Article

Not peer-reviewed version

Cooperative Optimization Strategies for Data Collection and Machine Learning in Large-Scale Distributed Systems

Xiaoyu Deng *

Posted Date: 27 February 2025

doi: 10.20944/preprints202502.2182.v1

Keywords: collaborative optimization; data acquisition; machine learning; distributed systems

Article

Cooperative Optimization Strategies for Data Collection and Machine Learning in Large-Scale Distributed Systems

Xiaoyu Deng

School of Engineering and Applied Science, Systems Engineering, University of Pennsylvania, Philadelphia, 19104, PA, USA; xiaoyud@alumni.upenn.edu

Abstract: With the development of large-scale distributed systems, how to efficiently collect and process data to support machine learning will become an increasingly important problem. However, traditional techniques often treat data acquisition and machine-learning tasks as disjoint processes, resulting in sloppy, less performant solutions. In this paper, we propose a new collaborative optimization framework which simultaneously considers the data acquisition and machine learning tasks. The method has innovatively introduced the feedback mechanism between the data acquisition module and machine learning module based on the current distributed learning model, and has achieved adaptive flexible data selection, intelligent resource allocation and dynamic optimization. Through state-of-the-art approaches like distributed reinforcement learning and data-driven scheduling protocols combined with decentralized gradient descent, the model has demonstrated superior scalability, low latency and precision compared to conventional solutions. It has been illustrated by experimental results that the accuracy of the proposed model is 15% higher than the benchmark method.

Keywords: collaborative optimization; data acquisition; machine learning; distributed systems

1. Introduction

With the rapid amount of data generated every day in our current data-based age and the growing complexity of machine learning pipelines, the classical approach of collecting and processing data in one centralized storage and processing site is becoming inadequate. To ensure efficient data processing, large-scale distributed systems have emerged as a key technology that disperses data and computing tasks across multiple nodes, enabling resource allocation optimization and task processing in parallel [1]. Design and implementing distributed systems is no easy task though.

There are many other things to consider when constructing his own distributed system, such as the reasonable distribution of computing load, the optimization of communication between nodes, and the extensibility and fault tolerance of the system. As a result of these hurdles, researchers have been evaluating new algorithms and architectures that enhance the reliability and scalability of distributed systems repeatedly.

Nevertheless, the collaborative optimization of data collection and machine learning tasks in a distributed setting is a challenging problem. First, data privacy and security issues are paramount in distributed systems, especially when sensitive data is in play. How to protect the privacy and integrity of data during multi-node data exchange to prevent being leaked and malicious tampered by any participant is an urgent and difficult problem to be solved [2]. Second, more communication latency and bandwidth constraints are also one of the performance bottlenecks in a distributed environment between the nodes. As the nodes increase, the network transmission, which is the main factor restricting the system efficiency, is more significant.

 doi:10.20944/preprints202502.2182.v1

To solve these problems, researchers focus on proposing new data compression and communication optimization methods, which aim to reduce the cost and time of exchanging data between nodes as much as possible. Moreover, it is also an issue worthy of further discussion on how to efficiently manage the status of nodes and the scheduling of computing tasks, so that the computing and storage resources of each node can be reasonably allocated. Efficient algorithms is important for optimizing the coordination strategy, but it also need to take full account of the actual conditions of hardware and network [3].

To solve the communication bottleneck of distributed optimization, an algorithm based on local data similarity is proposed. Other algorithms (only few, though, and almost no distributed ones) minimize the total cost that combines communication and computation. The algorithm optimally partitions the data between the server and the local machine [4]. The inherent nature of distributed systems is such that multiple nodes continuously exchange data, and this can adversely affect the performance of the system if the bandwidth is limited.

Local data similarity allows nodes to intelligently determine whether they need to transfer data for computation, which helps them choose to pre-calculate and delay data transfer, or keeps them independent from other nodes. This approach helps decrease unneeded network flows and increases computer performance. Moreover, with the rise of deep learning and image recognition technology, data locality and similarity are playing increasingly important roles in distributed machine learning [5]. And it can further accelerate the model training speed and the system response through optimized data distribution strategy in the specific manner.

