 doi:10.20944/preprints202001.0207.v1

Article

Averaging is Probably not the Optimum Way of Aggregating Parameters in Federated Learning

Peng Xiao [1], Samuel Cheng [1,2,*], Vladimir Stankovic [3] and Dejan Vukobratovic [4]

[1] Department of Computer Science and Technology, Tongji University, Shanghai, China; phd.xiaopeng@gmail.com
[2] the School of Electrical and Computer Engineering, University of Oklahoma, Tulsa, USA; samuel.cheng@ou.edu
[3] Department of Electronic and Electrical engineering, University of Strathclyde, Glasgow, UK
[4] Faculty of Technical Sciences, University of Novi Sad, Novi Sad, Serbia.
* Correspondence: samuel.cheng@ou.edu

Abstract: Federated learning is a decentralized topology of deep learning, that trains a shared model through data distributed among each client (like mobile phones, wearable devices), in order to ensure data privacy by avoiding raw data exposed in data center (server). After each client computes a new model parameter by stochastic gradient decrease (SGD) based on their own local data, all locally-computed parameters will be aggregated in the server to generate an updated global model. Almost all current studies directly average different client computed parameters by default, but no one gives an explanation why averaging parameters is a good approach. In this paper, we treat each client computed parameter as a random vector because of the stochastic properties of SGD, and estimate mutual information between two client computed parameters at different training phases using two methods in two learning tasks. The results confirm the correlation between different clients and show an increasing trend of mutual information with training iteration. However, when we further compute the distance between client computed parameters, we find that parameters are getting more correlated while not getting closer. This phenomenon suggests that averaging parameters may not be the optimum way of aggregating trained parameters.

Keywords: federated learning; federated averaging; mutual information; correlation

1. Introduction

Nowadays, more and more intelligent devices, such as smart phones, wearable devices and autonomous vehicles, are widely used [1],[2], generating a wealth of data. The generated data can be used to develop deep learning models powering applications such as speech recognition, face detection and text entry. With the increasing amount of generated data and increasing computing power of the smart devices[3], recent studies have explored distributed training of models at these edging devices [4],[5]. Considering that centralized storage of data in the central server, e.g., in the cloud, is often not feasible due to privacy of consumer data [6], *federated learning* [7] has been advocated as an alternative setting.

Federated learning[7] can be viewed as an extension of conventional distributed deep learning[8], as it aims to train a high quality shared model while keeping data distributed over clients. That is, each client computes an updated model based on his/her own locally collected data (that is not shared with others). After all model parameters are computed locally, they are sent to the server where an updated global model is computed by combining the received local model parameters.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 January 2020 doi:10.20944/preprints202001.0207.v1

Federated learning plays a critical role in various privacy-sensitive scenarios, such as power intelligence applications like keyboard prediction and emoji prediction in smart phone [9] [10], optimising patient's healthcare in hospitals[11], helping internet of things (IoT) systems adapt to environmental changes[12] etc. Improving communication efficiency is a key aspect in federated learning. Proposed approaches include allowing local updates to reduce the total number of communication rounds [7][13], quantization[14] and sparse compression[15] to reduce the size of messages at each round. Active sampling[16][17] and fault tolerance[18] are discussed in some studies to handle systems heterogeneity. Some research has focused on further enhancing the privacy using various cryptographic protocols[19][20], or applying differential privacy[21] to confuse attackers. For a more complete review on federated learning we refer readers to [22].

Specifically, assume there are Q participating clients over which the data is partitioned, with P_q being the set of indices of data points at client q, and $n_q = |P_q|$. A federated learning system optimizes a global model by repeating the following steps:

1) all participating clients download the latest global model parameter vector w^r.
2) each client improves downloaded model based on their local data using, e.g., stochastic gradient descent (SGD)[23] with a fixed learning rate η: $w_q = w^r - \eta g_q$, where $g_q = \bigtriangledown F_q(w^r), F_q(w^r) = \frac{1}{n_q}\sum_{i\in P_q} l(data_i, label_i; w^r)$, where $l(\cdot)$ is the loss of the prediction on the subset P_q made with the model parameter vector w^r.
3) each improved model parameter w_q is sent from the client to the server.
4) the server aggregates all updated parameters to construct an enhanced global model via $w^* = \sum_{q=1}^{|Q|} \frac{n_q}{n} w_q$, where $n = \sum_{q=1}^{|Q|} n_q$.

