



Article

# Averaging Is Probably Not the Optimum Way of Aggregating Parameters in Federated Learning

Peng Xiao <sup>1</sup> , Samuel Cheng <sup>1,2,\*</sup>, Vladimir Stankovic <sup>3</sup> and Dejan Vukobratovic <sup>4</sup>

<sup>1</sup> Department of Computer Science and Technology, Tongji University, Shanghai 201804, China; phd.xiaopeng@gmail.com

<sup>2</sup> the School of Electrical and Computer Engineering, University of Oklahoma, Tulsa, OK 73019, USA

<sup>3</sup> Department of Electronic and Electrical engineering, University of Strathclyde, Glasgow G1 1XW, UK; vladimir.stankovic@strath.ac.uk

<sup>4</sup> Faculty of Technical Sciences, University of Novi Sad, 21000 Novi Sad, Serbia; dejanv@uns.ac.rs

\* Correspondence: samuel.cheng@ou.edu

Received: 3 March 2020; Accepted: 9 March 2020; Published: 11 March 2020



**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 descent (SGD) based on their own local data, these locally-computed parameters will be aggregated to generate an updated global model. Many current state-of-the-art studies aggregate different client-computed parameters by averaging them, but none theoretically explains 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; decentralized learning; 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.

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. 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 summarized as follows:

- (1) All participating clients download the latest global model parameter vector  $w$ .
- (2) Each client improves downloaded model based on their local data using, e.g., stochastic gradient descent (SGD) [13] with a fixed learning rate  $\eta$ :  $w_q = w - \eta g_q$ , where  $g_q = \nabla F_q(w)$ , and  $F_q(w)$  is the local objective function for the  $q$ th device.
- (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  $w^*$ .

Current studies on federated learning and its variants address a series of challenges. In [14], they compress the model in both upstream and downstream links to promote communication efficiency. In [15], they demonstrate user-level differential privacy in large recurrent language models. In [16], they propose algorithms that achieve both communication efficiency and differential privacy. In [17], they propose a decentralized training framework to overcome issues related to heterogeneous devices. In [18], they suggest a general oracle-based framework in parallel stochastic optimization. In [19], they investigate a pluralistic solution to address the cyclic patterns in data samples during federated training. In these state-of-the-art studies, at step 4) above, the updated model parameters are calculated by directly averaging the models received from all clients via  $w^* = \sum_{q=1}^{|Q|} \frac{n_q}{n} w_q$ , where  $n = \sum_{q=1}^{|Q|} n_q$ . However, after each client computes  $w_q$  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 model 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 [20]. MI can detect any kind of relationship between random variables even those that do not manifest themselves in the covariance [21], and has a straightforward interpretation as the amount of shared information between datasets (measured in, for example, bits) [22]. In this paper, we treat each client's computed parameter vector  $w_q$  as a 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) [23] designed for a classification task and (2) recurrent neural network (RNN) [24] on a regression task. By using two methods, one is base on the MI derivative formula under the multi-dimensional Gaussian distribution assumption, and another is a continuous-continuous MI estimator based on k-nearest neighbor (KNN) distances described in [21]. 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 [25], Manhattan distance [26], Chebyshev distance [27]. 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 provide an overview of related work. In Section 3, we describe the details of estimating MI. Section 4 shows the experiment results. Conclusion and future work are discussed in Section 5.

## 2. Related Work

In [28], the authors represent Deep Neural Networks (DNNs) as a Markov chain and propose to use information theoretical tools, and in particular, Information Plane (IP)—the plane of the MI values of any other variable with the input variable (input data) and desired output variable (label), to study and visualise DNNs. In [29], they extend the approach of [28] demonstrating the effectiveness of the Information Plane visualization of DNNs. In particular, the authors of [29] suggest that the training process of DNNs is composed of an initial fitting phase and a representation compression phase, where the goal of the network is to optimize the Information Bottleneck (IB) tradeoff between compression and prediction. These results are later questioned by [30], who argue that the compression phase is not general in the DNN training process. However, [31] propose ensemble dependency graph estimator (EDGE) to estimate MI supporting the claim of the compression phase. Following this line of work, [32] propose a method for performing IP on arbitrarily-distributed discrete and/or continuous variables, while [33] use matrix-based Rényi's entropy to provide comprehensive IP analysis of large DNNs.

