REPRODUCIBLE REPORT OF ON THE INSUFFICIENCY OF EXISTING MOMENTUM SCHEMES FOR STOCHASTIC OPTIMIZATION

Zhong H, Sui Y, & Gao L
Department of Electrical & Computer Science, University of Southampton

ABSTRACT

This report is written for ICLR reproducible challenge 2019. In this report, we reproduced the results in *On the Insufficiency of Existing Momentum Schemes for Stochastic Optimization* published in ICLR conference 2018.

1 INTRODUCTION

In the original paper, a point was introduced that both heavy ball (HB) and Nesterov's accelerated gradient descent (NAG) method are not optimal in a stochastic first order (SFO) model Kidambi et al. (2018). Meanwhile, the accelerated stochastic gradient descent (ASGD) method can improve these two method by a factor of $\sqrt{k}$. In a SFO model, the condition number k describes the complexity of the model. With larger value of k, the model is more difficult to train Jain et al. (2017). In section 3 of the original paper, the authors mathematically proved that the learning rate of HB and NAG have a relationship with the condition number k with a scale factor $\frac{1}{k}$, which is the same as stochastic gradient descent (SGD), and the ASGD has a scale factor of $\frac{1}{\sqrt{k}}$. Hence, according to the authors, ASGD algorithm improves the rate of SGD with a factor of $\sqrt{k}$. HB and NAG do not give this improvement. Then, in section 5 of the original paper, the authors proved their theory by simulations.

In this report, we reproduced some of the results given in section 5 of the original paper [1]. The authors didn't give the source code of simulation, besides the ASGD algorithm. Hence, in our simulation, we use the PyTorch build-in function 'optim.SGD' as our SGD, HB and NAG optimization algorithms. As a result, our HB and NAG are slightly different with the HB and NAG algorithms given in Algorithm 1 and 2 from the original paper. For ASGD, we use the code given with the paper downloaded from Git-hub.

The following sections will give our reproducing results and analysis. For each section, we give the reproducing results of one sub-section from section 5 in the original paper.

2 LINEAR REGRESSION

In this section, we give our reproducing results of section 5.1, linear regression. We compared the relationship of these four algorithms (i.e. SGD, HB, NAG and ASGD) for both the cases that the training data follows a discrete distribution and a Gaussian distribution. For both training set and testing set, we generated 500 samples varying from k. The weight $w \in \mathbb{R}^2$ was from a normal distribution $N(0,5)$.

As in the original paper, we did a grid search for HB and NAG to find the optimal parameters by a 10×10 grid in $(0,1] \times (0,1]$. The optimal parameters we found was shown in Table 1. For SGD and ASGD, the given learning rates in the paper for both discrete case and Gaussian case will cause a divergence problem in our model (i.e. after several iterations, the loss goes to infinity). Thus, for discrete case, we set the learning rates of both SGD and ASGD as 0.2. For Gaussian case, we set them to 0.1. For other parameters (e.g. advantage parameter etc.), we set them as same as the original paper, from appendix C.1.

[1] The source code is available in https://github.com/COMP6248-Reproducability-Challenge

Table 1: Optimal parameters of HB and NAG

Parameters	HB (discrete)	NAG (discrete)	HB (Gaussian)	NAG (Gaussian)
Learning rate	0.2	0.6	0.2	0.1
Momentum	0.2	0.9	0.9	0.7