Note that Step 4, as most current studies[9][10][12][7][13][14][15], calculate the updated model parameters by directly averaging the models received from all clients. However, after each client computed w_q by optimizing their local objective function F_q, there is no evidence w^* will be close to the optimized value of the global objective function $\sum_q \frac{n_q}{n} F_q(w)$. That raises a question: is averaging parameters a reasonable approach in federated learning? With this question we begin our research.

Mutual information (MI) is a powerful statistic for measuring the degree of relatedness[24]. MI can detect any kind of relationship between random variables even those that do not manifest themselves in the covariance[25], and has a straightforward interpretation as the amount of shared information between datasets (measured in, for example, bits)[26]. In this paper, we treat each client's computed parameter vector w_q as an random vector because of the stochastic properties of SGD, and train each client in one training epoch many times to obtain a series of parameter samples. We firstly calculate the MI between different parameters in two learning tasks: (1) convolutional neural network (CNN) [27] designed for a classification task and (2) recurrent neural network (RNN) [28] on a regression task. By using two methods, one is an analytical method through the MI derivative formula under multi-dimensional Gaussian distribution assumption, and another is a discretization estimator which calculates MI between arbitrary datasets with unknown distribution based on k-nearest neighbor (KNN) distances described in[25], we estimate MI at different training phases. The results confirm the correlation between the model parameters at different clients, and show an increasing trend of MI. We further calculate the distance variation between different parameters using three regular distance metrics: Euclidean distance[29], Manhattan distance[30], Chebyshev distance[31]. By comparing the distance variation with MI variation, we show that the parameters are getting more correlated while not getting closer. A proposition derived in Section 3 implies that averaging parameters is not supported by theory and requires further scrutiny.

The rest of the paper is organized as follows. In Section 2, we describe the details of estimating MI. Section 3 shows the experiment results. Conclusion and future work are discussed in Section 4.

2. Estimating the Mutual Information

In this section we show two methods to estimate MI between model parameters at two clients. To do that, we need to perform training at each client many times to acquire enough samples of each

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 January 2020 doi:10.20944/preprints202001.0207.v1

parameter vector. Generally speaking, let X and Y be the model parameters computed at two different clients, $X, Y \in \mathbb{R}^t$. Let $Z = (X, Y), Z \in \mathbb{R}^{2t}$ be the joint variable of X and Y. After training at the two clients N times, we get N samples of X and Y, denoted by $x_1, x_2, \ldots, x_N$ and $y_1, y_2, \ldots, y_N$, and also N samples of Z, $z_1, z_2, \ldots, z_N, z_i = (x_i, y_i)$.

2.1. MI Derivative Formula under Multi-D Gaussian

In the first case, we assume that both X and Y satisfy the multi-dimensional Gaussian distribution, i.e., $X \sim \mathcal{N}(\mu_x, \Sigma_x), Y \sim \mathcal{N}(\mu_y, \Sigma_y)$, which implies that Z also satisfies the multi-dimensional Gaussian distribution $Z \sim \mathcal{N}(\mu_z, \Sigma_z)$.

Firstly, we calculate Z's covariance matrix Σ_z according to the N samples:

$$\Sigma_z = \begin{pmatrix} cov(X^1, X^1) & \ldots & cov(X^1, X^t) & cov(X^1, Y^1) & \ldots & cov(X^1, Y^t) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ cov(X^t, X^1) & \ldots & cov(X^t, X^t) & cov(X^t, Y^1) & \ldots & cov(X^t, Y^t) \\ cov(Y^1, X^1) & \ldots & cov(Y^1, X^t) & cov(Y^1, Y^1) & \ldots & cov(Y^1, Y^t) \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ cov(Y^t, X^1) & \ldots & cov(Y^t, X^t) & cov(Y^t, Y^1) & \ldots & cov(Y^t, Y^t) \end{pmatrix}, \tag{1}$$

where $X^i, Y^j, i, j = 1, 2, \ldots, t$ denote ith and jth dimension of X and Y, and $cov(X^i, Y^j) = E[X^i - E[X^i]][Y^j - E[Y^j]]$ denotes the covariance between X^i and Y^j. The mean of each variable is estimated from the collected samples.

