Variants of Stochastic Gradients Algorithms

Ruicheng Ao 1900012179

March 1, 2022

1 Problem description

In this homework, we focus on the unconstrained optimization algorithm in which the objective function can be de formulated into the form

$$\min \quad L_\lambda(x,y) = \frac{1}{n}\sum_{i=1}^n f_i^(w) + \lambda\|w\|_2^2, \tag{1}$$

wherer $f_i(w) = 1 - tanh(y^i w^\top x^i)$ and $\lambda > 0$. With a simple computation we get

$$\nabla_w f_i(w) = -1 + y^i x^i \tanh(y^i w^\top x^i)^2. \tag{2}$$

The gradient ∇L_λ involves the sum of n different terms, which makes the computation costly. As a result, several stochastic methods using the technique of sampling are widely used [2, Section 8]. In our experiments, we mainly focus on four of the first-order algorithms : Adagrad, Adam, SGD with momentum & linesearch, RMSprop and show their efficiency. The first two are required and SGD with momentum & linesearch, RMSprop are as extra credit.

2 Algorithm description

In this section, we introduce Adagrad, Adam, SGD with momentum, SGD with line search and RMSprop, and give the general frameworks.

2.1 Adagrad

Adagrad is one of the most famous algorithms in deep learning, named by adaptive gradient algorithm. It considers adding an accumulated second-order moment of gradient in order to accelerate the convergence. This scheme can adaptively adjust the gradient without fine tuning, which enjoys good performance on many problems [2]. The framework of Adagrad is given in Algorithm 1.

Algorithm 1 Adagrad

```
Input  Learning rate $lr$, regularization parameter $\delta$, batch size $bs$
1: Set $k = 0$
2: Set $r = \vec{0}$
3: while Do not converge do
4:     Sample a minibatch set B of size $bs$ from the training set
5:     Compute sampled gradient $g^k = \frac{1}{bs}\nabla_w \sum_{i\in B} f_i(w) + 2\lambda w$
6:     Update $r = r + g^k \odot g^k$
7:     Update $w^{k+1} = w^k - \frac{lr}{\sqrt{\delta + r}} * g^k$
8:     $k = k + 1$
9: end while
Output  $w^k$
```

2.2 Adam

Adam is one of the benchmark algorithms in deep learning, which is named of adaptive moment estimation. It considers adding a second-order moment of gradient in order to accelerate the convergence. The framework of Adam is stated in Algorithm 2.

Algorithm 2 Adam

```
Input  Learning rate $lr$, batch size $bs$, momentum parameters $\beta_1, \beta_2$, safefy gurantee $\epsilon$
1: Set $\beta_1^0 = \beta_1, \beta_2^0 = \beta_2, k = 0$
2: while Do not converge do
3:     $k = k + 1$
4:     $\beta_1^k = \beta_1^{k-1} * \beta_1, \beta_2^k = \beta_2^{k-1} * \beta_2$
5:     Sample a minibatch set $B$ of size $bs$ from the training set
6:     Compute sampled gradient $g^k = \frac{1}{bs}\nabla_m \sum_{i\in B} f_i(w) + 2\lambda w^k$
7:     Compute the first moment $\nu = \beta_1 \nu + (1 - \beta_1)\nu$
8:     Compute the second moment $m = \beta_2 m + (1 - \beta_2) m. \times m$
9:     Eliminate bias via $\hat{\nu} = \nu/(1 - \beta_1^k)$
10:    Eliminate bias via $\hat{m} = m/(1 - \beta_2^k)$
11:    Update $w^k = w^{k-1} - lr * \hat{\nu}/\sqrt{\hat{m} + \varepsilon \mathbf{1}}$
12: end while
```

2.3 SGD with momentum

Momentum algorithm is an modification of traditional gradient descent by utilizing the memory of updating directions in history to accelerate the convergence. The general framework of the Momentum algorithm is given in Algorithm 3.