There are few other attempts to study federated learning from an information-theoretic perspective. In [34], they apply information theory to understand the aggregation process of DNNs in the federated learning setup. The authors use EDGE [31] to measure the MI between representation and inputs as well as representation and labels in each local model, and compare them to the respective information contained in the representation of the averaged model. In [34], they leave as an open question for the theoretical analysis of when and how to aggregate local models. In contrast to [34] and the other work reviewed above, in this paper, we treat each local model as a continuous random vector, and directly measure the MI between different local models through the training phase using two methods. By doing so we explore the correlation between each local model. By further comparing with the distance variation between local models, we propose that averaging may not be the optimum way of aggregating trained parameters.

## 3. 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 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  $x_1, x_2, \dots, x_N$  and  $y_1, y_2, \dots, y_N$ , and also  $N$  samples of  $Z, z_1, z_2, \dots, z_N, z_i = (x_i, y_i)$ .

### 3.1. 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  $\Sigma_z$  according to the  $N$  samples:

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

where  $X^i, Y^j, i, j = 1, 2, \dots, t$  denote  $i$ th and  $j$ th 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 [35]:

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

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:

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

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

Finally, through the relationship formula between entropy and MI[35],

$$I(X, Y) = H(X) + H(Y) - H(X, Y), \quad (5)$$

we calculate the mutual information between  $X$  and  $Y$ .

### 3.2. 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[21], which is one of the most commonly used continuous-continuous MI discretization estimators. Note that there are other methods to estimate MI, such as EDGE [31]; however, EDGE method [31] is better for estimating a continuous-discrete mixture MI.

Specifically, let us denote by  $\epsilon(i)$  the distance from  $z_i$  to its  $k$ th 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:

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

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  $k$ th nearest distance, to approximately approach the true probability density, and has an advantage in vastly reducing systematic errors. For more details please refer to [21].

## 4. Simulation Results

In simulations, considering the time it takes to train a model and the limitations of our hardware, we selected two relatively small tasks and perform them in local iid dataset setting. 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[36] digit recognition dataset. 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.

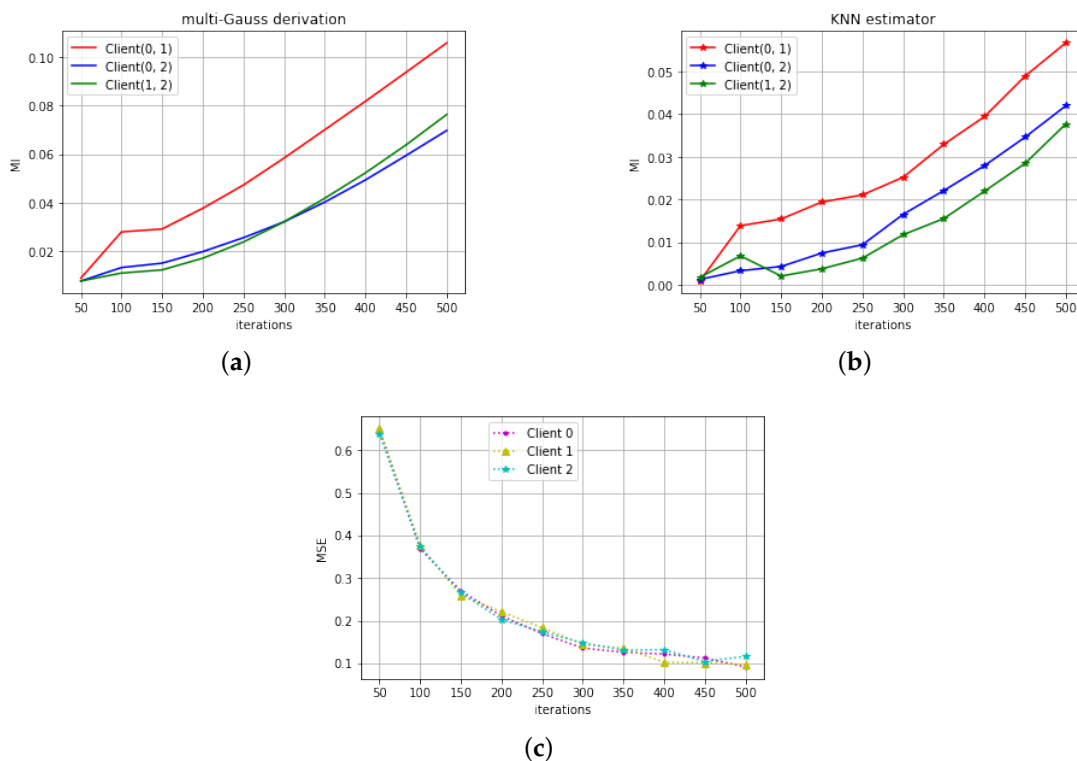
### 4.1. RNN on Signal Variation Prediction