Then, to calculate $H(Z) = H(X, Y)$, we use the entropy formula for multi-dimensional Gaussian distribution [32]:

$$H(Z) = E[-\log p(Z)] = \log \sqrt{det(2\pi e \Sigma_z)}, Z \sim \mathcal{N}(\mu_z, \Sigma_z). \tag{2}$$

Since Z is the joint distribution of X and Y, from Eq. (1) we can infer that $\Sigma_x = \Sigma_z[1:t, 1:t], \Sigma_y = \Sigma_z[t+1:2t, t+1:2t]$, and through Eq. (2), we then calculate $H(X)$ and $H(Y)$ as:

$$H(X) = \log \sqrt{det(2\pi e \Sigma_z[1:t, 1:t])}, \tag{3}$$

$$H(Y) = \log \sqrt{det(2\pi e \Sigma_z[t+1:2t, t+1:2t])}. \tag{4}$$

Finally, through the relationship formula between entropy and MI[32],

$$I(X, Y) = H(X) + H(Y) - H(X, Y), \tag{5}$$

we calculate the mutual information between X and Y.

2.2. KNN Discretization Estimator

In the previous subsection, we estimated MI under multi-Gaussian assumption. In this subsection, we show how to estimate MI with unknown distribution by the averaging k-nearest neighbor distance[25], which is one of the most commonly used continuous-continuous MI discretization estimators.

Specifically, let us denote by $\epsilon(i)$ the distance from z_i to its kth neighbor. We count the number $n_x(i)$ of points, x_j, whose distance to x_i is strictly less than $\epsilon(i)/2$, and similarly for Y. The estimate for MI is then:

$$I(X, Y) = \psi(k) - \frac{1}{N} \sum_{i=1}^{N} (\psi(n_x(i) + 1) + \psi(n_y(i) + 1)) + \psi(N), \tag{6}$$

 doi:10.20944/preprints202001.0207.v1

where $\psi(\cdot)$ is the digamma function, and k is a hyperparameter. In this paper we choose k to 3 after tuning. This method is mainly using the probability of points being in the kth nearest distance, to approximately approach the true probability density, and has an advantage in vastly reducing systematic errors. For more details please refer to [25].

3. Simulation Results

In simulations, considering the time it takes to train a model, we picked two relative small tasks. One is training a simple RNN model to predict the shift and scale variation of a signal sequence. The second one is applying transfer learning to train a CNN model on the MNIST[33] digit recognition task. For both tasks we train model parameters at three clients (indexed by 0, 1, 2) in a whole training epoch many times, and estimate the MI between three model parameters at different training phases.

3.1. RNN on Signal Variation Prediction

In the first experiment, we consider training an RNN model to predict the shift and scale variation of an arbitrary randomized 1-d signal sequence. Each client contains 100 training examples, where each training example is a signal sequence of length 64, generated by randomizing the source. The total samples obtained for each client is 22000.

The model architecture we use is a simple RNN with 10 layers and 4 units, where each layer is wrapped by Dropout function[34], for a total of 394 parameters. The metric we use is the *mean squared error (MSE)*. Though this model is not state of the art, it is sufficient for our purpose, as our goal is to estimate the MI between clients, not to achieve the best possible performance for this task.

