Scalable Distributed Machine Learning with Huge Data for IoT and Scientific Discovery

Tie (Tony) Luo Sajal K. Das
Department of Computer Science, Missouri University of Science and Technology
{tluo , sdas}@mst.edu

April 10, 2020 | Chicago, IL, USA | NSF Huge Data Workshop

I. Introduction

Data generation rates in the order of petabyte, exabyte, and even zettabyte per hour are becoming increasingly common and will soon go beyond the capacity of "Big Data" technologies. In fact, this is already a reality in some scientific fields, such as astronomy, physics, and biological sciences, and has posed a significant bottleneck to scientific discovery. Another example is Internet of Things (IoT), where billions of sensors and devices are interconnected and producing a huge deluge of data unceasingly.

Our ambitious goal is to solve this "Huge Data" problem in the context of machine learning such that models can be efficiently trained on a huge amount of data and can make fast predictions on new data samples. To this end, we sketch below a novel scalable distributed machine-learning framework.

II. Proposed Framework

A given data set or stream can be huge in either or both of the vertical and horizontal dimensions. Vertically huge means that the dataset contains a huge number of data samples (i.e., rows) or that the influx streaming rate is extremely high, while horizontally huge means that each data sample contains a huge number of features (i.e., columns). Formally, let the original data set or stream be denoted as $S = \{x_1, x_2, x_3, \ldots\}$ where x_i is the i-th data sample and $|x_i| = D$ is the size (width) of each sample, and both $|S|$ and D can be potentially huge.

To efficiently train a machine-learning model on such a data set or stream, and also make fast runtime predictions, our proposed framework consists of three layers as described below.

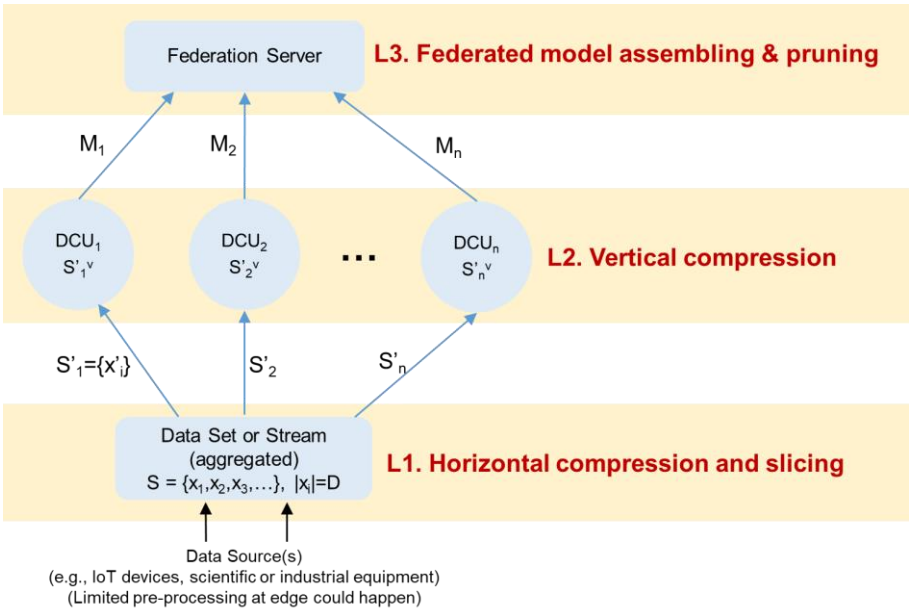

Figure 1. Proposed framework.