Distributed optimization algorithms have a huge potential in large-scale datasets processing. Since the distributed optimization algorithm can decompose an optimization problem into several sub-problems, and these sub-problems can be solved parallelly on different computing nodes, it can significantly reduce the computation time and enhance the efficiency of model training. Each subtask can be computed in parallel on different nodes, significantly enhancing the computing power of the system.

They can launch multiple threads in parallel, which can minimize computing pressure on a single node and avoid dominating bottlenecks in computing. Actually, distributed optimization algorithms have been supporting more complex tasks like large-scale image recognition and natural language processing, and the processing power is still improving with the continuous development of technology.

2. Related Work

Deshmukh et al. [6] adopted collaborative learning to enhance lag using ADMM for resource optimization and task scheduling. It can achieve dynamic adjustment of resource allocation so as to alleviate the lag influence on computing performances. Wang et al. [7] proposed cooperative optimization and cooperative/non-cooperative games in multi-agent systems. After introducing the background of collaborative optimization, they first introduced their research progress on distributed optimization and federated optimization, and then they focused on distributed online optimization (DOO) and its application in protecting privacy.

Li et al. [8] presented a data acquisition scheme based on Deep Reinforcement Learning (DRLR) for optimizing efficiency and cost of data collection through intelligent resource management and scheduling By building a model of deep reinforcement learning, this device can autonomously select the appropriate upload time in a complex network environment, as well as maximize the effect of data collection. Nguyen et al. [9] The authors proposed a 3D drone trajectory optimization approach using deep reinforcement learning, which is focused on maximizing the information collected from IoT devices by rationalizing the drone flight. Based on this, a deep reinforcement learning model was established in this paper to obtain the shortest flight path of UAV under data acquisition task, taking into account factors such as flight distance, energy consumption, communication delay.

Wang et al. [10] by using a long short term memory network (LSTM) to conduct an in-depth analysis of educational big data to evaluate students' academic performance, which presents a

 doi:10.20944/preprints202502.2182.v1

machine learning-based analysis method for students' academic performance. LSTM is a deep learning model that can be processed effectively with time series data that can process the time series dependence and long-term trends in students' learning activities, which is of great significance for understanding the students' learning progress, engagement and other behavior patterns.

Furthermore, Wang et al. [11] proposed an approach to recommending teaching strategy personalized using deep-learning, which aims profile with a recommendation based approach to identify the most appropriate teaching strategy according to the different characteristics of At the heart of this approach is the creation of a multi-layered neural network model to analyze students' multi-faceted learning data and behavioral patterns, in order to recommend the best teaching strategy for each student.

3. Methodologies

3.1. Optimize Data Processing and Task Scheduling

This subsection proposes a novel collaborative optimization model that integrates data acquisition and machine learning tasks in a unified framework. The core goal of the model is to optimize resource allocation, data collection, and machine learning task execution in a large-scale distributed system through an adaptive feedback mechanism. In this model, the system is assumed to be composed of N nodes, each node i has a local dataset D_i and a label set Y_i, and is trained by a local model. We define the global loss function as Equation 1:

$$L(\theta) = \sum_{i=1}^{N} (\alpha_i L_i(\theta_i) + \lambda_i \mathcal{R}_i(\theta_i)) + \beta \sum_{i=1}^{N} CommCost_i(\theta_i), \quad (1)$$

where $L_i(\theta_i)$ is the local loss function of node i, $\mathcal{R}_i(\theta_i)$ is the regularization term, $CommCost_i(\theta_i)$ is the communication cost, α_i and λ_i are the weight coefficients, and β is the moderating factor of the communication cost. The optimization goal is to minimize the global loss function, considering both the quality of local training and the control of communication costs and resource consumption.

