Demo Abstract: Distributed Machine Learning at Resource-Limited Edge Nodes

Tiffany Tuor*, Shiqiang Wang†, Theodoros Salonidis†, Bong Jun Ko†, Kin K. Leung*
*Imperial College London, UK. Email: {tiffany.tuor14, kin.leung}@imperial.ac.uk
†IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. Email: {wangshiq, tsaloni, bko}@us.ibm.com

***Abstract*—The Internet of Things (IoT) produces an unprecedented amount of data, generated by billions of connected devices. Due to the distributed nature of IoT devices, datasets are distributed and it is often infeasible to move all the locally collected data to a centralized location. Bandwidth and storage are too limited for the transmission of raw data, or such transmission can be prohibited due to privacy constraints. Due to these constraints, distributed machine learning algorithms which work on local datasets with limited global coordination are needed. In this demonstration, we present an distributed learning system that enables edge devices to collaboratively learn a shared model while keeping all the raw data stored distributedly at the edge. The system estimates parameters related to data distribution and resource consumption, and adapts the learning process based on these estimations in real time.**

I. Distributed Machine Learning

One of the major challenges the Internet of Things (IoT) faces today is how to manage and make sense of the myriad of data generated by the billions of connected devices. Machine learning (ML) is a useful tool to extract information from large-scale data (e.g. videos, pictures). When datasets are incredibly large and distributed, traditional machine learning algorithms cannot be used, because they require the entire training data to be centralized at one location. In distributed mobile edge systems, it is impossible to send all the data to a centralized location, because the volume of data generated by the multitude of devices is too large to be sent through the communication network. Furthermore, for privacy reasons, one may not want to send his/her raw data to a central cloud. This motivates the need for distribute machine learning. Typically, distributed machine learning works as follows: each edge device has a local model and improves it by learning from the local data; the improvement is captured in the local model weights. The local weights, obtained at different edge nodes, are then sent to a single location, referred to as the *aggregator*, that aggregates and sends the updated weights back to the edge nodes for the next iteration.

In a distributed edge environment, training a learning model on data spread across nodes can be challenging. A distributed learning system requires a significant amount of resources to run over distributed data and exchange updates, to keep the global model sufficiently consistent with all the local datasets.

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

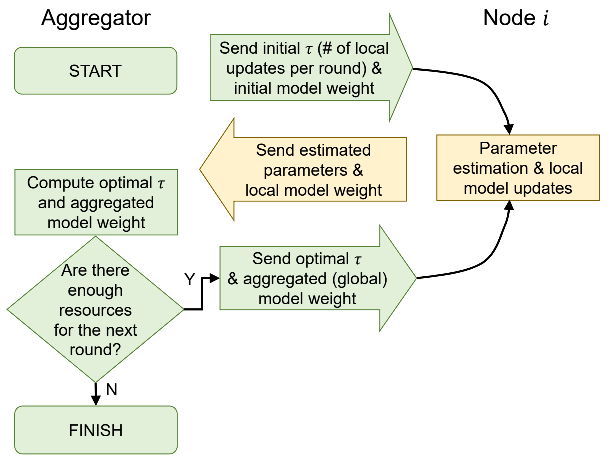

Aggregator
Node i
START
Send initial τ (# of local updates per round) & initial model weight
Parameter estimation & local model updates
Send estimated parameters & local model weight
Compute optimal τ and aggregated model weight
Are there enough resources for the next round?
Y
Send optimal τ & aggregated (global) model weight
N
FINISH

Fig. 1: Protocol for finding optimal τ.

Unfortunately, at the edge, communication and computation resources are scarce. This underlines the need for intelligent distributed learning mechanisms that efficiently uses limited resources. A distributed learning system that adapts the global aggregation frequency is presented in [1], where aggregation is performed whenever the model weight varies beyond an empirically selected threshold. The performance is evaluated using multiple cloud services running in data centers distributed across a large geographical area, where the main resource bottleneck is the communication bandwidth in the wide-area network. Our work in [2] addressed the unique challenges in edge computing environments, including: (1) both the communication and computation resources are limited; (2) the data at different edge nodes may be distributed non-uniformly due to the limited storage and strong coupling with the geographical location of edge nodes. We proposed a control algorithm for distributed learning that is derived from theoretical analysis, which learns the system and data characteristics in real time and dynamically determines the global aggregation frequency to maximize the learning accuracy for a given resource budget.

