Design of asynchronous federated learning incentive mechanism

Bingze Li[1,a]

[1]School of Management Engineering and Business, Hebei University of Engineering, Handan, China
[a]bingze1108@163.com

Abstract: *Federated learning has been widely paid attention as a new type of distributed machine learning that can protect data privacy and ensure data security. Asynchronous federation learning, a variant of traditional federation learning, can effectively improve model training efficiency. The introduction of incentive mechanism can help asynchronous federation learning to improve model training utility effectively. A federated learning incentive mechanism is constructed using the Stackelberg game, which optimizes the central server and data owner utilities, respectively, of the Stackelberg game. Based on this, we derive the equilibrium solution of the whole game, and finally analyze the feasibility of the model by arithmetic examples to obtain the optimal incentive effect.*

Keywords: *Federated learning, Stackelberg game, Robust optimization, Data quality, Nash equilibrium*

1. Introduction

In the past decades, machine learning has achieved unprecedented success with the development of artificial neural network algorithms and cloud computing environments. Although data and information development has brought many conveniences to productive development, at the same time data security and information privacy have become the primary concerns of data users. Users are more cautious about data privacy risks, more sensitive to the interaction information in the system, and hope that personal private data will not be leaked for other uses. Countries are starting to pay more attention to users' personal privacy, business secrets, and data security. The establishment of various regulations and the increase of users' privacy awareness are new challenges for traditional distributed machine learning. At the same time, as the awareness of data rights increases, more and more entities emphasize the ownership and usage of data, especially in some key industries, such as hospitals and banks, and do not allow exchanging data among themselves for machine learning. This private data cannot be used for analysis without the consent of the user. This has created a phenomenon called " data island", which refers to the phenomenon of data closure and isolation due to the reduced data flow rate and the inability to effectively exchange and share data between different entities. This phenomenon makes simple distributed machine learning unable to effectively solve the training problem of large-scale data sets.

Federated learning is proposed as an emerging distributed machine learning model[1]. Federated learning is considered to be the most capable machine learning model that can break through "data silos" and has received a lot of attention[2]. Asynchronous federated learning, a variation of traditional federated learning, implements model training through asynchronous communication of local model parameters on different devices. Unlike traditional federation learning, asynchronous federation learning does not need to wait for all devices to complete local training before sending updated parameters, thus allowing for faster training completion. Currently, studies related to asynchronous federation learning mainly consider issues such as heterogeneity, asynchronous federation model optimization[3], in order to improve model efficiency, ensure model validity, and guarantee privacy security. However, these studies are based on an optimistic premise that all data owners will unconditionally participate and upload updated parameters to the central server. In fact, the effectiveness of training asynchronous federated learning models depends heavily on the quality of data that data owners can provide, and data owners consume a lot of computational and communication resources, and without some incentives, these data owners may be reluctant to use their own data to participate in asynchronous federated learning, which involves multiple participants and differences in data quality between participants. This leads to the reluctance of data owners to participate or to put themselves in an unfair position when participating in federated learning. This paper considers the introduction of the Stackelberg game to design an incentive mechanism for asynchronous federated learning that provides an effective incentive for both data owners and task publishers.

2. System model

We consider asynchronous federated learning consisting of a task publisher and multiple data owners as shown in Figure 1 below. Where the first step of asynchronous federated learning distributes the model training task to different data owners, each of which runs local training independently and sends its local model updates to the central server periodically during the training process. After receiving these updates, the central server can merge them into a global model, analyze the updated global model, and resend a new model training task to the data owners, and each device can use the updated global model to continue local training based on the new training requirements. Specifically, in asynchronous federated learning model training, each data owner stores its respective private data locally and uses the local data for model training according to the initial model and parameter requirements set by the central server. In federated learning, each data owner, after completing the local model training task, transmits the model parameters obtained from the training back to the central server of federated learning. The server aggregates these updated parameters to update the global model and issues a new iteration task to the data owner. The data owner needs to continuously iterate the training as required until the target performance or a preset number of iterations is reached. This approach avoids data privacy issues in centralized training while speeding up model training.