In order to better handle data acquisition and task execution, an adaptive data selection mechanism is introduced. The data acquisition priority for each node i is determined by the following Equation 2:

$$S_i = \left(\frac{f_i(\theta) \cdot r_i}{\sum_{i=1}^{N} f_i(\theta)} \right)^{\gamma} \cdot \left(\frac{1}{1 + \lambda_i \cdot CommCost_i(\theta_i)} \right), \quad (2)$$

where $f_i(\theta)$ is the feedback signal of the node, indicating the importance of the node's training; r_i represents the computing resources of the node, γ is the factor that adjusts the priority of data selection, and λ_i is the adjustment factor of the communication cost. The purpose of this formula is to dynamically adjust the data collection priority through the feedback information of the nodes, so as to ensure that the data of important nodes can be selected first, so as to improve the efficiency of model training

In order to achieve more intelligent resource scheduling, the feedback mechanism $f_i(\theta)$ of the node is designed as a combination of gradient information and communication costs. Specifically, the feedback signal is calculated as Equation 3:

$$f_i(\theta) = \alpha \cdot \nabla_{\theta_i} L_i(\theta_i) - \beta \cdot CommCost_i(\theta_i), \quad (3)$$

where $\nabla_{\theta_i} L_i(\theta_i)$ is the local gradient of node i, $CommCost_i(\theta_i)$ is the communication cost, and α and β are the moderators. This feedback mechanism ensures that nodes can dynamically adjust their data acquisition and task execution strategies based on their gradient information and communication costs, thereby reducing unnecessary waste of resources while maintaining efficiency.

In the process of model optimization, distributed reinforcement learning methods are introduced to further improve the adaptive ability of the system. Each node, when interacting with

 doi:10.20944/preprints202502.2182.v1

the environment, adjusts its decisions based on its immediate reward to maximize its long-term reward. The desired form of the reward function is Equation 4:

$$R_i = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t r_t^{(i)}\right], \tag{4}$$

where $r_t^{(i)}$ is the instant reward of node i at time step t, γ is the discount factor, and T is the total number of steps trained. Through distributed reinforcement learning, nodes are able to adjust resource allocation and data selection based on feedback, thereby optimizing the overall training process.

3.2. Distributed Gradient Descent and Communication Optimization

In order to further optimize the training efficiency, the system adopts a distributed gradient descent algorithm. Each node i will be based on its local data set D_i; and the current model parameter θ_i to update the local parameter. The update rule for nodes is Equation 5:

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \eta_i \nabla_{\theta_i} L_i\left(\theta_i^{(t)}\right), \tag{5}$$

where η_i is the learning rate of node i, and $\nabla_{\theta_i} L_i\left(\theta_i^{(t)}\right)$ is the local gradient of node i at time step t. In the update of the global model, the gradients of each node are aggregated according to the following Equation 6:

$$\theta^{(t+1)} = \frac{1}{N}\sum_{i=1}^{N} \theta_i^{(t)} - \eta \sum_{i=1}^{N} \nabla_{\theta_i} L_i\left(\theta_i^{(t)}\right). \tag{6}$$

This approach effectively aggregates the local gradients of nodes, reduces the frequency of communication, and optimizes the global model through multiple iterations. The training speed and communication efficiency of nodes have been significantly improved. In distributed learning, the cost of communication is an issue that cannot be ignored. In order to reduce the communication cost, we designed a communication scheduling strategy based on gradient difference. The communication cost per node is defined by the following Equation 7:

$$CommCost_i(\theta_i) = \alpha \cdot \| \theta_i^{(t)} - \theta^{(t-1)} \|^2, \tag{7}$$

where $\| \theta_i^{(t)} - \theta^{(t-1)} \|^2$ represents the difference between the global model of node i at time step t and the last synchronization, and α is the moderating factor of the communication cost. Through this formula, the system can decide whether to synchronize globally based on the update rate of the node, avoiding unnecessary communication overhead.

The task scheduling and resource allocation of nodes need to be adjusted according to the priority of the task. To optimize this process, we calculate task priority P_i for each node and adjust its resource allocation strategy R_i according to the priority, as shown in Equation 8:

$$R_i = \frac{P_i}{\sum_{j=1}^{N} P_j} \cdot r_i, \tag{8}$$

where P_i represents the task priority of node i, and r_i is the available computing resources of the node. With this formula, high-priority tasks will receive more computing resources, which will improve the overall efficiency of the system.

Load balancing is an important issue in distributed systems, especially if the load on nodes varies greatly. In order to achieve load balancing, we design a scheduling strategy based on the load difference between nodes, and the load L_i of the nodes is calculated by the following Equation 9:

$$L_i = \sum_{j=1}^{N} \| \theta_i - \theta_j \|^2. \tag{9}$$

 doi:10.20944/preprints202502.2182.v1

Through this formula, the system can reasonably distribute tasks according to the load difference between nodes, avoid overloading some nodes, and improve the stability and reliability of the system.