Figure 1. (a)(b) shows the MI estimations in the regression task by two methods. The unit of MI is *bit*. Client(0, 1), Client(0,2), Client(1, 2), respectively, indicate MI between model parameters as Client 0 and 1, Client 0 and 2, Client 1 and 2. From the results of both methods, we can see that the MI between all of three clients are always larger than zero throughout the whole training epoch, which shows the correlation between different clients in the federated mode. **Figure 1. (c)** shows the decrease of MSE with training iteration. All three models were trained well in their own data, and all of them eventually converge to a similar MSE with similar training patterns. Comparing to similar MSE variations, **Figure 1. (a)(b)** show the difference of MI between different client-pairs.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 January 2020 doi:10.20944/preprints202001.0207.v1

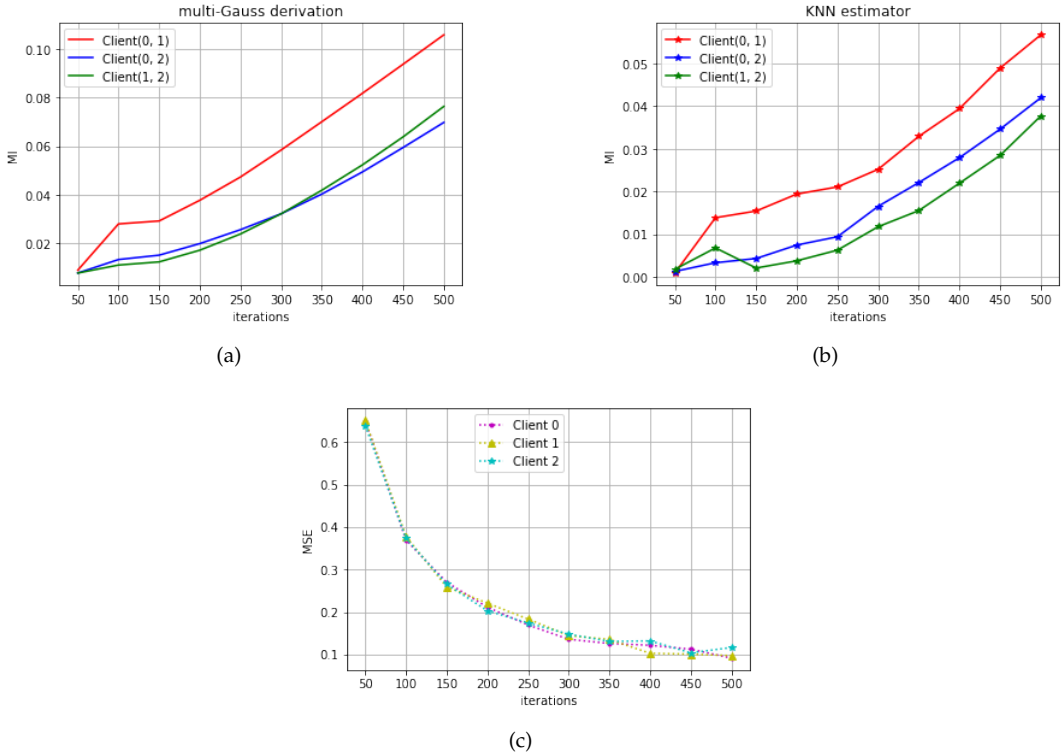

Figure 1. (**a**) The MI in the signal variation prediction task calculated by the first method (Subsection 2.1). (**b**) The MI in the signal variation prediction task calculated by the second method (Subsection 2.2). (**c**) The MSE variation with the training iterations.

3.2. CNN on Digit Recognition

In the second experiment, we firstly train the whole *Lenet* model[35] on the MNIST dataset, and then apply transfer learning by keeping the trained *Lenet* model unchanged except the last fully connected layer and apply it on augmented data. This way we reduce the number of training parameters to 850. Each client contains 200 training examples and each training example is an augmentation of a randomized selection of an original image. The augmented operation includes rotation, zoom, shift (all of these operations are performed in a certain range). The total number of samples obtained at each client is 26400. The metric we use is recognition *accuracy*.