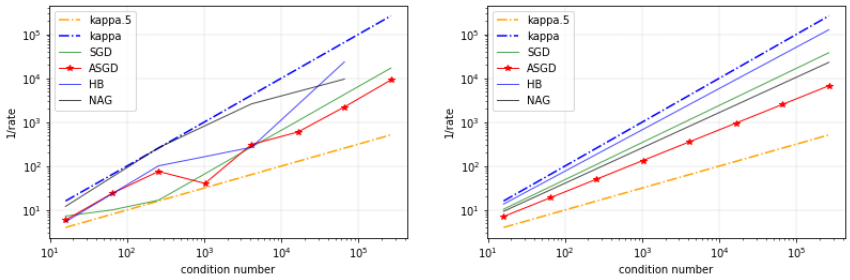


Figure 1: Discrete distribution

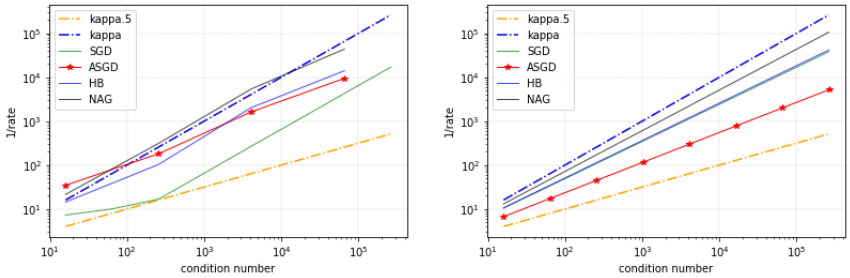


Figure 2: Gaussian distribution

Then, we run our simulations. Due to our computational recourse limitation, we only run a part of the situations (e.g. condition numbers etc.) given in the original paper. For SGD and ASGD in discrete distribution case, we choose k from $\{2^4, 2^6, ..., 2^{18}\}$. For each simulation, we run 50 times and took the mean as our result. For HB and NAG in discrete distribution case, we choose k from $\{2^4, 2^8, ..., 2^{16}\}$. For each simulation, we run 5 times. For Gaussian distribution case, we choose k from $\{2^4, 2^6, ..., 2^{18}\}$ with 50 simulations each for SGD, $\{2^4, 2^8, ..., 2^{16}\}$ with 5 simulations each for HB and NAG and $\{2^4, 2^8, ..., 2^{16}\}$ with 50 simulations each for ASGD. Figure 1 and 2 present the original results for discrete case and Gaussian case on the left. Due to we didn't run enough simulations for HB and NAG, we fit the points into a line and redraw the curves on the right with setting intercepts to 0. The slopes of them are represented in Table 2.

It was considered that if the line is closer to $kappa$, it is more likely to be a linear relationship between logarithm of conditional number and the logarithm of $1/rate$. If the line is closer to $kappa^{0.5}$, the relationship is more likely to be square root (i.e. $\sqrt{k}$). From our result, the line of ASGD doesn't superposed with $kappa^{0.5}$ like the original paper, which was considered as we didn't run enough simulations. However, in our result, it is obverse that the line of ASGD is closer to $kappa^{0.5}$ and the slope is closer to 0.5. The lines of SGD, HB and NAG are closer to $kappa$. Hence, from our result, the theorem in the original paper and the reproducible of the result were proved.

Furthermore, the reason of we set the intercepts to 0 in the re-draw graphs is that we have considered that only the slopes affect the relationships between condition number and rate. Since we didn't choose the optimal leaning rates for SGD and ASGD, the intercepts of them are not 0. With the optimal leaning rates, their intercepts are supposed to be 0 as in the original paper.

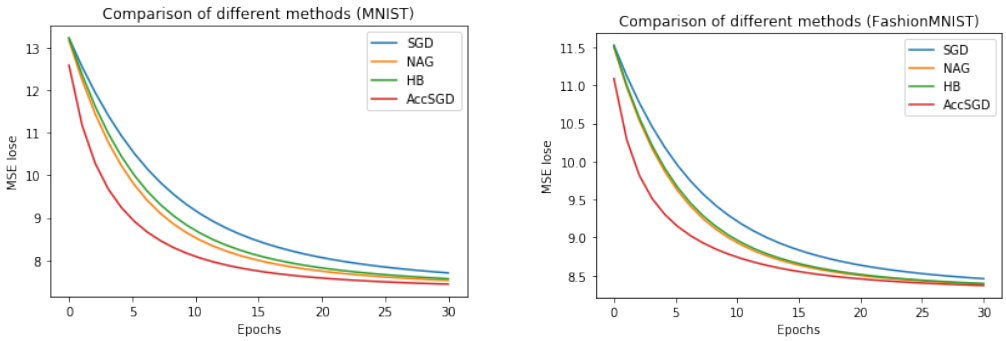
Table 2: Slopes of SGD, HB, NAG and ASGD

	SGD	**HB**	**NAG**	**ASGD**
discrete	0.8453	0.9417	0.8048	0.7064
Gaussian	0.8453	0.8528	0.9272	0.6856

3 DEEP AUTO-ENCODERS FOR MNIST AND FASHIONMNIST

MNIST dataset and a deep auto-encoder with architecture of 784-1000-500-250-30-250-500-1000-784 are implemented in the original paper. As a result, a high computational resource is required when training the models with batch size of 1. Considering batch size of 1 is an important experiment to prove the theorem, here we choose a network of 784-500-250-30-250-500-784 connections, and reduce the training set to 10000 images (i.e. randomly select 1/6 of the training set).

Grid search is applied to find the optimal parameters for each method. In our grid search, we choose our optimal parameters from a part of the parameters given in the original paper and some other reasonable parameters to save computational resource. Fig. 3 (a) gives a comparison among all the four methods with their optimal parameters with a batch size of 1. From our result, ASGD indeed outperforms SGD, HB and NAG, as the same as it in the original paper. Our loss is slightly higher than the original paper. It is because our model is simpler and our training set is smaller.



(a) MNIST (b) FashionMNIST

Figure 3: Comparison of different methods

Moreover, we change the MNIST dataset into FashionMNIST dataset to test this method, and the result is shown in Fig. 3 (b). It almost gives the same performance, and ASGD is still the best among the 4 methods.

