Unexpectedly Useful: Convergence Bounds And Real-World Distributed Learning

Unexpectedly Useful: Convergence Bounds And Real-World Distributed Learning

Francesco Malandrino
CNR-IEIIT and CNIT
Torino, Italy

Carla Fabiana Chiasserini
Politecnico di Torino, CNR-IEIIT, and CNIT
Torino, Italy

ABSTRACT

Convergence bounds are one of the main tools to obtain information on the performance of a distributed machine learning task, before running the task itself. In this work, we perform a set of experiments to assess to which extent, and in which way, such bounds can predict and improve the performance of real-world distributed (namely, federated) learning tasks. We find that, as can be expected given the way they are obtained, bounds are quite loose and their relative magnitude reflects the training rather than the testing loss. More unexpectedly, we find that some of the quantities appearing in the bounds turn out to be very useful to identify the clients that are most likely to contribute to the learning process, without requiring the disclosure of *any* information about the quality or size of their datasets. This suggests that further research is warranted on the ways – often counter-intuitive – in which convergence bounds can be exploited to improve the performance of real-world distributed learning tasks.

ACM Reference Format:
Francesco Malandrino and Carla Fabiana Chiasserini. 2023. Unexpectedly Useful: Convergence Bounds And Real-World Distributed Learning. In *Proceedings of International Conference on Machine Learning and Computing (ICMLC '23).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

It would be hard to overstate the importance of machine learning (ML) for a growing number of aspects of technology and, indeed, of our daily lives. Furthermore, owing to the growing complexity of the learning tasks to perform, to the ever-increasing amount of resources they require, and to the need to keep data local, a lot of today's learning is *distributed*, i.e., it requires the cooperation of multiple *learning nodes*, leveraging the help of a *learning server*.

A prominent example of distributed learning is represented by the Federated Learning (FL) paradigm, which operates [1] by performing five main steps, as summarized in Fig. 1:

(1) each learning node trains a *local* model, leveraging on-device data;
(2) after one or more local epochs, learning nodes send their current model to the server;

ICMLC '23, February 2023, Zhuhai, China

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

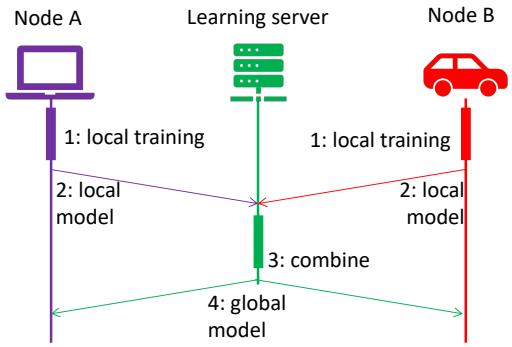

Figure 1: Main steps of each iteration of the federated learning paradigm: learning nodes train their local model (1) and send the local parameters to the server (2); the server performs a (weighted) averaging of the model (3) and sends the global parameters back to the learning nodes (4).

(3) the server creates a *global* model by combining, e.g., averaging [1], the local models it receives;
(4) the server sends the global model back to the learning nodes;
(5) the learning nodes replace their local models with the global one, and resume training from step 1 above.

In FL – as in all types of distributed optimization – the overall performance of learning chiefly depends upon three factors [2, 3]:

(1) how long each iteration (step 1 in Fig. 1) takes;
(2) how much network delay (steps 2 and 4 in Fig. 1) is incurred;
(3) how much the learning progresses at each iteration, hence, how many iterations are needed.

The first two factors are widely studied, comparatively well understood, and relatively easy to estimate with a good level of accuracy. The third factor, instead, is much harder to assess; indeed, how well a learning model (e.g., a deep neural network (DNN)) can learn depends upon many factors, several of which are unknown *a priori*.

The most promising efforts towards modeling and estimating the progress of learning tasks focus on *convergence bounds*, i.e., upper bounds on the *loss* achieved by a given model by a certain training epoch. Such bounds may account for features of the model being trained (e.g., the number of parameters therein), the loss function (e.g., its smoothness), and the datasets being learned from (e.g., their size). Since they establish upper bounds on the loss, works on convergence analysis must account for the *worst-case* scenario, i.e., they describe the behavior of the model under the most unfavorable possible conditions.

