Federated Learning Optimization

Miguel Fernandes
mfernandes@student.dei.uc.pt
Joel P. Arrais
jpa@dei.uc.pt
Catarina Silva
catarina@dei.uc.pt
Alberto Cardoso
alberto@dei.uc.pt
Bernardete Ribeiro
bribeiro@dei.uc.pt

University of Coimbra
CISUC - Centro de Informática e Sistemas
FCTUC-DEI - Departamento de Engenharia Informática
Coimbra, Portugal

Abstract

In a recent approach defined as Federated Learning (FL), a single model is shared between a server and the clients instead of the data itself, reducing the amount of data transferred. In addition, FL attenuates the privacy concerns since each model is computed locally by their respective client and only the model is shared.

Federated Learning is still a recent technology and, as such, much research is yet to be done. This work presents the proposal and implementation of two Federated Learning algorithms and comparison with state of the art.

1 Introduction

Federated Learning (FL) [1] is a rising decentralized learning technology. While conventional Machine Learning methods require data to be centralized, Federated Learning allows multiple clients to learn a shared global model without needing to send their local data to a server. In addition, all of the model's training is done locally and coordinated by a central server.

In a FL setting, the clients receive a shared model from the server θ_t and train it with data which is only accessible to it. Afterwards, each client sends the updated models to the server. In the server, the uploaded models are aggregated in order to form a new model.

This work proposes two new methods which outperform the state of the art (Federated Averaging and Federated Proximal) of Federated Learning: Federated Directional Congruent Learning (FedCong) and Federated Momentum (FedMom). While the first is based on the directions of the models' updates, the second algorithm is based on the momentum of the global model.

1.1 Federated Averaging

Federated Averaging (FedAvg) [1] is a FL algorithm which generates the global model by periodically averaging the clients' locally trained models [1].

The algorithm starts by initializing a global model θ_t. Afterwards, at the t Communication Round (CR), the server selects a random subset of clients, K, and uploads the current global model to the clients. The chosen clients then train θ_t by performing stochastic gradient descent (SGD) locally for E epochs. Lastly, the clients upload the resulting model to the server where they are aggregated using a weighted average given by:

$$\theta_{t+1} = \sum_{k \in K} \frac{n_k}{n} \theta^k_{t+1} \tag{1}$$

where n is the sum of all clients' local data n_k. It is empirically shown in the work by H.Brendan McMahan [1] that the tuning of the number of local updates is of major importance for FedAvg to converge. It is clear that more local updates cause the model to be fitter for the local optimization problem and move further away from the initial model, possibly causing divergence.

1.2 Federated Proximal

Federated Proximal (FedProx) [4] was developed with the purpose of restricting the amount of divergence of the local model with regard to the global model, removing the need for heuristically limiting the number of local updates.

FedProx is similar to FedAvg with the difference being that each client local optimizer minimizes the objective given by:

$$\min h_k \quad \text{where} \quad h_k = F_k + \frac{\mu}{2} ||\theta_t - \theta^k_{t+1}||^2 \tag{2}$$

where $\frac{\mu}{2}||\theta - \theta^t||^2$ corresponds to the *proximal term*, which reduces the effect of local updates by making the local model θ^k_{t+1} closer to the global model θ_t. A cautious reader will note that if $\mu = 0$, then this algorithm is the same as the FedAvg algorithm.

2 Proposed Algorithms

The next sections contain the new Federated Learning algorithms proposed and implemented in this work, namely Federated Congruent Directional Learning (FedCong) and Federated Momentum Learning (FedMom).

2.1 Federated Congruent Directional Learning

In this section, the FedCong algorithm will be presented. This algorithm tries to mitigate a problem in FedAvg. As it was previously explained, in FedAvg, the bigger the number of updates, the more fitted the model is to the local optimization problem, potentially causing divergence. This divergence can lead to a decay in the model's convergence speed.