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 model is initialized by the normal initializer, and optimized by the Adam [37] optimizer with learning rate 0.001. The mini-batch size is 64. The total number of samples obtained for each client is 22,000.

The model architecture we use is a simple RNN with 10 layers and four units, where each layer is wrapped by Dropout function [38], 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 1a,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 1c 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 1a,b show the difference of MI between different client-pairs.



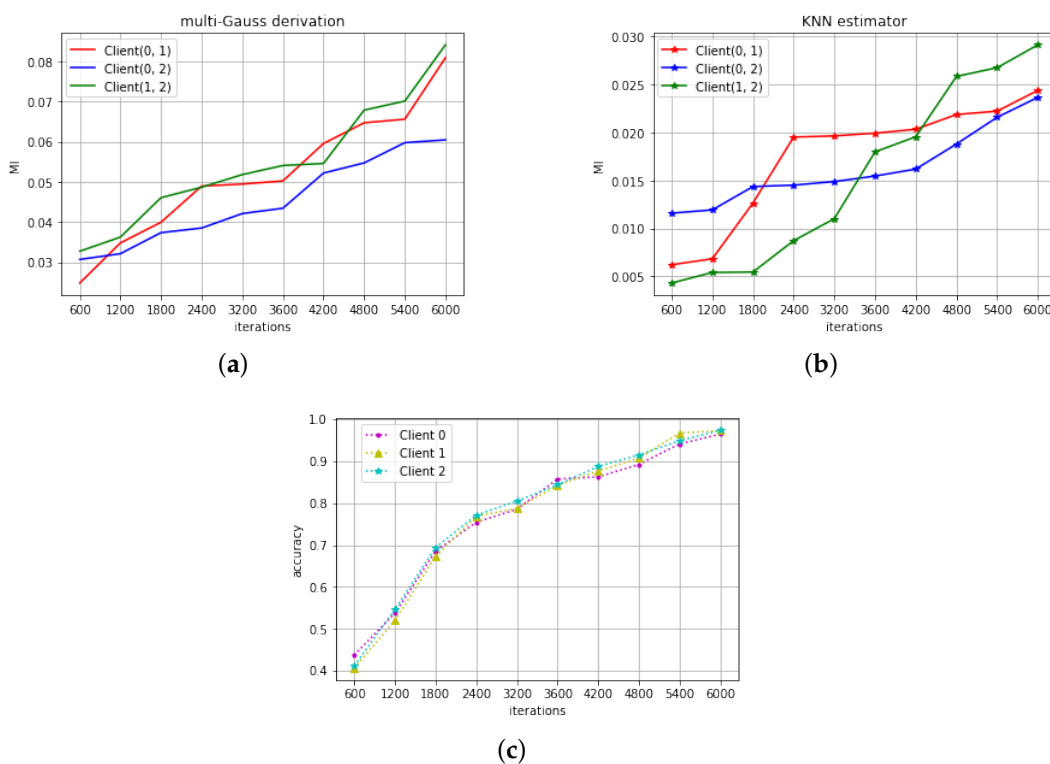
**Figure 1.** (a) The MI in the signal variation prediction task calculated by the first method (Section 3.1). (b) The MI in the signal variation prediction task calculated by the second method (Section 3.2). (c) The MSE variation with the training iterations.