In this paper, we aim at bridging the gap between theoretical works on convergence and real-world distributed learning. Our main contributions are twofold:

- first, we assess how accurately convergence bounds capture the qualitative *and* quantitative behavior of concrete distributed ML;
- second, we find that, while the bounds themselves have a loose relationship with the actual loss evolution, the quantities needed to compute the bounds can identify the learning nodes where local iterations yield the most substantial learning improvement [4–7].

The latter aspect is linked to the problem of *selecting* the nodes that can best contribute to the distributed training, whilst reducing the overhead [7].

In the remainder of this paper, Sec. 2 describes the convergence bounds we consider as a reference and our experimental setup, while Sec. 3 presents our experimental analysis and discusses our main findings. Finally, Sec. 4 concludes the paper and sketches directions for future research.

2 REFERENCE BOUNDS AND EXPERIMENTAL SETUP

In the following, we discuss the convergence results we compare against, as well as our experimental setup.

2.1 Convergence bounds

Among the many works dealing with distributed learning convergence, we take [8] as a reference. The main reason for our choice is that the bounds presented in [8] account for multiple aspects of the learning scenario, hence, they are (i) more suited to assess the impact of each factor, and (ii) potentially, tighter.

Under the assumptions that all learning nodes participate in the learning process, they are equally weighted, and one local epoch is performed for each FL iteration, [8] proves that the difference between expected loss $\mathbb{E}\left[F(t)\right]$ at iteration t and minimum loss F^* is given by:

$$\frac{8L/\mu}{(t-1+8L/\mu)}\left(\frac{16G^2}{\mu}+4L\mathbb{E}\|\mathbf{w}_1-\mathbf{w}^*\|\right). \tag{1}$$

In (1), bold letters denote vectors; also,

- μ is a non-negative quantity such that loss function F is μ-strongly convex;
- L is a non-negative quantity such that loss function F is L-smooth;
- G is a non-negative quantity such that the squared norm of the gradients of loss function F is bounded by G^2.

Recall that, as reported in [8], a loss function, F, is μ-strongly convex if there exists a quantity $\mu \geq 0$ such that, for any possible model parameters $\mathbf{u}$ and $\mathbf{v}$,

$$F(\mathbf{u}) \leq F(\mathbf{v}) + (\mathbf{u}-\mathbf{v})^T\nabla F(\mathbf{v}) + \frac{L}{2}\|\mathbf{u}-\mathbf{v}\|_2^2. \tag{2}$$

Similarly, F is L-smooth if there exists a quantity $L \geq 0$ such that, for any possible model parameters $\mathbf{u}$ and $\mathbf{v}$,

$$F(\mathbf{u}) \geq F(\mathbf{v}) + (\mathbf{u}-\mathbf{v})^T\nabla F(\mathbf{v}) + \frac{\mu}{2}\|\mathbf{u}-\mathbf{v}\|_2^2. \tag{3}$$

As better detailed below, quantities μ, L, and G can be computed locally at each node, by repeatedly choosing $\mathbf{u}$ and $\mathbf{v}$, and studying how the corresponding model instances perform over the local datasets. However, the bound in (1) depends upon their global values; e.g., the global G will be the largest of the G-values computed by each learning node. Additionally, it is worth noting that the local values of μ, L, and G can be exploited to predict the loss improvement achieved by each individual node during local epochs, as set forth below.

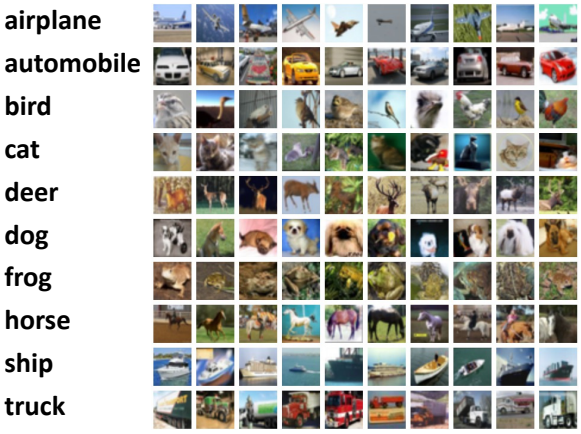

Figure 2: Classes and example images of the CIFAR-10 dataset [9].