The FedCong algorithm was developed taking these facts into consideration. It is similar to the other methods with the main difference being that at the server, for each local model θ^k_{t+1} received, each weight w^k_{t+1} update for the local problem is analysed. On the one hand, in case $w^k_{t+1} < w_t$, then it can be concluded that w^k_{t+1} had a negative update. On the other hand, in case $w^k_{t+1} > w_t$, then it can be concluded that w^k_{t+1} had a positive update.

After this process in completed, for each weight this algorithm calculates the number of positive and negative updates of all the local models. Afterwards, one of three possible situations will occur:

$$\begin{cases} \text{if} \quad P \geq K * \alpha, & \text{then average the positive clients} \\ \text{if} \quad N \geq K * \alpha, & \text{then average the negative clients} \\ \text{otherwise}, & \text{then average all the clients} \end{cases} \tag{3}$$

where K is the number of selected clients, P and N are the number of positive and negative updates for a specific weight, respectively, and $\alpha(0,1)$ is a control parameter which specifies the minimum number of positive or negative updates that are necessary to average a weight using only the positive or negative updates.

2.2 Federated Momentum Learning

The FedMom algorithm was inspired by the Momentum optimizer [3, 5], having the objective of maximizing the training speed of the FedAvg algorithm. Momentum is known to help the gradient vector pointing to the right direction, damping oscillations and taking more straightforward paths to the local minimum.

The following equations (4, 5, 6) show how the momentum update can be reformulated into a Federated Learning setting. Firstly, FedMom

starts by initializing a global model, θ_t. Afterwards, similarly to FedAvg, the server selects K clients and uploads the current global model, θ_t. Subsequently, the clients selected take one or multiple SGD steps locally as follows:

$$\theta_{t+1}^k = \theta_t - \eta a_k \quad where \quad a_k = (g_e^k + g_{e-1}^k + ... + g_0^k) \tag{4}$$

where θ_t^k is the local model of the $k-th$ client, η is the learning rate, t represents the global model's timestamp, e represents the local model's timestamp, g_e^k are the error gradients of θ_e^k, and a_k is the sum of all the gradient updates of the $k-th$ client's local model.

Afterwards, the clients upload the resulting model θ_{t+1}^k to the server. In the server, for every θ_{t+1}^k the server calculates the local update. Then, it calculates the global update of the model using each local update as follows:

$$\begin{aligned} -\eta a_k &= \theta_{t+1}^k - \theta_t \\ \alpha &= \sum_{k=1}^{K} \frac{n_k}{n}(-\eta a_k) \end{aligned} \tag{5}$$

where α is the global model update, n represents the total number of data points and n_k represents the number of data points of the $k-th$ client.

After this process is completed, the server stores the momentum variable and updates the global model as follows:

$$\begin{aligned} v_{t+1} &= \delta v_t + \alpha \\ \theta_{t+1} &= \theta_t + v_{t+1} \end{aligned} \tag{6}$$

where δ is the momentum term. As it can be observed, the momentum update (v_{t+1}) is calculated by summing a fraction of the previous update and the global model's update. Afterwards, the global model is updated by summing the previous model with the momentum update.

3 Experimentation

In this section, the experimental results of the Federated Learning optimizers will be presented. The algorithms tested were: FedAvg, FedProx, FedCong and FedMom.

In the scope of this work, the baseline model considered is the FedAvg since it is the most widely used algorithm for Federated Learning. In addition, FedProx is also evaluated. The primary objective of this work is that FedCong and FedMom outperform FedAvg by increasing the convergence speed of the models while maintaining the Mean Absolute Error (MAE). The best algorithm is the one whose CR of stabilization is the lowest without increasing the MAE.

The experimentation was done using the Turbofan Dataset. The Turbofan dataset is composed of four sub-datasets (FD001, FD002, FD003 and FD004). Each dataset has a time series readings of 26 features, such as Operational Settings, unit number, time indicator and 21 sensors' values regarding the turbofan engine components. At the start of each series, the system operates in a healthy condition until some point in time where it enters a failure state and can no longer function. This degradation is captured by the time indicator feature.

In addition, after some literature review, it was concluded that only 14 out of the 21 sensors presented valuable information to be used as input features.