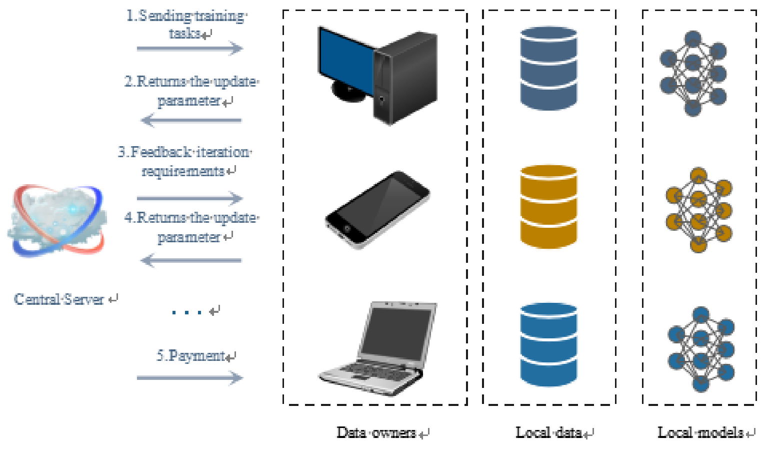
1.Sending training tasks
2.Returns the update parameter
3.Feedback iteration requirements
4.Returns the update parameter
Central Server
. . .
5.Payment
Data owners
Local data
Local models

Figure 1: Asynchronous federated learning model training process

Assume that each data owner $n \in N$ has the same sample size of local data to participate in federated learning. However, each data owner uses a different CPU cycle frequency f_n to train the local model. The number of CPU cycles to complete a data training is denoted by c_n. Thus, the time required for local model training computation is $T_n = (sc_n / f_n)$, and the CPU energy consumed in completing a local data training is: $D_n(f_n) = \varepsilon s c_n f_n^2$ [4]. The training efficiency of the data owner is denoted as λ_n, and the number of iterations of local training $\log(1/\lambda_n)$ is used to denote the number of iterations of local training, then the computation time of model update for global iterations is: $T_n^c = \log(1/\lambda_n) T_n$. During data transmission, the transmission rate can be expressed as: $r_n = B\ln(1+(\rho_n h_n / N_0))$, B is the transmission bandwidth, ρ_n is the transmission power of the data owner, h_n is the transmission channel gain, and N_0 is the background noise. Then the transmission time of local model update is: $T_n^t = s/(B\ln(1+(\rho_n h_n / N_0)))$.

The data owner is assumed to receive a payoff of $R_n = q_n f_n$, where q_n is the price per unit for the data owner to work with CPU frequency f_n [5]. The more computational resources the data owner provides, the faster the local model is trained, resulting in a higher payoff. The data owner is free to choose to sign a contract to complete the federated learning task. However, if the federated learning task cannot be completed according to the chosen contract, the given payoff is not available.

Set the task publisher utility to the global iteration duration: $U_T = \log(1/\lambda_n)T_n + T_n^t$. When the data owner n receives the corresponding payoff R_n from the task publisher, it takes into account the energy loss incurred in participating in the learning process, which depends on the CPU power level. Therefore, for each data owner whose goal is to maximize its own profit, $U_D = q_n f_n - \mu \log(1/\lambda_n) D_n$, the equation is bounded by $f_n \le f_{\max}$, where $f_{\max}$ is the upper limit of the CPU power of the data owner and μ is a predefined energy consumption weight parameter.