Figure 2. (a)(b) shows the obtained MI estimation. The unit of MI is *bit*. Similar to the previous experiment, the MI between all of three clients is larger than zero throughout the training phase, which shows that there exists correlation between the clients' parameter models. **Fig. 2 (c)** shows improvement in recognition accuracy with training iteration. All of three models also have similar training pattern like in regression tasks, but still looks different in MI estimations. These implies although different clients in a same learning task have similar training trajectory, the MI can still the subtle difference between them.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 January 2020 doi:10.20944/preprints202001.0207.v1

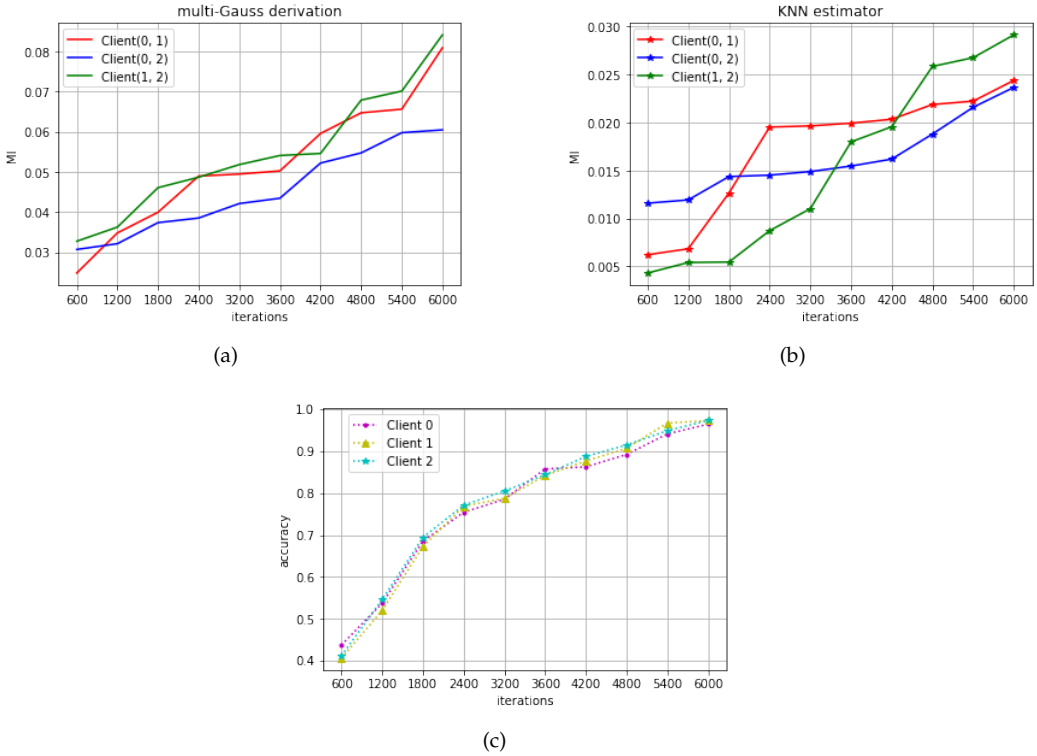

Figure 2. (**a**) The MI in the digit recognition task calculated by the first method (Subsection 2.1). (**b**) The MI in the digit recognition task calculated by the second method (Subsection 2.2). (**c**) The accuracy variation with the training iterations.

3.3. Distance Metrics

The MI results presented in the previous section indicate an upward trend of correlation between different client parameters with the training iterations. The increase in correlation seems to imply that the parameters trained from different clients converge. To verify this, in this subsection, we measure directly the difference of the parameter models at different clients using three different metrics.

We detect the distance between the model parameters at different clients using three regular distance metrics: Euclidean distance, Manhattan distance, Chebyshev distance, and refer to them as D_{Eu}, D_{Man}, D_{Che}, respectively. Specially, to calculate the distance between two model parameters, we firstly calculate the distance between two samples of the client computed parameters, and then average the result over all samples, i.e.,:

$$D_{Eu} = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\sum_{k=1}^{t}(x_i^k - y_i^k)^2}, \tag{7}$$

$$D_{Man} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{t}|x_i^k - y_i^k|, \tag{8}$$

$$D_{Che} = \frac{1}{N}\sum_{i=1}^{N}\max_k(|x_i^k - y_i^k|). \tag{9}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 January 2020 doi:10.20944/preprints202001.0207.v1

