The effect of delay and momentum in Asynchronous Stochastic Gradient Descent

Yanbo Xu, Zhiyang Wu, Guillaume Wang
EPFL, Switzerland

Abstract—**Training large machine learning models on large datasets is notoriously computationally heavy. When multiple workers are available, a common algorithm to perform the training is Asynchronous Stochastic Gradient Descent (ASGD), wherein gradients of the loss on mini-batches are computed by workers and added to the model weights by the server. The workers are not fully synchronized with the server, which enables speedup by parallelization. However, this asynchronicity causes the gradients to be computed with a "delay" compared to the actual model on the server. In this work, we simulate a simple version of ASGD allowing us to study the effect of delay only. Based on experiments on real-life data, we draw the following insights on the behavior of ASGD. We confirm the adverse effect of delay on the training process. We observe that past a certain threshold, delay may prevent training entirely. For large delays, using momentum improves the performance of ASGD dramatically.**

I. Introduction

In modern machine learning practice, with the advent of deep learning on big data, the task of fitting models such as neural networks becomes more and more computationally expensive. To deal with the massive amount of training data, the most well-established optimization algorithm for deep learning is Stochastic Gradient Descent (SGD), which operates by only using a mini-batch of data at each iteration. In order to further speed up the training by leveraging distributed computing systems, a natural extension of SGD to multiple workers is Asynchronous Stochastic Gradient Descent (ASGD) [1].

In ASGD, workers compute the gradients over their own mini-batches of data, fetch and add their gradients to the global model stored on the server, and then immediately continue their training process on the next mini-batch. Previous work has shown that ASGD can significantly improve training efficiency of deep neural networks [2] compared to (synchronous) SGD. However, it also suffers from the problem of "delayed gradient" [3]. Indeed, when a worker adds its gradient $g(w_t)$ (calculated based on global model weights w_t) to the global model stored on the server, the global model may have already been updated τ times by other workers and becomes $w_{t+\tau}$.

Mathematically, adding $g(w_t)$ to $w_{t+\tau}$ does not make sense. In this paper, we want to study the effect of such "delays" on the training results. We conduct experiments to simulate a multiple-worker environment, and by controlling number of workers, we can control the length of delay. We further investigate whether adding momentum to ASGD can mitigate the negative effects of delayed gradients, and finally we compare the results of ASGD to SGD with drop-out, to learn whether ASGD acts as a regularizer.

In Section II, we present our experimental setup. In Section III, we present our results. We discuss the limitations of our experiments and propose directions for future work in Section IV. Finally, the paper is concluded with Section V.

II. Experimental setup

We consider the following simple ASGD setting [3]. A central server trains a machine learning model, e.g., a neural network, aided by m workers. Each worker i possesses a copy of the whole model with its own set of weights, $w^{(i)}$, which may not be the same as the one stored on the central server, W, at all times. At each timestep, the server communicates with one worker:

- The worker sends to the server the gradients it computed since the previous communication;
- The server updates the model by taking one step of "stochastic gradient descent", using the gradients it received;
- The server sends to the worker the updated model weights, as well a new mini-batch of training data.

Furthermore, the server communicates with the workers sequentially, i.e., at time t the server communicates with worker $i = t \mod m$. This scheme clearly simulates the ASGD algorithm described in the introduction.

In this project, we simulate the behavior of ASGD on a single machine using a simplified sequential scheme, making our experiments feasible with limited resources. However, this requires storing $m + 1$ distinct copies of the entire network, which somewhat limits the scope of our experiments to small models. Moreover, the practical computational advantage of ASGD compared to SGD is totally lost, since all computations are performed on the same machine. To properly compare the performances of different choices of m, it should be taken into account that the *real training time* when using m workers is approximately equal to the *total compute time* divided by m – assuming no communication overhead. We emphasize that our main focus is to investigate the effect of delay on the training process itself, which is why we will report the total computation time in our experiments.

We benchmark the performance of ASGD for different choices of the relevant hyperparameters using a simple toy task, training a small convolutional neural network for classification of the CIFAR10 dataset [4]. The network consists of

- A convolutional layer with 6 output layers and kernel size 5×5;
- A max-pool layer with kernel size 2×2;
- A convolutional layer with 16 output layers and kernel size 5×5;
- Three fully-connected layers of respective widths 120, 84 and 10 (the number of classes in the classification task).

Each layer is followed by a ReLU activation applied element-wise. The loss function optimized with ASGD is the cross-entropy loss of the predictions. We use mini-batches of 4 samples.

Pytorch code for the ASGD simulator, as well as to reproduce our experiments, can be found in the appendix to this report.