L1. Horizontal compression and slicing

This layer serves two purposes. First, it compresses the data S horizontally (i.e., reduce the number of rows). We make a key observation here: oftentimes, consecutive data samples have little variation except for a very small set of features, and even for those features, the variation can be characterized by a well-defined function

or probability distribution. For example, data samples are often timestamped, and each may also be associated with a counter; both the timestamp and the counter will typically have a fixed increment one after another. Therefore, a novel compression method could be developed by only keeping a "seed sample" (say, the first sample) and associating it with a small set of functions, each describing the variation of a varying feature. Thus, all the subsequent samples are drastically condensed into a single number, until another "seed sample" appears, which indicates a change that the above functions cannot describe. This way, data S is compressed into a substantially smaller set $S'=\{x'_1, x'_2, x'_3,\ldots\}$ together with a metadata set $Q = \{\{F_1, k_1\}, \{F_2, k_2\}, \{F_3, k_3\},\ldots\}$, where x'_j is a seed sample, F_j is its associated descriptive function set, and k_j is the number of data samples described by the tuple (x'_j, F_j). Therefore, the compression ratio is $r = 1 - \frac{m}{\sum k_j}$ where $m = |S'|$ is the number of seed samples. We anticipate r can be close to 1 (i.e., a large compression ratio) in many cases.

The second purpose of this layer is to slice the compressed data S' horizontally, into $S'_1, S'_2,\ldots, S'_n$ while keeping intact the width of each data sample D. The number n of these subsets S'_i is determined by the number of distributed computing units (DCUs) available, and the size of each dataset S'_i is determined by the computation and storage capacity of DCU_i and the network bandwidth between DCU_i and the host machine where S resides.

Note that, when S is a data stream rather than a static dataset, the slicing procedure is substituted by a dispatching mechanism: it does not wait for each subset S'_i to be complete but immediately sends it to a DCU once the size of S'_i reaches a certain value (can be as small as one), and this repeats. The choice of DCU can be in a round-robin manner, or adaptively based on network bandwidth and DCU workload.

L2. Vertical compression

This layer of function is performed at each DCU, serving the purpose of reducing the dimension of each sample $x'_j \in S'_i$ from $|x'_j| = D$ to d where $d \ll D$. One method is to use principal component analysis (PCA) which is a classic dimensionality reduction technique. However, such a technique is deterministic, meaning that it will always result in the same subset of reduced features for all the S'_i. This is not desired by some machine learning tasks such as *ensemble learning*, which needs *model diversity* to boost performance. Therefore, other dimensionality reduction methods are needed to address this issue. One such method we propose to explore is based on random Johnson-Lindenstrauss projection.

After vertically compressing S'_i into S'^v_i whose dimension $d \ll D$, a sub-model M_i is trained over S'^v_i on the same DCU. Here the training process needs to be tailored to accommodate the metadata introduced by our proposed horizontal compression (L1), for which we anticipate an additional "virtual expansion" procedure.

L3. Federated model assembling and pruning

This layer serves the purpose of assembling the n sub-models M_i into one global model M. The challenge is that the sub-models are heterogeneous, as they are trained on possibly non-i.i.d. datasets due to the horizontal slicing (L1) and likely have different features due to vertical compression (L2). As such, conventional distributed machine learning is no longer applicable. The recently emerged federated averaging can handle non-iid data, but does not solve the feature heterogeneity incurred by L2. Therefore, a novel method is needed which we present as an open challenge.

Due to the huge data setting, the global model M may still contain a large number of parameters and, thus, may need a *pruning* process before deployment in order to achieve fast runtime predictions. This needs to be achieved without sacrificing model performance, such as accuracy, F1-score, and AUC. There exist works on pruning deep neural networks, which can serve as a good starting point.

III. Broader Impact

Given the accelerated growth of AI and proliferation of IoT devices as well as their penetration into each other, our proposed framework and preliminary ideas would advance the state-of-the-art in scalable distributed machine learning in the emerging large-scale networking (LSN) paradigm for huge data analytics. The hurdles of storing, transferring, and processing huge amount of data could be alleviated by our approach, to the extent that scientific research activities at petabyte and exabyte data scale, as well as the gigantic IoT systems, could substantially benefit from AI which was not able to handle the huge data previously. As such, we anticipate new breakthrough in scientific discovery and technology developments that directly or indirectly benefit from this proposed framework.