Variants of Stochastic Gradients Algorithms

Ruicheng Ao 1900012179

February 24, 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

$$\min \quad L_\lambda(x,y) = \frac{1}{n}\sum_{i=1}^{n} f_i^(w) + \lambda\|w\|_1, \tag{1}$$

where $f_i(w) = \log(1+\exp(-y^i w^\top x^i))$ and $\lambda > 0$. With a simple computation one can get

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

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 three of the first-order algorithms : Momentum, Adam and SGD with linesearch and show their efficiency.

2 Algorithm description

In this section, we introduce Momentum, Adam and SGD with linesearch, give the general frameworks.

2.1 The Momentum algorithm

Momentum algorithm is an modification of traditional gradient descent by utilizing the memory of updating directions in history to accelerate the convergence. We apply subgradient in the algorithm in the ℓ_1-regularized problem since the norm is non-differentiable. The general framework of the Momentum algorithm is given in Algorithm 1.

Algorithm 1 The Momentum algorithm

Input Learning rate lr, momentum weight mom, batch size bs

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 $g^k = \frac{1}{bs}\nabla_m \sum_{i\in B} f_i(w) + \lambda\partial\|w\|_1$
5: Update momentum $\nu = mom * \nu + (1-\nu)g^k$
6: Update $w^{k+1} = w^k - lr * \nu$
7: $k = k+1$
8: **end while**

Output w

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. We use subgradient instead of gradient in the experiments for the same reason. The framework of Adam is stated in Algorithm 2.

Algorithm 2 Adam

Input Learning rate lr, batch size bs, momentum parameters β_1, β_2, safefy gurantee ϵ

Set $\beta_1^0 = \beta_1, \beta_2^0 = \beta_2, k = 0$
while Do not converge **do**
  $k = k + 1$
  $\beta_1^k = \beta_1^{k-1} * \beta_1, \beta_2^k = \beta_2^{k-1} * \beta_2$
  Sample a minibatch set B of size bs from the training set
  Compute sampled gradient $g^k = \frac{1}{bs}\nabla_m \sum_{i\in B} f_i(w) + \lambda\partial\|w\|_1$
  Compute the first moment $\nu = \beta_1\nu + (1-\beta_1)\nu$
  Compute the second moment $m = \beta_2 m + (1-\beta_2)m. \times m$
  Eliminate bias via $\hat{\nu} = \nu/(1-\beta_1^k)$
  Eliminate bias via $\hat{m} = m/(1-\beta_2^k)$
  Update $w^k = w^{k-1} - lr * \hat{\nu}/\sqrt{\hat{m} + \varepsilon\mathbf{1}}$
end while

2.3 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 3.

Algorithm 3 SGD with linesearch

Input Learning rate lr, acception parameter γ, decay rate ρ, maximal number of iterations in linesearch N

Set $k = 0$
while Do not converge **do**
  Sample a minibatch set B of size bs from the training set
  Compute sampled gradient $g^k = \frac{1}{bs}\nabla_m \sum_{i\in B} f_i(w) + \lambda\partial\|w\|_1$
  Set $i = 0, lr' = lr$
  while $f(w - lr'g^k) \geq f(w) - lr' * \gamma\|g^k\|_2^2$ and $i < N$ **do**
    $lr' = lr' * \rho$
    $i = i + 1$
  end while
  $w^{k+1} = w^k - lr'g^k$
  $k = k + 1$
end while

3 Numerical experiments

In our numerical experiments, we test all three algorithms proposed above on two datasets MNIST and Covtype. For MNIST, the numbers are divided according to parity, while for Covtype we classify the 2nd class and others. More

detailed descriptions of the datasets can be found in [1]. For each traindata (x_i, y_i), x_i is normalized in ℓ_2 norm. 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 and 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.