2.2 Experimental setup

We carry out our experiments by performing an image classification task over the CIFAR-10 dataset [9], containing a total of 60,000 images belonging to 10 different classes. To perform the classification, we leverage a DNN including two convolutional layers and three fully-connected ones, as per [10], for a total of over 60,000 parameters. The dataset is partitioned into a *testing* set of 6,000 images and a *training* set of 54,000 ones; the latter is further partitioned into *local datasets* associated with the individual learning nodes.

At each local node, we compute μ, L and G as follows:

(1) we extract two random sets of parameters $\mathbf{u}$ and $\mathbf{v}$;
(2) we compute the resulting loss values $F(\mathbf{u})$ and $F(\mathbf{v})$ over the local datasets;
(3) we compute the gradients of the loss $\nabla F(\mathbf{u})$ and $\nabla F(\mathbf{v})$;
(4) we compute the value $m = 2\frac{F(\mathbf{u})-F(\mathbf{v})+(\mathbf{v}-\mathbf{u})^T\nabla F(\mathbf{v})}{\|\mathbf{u}-\mathbf{v}\|_2^2}$ and store it;
(5) we compute the value $g = \sqrt{\|F(\mathbf{v})\|^2}$ and store it;
(6) we repeat the above steps starting from (1) until a sufficiently large number of samples has been collected.

After all samples have been collected, (i) as per [8, Assumption 1], μ is set to the smallest of the m-values; (ii) as per [8, Assumption 2], L is set to the largest of the m-values, and (iii) as per [8, Assumption 4], G is set to the largest of the g-values.

We compare three different learning scenarios, changing the number of nodes and the quality of their data. In the basic scenario (labelled as "5 nodes" in the plots shown in Sec. 3), there are five learning nodes, each with 2,000 images drawn from the CIFAR-10 dataset. We then consider a richer scenario (labelled as "10 nodes" in the plots) where we double the number of learning nodes. Finally,

we consider a more challenging scenario (labelled as "5 nodes, missing class" in the plots), where there are five learning nodes, each has 2,000 images, *and* all samples of one class (namely, *ship*) are missing from all training sets.

3 EXPERIMENTAL ANALYSIS AND MAIN FINDINGS

The first aspect we are interested in is the extent to which bounds match, qualitatively and quantitatively, the behavior of the actual loss. To this end, Fig. 3 shows the evolution of the training loss (Fig. 3(a)) and of the testing loss (Fig. 3(b)), and the bounds thereto (Fig. 3(c)). Looking at Fig. 3(a) and Fig. 3(b) and comparing the blue solid line and the red dotted line therein, we can observe that, as expected, having more learning nodes increases the training loss (i.e., intuitively, it is harder to converge to a good model) but decreases the testing one (i.e., the resulting model works better with hitherto unknown data). The yellow dashed lines, describing the effect of removing a whole class from all training datasets, show very different effects on the testing and training loss. Having fewer classes to learn makes training easier (hence, a lower training loss in Fig. 3(a)). However, the resulting model performs very poorly over the testing set (hence, a higher loss in Fig. 3(b)). Both these effects make intuitive sense and are routinely observed in similar scenarios.

More interestingly, Fig. 3(c) depicts the loss bounds, i.e., the value of (1), for the three scenarios. By looking at the scale of the y-axis, the first thing we can notice is that bounds are orders of magnitude larger than the corresponding loss values – which is to be expected, as bounds have to account for the worst possible conditions over *all* choices of $\mathbf{u}$ and $\mathbf{v}$. Perhaps more relevant, the *qualitative* relationship between the bounds of different scenarios follows neither the train losses in Fig. 3(a) nor the testing losses in Fig. 3(b). Furthermore, the bounds provide no warning about the serious problems arising from whole categories missing in the training set (yellow lines in Fig. 3(a) and Fig. 3(b)).

However, a more detailed analysis surprisingly shows that, although the bounds themselves cannot be directly used to predict and improve the performance of real-world ML tasks, some of their components can be very useful. In Fig. 4, we examine the relationship between the three quantities we compute to determine the bounds, i.e., μ, L, and G, and the *usefulness* of each node within the cooperative training. The usefulness metric is defined [7] as average improvement in testing loss achieved by learning nodes during their local iterations; the underlying intuition is that nodes with a larger usefulness "push" the learning further during their local epochs.