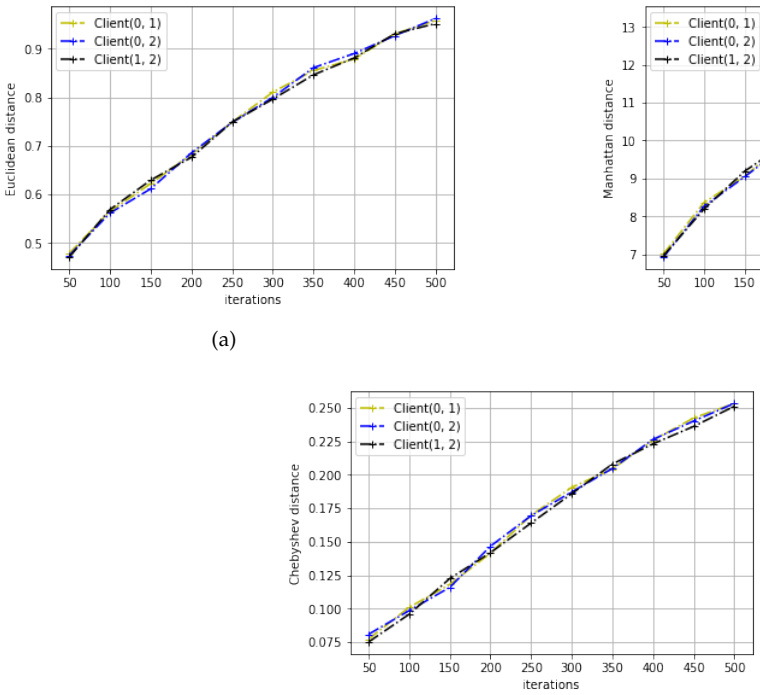

Figure 3. The distance metrics in the signal variation prediction task (**a**) Euclidean distance. (**b**) Manhattan distance. (**c**) Chebyshev distance.

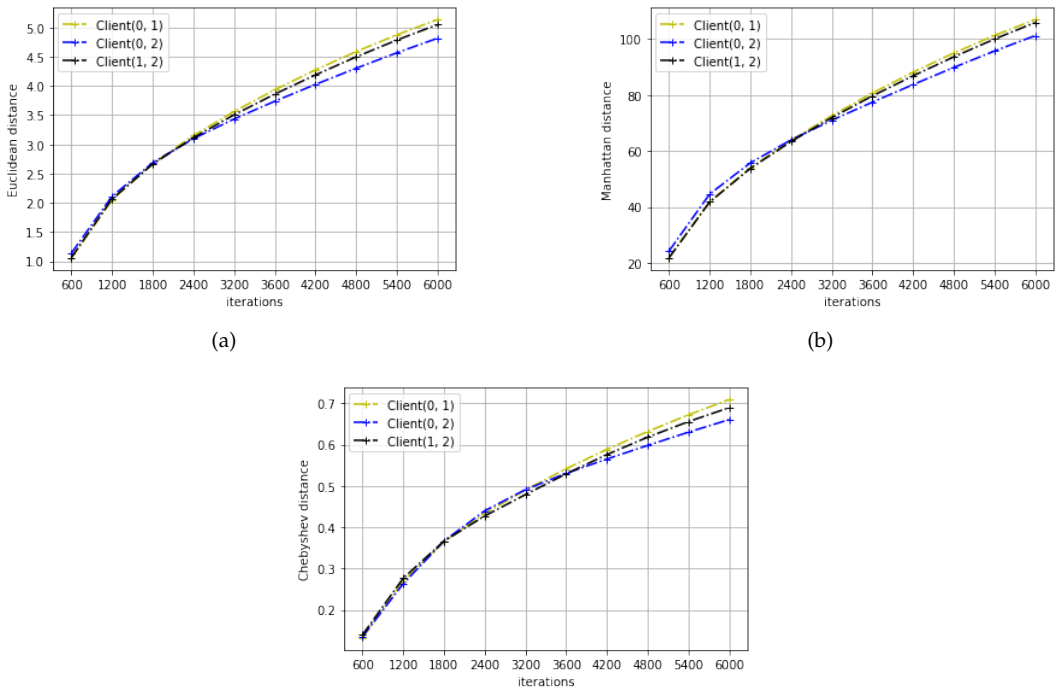

Figure 4. The distance metrics in the digit recognition task (**a**) Euclidean distance. (**b**) Manhattan distance. (**c**) Chebyshev distance.

