t \sim D$
7      Let $v^{(t)} = \beta w^{(t-1)} + (1-\beta) u^{(t-1)}$
8      Let $g_t = \nabla_w f_{B_t}(v^{(t)})$
9      Let $\tilde{g}_t^2 = \rho \tilde{g}_{t-1}^2 + (1-\rho) g_t^2$
10     Let $\Lambda_t = \frac{\eta}{1-\beta} \mathrm{diag}((\tilde{g}_t + \epsilon)^{-1})$
11     Let $\tilde{\alpha}_t = \tilde{\alpha}_{t-1} - \Lambda_t g_t$
12     Let $\tilde{\Lambda}_t = \tilde{\Lambda}_{t-1} + \Lambda_t$
13     Let $u^{(t)} = \mathrm{prox}_{\tilde{\Lambda}_t g}(\tilde{\alpha}_t)$
14     Let $w^{(t)} = \beta w^{(t-1)} + (1-\beta) u^{(t)}$
Return: $w^{(T)}$
```

Algorithm 9: Prox-AdaACCL-v3

```
Input: $\phi(\cdot)$, learning rates $\eta$ , $\beta$ (default is 0.9), $\rho$ (default is 0.999), $\epsilon > 0$ (default is $10^{-8}$ ,
       $w^{(0)}$
Output: $w^{(T)}$
1  Let $\tilde{g}_0^2 = 0$
2  Let $u^{(0)} = w^{(0)}$
3  Let $\tilde{\alpha}_0 = 0$
4  Let $\tilde{\Lambda}_0 = 0$
5  for $t = 1, 2, \ldots, T$ do
6      Randomly pick $B_t \sim D$
7      Let $v^{(t)} = \beta w^{(t-1)} + (1-\beta) u^{(t-1)}$
8      Let $g_t = \nabla_w f_{B_t}(v^{(t)})$
9      Let $\tilde{g}_t^2 = \rho \tilde{g}_{t-1}^2 + (1-\rho) g_t^2$
10     Let $\Lambda_t = \frac{\eta}{1-\beta} \mathrm{diag}((\tilde{g}_t + \epsilon)^{-1})$