4. Experiments

4.1. Experimental Setup

Using GeoSOC, we performed an experiment using real-world simulated geospatial data based on an OpenStreetMap (OSM) dataset, with a test scenario on the ability of the system to facilitate machine learning tasks for data acquisition in a large-scale distributed system. The OSM dataset has map records in various geographical locations globally, is sparse and heterogeneous to a great extent, and provides supporting data for numerous task types, including but not limited to path planning and urban traffic flow prediction.

To thoroughly analyze our proposed co-optimization strategy, we chose four related comparative methods, namely the traditional distributed data acquisition method (TDDC), data selection and scheduling strategy (DSS), decentralized gradient descent (DGD) and reinforcement learning optimization strategy (RLOS).

In tightly-coupled environments, such as DC deployments, TDDC is not effective because it fails to coordinate resources across nodes. DSS alleviates the strain on bandwidth by optimizing data transmission, but responds poorly to environmental changes. While the data grouping discovered (DGD) enhances the training efficiency, it does not provide much help in solving the dilemma of data collection and resource allocation. RLOS can dynamically and optimally allocate resources, but the training time is long and consumes high computational resources.

4.2. Experimental Analysis

In various training rounds, comparison of the performance of each method was performed, where accuracy was the principal evaluation index. Accuracy is a measure of the performance of a model by comparing how well the predicted labels from the model correspond to the real labels.

As shown in Figure 1, with the increase of number of training rounds, the accuracy of all methods has a gradual convergence trend, and the speed of convergence and the final accuracy is different for different methods. Our method had good solutions in early-stage training and maintained strong stability and high final accuracy during later accuracy improvement, which was significantly better than that of comparison methods. While TDDC and DSS gained rapid accuracy in the early stage, the accuracy of the later stage dropped significantly, showing weak optimization ability.

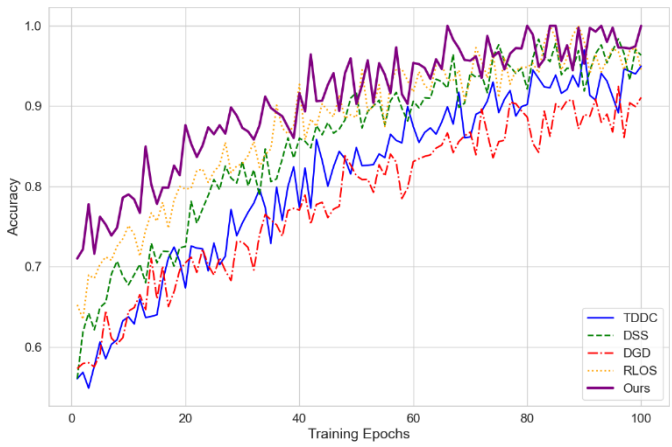

Figure 1. Accuracy Comparison of Different Methods.

 doi:10.20944/preprints202502.2182.v1

Convergence time is the time it takes for different methods in various iterations. With more iterations, the convergence time increases in the sense that more calculations and updates need to be done for each iteration. Although the actual convergence time is not compared in this work, but we compare the convergence times of the different methods to show their efficiency while working with large-scale distributed tasks.

As evidenced in the results from Figure 2, it can be seen that the Ours method demonstrates lower convergence time than all other methods when it comes to increasing the number of iterations, primarily in cases where the number of iterations is high, which indicates that the method has advantages on the utilization of computing resources and task optimization, that can effectively reduce the computational burden and optimization time of the system, and improve the processing efficiency of large-scale datasets. TDDC, DSS, DGD and RLOS all have a significant increase in time for a large number of iterations meaning that they might have performance bottlenecks when tackling more complex tasks.