III. Results

A. Effect of delay

We first compare ASGD with m workers for different values of m, and with synchronous SGD (corresponding to $m = 1$), shown in Fig. 1.

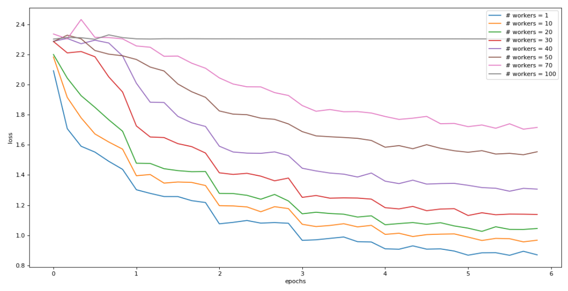
We find that as we use more and more workers, i.e., we introduce more delay, the loss decreases more slowly with respect to the total compute time. The negative effect of delay is however relatively mild for small values of m (the difference in accuracy between $m = 1$ and $m = 10$ is of the order of 5% at any time during training), which justifies the use of ASGD in practice, since the real compute time is the total compute time divided by m.

Furthermore, for very large values of m, the model seems not to improve at all over training ($m = 100$ in the reported experiment). It is striking that in this case, the loss of the model remains constant, and that the accuracy remains constant at the level of random guessing, as opposed to varying without converging.

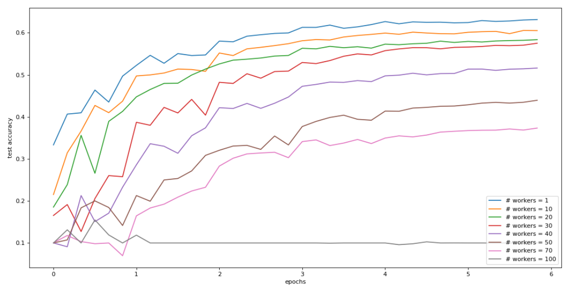
B. Effect of momentum

In ASGD, we add the gradient computed by the local worker $g(w_t)$ to the global model $w_{t+\tau}$, which makes the gradient less reliable especially when τ is large. By introducing a momentum term, we expect to reduce the negative effect when the global model receive an outdated gradient, and therefore mitigate the negative effect of delayed gradient in ASGD training. The result is shown in Fig. 2.

By adding the momentum term to ASGD and comparing it with ASGD without momentum, we find that when the number of workers is relatively small (m = 1, 10, 20, 30), there is no significant difference between these two methods. But as we continue to increase the number of workers, i.e., introduce more delays, adding momentum term can significantly mitigate the negative effect of delayed gradients, which is consistent with our expectation. Specifically, when m = 100, the model still converges, while in ASGD without momentum, the model fails to converge.



(a) Loss



(b) Test accuracy

Fig. 1: Loss and test accuracy of the model for different values of the delay m

C. Comparison of ASGD with SGD+dropout

In this section, we study whether ASGD has a similar effect as introducing a regularizer, so we compare the result of ASGD and SGD with drop-out, shown in Fig. 3.

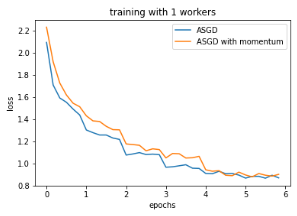
The experimental result implies that using of more workers (ie, increase m) has a very similar effect as increasing the drop-out rate (p). One possible explanation may be that ASGD and drop-out both introduce some degree of randomness to the model training, which slows down the convergence speed. And as the degree of randomness increases, the training becomes unstable and leads to an unsatisfactory training result in our experiments.

The result also suggests that ASGD may be able to prevent overfitting as a regularizer.

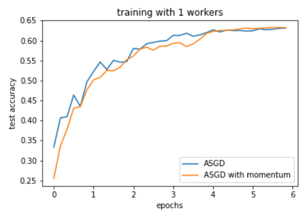
IV. Directions for future work

In future work, it would be interesting to explore the following directions.

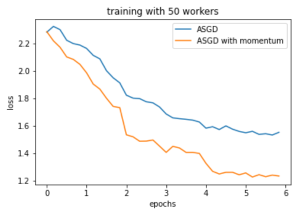
- *Verifying our observations with more reliability*: The experimental results presented in this paper, while already providing good insights about the behavior of ASGD and the effect of delay, are too limited to make a strong quantitative statement.
  We observed some variability across runs, due to the randomness in the draws of mini-batches of training data. For example, in some runs, the phenomenon where the



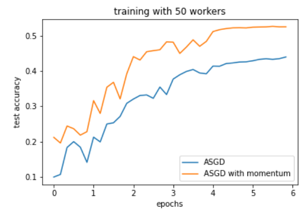
(a) Loss ($m = 1$)