3. Solution of incentive mechanism based on Stackelberg game

Using Stackelberg game to construct the model, the lower game is:

$$\max_{f_n} U_D = q_n f_n - (\overline{\sigma}_n + \widehat{\sigma}_n \psi_n)\mu\varepsilon s c_n f_n^2$$

$$\text{s.t.} f_n \le f_{\max}$$

The top game is:

$$\min_{q_n} U_T = \log(1/\lambda_n)(sc_n / f_n) + s/[B\ln(1+[\rho_n h_n / N_0])$$

$$\text{s.t. } q_n f_n \le R_{\max}$$

In general, Stackelberg game equilibrium can be obtained by finding its optimal Nash equilibrium. In the game constructed in this paper, given the unit price of CPU power of the data owner, there is a non-cooperative game management system between the data owners, a Nash equilibrium is defined as one in which no player can change their strategy to increase their return. For non-cooperative games, if the game satisfies: (1) the set of the two players is limited (2) the strategy space of the game is bounded by the closed set of the dominant space of the data set (3) the profit function of the non-cooperative game is continuous and satisfies the concave function in the strategy space. Then there is a Nash equilibrium in the game, and the utility of each player will be maximized, and no player can get a higher return by selfishly changing their strategy. In this game, the number of players is limited, and the optimal CPU frequency price provided by the central server is a bounded closed set in European space, moreover, the utility function of the lower game changes continuously with the independent variable, and the profit function satisfies the concave function. By calculating the first derivative and the second partial derivative of CPU power, we know that the second partial derivative is negative and the profit function satisfies the strict concave function, therefore, the Stackelberg game model has a sub-game Nash equilibrium solution.

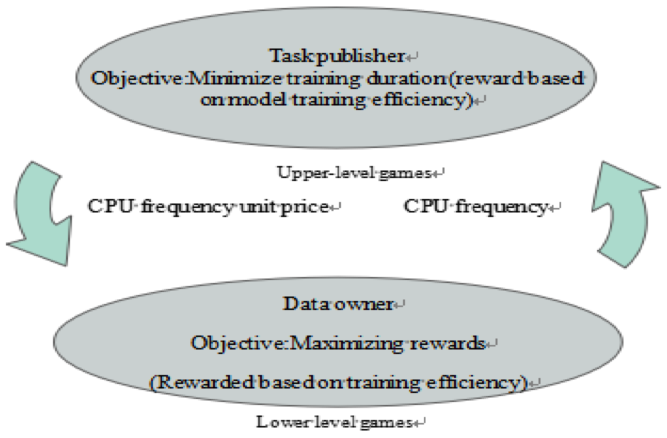
Task publisher
Objective:Minimize training duration(reward based on model training efficiency)
Upper-level games
CPU frequency unit price
CPU frequency
Data owner
Objective:Maximizing rewards
(Rewarded based on training efficiency)
Lower level games

Figure 2: Stackelberg Game Model for Asynchronous Federated Learning

For the above-mentioned game model, we consider the Stackelberg equilibrium problem of task publisher and data owner. For Stackelberg game with unknown data quality, the upper and lower game equilibrium solutions are determined by backward induction method[6]. First of all, according to the first-order optimality conditions for the lower game to find its equilibrium solution. Then, the lower equilibrium solution is brought into the upper game to seek the game solution. The Stackelberg game of asynchronous federated learning is shown in Figure 2:

In order to find the equilibrium solution of the data owner in the lower level game, take the profit function of the lower level game to the CPU power, and obtain the first derivative $\frac{\partial U_D}{\partial f_n} = \frac{\partial [q_n f_n - \log(1/\lambda_n)\mu\varepsilon s c_n f_n^2]}{\partial f_n} = q_n - 2\log(1/\lambda_n)\mu\varepsilon s c_n f_n$ .By setting the above equation to zero, the CPU power of the data owner can be obtained as:

$$f_n = \begin{cases} \frac{q_n}{2\mu\varepsilon s c_n \log(1/\lambda_n)} & if \frac{q_n}{2\mu\varepsilon s c_n \log(1/\lambda_n)} \le f_{\max} \\ f_{\max} & if \frac{q_n}{2\mu\varepsilon s c_n \log(1/\lambda_n)} \ge f_{\max} \end{cases}$$