4 DEEP RESIDUAL NETWORK FOR CIFAR 10

We trained four learning algorithms with Cifar10. In our model, we chose our network as ResNet-18 for a better computational efficiency, instead of ResNet-44 in the original paper. Meanwhile, we want to see the results under the different network.

In the first experiment, we fixed our learning rates as the same for the 4 algorithms. We train the model for a few epoch to see the loss and to choose the appropriate learning rate. We choose learning rate 0.033 and the advanced parameter, kappa are 10 and 1000.

When the batch size is 128, we run 100 epoch. In figure 4, the left figure shows that ASGD has the best performance in training and we can see four loss lines are obviously decreased at epoch 60, 90 since we cut the learning rate at epoch 60 and 90. However,the right figure shows that ASGD performs bad in validation loss when after epoch 60. Through the analysis, we choose the fix learning rate 0.033 only according to the first few (10) epochs' loss. The ASGD becomes overfitting after epoch 60 because of the bad choice of learning rate.

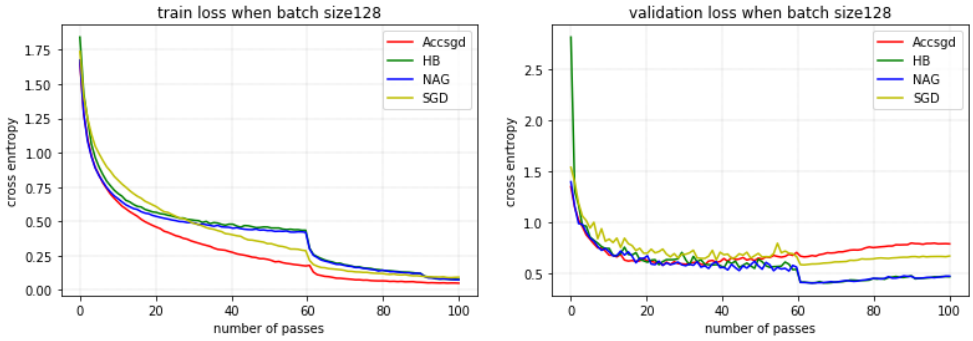


Figure 4: loss figure for minibatch size 128

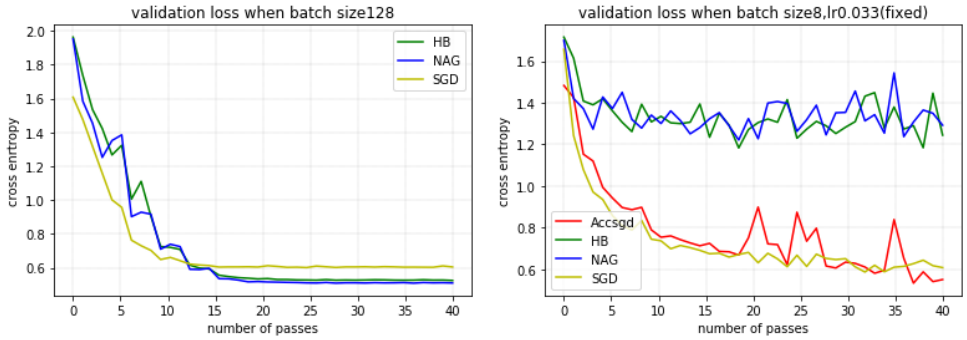


Figure 5: Effect of minibatch size.(lefet:128,right:8)

Then we change the learning rate over time. In every 3 epochs, we compare the validation loss. If validation loss cannot reduce more than 0.01 or loss is increasing, then we reduce the learning rate and divided the learning rate by 3. The initial learning rate is 0.33 and then learning rate would decay based on the loss. The minimal learning rate is 0.0000625 and advanced parameter, kappa in ASGD are 10 and 1000 respectively. From Figure 5 it can be observed that the NAG and HB outperform the SGD when minibatch size is 128 but oscillate when minibatch size is 8. The initial set of learning rate might be too big for HB,NAG when batch size 8, evev if the learning rate is decayed.

Accordingly, the effect of minibatch size can influence the performance of HB and NAG. The loss of HB,NAG can be converged well but performance are varied a lot when batch size is 8. The performance of HB,NAG are varied a lot for the different minibatch size. Also, NAG and HB are sensitive for the learning rate. Figure4 shows HB, NAG can further decrease the loss compared with ASGD and SGD when we cut the learning rate at eporch 60. But the improper initial learning rate makes it oscillate.

In conclusion, according to our results in this and previous sections, we have proved that the results given by the original paper is reproducible.

REFERENCES

Prateek Jain, Sham M Kakade, Rahul Kidambi, Praneeth Netrapalli, and Aaron Sidford. Accelerating stochastic gradient descent. *stat*, 1050:26, 2017.

Rahul Kidambi, Praneeth Netrapalli, Prateek Jain, and Sham Kakade. On the insufficiency of existing momentum schemes for stochastic optimization. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9. IEEE, 2018.