Neural Network Training in Distribution Computing Method

Zhongyu Zhao*
Department of Information Engineering Wuhan University of Technology, Wuhan, China

*Corresponding author: 276234@whut.edu.cn

Abstract. This paper raises some method to make neural network training can work on distributed computing clusters well.

Keywords: Neural Network, Distribution Computing, Network Training.

1. Introduction
With computing power developing, artificial intelligence, which leads by machine learning, plays an increasing role in social life. People have been exploring how to extract rules from big data while processing big data became the main aim of many studies.

2. Distributed Computing Scheme of Neural Network
2.1. Hyperparameters in deep learning
A deep neural network is the most commonly used network in machine learning. However, the parameter in a deep neural network is enormous, and mostly there is more than one hidden layer in the network. Thus, there is a significantly high cost to train a fully connected network.

Therefore, people have optimized the structure and connect method and proposed several models, including convolution neural network and impulsive neural networks. Compared with the fully connected network, the above structures have a considerable application effect and a decent number of parameters, which are advantages to training and deploying.

Besides, some functions could influence the model, whether it could fit the data well or not. For example, cross-entropy often be used in classification and a loss function, reflecting the distance, such as Minkowski distance. As a result, users should select an appropriate function is essential. Moreover, the hyperparameters in function also considerable; for instance, the dimension of Minkowski distance.

Hyperparameters have a significant influence on neural networks. Though there is some experience in hyperparameter tuning, it is still randomness.

A proper set of parameters frequently need multiple training and comparison. Furthermore, this set matches a given training set, which means if the question has changed, the model's applicability will be significantly reduced, and new training is needed.

2.2. Synchronous training mechanism with dynamic adjustment
TensorFlow proposes a synchronous training mechanism. The main node distributes the computing task process on different computing nodes. While all nodes have completed their task, main node averages
then update the parameters, as figure 1. It is seen that the worst node will limit the efficiency, and this method not suits for cheap clusters.

![Diagram of the synchronous training mechanism]

**Figure.1.** Synchronous training mechanism

We raise a solution for that question. At a moment, there are $N$ computing nodes and $M$ tasks for each node. First, we set a threshold time $T$ and a completion ratio $P$. While $N \cdot P$ nodes finish their task and report to the main node, the cluster will wait up to $T$. If there are some nodes in computing, stop them, and keep (or abandon) the result they upload. At next epoch, add $N \cdot P$ nodes in the cluster; each node bears the $M/2$ tasks for each node that did not finish their previous epoch task, as figure 2.
2.3. Training model of saliency samples for main-detail structure

For a big data set, the data can significantly improve the model and only make up a small proportion. Therefore, we could build up several auxiliary computing clusters to find the most contributing part in a given data set. After selecting a proportion of sorted data set, those clusters computing with a bigger learning rate could accelerate finding the parameter vector. At the end of each epoch, the main node compares the loss among all nodes, finds the minimal one, and copies the parameter to the main computing node, as Figure.3.

In order to avoid the saddle point, it is not supposed to set a small proportion that could keep enough noise in this set.

2.4. Training model of saliency samples for main-detail structure

For convex optimization, once training has the probability of convergence at a locally optimal solution instead of a globally optimal solution. In the field of traditional deep learning, we often start training with different parameters repeatedly in order to avoid local optimal solutions.
Those tasks can deploy on computing clusters. Different groups of computing nodes can undertake a task simultaneously using the method in 2.2.

2.5. **Speculation in model training**

There is already a speculative execution in Spark, which depends on the directed acyclic graph. Generally, training in one epoch is also a directed acyclic graph. So, speculation can fit deep learning. Speculative execution is the method that allows the main node redistributes to other lightly loaded nodes. First, set a detection time and detect the proportion of finished nodes in each epoch. If the proportion exceeds a certain scale, the main node will judge the unfinished node, whether it spends too much time on this task (a coefficient multiplies the average complete time). For the unfinished node, the main node distributes the uncomplete task to some lightly loaded nodes. Once a node or a group of nodes finish the task, use the result, and kill all nodes computing this task.

3. **Results**

3.1. **The establishment of experiment environment**

We use three computers to establish the environment. Those clusters are used to simulate the indeterminate computing clusters.

3.2. **Analysis of experimental results**

To measure the efficiency of the algorithm, we generate lists of data for multiple linear regression. We use dynamic adjustment method in 2.2, and the result as shown in Table.1.

| Task | Only main computing node | Computing cluster |
|------|--------------------------|-------------------|
| A    | 1570 seconds             | 818 seconds       |
| B    | 1264 seconds             | 753 seconds       |
| C    | 782 seconds              | 494 seconds       |

From the comparison between computing on main node and on cluster, we find that algorithm could improve the effectiveness of training.

4. **Conclusions**

Nowadays, the computing power of a single computer is more and more difficult to increase. People have proposed many methods to make deep learning can work on distributed clusters effectively. This paper also raises some feasible distributed computing schemes, and their advantages, disadvantages are briefly described.

**References**

[1] Deng L, Yu D. Deep Learning: Methods and Applications [J]. Foundations & Trends in Signal Processing, 2014, 7(3).
[2] Tian Bin. Big data machine learning under the framework of distributed computing [J]. Electronic Technology and Software Engineering, 2019(20).
[3] He Majun. Research on Efficient Distributed Parallel Algorithm of Deep Learning Framework TensorFlow [D]. 2019.
[4] Huang, Xiaohan, Chunlin Li, and Youlong Luo. "Optimized speculative execution strategy for different workload levels in heterogeneous spark cluster." Proceedings of the 2019 4th International Conference on Big Data and Computing. 2019.