Optimization for Machine Learning: Mini-Project
Convergence of Decentralized SGD under Various Topologies

Adrien Vandenbroucque (258715), Paul Griesser (274918), Robin Zbinden (274236)
EPFL, Switzerland

***Abstract*—Decentralized machine learning optimization, where training data is distributed over several computers, is a very promising paradigm both in term of data privacy and scalability. There already exists algorithms for this purpose, but understanding how they behave in term of convergence given different settings is essential for further optimization. In this report, we review Choco-SGD [1], an algorithm for decentralized stochastic gradient descent, and explore how different communication topologies and number of machines influence the convergence of this algorithm.**

I. Introduction

Decentralized machine learning is a major topic where research has been very active in the last few years. Recently the algorithm "Choco-SGD" was proposed in [1], which outperforms previous state-of-the-art algorithms in terms of convergence. It is a gossip-based stochastic gradient descent algorithm that runs in a decentralized setup with data samples being distributed over n machines which can only communicate to their direct neighbors on a fixed communication graph. At each iteration of the algorithm, every machine (i) computes locally a stochastic gradient step and updates its model parameters accordingly, (ii) communicate a compressed version of this update to their neighbors, (iii) and finally re-update their parameters according to the messages received. The complete algorithm can be found in [1, Algorithm 2].

The goal of this work is to gain more insight on how the Choco-SGD algorithm behaves in term of convergence for different communication graphs (a.k.a topologies). This knowledge is essential in order to either build reliable communication networks, or to predict the behavior of the algorithm on already existing ones.

To this end, we will run several experiences simulating Choco-SGD for different communication graph topologies and discuss their convergence. Firstly, we will explore convergence behavior for large number of machines on topologies considered in [1], and extend the knowledge to yet unexplored topologies (section II-C). Secondly, we analyze the convergence of the algorithm for topologies that contain sparse cuts in order to gain a greater understanding of how topology may affect the convergence quality (section II-D). Finally we explore convergence for more general topologies in order to extend practical results from [1]. In particular, we consider non-symmetric gossip matrices (section II-E).

II. Experiments

A. Dataset

In our experiments, we use a dataset from the CERN simulating collision of protons, and the machine learning task is to identify whether a signal of the Higgs boson is detectable [2]. The dataset contains pairs $\{(\mathbf{a}_i, b_i)\}_{i=1}^N$, $\mathbf{a}_i \in \mathbb{R}^D, b_i \in \{-1, 1\}$ where $N = 250,000$ and $D = 30$.

B. Setup

To asses the performance of the decentralized optimization, we perform logistic regression defined as $\frac{1}{N}\sum_{i=1}^{N} \log(1+\exp(-b_i \mathbf{a}_i^T \mathbf{x})) + \frac{1}{2N}||\mathbf{x}||^2$. The dataset is randomly and evenly distributed over n machines, where n depends on the experiments.

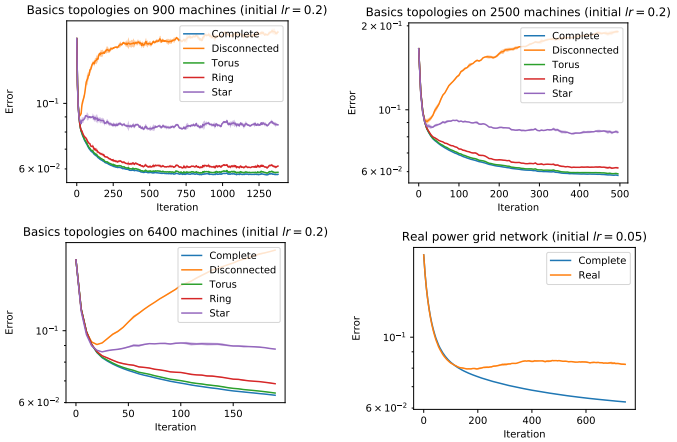

Figure 1. On the top row and in the bottom-left corner, convergence on various topologies and number of nodes. At the bottom-right corner, convergence on real power grid network topology and on the complete graph with same number of nodes ($n = 4941$).

In all subsequent experiments, we compare the loss of the decentralized method in different setups against a baseline classifier from the scikit-learn package [3] which performs the optimization using stochastic gradient descent with logistic loss. On the plots, we always show the difference between the loss obtained by the decentralized optimization and the loss obtained by the baseline as a function of the iteration number. Each experiment is repeated three times and the plots show the mean together with the standard deviation which is represented by the area around the mean value. The learning rate is updated at each iteration following the "Bottou" update [4]. For the Choco-SGD algorithm, we focus on full communication of the gradient in step (ii) of the algorithm, as we are more interested to see the general behavior of the decentralized algorithm when changing the number of nodes and the topology. Moreover, we set the "consensus learning rate" to 1. Finally for all topologies, unless stated otherwise, we assume that there are self-loops at each node in the underlying Markov chain.