It is interesting to notice how L and (to a lesser extent) G have a strong correlation with node usefulness. It follows that computing *local* values of such quantities can significantly help identify the nodes that are more likely to give a better contribution to the cooperative learning, a very important problem in all distributed learning scenarios. Even more importantly, computing and sharing these quantities require nodes to disclose *no information* about the size and quality of their dataset, which is instead required by many existing node selection schemes and may result in privacy leakage.

Another very interesting aspect we can notice from Fig. 4 is that *higher* values of both L and G are associated with *higher* usefulness; however, as per (1), high values of both L and G make the value of the bound larger, i.e., indicate a worse learning. We can make sense of this apparent contradiction by remembering to what exactly the bound in (1) refers, that is, the *training* loss. Intuitively, a good way to obtain a low training loss is to have a small training dataset, with samples that are not too different from each other. Indeed, moving to a degenerate scenario, a dataset with only *one* class represented therein can be learned with zero training loss by a DNN always predicting that class. On the other hand, generalization (hence, good performance over the testing set) requires larger datasets of higher quality, which may require more training epochs, thus, incur a higher training loss.

This discrepancy also points at a higher-level aspect that it is essential to keep in mind, in order to understand and leverage convergence results: bounds are based upon the analysis of the behavior of the stochastic gradient descent (SGD) *optimization* algorithm. While optimization is a fundamental part of ML, ML is much more than optimization; therefore, there are many aspects of ML that convergence bounds, by their nature, cannot capture.

A further example is shown in Fig. 5, presenting the cumulative density function (CDF) of the quantity G obtained by selecting random $\mathbf{u}$ and $\mathbf{v}$ values (blue dotted curve in the plot) and the G-values observed during actual training (red solid curve). We can immediately see that the values obtained from random $\mathbf{u}$ and $\mathbf{v}$ values are over an order of magnitude larger than those actually observed during training. Recalling that bounds must hold for even the largest possible values of G, i.e., the top point of the blue curve, this explains why the bounds in Fig. 3(c) are as loose as they are.

This also ties with our earlier remark about the intrinsic limit of convergence studies, i.e., there are aspects of ML that simply cannot be captured by convergence studies. In the case of Fig. 5, the gradients (hence, the G-values) encountered during training are relatively small precisely because a lot of effort and research in the field of ML, e.g., DNN initialization schemes, learning rate adaptation algorithms, etc., have been devoted to keeping gradients low. In other words, one may say that convergence bounds capture the optimization aspect of ML, but not the many techniques used in ML to make the optimization perform better.

4 CONCLUSION AND FUTURE WORK

In the context of distributed learning, it is of paramount importance to estimate how many training epochs will be needed to reach the target learning performance; this, in turn, depends upon how much the loss function can be reduced in a single epoch. Convergence analysis results, based on the performance and behavior of SGD, are a very valuable tool to estimate this important quantity *a priori*, that is, before actually starting to train the network.

In this work, we have leveraged a set of experiments based on federated learning to (i) assess to which extent the bounds reflect, qualitatively and quantitatively, the behavior of actual DNN training, and (ii) whether the quantities appearing in the bounds can be leveraged to improve the performance of distributed learning. Our major findings can be summarized as follows:

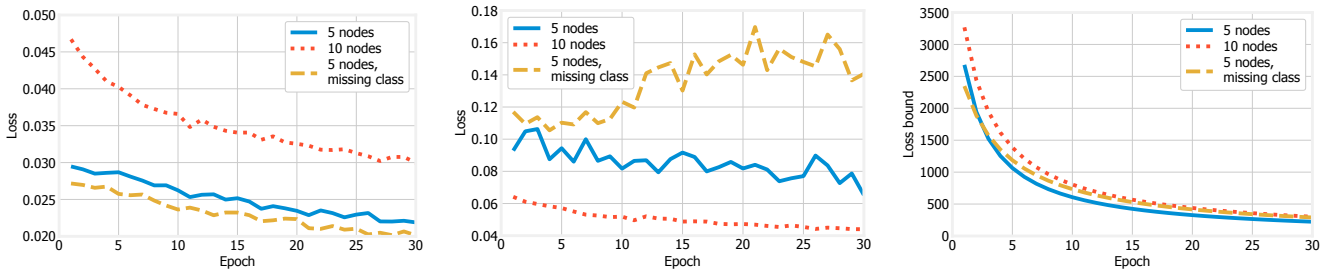

Figure 3: FL experiments: loss achieved during the training (a) and testing (b) phase; bounds thereto (c).