#### 4.2. CNN on Digit Recognition

In the second experiment, we firstly train the Lenet model [39] from scratch on the whole MNIST dataset. After getting the pretrained Lenet model, at each client, we keep the trained Lenet model unchanged except the last fully connected layer and apply it on augmented MNIST data. This way, we reduce the number of training parameters to 850. Each client contains 200 training examples (20 for

each digit) and each training example is an augmentation of a randomized selection of an original digit image in MNIST. The augmented operation includes rotation, zoom, shift (all of these operations are performed in a certain range). The model is optimized by Adam optimizer with learning rate  $1e-5$ . The mini-batch size is 50. The total number of samples obtained at each client is 26,400. The metric we use is recognition accuracy.

Figure 2a,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. Figure 2c 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 imply that although different clients in a same learning task have similar training trajectory, the MI can still the subtle difference between them.



**Figure 2.** (a) The MI in the digit recognition task calculated by the first method (Section 3.1). (b) The MI in the digit recognition task calculated by the second method (Section 3.2). (c) The accuracy variation with the training iterations.

### 4.3. 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.,:

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

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

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

The results of distance variation for the two learning tasks can be seen in Figures 3 and 4. As we can see, for both learning tasks, all three distance metrics between any client pairs increase rather than decrease with the training iteration. This may not be surprising as we initialize the parameters to be the same across all clients. However, 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 Figures 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.

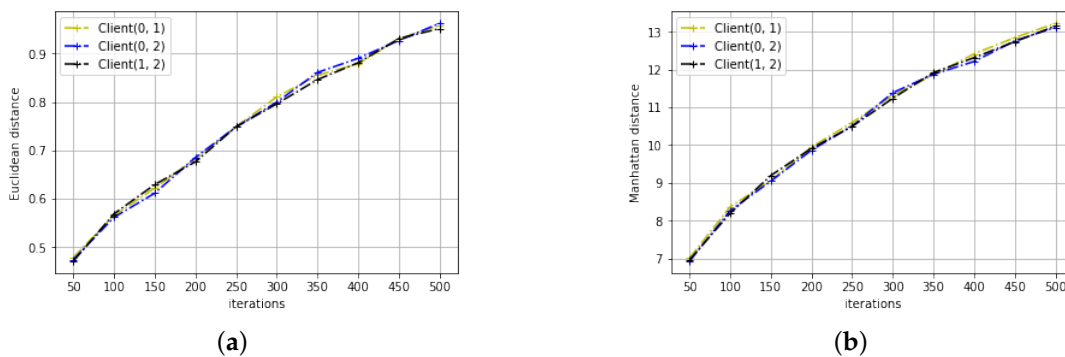
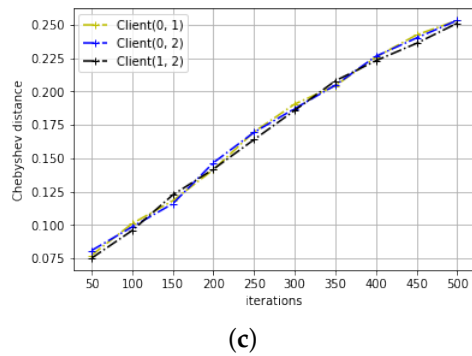
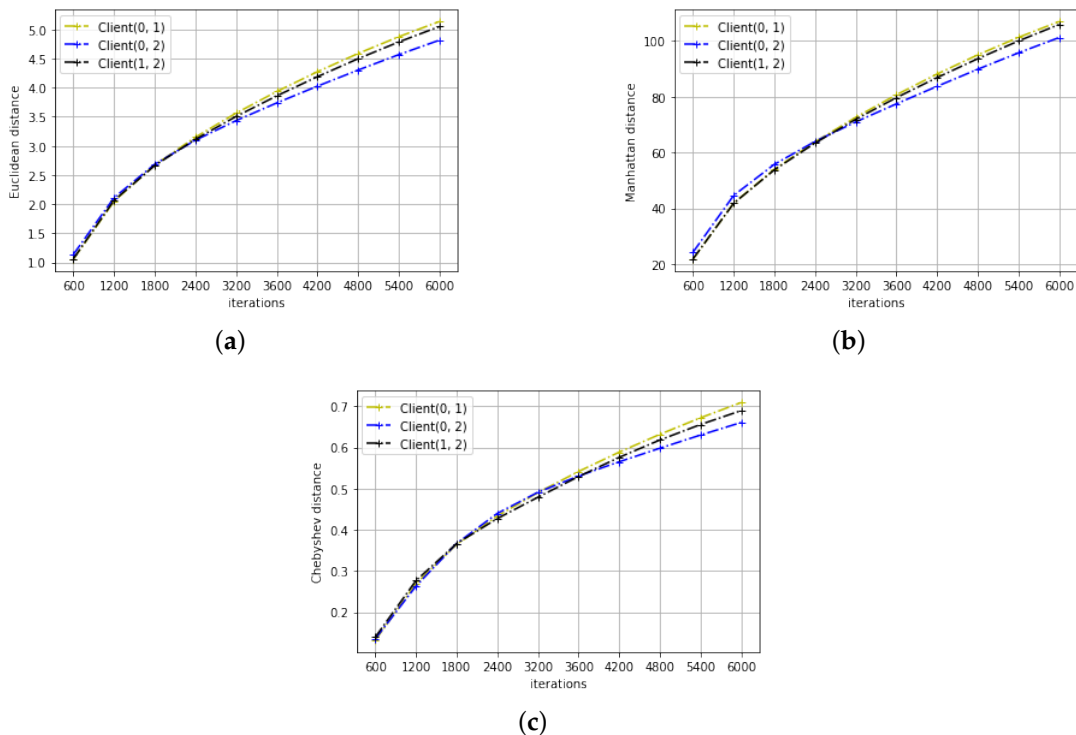


