Always Synchronize All Workers for Asynchronous Parallel Scheme via Broadcast

Yang Yutong, Chen Kailin, Tan Zhiren, Shi Hao
National University of Singapore, School of Computing, CS5260

Introduction

Federated Learning (FL) has enabled multiple mobile devices to collaboratively train a centralized machine learning model without sharing their private dataset. Most commonly used FL algorithms are synchronous(e.g. Local SGD), where the server aggregates the updated models from all involved workers and returns the aggregated global model back to them via broadcast communication. However, such synchronous algorithms are also susceptible to performance degradation in heterogeneous environments. Although asynchronous algorithms(e.g. Asynchronous Local SGD) break the limitation of heterogeneity, they can not benefit from the broadcast acceleration. Considering that broadcast is much more efficient than point-to-point in the edge learning scenario, we tried to apply broadcast to A-LSGD and proposed APSB.

Background

Problem Definition

Consider federated learning with n devices. Each device $i \in [n]$ stores a local database D_i. The overall goal is to train a global model $w \in R^d$ using data from all devices. Let $f(w; z)$ be the loss function on sample z, we define global loss function

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} E_{z^i \sim D^i} f(w; z^i)$$

so that the optimization problem could be described as $\min_{w \in R^d} F(w)$

Related Work

In A-LSGD, we denote K as the communication interval, so that all workers iterate locally for K times to generate a sequence of gradient $\{g_k^i\}$ for worker $i \in [n]$ and iteration $k \in [K]$. After a round of K iterations, worker calculates accumulated gradient $G^i = \sum_{k=1}^{K} g_k^i$ and uploads to server, where the server would use G^i to update global model $w = w + G^i$, then send the newest model w to replace the local model w_i on worker i.

Similarly, Local SGD (LSGD) is the synchronous version of A-LSGD, where the communication process is synchronized. In each round, server would wait until finish receiving and updating global model by G^i from every worker i, then broadcast new model to all workers at the same time.

NUS | Computing
National University of Singapore

Methods

APSB

Consider a single execution of federated learning with P global epochs. In round $t \in [P]$, the server receives an accumulated gradient G_i^t from an arbitrary worker $i \in [n]$, and updates the global model as following:

$$w_{t+1} = w_t + \gamma G_t^i$$

Where γ is the learning rate. Once the global model is updated, the server immediately broadcast new model to all n devices.

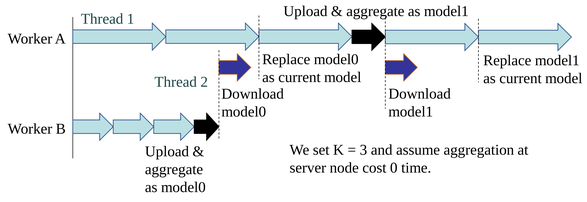
On arbitrary device i, worker uploads accumulate gradient G_t^i for every K iterations, which contributes to global model w_{t+1}. G_t^i is calculated by

$$G_t^i = \sum_{k=0}^{K-1} \nabla f\left(w_{t,k-1}^i, z_{t,k}^i\right)$$

where $z_{t,k}^i \sim D^i$ denotes the sample randomly selected in local dataset D^i. In iteration $k \in [K]$ local model $w_{t,k}^i$ is defined as

$$f(w_{t,k}^i) = \begin{cases} w_\tau, & receive\ w_\tau\ from\ server \\ w_{t,k-1}^i - \gamma \nabla f\left(w_{t,k-1}^i, z_{t,k}^i\right), & otherwise \end{cases}$$

Note that the subscript t in G_t^i and $w_{t,k}^i$ is used to distinguish which version of global model w_t they finally contribute to, and the exact value of t is unknown in practice.