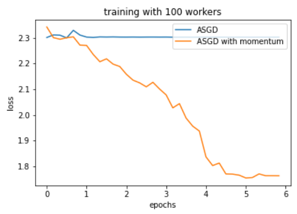
(b) Test accuracy ($m = 1$)



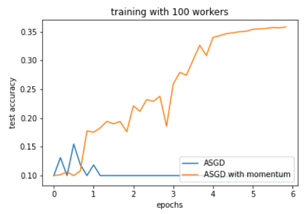
(c) Loss ($m = 50$)



(d) Test accuracy ($m = 50$)



(e) Loss ($m = 100$)



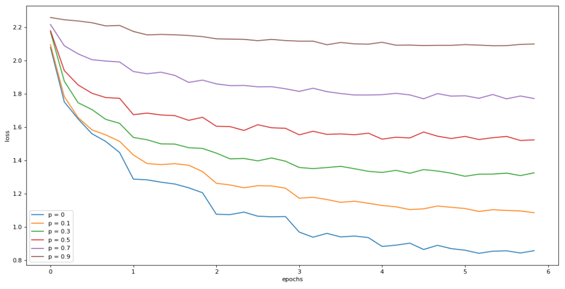
(f) Test accuracy ($m = 100$)

Fig. 2: Comparison of the loss and test accuracy between ASGD and ASGD with momentum, for different values of the delay m

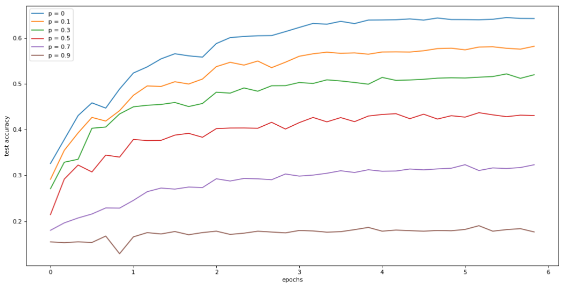
model remains constant for large m occurs already for $m = 70$.
Therefore, in order to convincingly confirm our observations, it would be necessary to run the experiments several times and to plot the mean loss and accuracy, as well as their variance.

- *Investigating the effect of the batch size*: A theoretical analysis of ASGD would be hindered by the stochasticity coming from the draw of the mini-batches. So from a theoretical perspective, it could be interesting as a first step to study the full-batch case, or Asynchronous Gradient Descent. (This would also remove stochasticity from the experiments, making them easier to interpret.) Note that Asynchronous Gradient Descent is not practical because each worker would need to do a lot of work in order to compute the gradient of the full-batch loss (this is the same reason why SGD was introduced as a replacement of full-batch GD in the first place).
  Beyond its computational advantage, training neural networks with SGD was found to yield better generalization (test accuracy) than with GD. Thus, beyond the full-batch case, it would be interesting to investigate the effect of the batch size, along with the delay m, on how well the solutions found by ASGD generalize.

An obstacle to exploring these directions would be that our simulator is quite slow. We can only train networks of limited size in reasonable time. Thus, a first step would be to actually implement ASGD with different workers, rather than simulating it with a single machine. This would also open the way to experimenting with larger models, such as ResNet.



(a) Loss



(b) Test accuracy

Fig. 3: Loss and test accuracy of SGD+dropout with different values of the dropout rate p

V. Conclusion

In this project, we investigated the effect of delay in Asynchronous Stochastic Gradient Descent (ASGD), and how it interacts with momentum. In order to isolate the effect of delay, we implemented a simple version of ASGD where the server communicates with the workers in a sequential order, which ensures that the delay is consistently proportional to the number of workers. Our experiments confirm the adverse effect of delay for the training dynamics of ASGD already observed in the literature. Furthermore, the behavior of ASGD is dramatically improved by adding momentum.

References

[1] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, M. a. Ranzato, A. Senior, P. Tucker, K. Yang, Q. Le, and A. Ng, "Large scale distributed deep networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012.

[2] J. Chen, R. Monga, S. Bengio, and R. Józefowicz, "Revisiting distributed synchronous SGD," *CoRR*, vol. abs/1604.00981, 2016. [Online]. Available: http://arxiv.org/abs/1604.00981

[3] S. Zheng, Q. Meng, T. Wang, W. Chen, N. Yu, Z. Ma, and T. Liu, "Asynchronous stochastic gradient descent with delay compensation for distributed deep learning," *CoRR*, vol. abs/1609.08326, 2016. [Online]. Available: http://arxiv.org/abs/1609.08326

[4] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.