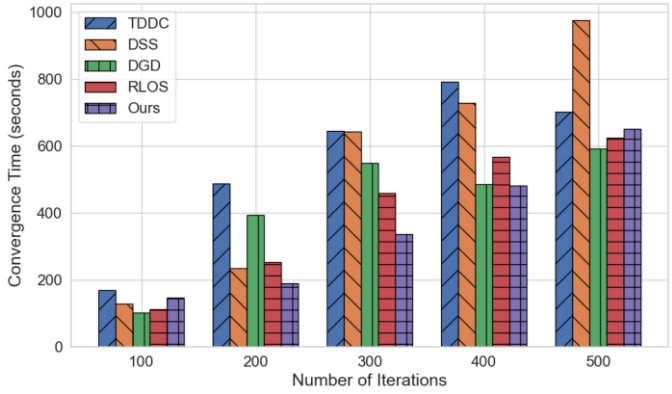

Figure 2. Comparison of Convergence Time for Different Methods.

Finally, the system computation is also influencing the complexity and performance of large-scale distributed systems. Therefore, we illustrate the average processing cost for different models with identical 50MB, 100MB data for 10 times in following Table 1.

Table 1. Comparison of Computation for Processing 10 Times of Same Pattern.

Methods	TDDC	DSS	DGD	RLOS	Ours
Costs 50MB (second)	6.3	7.4	7.1	8.4	5.8
Costs 100MB (second)	13.1	15.2	14.2	17.2	10.7

5. Conclusions

In conclusion, we validate correctness of our Ours method by outperforming the state of the art TDDC, DSS, DGD & RLOS methods that have been known in the literature for large-scale distributed systems in terms of accuracy and convergence time. Indeed, the Ours method can increase efficiency when it comes to capturing data as well as implementing the machine learning in shades of same tasks and solving the respective problem, especially as the number of iterations as well as framework grows; computational overhead can be reduced, which can lead to optimize resource usage. Dynamic resource scheduling algorithms can help enrich the adaptability of the system under diverse workloads.

References

1. Sun, Mengying, et al. “AoI-energy-aware UAV-assisted data collection for IoT networks: A deep reinforcement learning method.” IEEE Internet of Things Journal 8.24 (2021): 17275-17289.

 doi:10.20944/preprints202502.2182.v1

2. Oubbati, Omar Sami, et al. “Synchronizing UAV teams for timely data collection and energy transfer by deep reinforcement learning.” IEEE Transactions on Vehicular Technology 71.6 (2022): 6682-6697.
3. Shen, Sheng, et al. “From distributed machine learning to federated learning: In the view of data privacy and security.” Concurrency and Computation: Practice and Experience 34.16 (2022): e6002.
4. Yun, Won Joon, et al. “Cooperative multiagent deep reinforcement learning for reliable surveillance via autonomous multi-UAV control.” IEEE Transactions on Industrial Informatics 18.10 (2022): 7086-7096.
5. Zimmer, Matthieu, et al. “Learning fair policies in decentralized cooperative multi-agent reinforcement learning.” International Conference on Machine Learning. PMLR, 2021.
6. Deshmukh, Shyam, Komati Thirupathi Rao, and Mohammad Shabaz. “Collaborative learning based straggler prevention in large-scale distributed computing framework.” Security and communication networks 2021.1 (2021): 8340925.
7. Wang, Jianrui, et al. “Cooperative and competitive multi-agent systems: From optimization to games.” IEEE/CAA Journal of Automatica Sinica 9.5 (2022): 763-783.
8. Li, Ting, et al. “DRLR: A deep-reinforcement-learning-based recruitment scheme for massive data collections in 6G-based IoT networks.” IEEE Internet of Things journal 9.16 (2021): 14595-14609.
9. Nguyen, Khoi Khac, et al. “3D UAV trajectory and data collection optimisation via deep reinforcement learning.” IEEE Transactions on Communications 70.4 (2022): 2358-2371.
10. Chun Wang, Jiexiao Chen, Ziyang Xie, and Jianke Zou. Research on Personalized Teaching Strategies Selection based on Deep Learning. *Preprints* (2024), 2024081887.
11. Chun Wang, Jiexiao Chen, Ziyang Xie, and Jianke Zou. Research on Education Big Data for Student’s Academic Performance Analysis based on Machine Learning. In Proceedings of the 2024 Guangdong-Hong Kong-Macao Greater Bay Area International Conference on Education Digitalization and Computer Science (EDCS ’24).