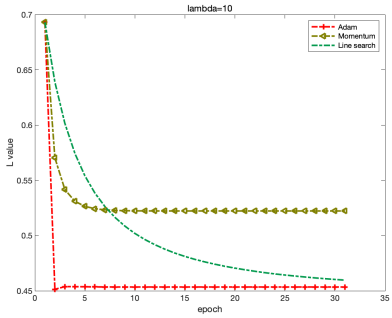

Figure 1: function value

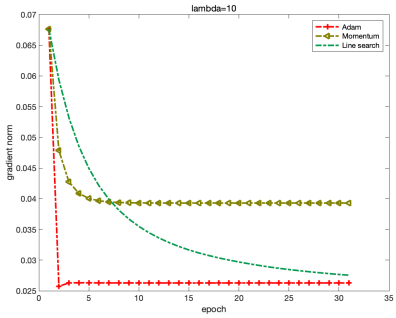

Figure 2: gradient norm

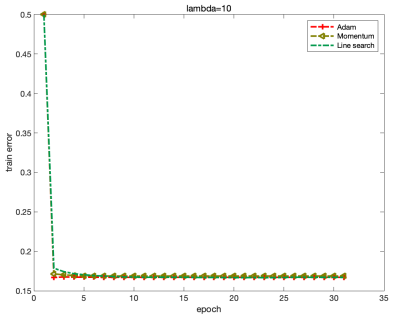

Figure 3: classification error

Results on MNIST when $\lambda = 10$

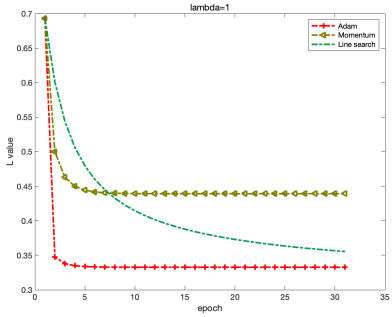

Figure 4: function value

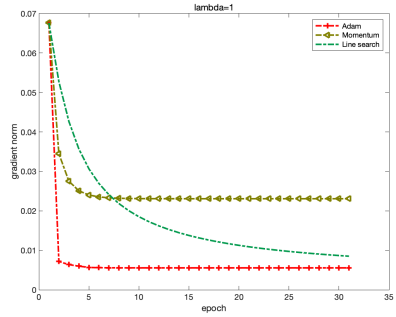

Figure 5: gradient norm

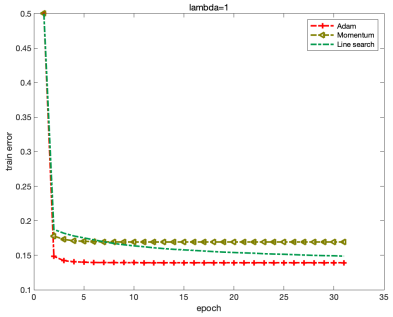

Figure 6: classification error

Results on MNIST when $\lambda = 1$

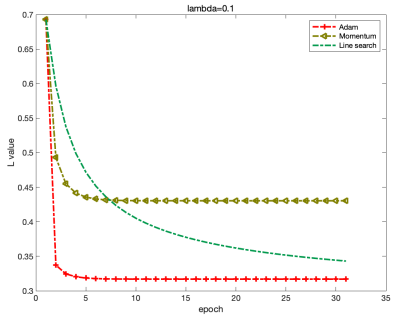

Figure 7: function value

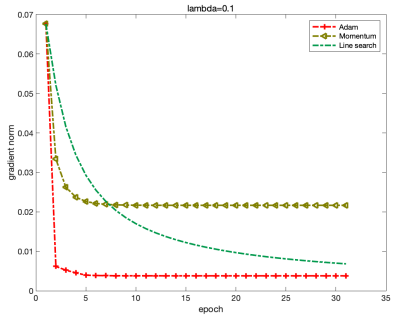

Figure 8: gradient norm

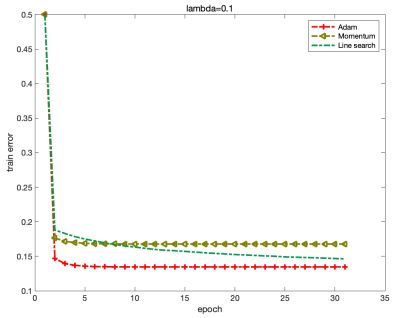

Figure 9: classification error

Results on MNIST when $\lambda = 0.1$

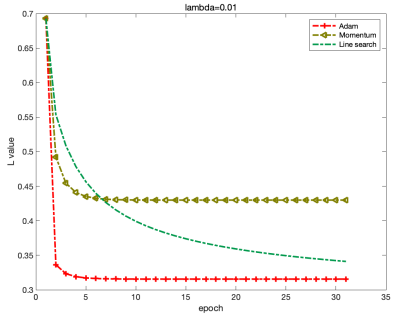

Figure 10: function value

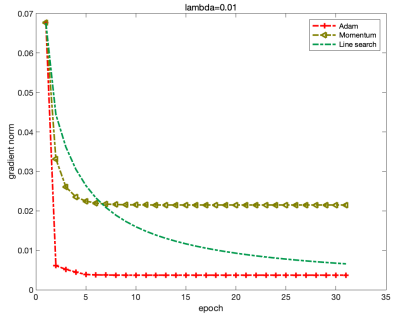