Algorithm 3 SGD with momentum

```
Input Learning rate $lr$, momentum weight $mom$, batch size $bs$
1: Set $k = 0$
2: Set $\nu = \vec{0}$
3: while Do not converge do
4:     Sample a minibatch set $B$ of size $bs$ from the training set
5:     Compute sampled gradient $g^k = \frac{1}{bs}\nabla_m \sum_{i \in B} f_i(w) + 2\lambda w$
6:     Update momentum $\nu = mom * \nu + (1 - \nu)g^k$
7:     Update $w^{k+1} = w^k - lr * \nu$
8:     $k = k + 1$
9: end while
Output $w^k$
```

2.4 SGD with linesearch

[3] has proposed a stochastic gradient descent with linesearch based on Armijo linesearch algorithm. We adopt one of the variants of such linesearch schemes, whose framework is shown in Algorithm 4.

Algorithm 4 SGD with linesearch

```
Input Learning rate $lr$, acception parameter $\gamma$, decay rate $\rho$, maximal number of iterations in linesearch $N$
1: Set $k = 0$
2: while Do not converge do
3:     Sample a minibatch set $B$ of size $bs$ from the training set
4:     Compute sampled gradient $g^k = \frac{1}{bs}\nabla_m \sum_{i \in B} f_i(w) + 2\lambda w$
5:     Set $i = 0, lr' = lr$
6:     while $f(w - lr'g^k) \geq f(w) - lr' * \gamma \|g^k\|_2^2$ and $i < N$ do
7:         $lr' = lr' * \rho$
8:         $i = i + 1$
9:     end while
10:    $w^{k+1} = w^k - lr'g^k$
11:    $k = k + 1$
12: end while
Output $w^k$
```

2.5 RMSprop

RMSprop is a modified version of Adagrad, named by root mean square prop. RMSprop uses the "mean square root" instead of the accumulating squre of gradient in order to prevent vanishing step size. The framework is given in Algorithm

Algorithm 5 RMSprop

```
Input Learning rate $lr$, regularization parameter $\delta$, batch size $bs$, weight $0 < \rho < 1$
1: Set $k = 0$
2: Set $r = \vec{0}$
3: while Do not converge do
4:     Sample a minibatch set B of size $bs$ from the training set
5:     Compute sampled gradient $g^k = \frac{1}{bs}\nabla_w \sum_{i \in B} f_i(w) + 2\lambda w$
6:     Update $r = \rho * r + (1 - \rho) * g^k \odot g^k$
7:     Update $w^{k+1} = w^k - \frac{lr}{\sqrt{\delta + r}} * g^k$
8:     $k = k + 1$
9: end while
Output $w^k$
```

3 Numerical experiments

In our numerical experiments, we test all three algorithms proposed above on two datasets Covtype and Gisette. For Covtype we classify the 2nd class and others, while for Gisette we classify with label -1 and 1. The dataset Covtype is splitted into train set and test set with ration 70% and 30%, while the default split of Gisette is utilized. More detailed descriptions of the datasets can be found in [1]. For traindata (x_i, y_i), x_is are normalized in ℓ_2 norm in each feature . The regularization parameter λ is chosen from the set $\{10, 1, 0.1, 0.01\}$.

We tune our parameters by a simple grid search. The initial learning rate lr is chosen to be $1e-3$ for Adam, Adagrad, RMSprop and SGD with momentum, $1e-2$ for SGD with linesearch. The momentum parameter mom is set to be 0.95 for each. We choose $\beta_1 = \beta_2 = 0.999$ for Adam and safety gurantee $\epsilon = 1e-5$. A maximal linesearch iterative number is shosen to be $N = 5$ with acception parameter $\gamma = 0.1$ and decay rate $\rho = 0.5$. We train the instances for fixed epochs. The results are established as below.

