Signal Processing for Big Data - Project

Stefano D'Arrigo

February 2022

1 Introduction

Distributed training of Neural Networks is nowadays a crucial topic. Neural networks are pervasive in many fields and they require an increasing resources' power to train, due to both their expanding complexity and the huge size of data sets which are fed to them. Furthermore, the success of cloud computing and distributed resources over the network force models to be handled and trained in a distributed fashion. Finally, among the many other reasons, also data privacy and localization has an important role, as pointed out in [1].
In this short report the simulation results of the distributed training of a neural network on a regression task are shown and commented, following [1]. The code is available at https://github.com/stdrr/distributed_training_NN.git.

2 Problem formulation

Let $f(\boldsymbol{w};\boldsymbol{x})$ be a neural network model having $\boldsymbol{w} \in \mathbb{R}^Q$ adaptable parameters and fed with $\boldsymbol{x} \in \mathbb{R}^d$ inputs. Such function f is non-convex due to the composition with some non-linear (activation) functions a_i.

Let also $\mathcal{G}(\mathcal{V}, \mathcal{E})$ be a graph of vertices $\mathcal{V} = \{1, ..., I\}$ and edges $\mathcal{E} = \{(i,j)|i,j \in \mathcal{V}, i \neq j\}$ representing the network of agents which optimize f on data partitions $\mathcal{X}_1, ..., \mathcal{X}_I$.

The optimization problem of the function $f(\boldsymbol{w};\boldsymbol{x})$ is

$$\min_{\boldsymbol{w}} \left\{ \sum_i^I g_i(\boldsymbol{w}) + r(\boldsymbol{w}) \right\} \tag{1}$$

where $g_i(\boldsymbol{w}) = \sum_{m \in \mathcal{X}_i} l\left(f(\boldsymbol{w};\boldsymbol{x}_m)\right)$ is the error term of the agent i, $r(\boldsymbol{w})$ is a regularization term, $l(\cdot)$ is the loss function. Since f is non-convex, the problem (1) is non-convex.

In order to solve the problem through a distributed consensus optimization algorithm, a convex approximation of g_i is considered:

$$\widetilde{g}_i(\boldsymbol{w}_i;\boldsymbol{w}_i[n]) = g_i(\boldsymbol{w}_i[n]) + \nabla g_i(\boldsymbol{w}_i[n])^T(\boldsymbol{w}_i - \boldsymbol{w}_i[n]) + \frac{\tau}{2}\|\boldsymbol{w}_i - \boldsymbol{w}_i[n]\|_2^2 \tag{2}$$

Since the loss function of the regression problem tackled in this project is

$$l(a,b) = (a-b)^2 \tag{3}$$

with L2-regularization, the surrogate problem admits a closed form solution:

$$\widetilde{\boldsymbol{w}}_i[n] = -\frac{1}{\lambda}\left(\nabla g_i(\boldsymbol{w}_i[n]) + \widetilde{\boldsymbol{\pi}}_i[n]\right) \tag{4}$$

where $\widetilde{\boldsymbol{\pi}}_i[n]$ is the local estimate of the gradient.

Finally, the surrogate problem is solved by the algorithm depicted in Figure 1.

Algorithm 1: NEXT Framework for Distributed Optimization of (3)

Data : $\mathbf{w}_i[0]$, $\mathbf{y}_i[0] = \nabla g_i[0]$, $\boldsymbol{\pi}_i[0] = I\mathbf{y}_i[0] - \nabla g_i[0]$, $\forall i = 1,\dots,I$, and $\{\mathbf{C}[n]\}_n$. Set $n = 0$.

(S.1) If $\mathbf{w}_i[n]$ satisfies a global termination criterion: STOP;

(S.2) `Local Optimization`: Each agent i

(a) computes $\widetilde{\mathbf{w}}_i[n]$ as:

$$\widetilde{\mathbf{w}}_i[n] = \arg\min_{\mathbf{w}_i} \widetilde{U}_i\left(\mathbf{w}_i;\mathbf{w}_i[n],\widetilde{\boldsymbol{\pi}}_i[n]\right), \tag{12}$$

(b) updates its local variable $\mathbf{z}_i[n]$:

$$\mathbf{z}_i[n] = \mathbf{w}_i[n] + \alpha[n]\left(\widetilde{\mathbf{w}}_i[n] - \mathbf{w}_i[n]\right).$$

(S.3) `Consensus update`: Each agent i

(a) collects $\mathbf{z}_j[n]$ and $\mathbf{y}_j[n]$ from neighbors,

(b) updates $\mathbf{w}_i[n]$ as:

$$\mathbf{w}_i[n+1] = \sum_{j=1}^{I} c_{ij}[n]\,\mathbf{z}_j[n],$$

(c) updates $\mathbf{y}_i[n]$ as:

$$\mathbf{y}_i[n+1] = \sum_{j=1}^{I} c_{ij}[n]\,\mathbf{y}_j[n] + \left(\nabla g_i[n+1] - \nabla g_i[n]\right),$$

(d) updates $\widetilde{\boldsymbol{\pi}}_i[n]$ as :

$$\widetilde{\boldsymbol{\pi}}_i[n+1] = I\cdot\mathbf{y}_i[n+1] - \nabla g_i[n+1].$$

(S.4) $n \leftarrow n+1$, and go to (S.1).