11     Let $u^{(t)} = \mathrm{prox}_{\Lambda_t g}(u^{(t-1)} - \Lambda_t g_t)$
12     Let $w^{(t)} = \beta w^{(t-1)} + (1-\beta) u^{(t)}$
Return: $w^{(T)}$
```

4 Empirical Studies

We study the smoothed hinge loss function $\phi_\gamma(z)$ with $\gamma = 1$, and solves the following $L_1 - L_2$ regularization problem:

$$\min_w \left[\underbrace{\frac{1}{n}\sum_{i=1}^n \phi_\gamma(w^\top x_i y_i)}_{f(w)} + \underbrace{\frac{\lambda}{2}\|w\|_2^2 + \mu\|w\|_1}_{g(w)} \right].$$

We compare different adaptive gradient algorithms.

References

[1] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR'15*, 2015.

[3] H Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and L1 regularization. In *AISTATS*, 2011.

[4] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, 2013.

[5] Lin Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *J. Mach. Learn. Res.*, 11:25432596, December 2010.

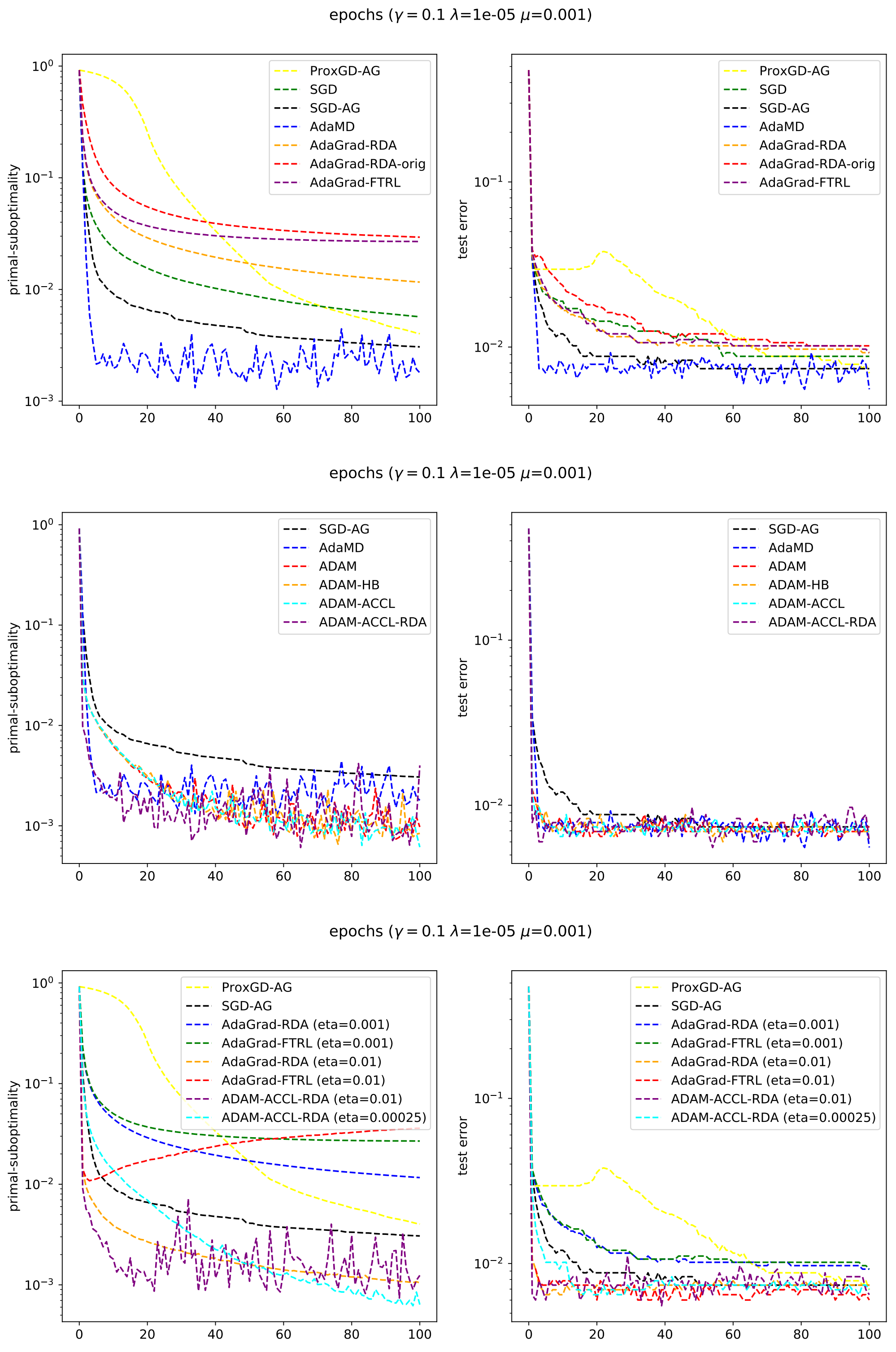

Figure 1: Comparisons of different stochastic algorithms (with proximal terms)

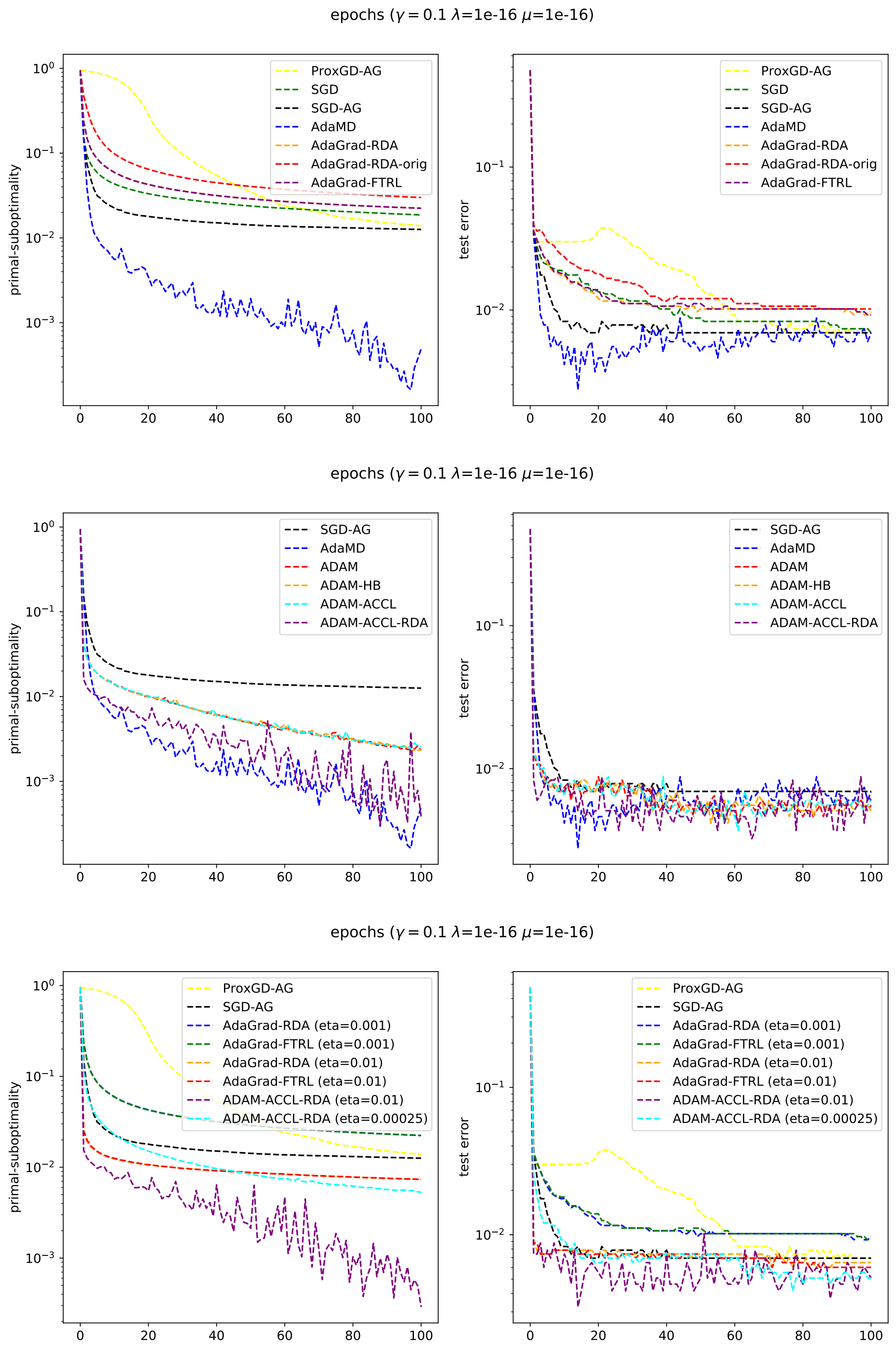

Figure 2: Comparisons of different stochastic algorithms (without proximal terms)