Figure 3. Cont.



**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.

### 5. Conclusions

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.

We encourage more studies to explore the federated learning through information-theoretic perspective in large learning scenarios and non-iid setting. 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.

**Author Contributions:** Data curation, P.X.; Formal analysis, P.X.; Supervision, S.C.; Writing—original draft, P.X.; Writing—review and editing, V.S. and D.V. All authors have read and agreed to the published version of the manuscript.

**Funding:** The work was supported in part by the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 734331.

**Acknowledgments:** During the process of this work, the former National Basketball Association player Kobe Bryant died in a helicopter accident. Kobe has been with his tenacity and indomitable fighting spirit to inspire a generation, and is also an important spiritual idol in the growing way of P.X. Here P.X. want thank to Kobe for his inspiration. May he rest in the peace.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Poushter, J. Smartphone ownership and internet usage continues to climb in emerging economies. *Pew Res. 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. *Healthc. Inform. Res.* **2017**, *23*, 4–15.
3. Taylor, R.; Baron, D.; Schmidt, D. The world in 2025—predictions for the next ten years. In Proceedings of the 2015 10th International Microsystems, Packaging, Assembly and Circuits Technology Conference (IMPACT), Taipei, Taiwan, 21–23 October 2015; pp. 192–195.
4. Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 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 Comput. Commun. Rev.* **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, USA, 2012; pp. 1–62.
7. McMahan, H.B.; Moore, E.; Ramage, D.; Hampson, S. Communication-efficient learning of deep networks from decentralized data. *arXiv* **2016**, arXiv:1602.05629.
8. Dean, J.; Corrado, G.; Monga, R.; Chen, K.; Devin, M.; Mao, M.; Ranzato, M.; Senior, A.; Tucker, P.; Yang, K. Large scale distributed deep networks. In Proceedings of the Advances in neural information processing systems, Lake Tahoe, NV, USA, 3–6 December 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* **2018**, arXiv:1811.03604.
10. Ramaswamy, S.; Mathews, R.; Rao, K.; Beaufays, F. Federated learning for emoji prediction in a mobile keyboard. *arXiv* **2019**, arXiv:1906.04329.
11. Huang, L.; Yin, Y.; Fu, Z.; Zhang, S.; Deng, H.; Liu, D. LoAdaBoost: Loss-Based AdaBoost Federated Machine Learning on medical Data. *arXiv* **2018**, arXiv:1811.12629.
12. Samarakoon, S.; Bennis, M.; Saad, W.; Debbah, M. Federated learning for ultra-reliable low-latency V2V communications. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, United Arab Emirates, 9–13 December 2018; pp. 1–7.
13. Bottou, L. Large-scale machine learning with stochastic gradient descent. In Proceedings of the 19th International Conference on Computational Statistics, Paris, France, 22–27 August 2010; pp. 177–186.
14. Sattler, F.; Wiedemann, S.; Müller, K.R.; Samek, W. Robust and communication-efficient federated learning from non-iid data. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, doi:10.1109/TNNLS.2019.2944481.