C. Basic Topologies and Real Network Topology

Here, we want to see the impact of a large number of node on the decentralized SGD. We compare the convergence of the algorithm for different topologies: torus, ring, fully connected, disconnected and star (see Figure 1). We use $n \in \{900, 2500, 6400\}$. The star topology consists of a central node connected to all the others, while other nodes are only connected to the central node, and disconnected topology corresponds to no communication between the nodes. The other topologies are similar to the ones in [1]. We train for 5 epochs, where one epoch corresponds to N/n iterations.

From the plots, we observe that for the disconnected graph, the algorithm is not able to reduce its loss anymore after a few iterations. For the ring and torus topologies, we get the same type of behavior observed in [1], that is, a better convergence for the torus than for the ring, but still not as good as the complete graph.

Unfortunately, in real life, the encountered topologies are often not that simple. We therefore choose to run Choco-SGD (with initial learning rate 0.05) on a network representing the topology of the Western States power grid of the United States [5], which consists of 4941 nodes and 6594 edges. We observe on Figure 1 that even on this unstructured graph, Choco-SGD obtains good results but seems to reach a limit that the complete graph topology exceeds. In fact, when running with a higher learning rate, one will see that both curves actually are almost the same, but cannot reach an optimum as good as when the learning rate is low. This shows again how the topology seems to impact the convergence of the optimization algorithm, but also how the learning rate has an important role to play.

D. Information Bottleneck in Graphs

The previous experiments reveal some relationship between topology and convergence speed. In this section, we try to understand this relationship better through further experimentation, and first build an intuition on why this convergence behavior is observed. We ask in particular the following question: If there are very sparse cuts in the graph, does the information "flows" well in the network and does it affect the convergence rate of decentralized SGD?

Note that usually, analyses are focused on understanding the mixing time of the underlying Markov Chain through the use of the spectral gap, and it is well-known that the spectral gap gives information about sparse cuts. Indeed, these are related through Cheeger's inequality [6]. A theoretical result of how the spectral gap is related to the mixing time is given in section II-E.

To check whether this is observed in practice, we decide to run Choco-SGD on four different graphs: the complete graph, the barbell graph, the torus graph and the path graph. The barbell graph on n vertices consists of two fully connected components of size $n/2$, which are connected together with one edge. The path graph corresponds to the case where the vertices form a chain, and so each can communicate with their two neighbors except the two nodes at each end of the chain. To make it more interesting, we choose a high number of nodes ($n = 8100$) in the hope of better observing the behavior of how the information flows through these networks. Our expectations are the following: The convergence speed should be best for the complete graph as nodes are fully connected, followed by the torus, then by the barbell graph since it has a cut of size one which may be a bottleneck, and finally the path graph since it has many cuts of small sizes. The results are shown in Figure 2 for two different initial learning rates.

On the left plot, when the initial learning rate is small enough and equals 0.1, the topology seems to not influence the convergence that much. This is probably due to the fact that since the optimization algorithm makes very little progress at each step, even the topologies which are not very well connected like the path graph are able to follow the same convergence such as the complete graph. However, things become more interesting when one chooses the initial learning rate to be a bit higher, for example 0.4. In this case, we see on the right plot the differences between topologies. For example, the path graph becomes worse since the information is not able to flow as fast as the optimization is making progress. The torus is substantially better, but something interesting can be seen for the complete and barbell graph. Indeed, they follow almost the exact same convergence

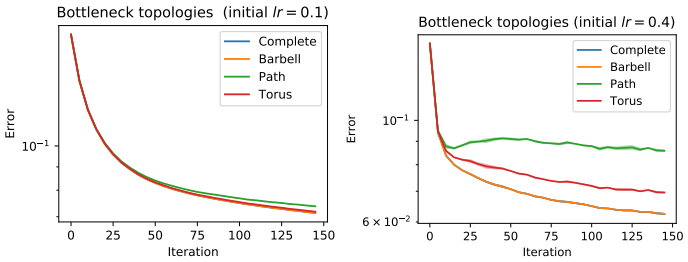

Figure 2. On the left, the convergence of various topologies for initial learning rate 0.1. On the right, the convergence of the same topologies for initial learning rate 0.4

Graph	Complete	Barbell	Torus	Path
Spectral gap ($n = 64$)	1	0.0018	0.0355	0.0008
Diameter	1	3	$2(\sqrt{n}-1)$	$n-1$

Table I
SPECTRAL GAPS AND DIAMETERS OF VARIOUS GRAPHS ON n VERTICES

behavior, which suggests that even though the barbell graph contains a cut of size one which splits it into two complete graph, the information still flows well between vertices. For reference, the spectral gaps of each of these topologies is shown in Table I for $n = 64$ nodes, from which we once again note that it does not fully explain the results obtained.