First, the optimal CPU power of the data owner is replaced with that of the upper game when the lower game reaches equilibrium Utility maximization problem. Since the constraints of the upper-level game are linear.Therefore, the Lagrangian method is used to solve the problem. The Lagrangian of the optimization problem are: $L(q,\alpha) = \log(1/\lambda_n)T_n + T_n^t + \alpha[\frac{q_n^2}{2\mu\varepsilon s c_n \log(1/\lambda_n)} \text{-} R_{\max}]$. In order to obtain the optimal solution, the first derivative of the above Lagrangian $L(\cdot)$ is obtained, and the value of the Lagrange multiplier α at the optimal point is obtained from the KKT condition as follows: $\alpha = \text{-}\frac{\partial U_T}{\partial q_n}\frac{\mu\varepsilon s c_n \log(1/\lambda_n)}{q_n}$.

The first derivative of the utility function of the upper game shows that the utility function of the upper game is negatively related to the CPU power f_n provided by the data owner. There is a positive correlation between the unit price of CPU power paid by the task publisher and the CPU power provided by the data owner. Therefore, the first term to the right of the above equation is positive, and since the second term is always positive. In conclusion, Lagrange multiplier $\alpha > 0$ stands.

According to the above KKT conditions $\alpha[\frac{q_n^2}{2\mu\varepsilon s c_n \log(1/\lambda_n)} - R_{\max}] = 0$, it can be seen that $\frac{q_n^2}{2\mu\varepsilon s c_n \log(1/\lambda_n)} = R_{\max}$,the Nash equilibrium solution of the game exists on the boundary according to the complementary relaxation condition of KKT. So the equilibrium solution for all $n \in N$ the upper-level games is: $q_n^* = \sqrt{2\mu\varepsilon s c_n \log(1/\lambda_n) R_{\max}}$.

4. Numerical Analysis

Table 1: Parameter setting in simulation experiment

Parameter	Description	Value
T_n^t	Transmission time of the local model	0.5
μ	Weight parameter for energy consumption	0.1
s	Local data sample size	20
c_n	Data owner CPU cycles	5
ε	Effective capacitance parameter for data owner	2
λ_n	Accuracy of the local data	[20%,90%]

In the simulation experiment, we use MNIST data set and TensorFlow, a widely used software environment, to perform the numerical classification task to evaluate the incentive scheme constructed above. In the Federated Learning Task, we set up 10 task publishers and 100 data owners, in which the data owners are divided equally into 10 different data quality nominal values, and there is uncertainty about the quality of the data provided. To truly reflect the heterogeneity of data owners when training the model locally, we randomly selected participating data owners by the number of repeated training sessions. We specify a maximum CPU power of 15 for a single data owner and a maximum CPU frequency unit price of 100 for a task publisher. The other parameters in the simulation experiment are given in Table 1.

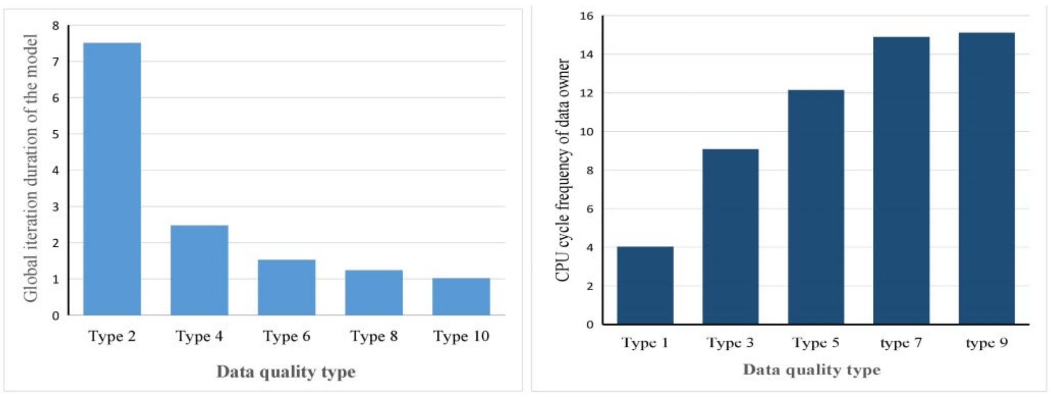

Figure 3: Global iteration duration and data owner CPU cycle frequency are affected by changes in data quality