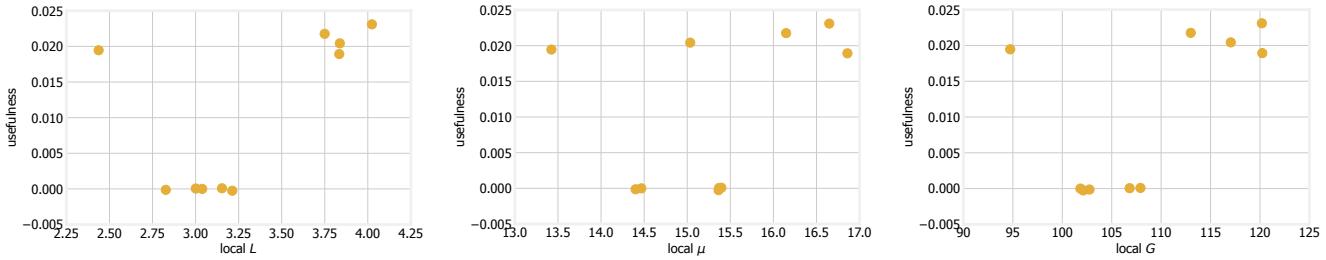

Figure 4: FL experiments: relationship between the node usefulness and the local values for the L (a), μ (b) and G (c) quantities.

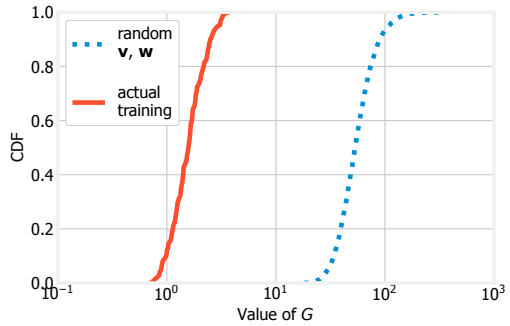

Figure 5: FL experiments: distribution of the values of G measured through random u and v values (blue) and during actual training (red).

- the full convergence bounds have only a loose relationship with the qualitative and quantitative evolution of the testing and training losses;
- nonetheless, the *quantities appearing therein can be very useful* to identify the learning nodes where local updates yield the largest loss reduction.

The latter metric is linked to how effectively each node can contribute to the learning process [7], hence, it is also useful towards more effective node selection.

Our results also highlight a fundamental feature of all convergence studies, i.e., that they can well capture the behavior of the optimization component of ML, while it is much harder for them to account for the techniques used in ML to improve the performance of optimization. In spite of this inherent limitation, as noted above, theoretical convergence studies can be very valuable in identifying the most suitable nodes to participate in the distributed ML process, thereby improving the performance of the learning itself.

Future work will focus on leveraging such insights to build a concrete algorithm for the selection of learning nodes, and evaluate its performance over a wide set of datasets and DNN architectures.

REFERENCES

[1] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," *arXiv preprint arXiv:1511.03575*, 2015.

[2] F. Malandrino, C. F. Chiasserini, N. Molner, and A. De La Oliva, "Network support for high-performance distributed machine learning," *IEEE/ACM Transactions on Networking*, 2022.

[3] F. Malandrino, C. F. Chiasserini, and G. Di Giacomo, "Energy-efficient training of distributed dnns in the mobile-edge-cloud continuum," in *IEEE/IFIP WONS*, 2022.

[4] T. Nishio and R. Yonetani, "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge," in *IEEE ICC 2019*, 2019.

[5] A. Imteaj and M. H. Amini, "Fedar: Activity and resource-aware federated learning model for distributed mobile robots," in *IEEE ICMLA*, 2020.

[6] C. W. Zaw, S. R. Pandey, K. Kim, and C. S. Hong, "Energy-aware resource management for federated learning in multi-access edge computing systems," *IEEE Access*, 2021.

[7] F. Malandrino and C. F. Chiasserini, "Federated learning at the network edge: When not all nodes are created equal," *IEEE Communications Magazine*, 2021.

[8] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," in *International Conference on Learning Representations*, 2019.

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

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.