Since this is not concluding, we deviate from the usual route which consists in analyzing the Markov Chain and try to explain the observed behavior in an alternative way using information-theoretic tools instead.

In [7], they consider the problem of distributed function computation over a network of point-to-point channels, and provide lower bounds on the computation time required by any algorithm. In particular, they provide what they call cutset analyses, and present new results by using a multi-cutset analysis, an interesting one being that the minimum number of steps T used by any algorithm is lower bounded by a function that scales linearly with respect to the diameter of the network. Recall that the diameter of a graph is a distance measure defined as the longest shortest path between any two vertices of the graph.

Going back to our experiment, let us check whether the graph diameter might explain the results obtained (Figure 2). Comparing to the diameters of the graph used in the experiment (shown in Table I), we see that the convergence behavior clearly relates to this measure. In particular, the behavior of the barbell graph following the convergence of the complete graph seems to be due to the fact that their diameters are almost the same and very small. Also, one can deduce why the convergence of the torus and the path graph is not able to reach results that are as good as the complete graph since their diameter is not constant but grows as a function of the number of vertices. We thus conclude that in the setup of this experiment, the graph diameter is a measure which explains well the results obtained.

E. Extending to More General Stochastic Matrices [1]

In Definition 1. of [1], it is assumed that the "gossip matrix" W representing the topology is symmetric, making it a doubly stochastic matrix. Since the matrix is doubly stochastic, this means that the stationary distribution of the Markov chain is the uniform one, which

[1]All results on Markov chains stated in this section come from [8].

$$W = \begin{bmatrix} p_1 & p_2 & 0 & \dots & 0 & 0 & p_3 \\ p_3 & p_1 & p_2 & \dots & 0 & 0 & 0 \\ 0 & p_3 & p_1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & p_1 & p_2 & 0 \\ 0 & 0 & 0 & \dots & p_3 & p_1 & p_2 \\ p_2 & 0 & 0 & \dots & 0 & p_3 & p_1 \end{bmatrix}$$

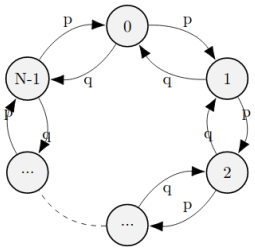

Figure 3. On the left, the general form of the transition matrix used. On the right a visual representation with N nodes for the case $p_1 = 0, p_2 = p$ and $p_3 = q$.

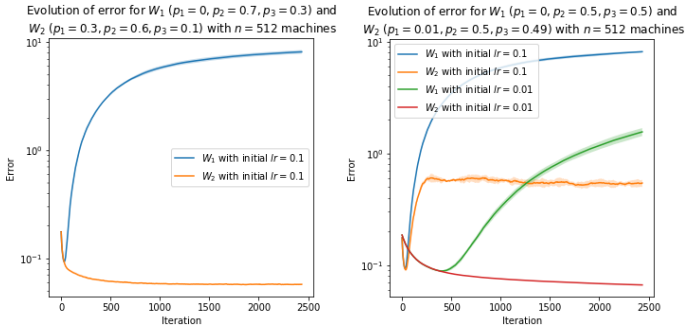

Figure 4. On the top, plot of the evolution of the error when considering W_1 with $p_1 = 0, p_2 = 0.7, p_3 = 0.3$. At the bottom, plot of the evolution of the error when considering W_2 with $p_1 = 0.3, p_2 = 0.6, p_3 = 0.1$.

is desired in this setup since we wish that all information is distributed uniformly in the network as information is exchanged. From the symmetry, the induced Markov chain is reversible, and in such a setup we can use the well-known result on the mixing time of a Markov chain which states that $\|\pi^{(0)}W^t - \pi\|_{TV} \leq C(1-\delta)^t$, where δ denotes the spectral gap, i.e. $\delta = 1 - \max\{|\lambda_2|, |\lambda_n|\}$ (and λ_i denotes the i^{th} eigenvalue when they are sorted in decreasing order). This result tells us that starting from any initial distribution $\pi^{(0)}$, the total variation distance between the distribution at step t and π (uniform distribution) decays exponentially fast as a function of t.

The question we ask now is the following: What happens when we allow more general matrices, i.e., when we do not require symmetry for example? Or when we do not assume that there is a self-loop at each node?