Figure 11: gradient norm

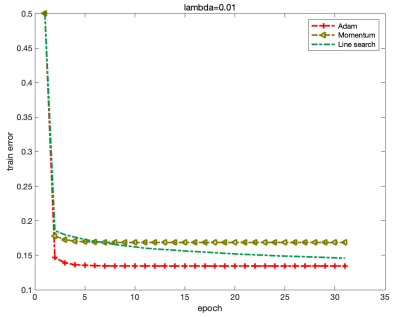

Figure 12: classification error

Results on MNIST when $\lambda = 0.01$

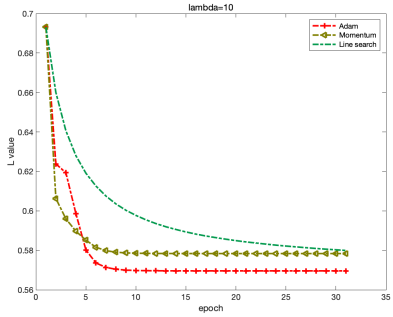

Figure 13: function value

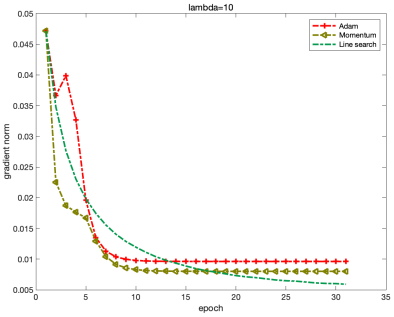

Figure 14: gradient norm

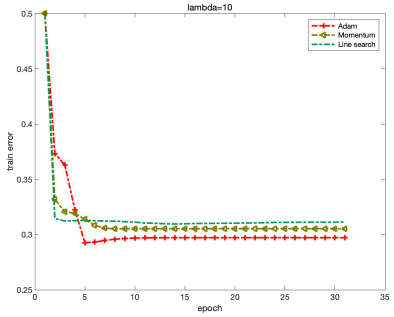

Figure 15: classification error

Results on Covtype when $\lambda = 10$

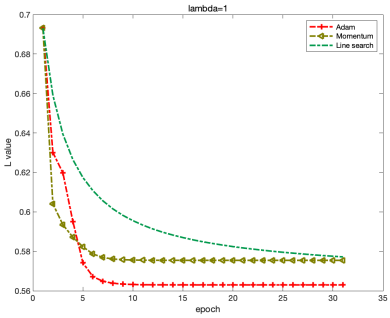

Figure 16: function value

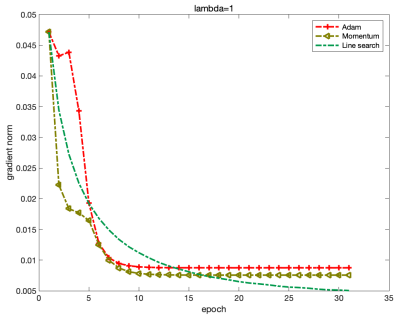

Figure 17: gradient norm

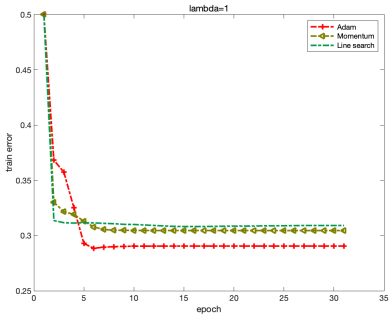

Figure 18: classification error

Results on Covtype when $\lambda = 1$

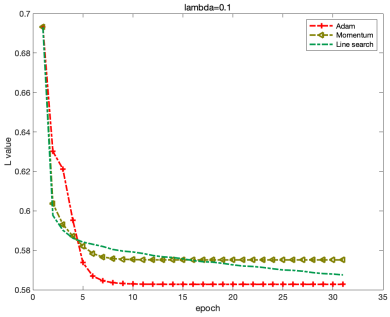

Figure 19: function value

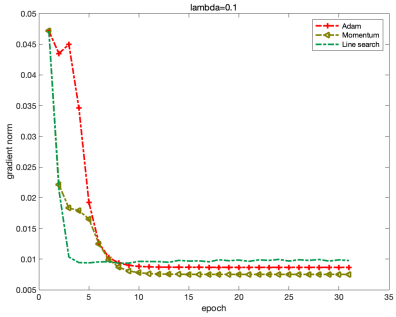

Figure 20: gradient norm

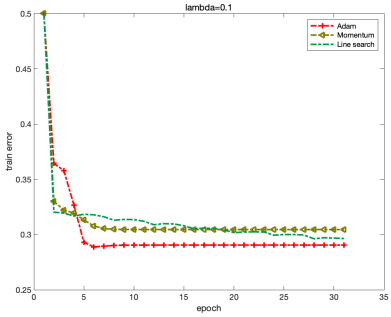

Figure 21: classification error