In this paper, the utility of the upper-level game is to minimize the global iteration time. We train 100 data owners classified by data quality to participate in the model, the effect of different data quality of data owner on task publisher's utility is analyzed by solving the mean of iteration time. As shown in the figure above, when the upper and lower levels of the game reach an equilibrium optimal solution, the CPU frequency provided by the data owner increases as the data quality improves, and the global iteration time of the task publisher's model decreases, the global iteration time of the task publisher's model is quasi-concave. The main reason is that with the improvement of the data quality of the data owner, the energy consumption of the data owner is reduced to achieve the same model precision, and in order to obtain higher reward, the data owner has to use less energy to achieve the same model precision, increase Your Own CPU frequency, so in Figure 3, the data owner's optimal CPU frequency is increasing as data quality improves, but the impact of data quality on model computation time is getting smaller and smaller, so that the data owner increases the CPU rate is also getting smaller.

The global iteration time of the task publisher also decreases as the optimal CPU frequency of the data owner increases, and when the optimal CPU frequency changes slowly, the task publisher's model global iteration time is also decreasing. Therefore, in selecting the data owners involved in model training in asynchronous federated learning, we should try to select the data owners with better data quality and willingness to provide them as model training, in order to improve the overall training effectiveness of the model, and improve the data owner's own reward, reduce energy consumption, so as to achieve the relative optimization of both sides.

5. Conclusions

This paper proposes a Stackelberg game-based incentive mechanism for asynchronous federated learning to analyze the distribution of benefits between task publishers and different data owners. What we have done is to introduce Stackelberg game into the design of incentive mechanism, which can better reflect the process of utility relationship and interaction between the two sides of the task, in order to seek the relative optimal solution of both the task publisher and the data owner. Specifically, we construct the utility objectives and constraints of the upper and lower level games based on the training model of Asynchronous Federation learning, and solve the equilibrium solutions of the lower level games and the upper level games. Through the simulation experiment, we prove and analyze the influence of different data quality data owners on the model iteration global time and its optimal CPU frequency. In this paper, we only consider the incentive mechanism design of asynchronous federated learning in Ideal State, but in reality, there may be some uncertain factors such as uncertain data quality, transmission loss and so on. These factors also affect the effect of incentive mechanism on asynchronous federated learning, so the design of incentive mechanism for asynchronous federated learning based on Stackelberg game with uncertain factors will be a further research direction.

References

[1] Roberts M, Driggs D, Thorpe M, et al. Common pitfalls and recommendations for using machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans [J]. Nature Machine Intelligence, 2021, 3(3):199-217.
[2] Gong L, Lin H, Li Z, et al. Systematically Landing Machine Learning onto Market-Scale Mobile Malware Detection [J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 7: 32.
[3] Yang Q, Liu Y, Chen T, et al. Federated Machine Learning: Concept and Applications [J]. ACM Transactions on Intelligent Systems and Technology, 2019, 10(2).
[4] Lu Y, Huang X, Dai Y, et al. Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT [J]. 2020, 16(6):4177-4186.
[5] Sattler F, Wiedemann S, Mueller K, et al. Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data [J]. IEEE Transactions on Neural Networks and Learning Systems, 2020, 31(9):3400-3413.
[6] Wang S, Tuor T, Salonidis T, et al. Adaptive Federated Learning in Resource Constrained Edge Computing Systems [J]. IEEE Journal on Selected Areas in Communications, 2019, 37(6): 1205-1221.