In this experiment, we choose ergodic Markov chains such that the transition matrix is still doubly stochastic to ensure that the limiting (and stationary) distribution is the uniform one. More specifically, we choose two matrices of the form depicted in Figure 3. The first one, which we call W_1, corresponds to the state transition diagram in Figure 3, i.e., $p_1 = 0$. In this case, to ensure ergodicity, we must restrict the number of nodes to an odd integer so that the chain is aperiodic. The second matrix we choose, W_2, is of the same type except that self-loops are added to each nodes ($p_1 > 0$). Note that in both cases, in the first experiment, weights are chosen so that neither W_1 nor W_2 is symmetric, the number of machines is $n = 512$, and the initial learning rate is set to 0.1. The convergence plot is shown on the top in Figure 4.

One can observe the following: In the case where self-loops are added (W_2), convergence does occur and results are comparable to those in the symmetric case. However, more interestingly, when no self-loop exists (W_1), convergence seems to happen in the first iterations but somehow it diverges from the optimum reached by W_2 and tends to grow but not too fast.

After experimenting with the parameters p_i's for W_1, whatever one chooses, the same "divergence" behavior is observed. It remains interesting to see what happens when we set p_1 to 0.5 and thus recover a symmetric matrix. Does convergence happen in that case? Also, we claim that this behavior might be due to the fact that there is no self-loop. To see this, we repeat experiment with W_2 too, but we set the self-loop weight to a low number, here 0.01. Error plots are shown at the bottom of Figure 4, for initial learning rates 0.1 and 0.01.

We see that for the symmetric matrix, divergence (from the optimum reached by topology with self-loops) happens once again regardless of the chosen learning rate. In fact, lowering the learning rate allows the algorithm to get to a better value in the first iterations but it still diverges eventually. We also note that setting the self-loops in W_2 to a low number prevents to reach the same convergence observed in the top plot when the initial learning rate is set to 0.1. However, we can recover the better convergence using initial learning rate 0.01. Note that even though reducing the initial learning rate may in the end allow convergence, it then becomes a difficulty practically as the rate of convergence is arguably slow. We thus conclude that self-loops may play an important role in the convergence of the distributed SGD for more general matrices in practice.

III. Discussion And Conclusion

From the first experiments, we were able to reproduce known results on basic examples using a larger number of nodes, and test their practical efficiency on a real life topology. In trying to understand the various observed behaviors, it was interesting to find alternate explanations using results from information theory. This proved our initial guesses to be incomplete, and pushed us to research for results that goes beyond usual analyses that study Markov Chains. In particular, we noted how the diameter of the graph corresponding to the topology might explain partly the convergence shown. It would be interesting to dig deeper in this direction to get a better understanding of which set of parameters yield a convergence which is related to the graph diameter, and which ones yield a convergence related to the spectral gap.

When extending the algorithm to more general matrices, the answer about whether convergence occurs in practice is mitigated. When self-loops are present, which is usually the case in real-life scenarios, good convergence can be observed. However, for the special case of the ring with no self-loops, convergence seems to fail to happen after a couple of iterations in practice. Also, recall that these results are taken for the special case of doubly stochastic matrices, so that the limiting distribution is uniform. However, one could imagine allowing any stochastic matrices, and depending on the limiting distribution, do some sort of re-weighting scheme so that the data can still somehow be uniformly distributed among the nodes. This way, one could potentially extend the convergence results presented in [1] to many more stochastic matrices.

In conclusion, through this project, we saw a glimpse of how powerful decentralized optimization algorithms can be. Not only they seem to perform well in practice, but they also work well even when generalizing known results through experiments. There is of course a lot of room for improvement as mentioned in the last few paragraphs, which is encouraging since this once again shows the potential of such new methods.

REFERENCES

[1] S. U. S. Anastasia Koloskova and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," *ICML Proceedings of the 36th International Conference on Machine Learning*, vol. 17, p. 3478–3487, 2019.

[2] Kaggle, "Higgs boson machine learning challenge," https://www.kaggle.com/c/higgs-boson, 2014.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[4] L. Bottou, *Stochastic Gradient Descent Tricks*, 01 2012, vol. 7700, pp. 10–11.

[5] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature 393*, pp. 440–442, 1998.

[6] J. Cheeger, "A lower bound for the smallest eigenvalue of the laplacian," in *Proceedings of the Princeton conference in honor of Professor S. Bochner*, 1969, pp. 195–199.

[7] A. Xu and M. Raginsky, "Information-theoretic lower bounds for distributed function computation," *CoRR*, vol. abs/1509.00514, 2015. [Online]. Available: http://arxiv.org/abs/1509.00514

[8] N. M. Olivier Lévêque, "Lecture notes of the 'markov chains and algorithmic applications' class given at epfl," 2019, https://moodle.epfl.ch/course/view.php?id=15016.