For each dataset, J clients were created so that each client has exactly two time series. As such, each client only has a small portion of the data. In the following experimentation, for each CR, K clients were randomly selected from the J clients. The value of K for these experiments was set to 20.

Each client of the FL setting is represented by a Feed Forward Neural Network which has the objective of predicting the system's health percentage. This neural network takes as input the preprocessed features' values. The network architecture is as follows: three hidden layers with 20, 30 and 20 neurons, respectively, with *tanh* activation functions; the output layer is a single neuron with a *sigmoid* activation function, which represents the predicted system's health percentage. The local optimizer used was SGD and the error function used was the Mean Square Error (MSE).

In Figure 1, the results of the proposed methods are presented for the four different datasets (FD001, FD002, FD003 and FD004). The step size η is similar between all the algorithms. The α and δ from FedCong and FedMom, respectively, were tuned in order to obtain the best performance. It can be observed that the two new proposed algorithms converge faster than the others while maintaining the same MAE.

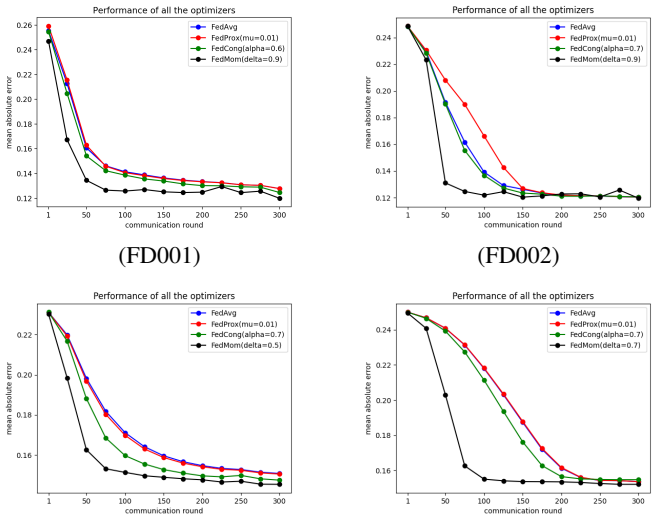

Figure 1: Performance comparison between FedAvg, FedProx, FedCong and FedMom

4 Conclusion

The main contribution of this work are the proposal and development of two new Federated Learning algorithms, namely FedCong and FedMom, and comparison with state of the art. These methods have the objective of improving the convergence speed of the Federated Learning models.

Although FedCong and FedMom greatly increased the convergence speed of the models, some limitations should be considered. Firstly, in FedCong, it is of most importance to tune the control term, α, with respect to the data distribution. If the α value is low and the current CR has a bad error representation, the global model will have difficulties to converge.

As for FedMom, the momentum parameter δ has to be tuned in order for the model to converge. During a model's update that has a poor representation of the global error, a large momentum term can cause the model to diverge even further and have constant fluctuations. This can cause difficulties in the convergence of the model.

For future work it is suggested to improve the FedCong algorithm by taking into consideration more than just the direction of the gradient descend step. For example, taking into consideration also the size of the step may help reducing the fluctuation of the global model.

In addition, it is suggested to compare the FedMom algorithm to the work proposed by Huo et al. [2] where the authors use the current model's update to estimate the new weight value, instead of using an exponentially weighted average.

References

[1] Daniel Ramage Seth Hampson Blaise Aguera y Arcass H.Brendan McMahan, Eider Moore. Communication-efficient learning of deep networks from decentralized data. In *In AAAI Fall Symposium*.

[2] Zhouyuan Huo, Qian Yang, Bin Gu, and Lawrence Carin. Heng Huang. Faster on-device training using new federated momentum algorithm, 2020.

[3] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*.

[4] Anit Kumar Sahu, Tian Li, Maziar Sanjabi, Manzil Zaheer, Ameet Talwalkar, and Virginia Smith. On the convergence of federated optimization in heterogeneous networks.

[5] Tianbao Yang, Qihang Lin, and Zhe Li. Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *arXiv: Optimization and Control*.

*/