Results on Covtype when $\lambda = 0.1$

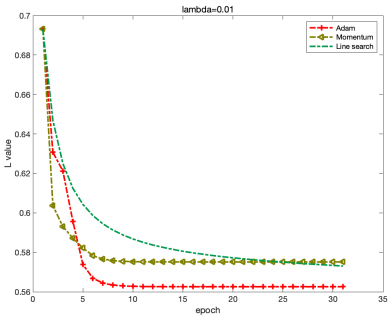

Figure 22: function value

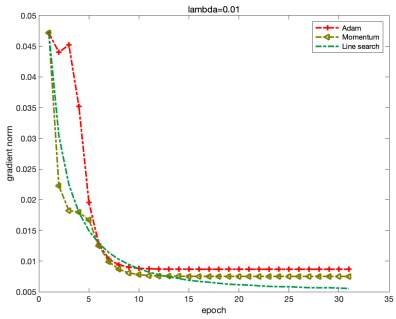

Figure 23: gradient norm

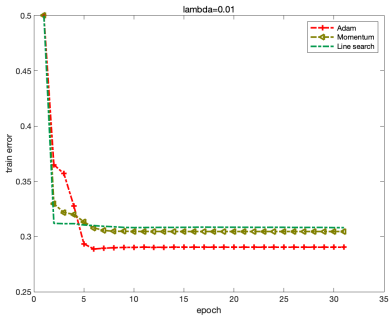

Figure 24: classification error

Results on Covtype 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 outperforms the others in both function value and classification error, which is consistent with our intuition that Adam utilizes a kind of self-adaptive stepsize on each component for acceleration. No wonder Adam is so widely used in deep learning. On the other hand, the SGD with linesearch converges faster in primal gradient norm and classification error than the Momentum algorithm. 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 function value (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.