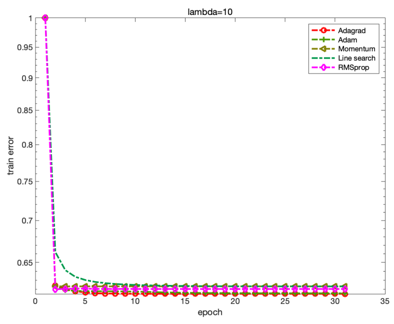

Figure 1: Training errer

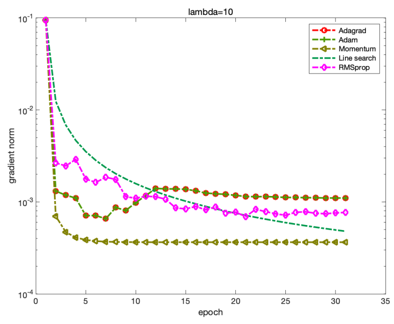

Figure 2: gradient norm

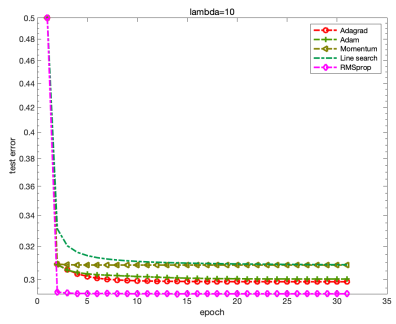

Figure 3: Testing error

Results on Covtype when $\lambda = 10$

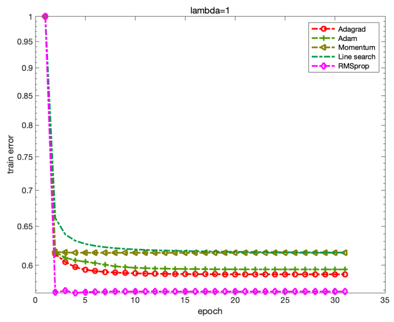

Figure 4: Training errer

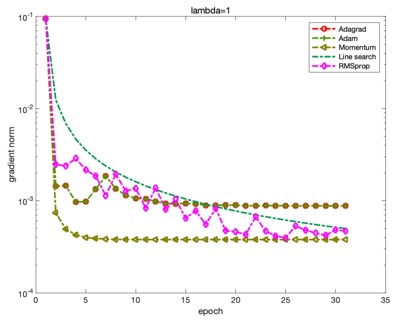

Figure 5: gradient norm

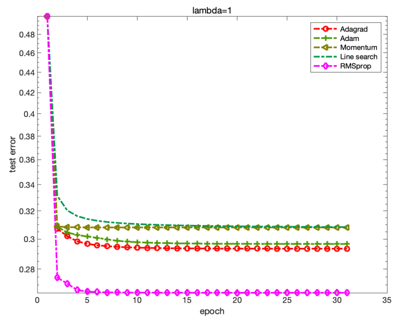

Figure 6: Testing error

Results on Covtype when $\lambda = 1$

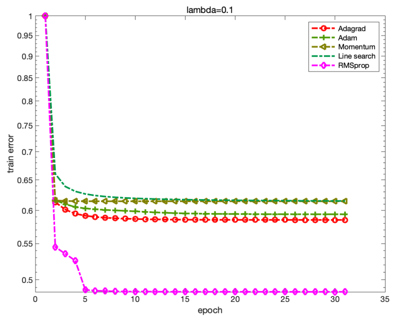

Figure 7: Training errer

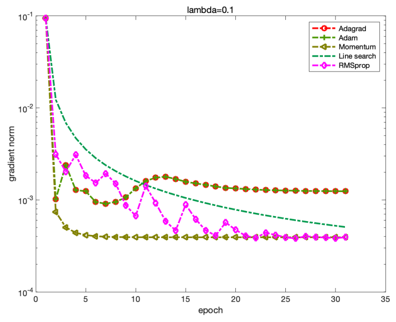

Figure 8: gradient norm

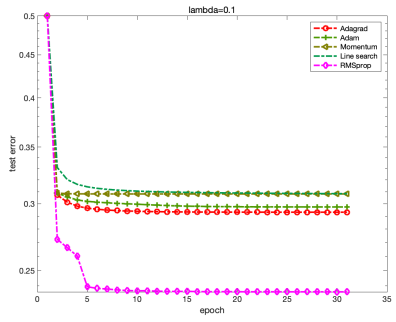

Figure 9: Testing error

Results on Covtype when $\lambda = 0.1$

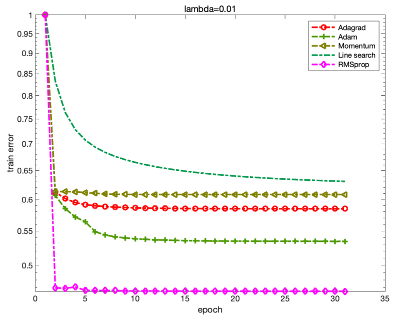

Figure 10: Training errer

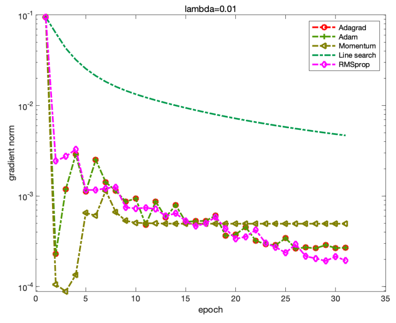

Figure 11: gradient norm

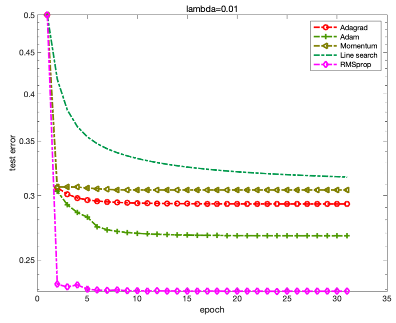

Figure 12: Testing error

Results on Covtype when $\lambda = 0.01$

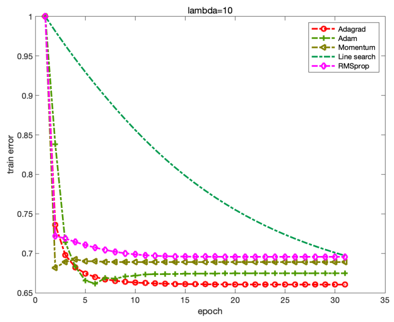

Figure 13: Training errer

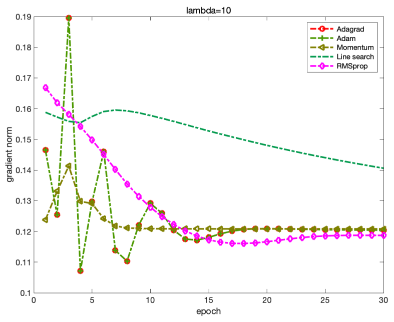

Figure 14: gradient norm

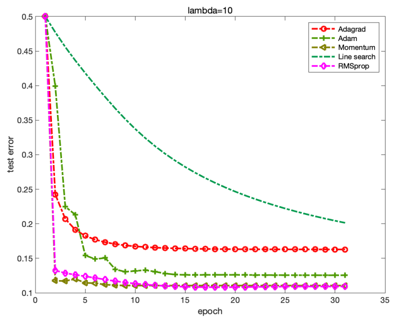

Figure 15: Testing error

Results on Gisette when $\lambda = 10$

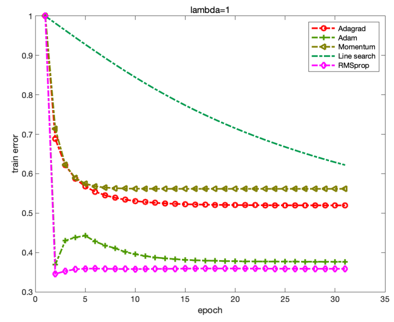

Figure 16: Training errer