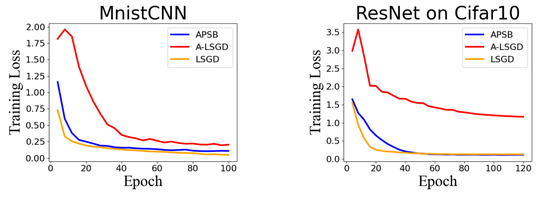


Algorithm 1 APSB

Input: Number of workers n, learning rate γ
Initialize: ω_1, $isPull = False$;
On worker i=1,…,n:
Parallel the following two threads:
Hearing Thread:
  repeat:
    Receive global model ω_t from server;
    Set the pulled flag $isPull = True$;
  until Convergence
Computation Thread:
  repeat:
    Initialize $G_{t,0}^i = 0$;
    if $isPull$ is $True$ **then**
      Update local model from server $\omega_{t,0}^i = \omega_\tau$;
      $isPull = False$;
    end if
    for $k = 0$ **to** $K - 1$:
      if $isPull$ is $True$ **then**
        Update local model from server $\omega_{t,k}^i = \omega_\tau$;
        $isPull = False$;
      end if
      Randomly sample a mini-batch $z_{t,k}^i$;
      Compute the stochastic gradient $\nabla f_i(\omega_{t,k}, z_{t,k}^i)$;
      Update param $\omega_{t,k+1}^i = \omega_{t,k}^i - \gamma \nabla f_i(\omega_{t,k}, z_{t,k}^i)$;
      Accumulate $G_{t,k+1}^i = G_{t,k}^i + \nabla f_i(\omega_{t,k}, z_{t,k}^i)$;
    end for
    Push $G_{t,K}^i$ to server;
  until Convergence
On server:
repeat:
  Receive G_t^i from any worker i;
  Update global model $\omega_{t+1} = \omega_t - \gamma G_t^i$;
  Broadcast global model ω_{t+1} to all workers;
until Convergence

Results

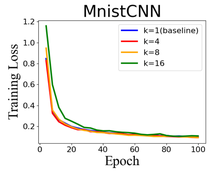
We conduct a series of experiment to elaborate the performance of APSB on CNN and ResNet. The experiment is implemented with Pytorch and execute them on heterogeneous cluster containing 9 virtual machines (with 8 workers and 1 server), with GCC Linux Red Hat 4.8.5-16 operating system and supporting OpenMPI.

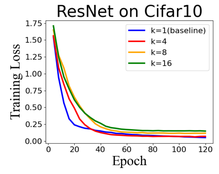


We first use MnistCNN and ResNet on Cifar10 to evaluate the performance between APSB and A-LSGD setting K = 8. Although A-LSGD could still converge for simple model like MnistCNN, for ResNet it has a high chance of convergence failure during experiment even when K = 2, making it infeasible for learning larger model like this. We also include LSGD as baseline to show that APSB can achieve a performance approaching to synchronous distributed algorithms with same setting.

Test Accuracy	APSB	LSGD	A-LSGD
MnistCNN	98.9%	99.2%	96.7%
ResNet	65.1%	68.7%	15.9%

We then compare the communication cost and accuracy in different K from 4 to 16. For Mnist all experiments with different K achieve a similar training loss and test accuracy. In ResNet model the convergence loss gets higher when K increases, which is because the gradient varieties increase when communication loss is less frequent. On the other hand, the communication cost is reduced significantly by 1/K comparing to baseline.





Comm Cost	K=1	K=4	K=8	K=16
MnistCNN (Mb)	427	104	61	26
ResNet (Gb)	260	68	32	17

Test Accuracy	K=1	K=4	K=8	K=16
MnistCNN	98.9%	99.0%	98.6%	98.4%
ResNet	72.4%	70.2%	66.7%	60.3%

Application

In the machine learning field, federated learning with wireless communications has attract increasing interest considering the availability of unprecedented amount of data and advancements in computing and parallel processing. Distributed learning scheme adapting wireless communication is particularly appealing because the traditional model-driven approaches are not rich enough to capture the growing complexity and heterogeneity of the modern wireless networks.

As an asynchronous algorithm, APSB is tolerant to devices with various computation speed to compute parallelly. The increase of communication interval K significantly alleviates the communication overhead where the data transfer speed in wireless network is usually limited. Utilizing the factor that all devices can receive broadcast operation reduces stimulatingly with the same cost of p2p transfer, APSB reduces the gradient latency and stabilizes the iteration step for large K.

Contact

Our Group: https://github.com/karnina/APSB

tswddd@gmail.com
kailinc@comp.nus.edu.sg
zhiren.tan@u.nus.edu
e0576183@u.nus.edu