Figure 1: Algorithm used to solve the surrogate problem.

3 Methods, Data and Results

The network of the agents is sampled at random so that the corresponding graph is scale-free. An example with 4 agents is shown in Figure 2.

As in the reference paper, the model f has been defined as a shallow fully-connected neural network with 12 units.

The data set on which the model has been trained is the Wine data set from UCI, containing 4898 samples of 11 features each and labeled with integers in $\{0, ..., 10\}$; even though the target variable is categorical, the problem that was tackled, as mentioned before, as a regression task. The data set has been split into train (80%) and test (20%) and then further partitioned among the agents.

The simulation of the distributed training has been carried out both sequentially by running the steps of the algorithm both within only one process and within parallel processes, one for each agent. Due to the data set size and the available computational resources, the number of agents was no more than 4.

The results of the training both in term of prediction error (Mean Square Error) and agent's disagreement are shown in Figures 3 and 4, respectively. From the plots, it can be observed that all the agents reach approximately the same value of prediction error as the number of epochs increases and the disagreement between their parameter estimates reduces to $\mathcal{O}(10^{-2})$.

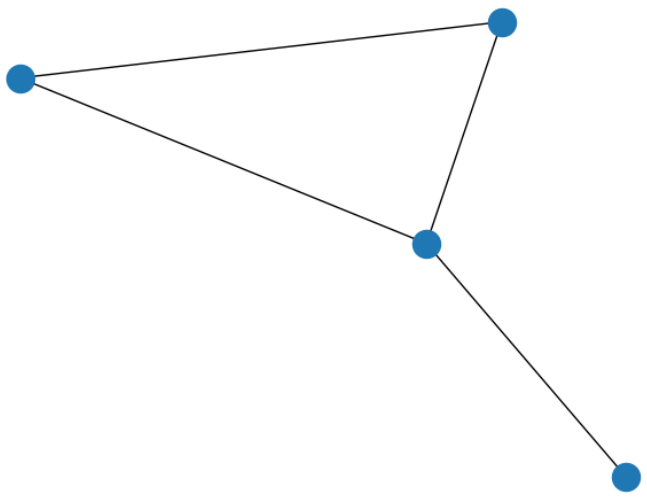

Figure 2: Network with 4 agents.

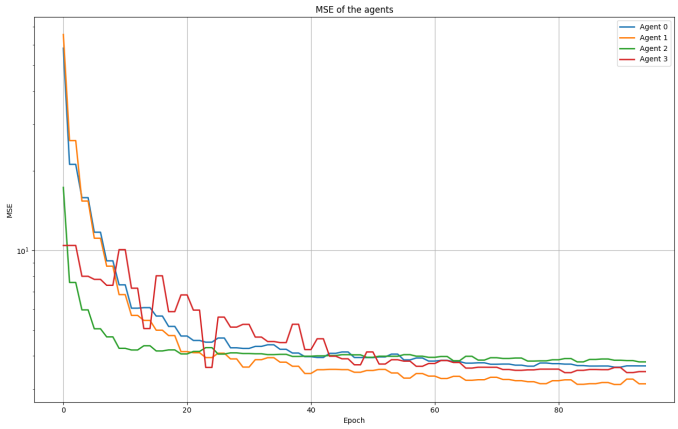

Figure 3: Mean Square Error of each agent across the epochs. The first 5 epochs have been discarded in order to improve the readability of the plot.

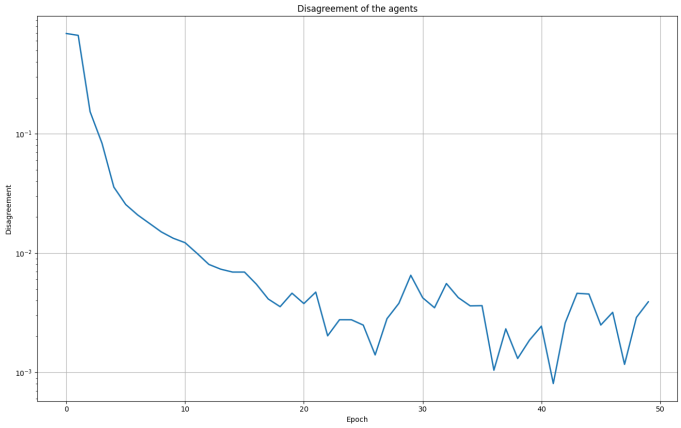

Figure 4: Disagreement of each agent across the epochs.

References

[1] Simone Scardapane and Paolo Di Lorenzo. "A framework for parallel and distributed training of neural networks". In: *Neural Networks* 91 (2017), pp. 42–54.