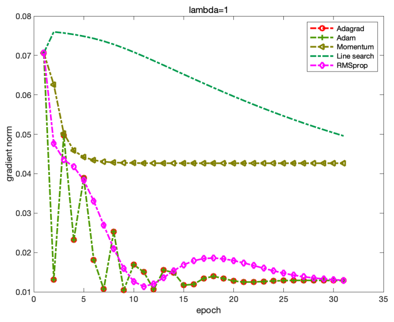

Figure 17: gradient norm

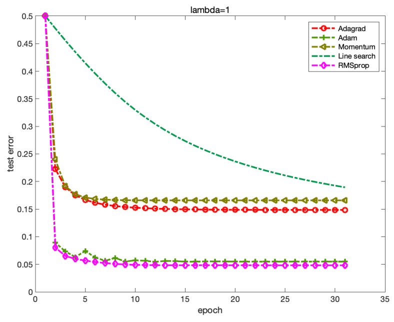

Figure 18: Testing error

Results on Gisette when $\lambda = 1$

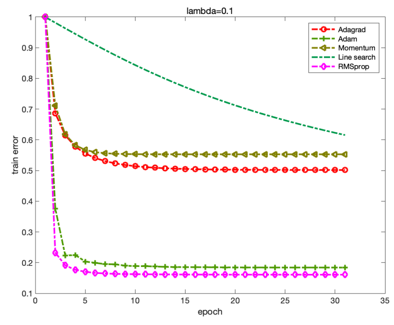

Figure 19: Training errer

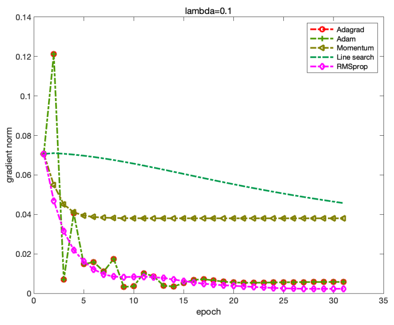

Figure 20: gradient norm

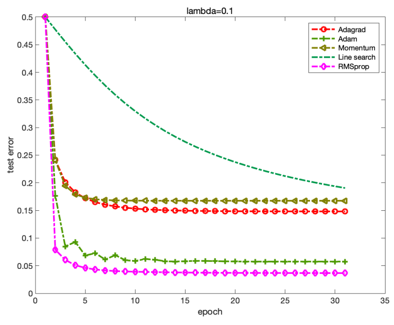

Figure 21: Testing error

Results on Gisette when $\lambda = 0.1$

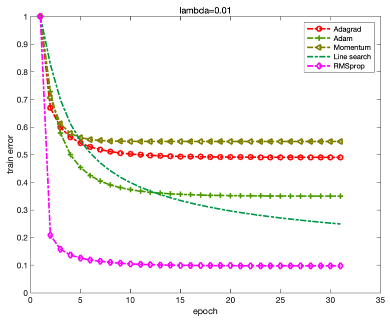

Figure 22: Training errer

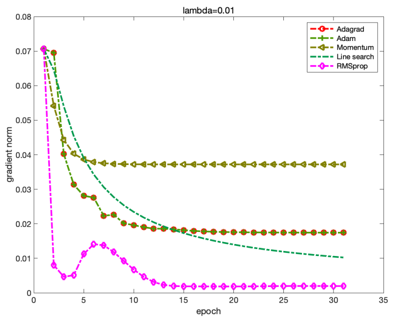

Figure 23: gradient norm

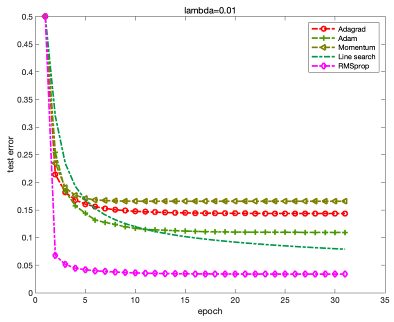

Figure 24: Testing error

Results on Gisette when $\lambda = 0.01$

Here in the gradient norm, we omit the ℓ_1 regularized part since it dominates when λ is large. We see all the three algorithms perform well on the datasets, yielding a relatively high quality solution within only a few epochs. A significant influence of regularization term is shown and the three algorithms have different performance in term of different standards. We observe that Adam and RMSprop outperforms the others in both training error and testing error, which is consistent with our intuition that Adam and RMSprop utilize a kind of self-adaptive stepsize on each component for acceleration. No wonder Adam and RMSprop are so widely used in deep learning. Additionally, Adagrad also behaves well. On the other hand, the SGD with linesearch converges faster in primal gradient norm and testing error than the momentum algorithm in the first dataset. This indicates its competitive stability and ability of restoration in the theoretical base of convergence analysis, as well as easy implementation. The faster convergence of the momentum in training error (with regularization term) compared to SGD with linesearch shows the superiority of momentum in preserving the descent direction, since we did not add a momentum term to SGD with linesearch. In this kind of median-size problems, these algorithms show a fast convergence. We leave further discussion of second-order algorithms in the future due to space limitation. Anyway, we have completed all requirements including extra-credits in this assignment.

References

[1] Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. Exact and inexact subsampled newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578, 2019.

[2] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[3] Courtney Paquette and Katya Scheinberg. A stochastic line search method with convergence rate analysis. *arXiv preprint arXiv:1807.07994*, 2018.