The results of distance variation for the two learning tasks can be seen in **Figure 3** and **Figure 4**. As we can see, for both learning tasks, all three distance metrics between any client pairs increase rather

 doi:10.20944/preprints202001.0207.v1

than decrease with the training iteration. This may not be surprising as we initialize the parameters to be the same across all clients. But this contradicts with the conclusion of the previous section that the parameters converge. Studying the trend of ML and distance change with the training iteration, the following proposition is derived:

Proposition 1. *In federated learning, the model parameters of two clients X and Y cannot be modelled by the following additive model: $X = P + N_1, Y = P + N_2$, where N_1 and N_2 are independent noises and independent of some optimum set of parameters P.*

Proof. Let us assume that X and Y satisfy the proposed additive model, then $X = Y + N_1 - N_2$. Mutual information between X and Y increases as noise variances of N_1 and N_2 decreases, which also leads to decreasing distance between X and Y. This contradicts our simulation results shown in Figs 3 and 4. Therefore, the model parameters of the two clients cannot be modelled in the proposed form. □

Averaging is a very restrictive estimate. According to Proposition 1, if the parameters cannot be the modelled using an additive noise model, there is no particular reason why averaging different parameters would result in the optimal or even a reasonably good set of parameters. Although averaging parameters achieved good results in some learning tasks, it is not a best or even reasonable choice in general federated learning.

4. Conclusion

This study explores the correlation between clients' model parameters in federated learning. Through estimating the MI between different client computed parameters in two learning tasks by two methods, we confirm the existence of correlation between different clients. By further studying variation of three distance metrics, we show that model parameters at different clients are getting more correlated while not getting closer as with training iterations. All of these implies that the aggregating parameters by averaging the local model parameters of each client directly may not be a reasonable approach in general.

The existence of correlation implies the possibility that distributed source coding can be applied to compress the information in federated learning. Thus we can explore compressing information to promote communication efficiency in federated learning. The derived Proposition also implies there maybe some better choice than averaging parameters in the federated mode, which can be a future direction in federated learning.

References

1. Poushter, J.; others. Smartphone ownership and internet usage continues to climb in emerging economies. *Pew Research Center* **2016**, *22*, 1–44.
2. Haghi, M.; Thurow, K.; Stoll, R. Wearable devices in medical internet of things: scientific research and commercially available devices. *Healthcare informatics research* **2017**, *23*, 4–15.
3. Taylor, R.; Baron, D.; Schmidt, D. The world in 2025-predictions for the next ten years. 2015 10th International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT). IEEE, 2015, pp. 192–195.
4. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012, pp. 13–16.
5. Garcia Lopez, P.; Montresor, A.; Epema, D.; Datta, A.; Higashino, T.; Iamnitchi, A.; Barcellos, M.; Felber, P.; Riviere, E. Edge-centric computing: Vision and challenges. *ACM SIGCOMM Computer Communication Review* **2015**, *45*, 37–42.
6. House, W. Consumer data privacy in a networked world: A framework for protecting privacy and promoting innovation in the global digital economy. *White House, Washington, DC* **2012**, pp. 1–62.

7. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S.; others. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629* **2016**.
8. Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Ranzato, M.; Senior, A.; Tucker, P.; Yang, K.; others. Large scale distributed deep networks. Advances in neural information processing systems, 2012, pp. 1223–1231.
9. Hard, A.; Rao, K.; Mathews, R.; Ramaswamy, S.; Beaufays, F.; Augenstein, S.; Eichner, H.; Kiddon, C.; Ramage, D. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604* **2018**.
10. Ramaswamy, S.; Mathews, R.; Rao, K.; Beaufays, F. Federated learning for emoji prediction in a mobile keyboard. *arXiv preprint arXiv:1906.04329* **2019**.
11. Huang, L.; Yin, Y.; Fu, Z.; Zhang, S.; Deng, H.; Liu, D. LoAdaBoost: Loss-Based AdaBoost Federated Machine Learning on medical Data. *arXiv preprint arXiv:1811.12629* **2018**.
12. Samarakoon, S.; Bennis, M.; Saad, W.; Debbah, M. Federated learning for ultra-reliable low-latency V2V communications. 2018 IEEE Global Communications Conference (GLOBECOM). IEEE, 2018, pp. 1–7.
13. Smith, V.; Chiang, C.K.; Sanjabi, M.; Talwalkar, A.S. Federated multi-task learning. Advances in Neural Information Processing Systems, 2017, pp. 4424–4434.
14. Konečnỳ, J.; McMahan, H.B.; Yu, F.X.; Richtárik, P.; Suresh, A.T.; Bacon, D. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492* **2016**.
15. Sattler, F.; Wiedemann, S.; Müller, K.R.; Samek, W. Robust and communication-efficient federated learning from non-iid data. *arXiv preprint arXiv:1903.02891* **2019**.
16. Nishio, T.; Yonetani, R. Client selection for federated learning with heterogeneous resources in mobile edge. ICC 2019-2019 IEEE International Conference on Communications (ICC). IEEE, 2019, pp. 1–7.
17. Kang, J.; Xiong, Z.; Niyato, D.; Yu, H.; Liang, Y.C.; Kim, D.I. Incentive Design for Efficient Federated Learning in Mobile Networks: A Contract Theory Approach. *arXiv preprint arXiv:1905.07479* **2019**.
18. Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C.; Konecny, J.; Mazzocchi, S.; McMahan, H.B.; others. Towards federated learning at scale: System design. *arXiv preprint arXiv:1902.01046* **2019**.
19. Bonawitz, K.; Ivanov, V.; Kreuter, B.; Marcedone, A.; McMahan, H.B.; Patel, S.; Ramage, D.; Segal, A.; Seth, K. Practical secure aggregation for privacy-preserving machine learning. Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. ACM, 2017, pp. 1175–1191.
20. Ghazi, B.; Pagh, R.; Velingker, A. Scalable and Differentially Private Distributed Aggregation in the Shuffled Model. *arXiv preprint arXiv:1906.08320* **2019**.
21. McMahan, H.B.; Ramage, D.; Talwar, K.; Zhang, L. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963* **2017**.
22. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated learning: Challenges, methods, and future directions. *arXiv preprint arXiv:1908.07873* **2019**.
23. Bottou, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*; Springer, 2010; pp. 177–186.
24. Cover, T.M.; Thomas, J.A. *Elements of information theory*; John Wiley & Sons, 2012.
25. Kraskov, A.; Stögbauer, H.; Grassberger, P. Estimating mutual information. *Physical review E* **2004**, *69*, 066138.
26. Ross, B.C. Mutual information between discrete and continuous data sets. *PloS one* **2014**, *9*, e87357.
27. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012, pp. 1097–1105.
28. Mikolov, T.; Karafiát, M.; Burget, L.; Černockỳ, J.; Khudanpur, S. Recurrent neural network based language model. Eleventh annual conference of the international speech communication association, 2010.
29. Danielsson, P.E. Euclidean distance mapping. *Computer Graphics and image processing* **1980**, *14*, 227–248.
30. Krause, E.F. *Taxicab geometry: An adventure in non-Euclidean geometry*; Courier Corporation, 1986.
31. Cantrell, C.D. *Modern mathematical methods for physicists and engineers*; Cambridge University Press, 2000.
32. Shannon, C.E. A mathematical theory of communication. *Bell system technical journal* **1948**, *27*, 379–423.
33. Deng, L. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine* **2012**, *29*, 141–142.
34. Gal, Y.; Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. international conference on machine learning, 2016, pp. 1050–1059.

 doi:10.20944/preprints202001.0207.v1

35. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P.; others. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **1998**, *86*, 2278–2324.