In this demonstration, we present a system that has the ability to train a machine learning model from data distributed at multiple nodes, without sending the raw data to a central place. We further demonstrate the effectiveness of our algorithm presented in [2]. This demonstration shows that our algorithm can estimate the parameters related to data distribution and resource consumption, and adapt the learning process based on these estimations in real time. The system includes a protocol (briefly summarized in Figure 1) for information exchange, which enables the distributed learning process to adapt to different data distributions and system conditions. The protocol

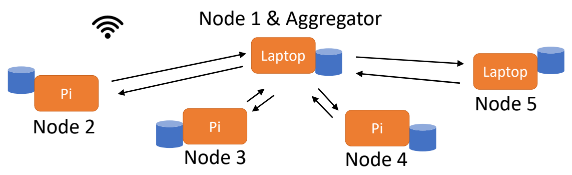


Fig. 2: Demonstration setup.

allows the nodes and the aggregator to exchange local/global model weights in order to keep the global model consistent across nodes with different data distributions. Additionally, the protocol enables the nodes and the aggregator to coordinate for adjusting the frequency of aggregating local model weights, or, equivalently, finding the optimal number of local updates (denoted by τ) between two rounds of aggregations.

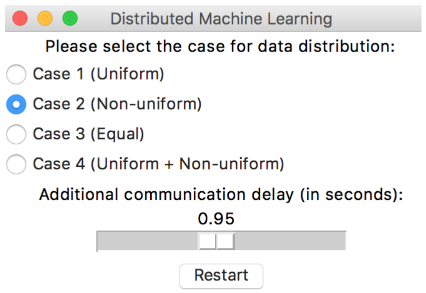
This frequency of aggregation is computed based on the estimated parameters related to the data distribution and resource usage pattern. Among these parameters, the *gradient deviation* captures the deviation between the local and global gradients evaluated on the same model weights. The larger the gradient deviation, the more frequently the system should perform aggregation. Two parameters related to the resource consumption for one local update at all nodes and for a step of global aggregation, respectively, are also estimated. Based on the estimations, the aggregator monitors the total resource consumption and compares it against the total resource budget. If the consumed resource has reached the budget limit, it stops the learning process. Note that all estimations are based on measurements taken at all edge nodes and the aggregator. The control algorithm behind the protocol is detailed in [2].

II. Description of the Demonstration

We demonstrate our distributed learning system on a networked prototype with 5 nodes (2 laptop computers and 3 Raspberry Pi devices), which are all interconnected via Wi-Fi. The system setup is shown in Figure 2. The 5 nodes have their own local datasets. As shown in Figure 2, the aggregator is located on one of the laptops and is hence co-located with one of the local datasets. The demonstration shows the training of a support vector machine (SVM) model to classify whether a handwritten digit is an even number or an odd number, where the MNIST dataset [3] is used. We consider time as the resource in our demonstration.

To demonstrate how the system adapts to different data distributions, we consider four different ways of distributing the data into nodes. In *Case 1*, data samples are randomly allocated into nodes following a uniform distribution. In *Case 2*, the data samples are non-uniformly distributed across nodes. Each node is only contains samples with two specific digits. For example, node 1 has all the samples with numbers 0 or 5. In *Case 3*, distribution is equal across nodes, the full dataset is replicated on each node. *Case 4* is a combination of uniform and non-uniform distributions. Data samples with digits 0–4 are distributed to the first three nodes following the approach of Case 1; the rest of the samples are distributed to the remaining nodes following the approach of Case 2.

An overview of the demonstration is presented in Figure 3. The user can interact with the learning system by switching



Distributed Machine Learning

Please select the case for data distribution:

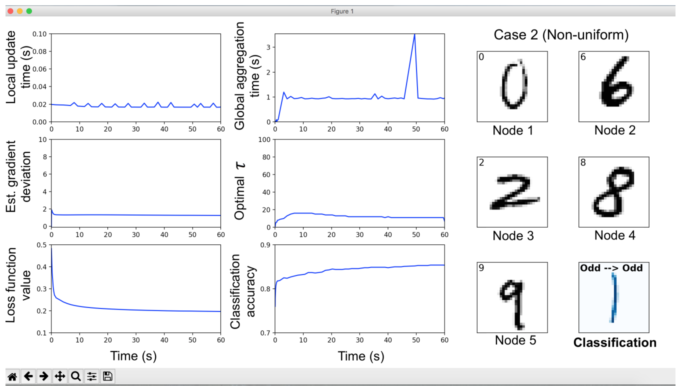
- ○ Case 1 (Uniform)
- ● Case 2 (Non-uniform)
- ○ Case 3 (Equal)
- ○ Case 4 (Uniform + Non-uniform)

Additional communication delay (in seconds):