15. McMahan, H.B.; Ramage, D.; Talwar, K.; Zhang, L. Learning differentially private recurrent language models. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
16. Agarwal, N.; Suresh, A.T.; Yu, F.X.X.; Kumar, S.; McMahan, B. cpSGD: Communication-efficient and differentially-private distributed SGD. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 7564–7575.
17. He, L.; Bian, A.; Jaggi, M. Cola: Decentralized linear learning. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 4536–4546.
18. Woodworth, B.E.; Wang, J.; Smith, A.; McMahan, B.; Srebro, N. Graph oracle models, lower bounds, and gaps for parallel stochastic optimization. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 8496–8506.
19. Eichner, H.; Koren, T.; McMahan, B.; Srebro, N.; Talwar, K. Semi-Cyclic Stochastic Gradient Descent. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 10–15 June 2019; pp. 1764–1773.
20. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*; John Wiley & Sons: New York, NY, USA, 2012.
21. Kraskov, A.; Stögbauer, H.; Grassberger, P. Estimating mutual information. *Phys. Rev. E* **2004**, *69*, 066138.
22. Ross, B.C. Mutual information between discrete and continuous data sets. *PLoS ONE* **2014**, *9*, e87357.
23. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
24. Mikolov, T.; Karafiát, M.; Burget, L.; Černocký, J.; Khudanpur, S. Recurrent neural network based language model. In Proceedings of the Eleventh Annual Conference of the International Speech Communication Association, Chiba, Japan, 26–30 September 2010.
25. Danielsson, P.E. Euclidean distance mapping. *Comput. Gr. Image Process.* **1980**, *14*, 227–248.
26. Krause, E.F. *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*; Courier Corporation: Chelmsford, MA, USA, 1986.
27. Cantrell, C.D. *Modern Mathematical Methods for Physicists and Engineers*; Cambridge University Press: Cambridge, UK, 2000.
28. Tishby, N.; Zaslavsky, N. Deep learning and the information bottleneck principle. In Proceedings of the 2015 IEEE Information Theory Workshop (ITW), Jerusalem, Israel, 26 April–1 May 2015; pp. 1–5.
29. Shwartz-Ziv, R.; Tishby, N. Opening the black box of deep neural networks via information. *arXiv* **2017**, arXiv:1703.00810.
30. Saxe, A.M.; Bansal, Y.; Dapello, J.; Advani, M.; Kolchinsky, A.; Tracey, B.D.; Cox, D.D. On the information bottleneck theory of deep learning. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
31. Noshad, M.; Zeng, Y.; Hero, A.O. Scalable mutual information estimation using dependence graphs. In Proceedings of the ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 2962–2966.
32. Kolchinsky, A.; Tracey, B.D.; Wolpert, D.H. Nonlinear information bottleneck. *Entropy* **2019**, *21*, 1181.
33. Wickstrøm, K.; Løkse, S.; Kampffmeyer, M.; Yu, S.; Principe, J.; Jenssen, R. Information Plane Analysis of Deep Neural Networks via Matrix-Based Rényi’s Entropy and Tensor Kernels. *arXiv* **2019**, arXiv:1909.11396.
34. Adilova, L.; Rosenzweig, J.; Kamp, M. Information-Theoretic Perspective of Federated Learning. *arXiv* **2019**, arXiv:1911.07652.
35. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423.
36. Deng, L. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Process. Mag.* **2012**, *29*, 141–142.
37. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.

38. Gal, Y.; Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of The International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1050–1059.
39. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).