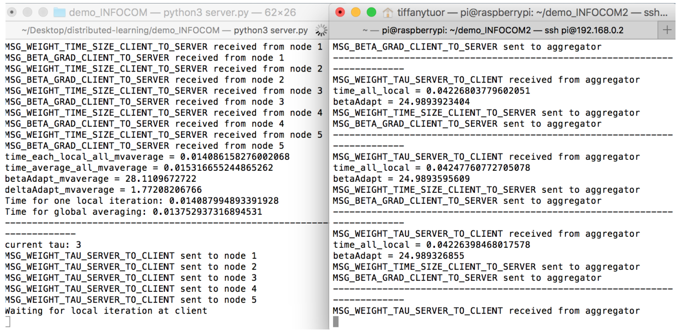
0.95

Restart

(a) Control panel



(b) Plot



```
demo_INFOCOM — python3 server.py — 62×26
~/Desktop/distributed-learning/demo_INFOCOM — python3 server.py
MSG_WEIGHT_TIME_SIZE_CLIENT_TO_SERVER received from node 1
MSG_BETA_GRAD_CLIENT_TO_SERVER received from node 1
MSG_WEIGHT_TIME_SIZE_CLIENT_TO_SERVER received from node 2
MSG_BETA_GRAD_CLIENT_TO_SERVER received from node 2
MSG_WEIGHT_TIME_SIZE_CLIENT_TO_SERVER received from node 3
MSG_BETA_GRAD_CLIENT_TO_SERVER received from node 3
MSG_WEIGHT_TIME_SIZE_CLIENT_TO_SERVER received from node 4
MSG_BETA_GRAD_CLIENT_TO_SERVER received from node 4
MSG_WEIGHT_TIME_SIZE_CLIENT_TO_SERVER received from node 5
MSG_BETA_GRAD_CLIENT_TO_SERVER received from node 5
time_each_local_all_mvaverage = 0.014086158276002068
time_average_all_mvaverage = 0.015316655244865262
betaAdapt_mvaverage = 28.1109672722
deltaAdapt_mvaverage = 1.77208206766
Time for one local iteration: 0.014087994893391928
Time for global averaging: 0.013752937316894531
--------------------------------------------------------------
-------------
current tau: 3
MSG_WEIGHT_TAU_SERVER_TO_CLIENT sent to node 1
MSG_WEIGHT_TAU_SERVER_TO_CLIENT sent to node 2
MSG_WEIGHT_TAU_SERVER_TO_CLIENT sent to node 3
MSG_WEIGHT_TAU_SERVER_TO_CLIENT sent to node 4
MSG_WEIGHT_TAU_SERVER_TO_CLIENT sent to node 5
Waiting for local iteration at client
```

```
tiffanytuor — pi@raspberrypi: ~/demo_INFOCOM2 — ssh...
~ — pi@raspberrypi: ~/demo_INFOCOM2 — ssh pi@192.168.0.2
MSG_BETA_GRAD_CLIENT_TO_SERVER sent to aggregator
--------------------------------------------------------------
-------------
MSG_WEIGHT_TAU_SERVER_TO_CLIENT received from aggregator
time_all_local = 0.04226803779602051
betaAdapt = 24.9893923404
MSG_WEIGHT_TIME_SIZE_CLIENT_TO_SERVER sent to aggregator
MSG_BETA_GRAD_CLIENT_TO_SERVER sent to aggregator
--------------------------------------------------------------
-------------
MSG_WEIGHT_TAU_SERVER_TO_CLIENT received from aggregator
time_all_local = 0.04247760772705078
betaAdapt = 24.9893595609
MSG_WEIGHT_TIME_SIZE_CLIENT_TO_SERVER sent to aggregator
MSG_BETA_GRAD_CLIENT_TO_SERVER sent to aggregator
--------------------------------------------------------------
-------------
MSG_WEIGHT_TAU_SERVER_TO_CLIENT received from aggregator
time_all_local = 0.04226398468017578
betaAdapt = 24.989326855
MSG_WEIGHT_TIME_SIZE_CLIENT_TO_SERVER sent to aggregator
MSG_BETA_GRAD_CLIENT_TO_SERVER sent to aggregator
--------------------------------------------------------------
-------------
MSG_WEIGHT_TAU_SERVER_TO_CLIENT received from aggregator
```

(c) Terminal output at the aggregator (left) and one of the nodes (right)

Fig. 3: Demonstration overview.

among the four cases of data distributions and adding some amount of additional communication delay. The model loss, accuracy, optimal τ, and other estimated parameters are plotted in real time, illustrating the process of learning. The plot also displays the content of the five nodes and the classification result of the global model on the test data. The protocol interactions between the aggregator and an edge node are shown in the terminal output. For more details, see the video at: https://youtu.be/34O7XEWgcvU

References

[1] K. Hsieh, A. Harlap, N. Vijaykumar, D. Konomis, G. Gander, P. Gibbons, and O. Mutlu, "Gaia: Geo-distributed machine learning approaching lan speeds," *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pp. 620–647, 2017.

[2] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. of IEEE INFOCOM*, 2018. [Online]. Available: https://ibm.co/2DaHbDs

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.