Communication-Efficient Learning of Deep Networks from Decentralized Data [1]

The introduction of Federated Learning

Jose A. Lorencio Abril

Machine Learning, M2 BDMA

Université Paris-Saclay, CentraleSupélec

Fall 2023

Table of Contents

Table of Contents

Introduction

Google was facing the situation:

- Lot of data, distributed across many devices.
- Privacy-sensitive data.
- A ML model to be trained using these data.

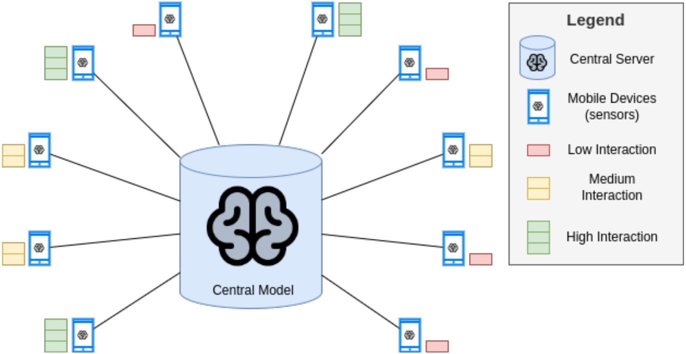
The **problem** then was:

> How to train the model on the distributed data, without centralizing it, preserving its privacy?

Introduction

Moreover, contrarily to usual ML settings, the data:

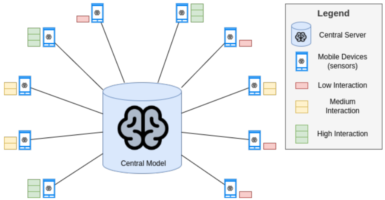
- Is Not-IID: each user can show different behavior.
- Is unbalanced: each user can have different usage.



Introduction

Moreover, contrarily to usual ML settings, the data:

- Is Not-IID: each user can show different behavior.
- Is unbalanced: each user can have different usage.



- Is Massively-Distributed: more devices than data points per device.
- The communication is limited (since the aim is to work with mobile devices).

Moreover, contrarily to usual ML settings, the data:

- Is Not-IID: each user can show different behavior.
- Is unbalanced: each user can have different usage.
- Is Massively-Distributed: more devices than data points per device.
- The communication is limited (since the aim is to work with mobile devices).

So, the problem becomes:

> How to train the model on massively-distributed, non-IID, unbalanced data, with limited communication, without centralizing it and preserving its privacy?

Table of Contents

Related work

Although there is **no previous work exactly tackling this problem**, there are precedents on decentralized training:

- MacDonald et al. [2]: trained a perceptron by averaging several perceptrons, trained in different subsets of data.

Related work

Although there is **no previous work exactly tackling this problem**, there are precedents on decentralized training:

- MacDonald et al. [2]: trained a perceptron by averaging several perceptrons, trained in different subsets of data.
- Povey et al. [3]: applied it to NLP tasks.

Related work

Although there is **no previous work exactly tackling this problem**, there are precedents on decentralized training:

- MacDonald et al. [2]: trained a perceptron by averaging several perceptrons, trained in different subsets of data.
- Povey et al. [3]: applied it to NLP tasks.
- Zhang et al. [4]: used a similar approach to train DNNs.

Related work

Although there is **no previous work exactly tackling this problem**, there are precedents on decentralized training:

- MacDonald et al. [2]: trained a perceptron by averaging several perceptrons, trained in different subsets of data.
- Povey et al. [3]: applied it to NLP tasks.
- Zhang et al. [4]: used a similar approach to train DNNs.

All this approaches considered IID and balanced data distribution across devices.

Related Work

- Neverova et al. [5]: discussed the advantages of keeping user data inside their devices.
- Balcan et al. [6] and Zhang et al. [7] tackled distributed learning, assuming convexity, few devices and IID data.
- Dean et al. [8] proposed a way to do distributed SGD, but this approach is very computationally expensive.

Table of Contents

FedAvg

Let C be the fraction of devices participating in each round, out of the total K devices, E the number of local SGD epochs, and B the local minibatch size.

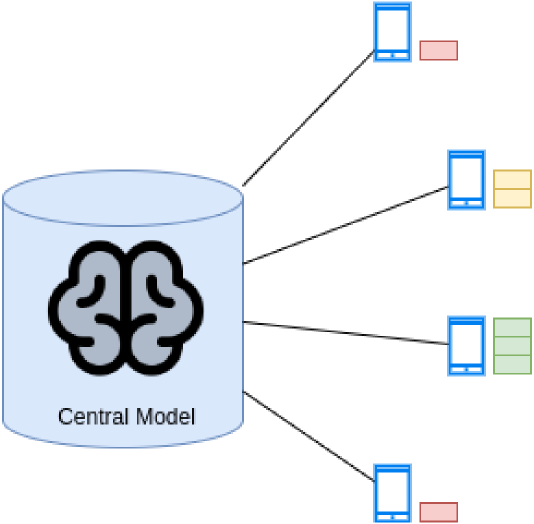
Server executes:

1. Initialize w_0
2. For each round t:
   - i. $m \leftarrow \max\{C \cdot K, 1\}$
   - ii. $S_t \leftarrow$ (m random clients)
   - iii. For each client $k \in S_t$:
     - a. $w_{t+1}^k \leftarrow \textbf{ClientUpdate}(k, w_t)$
   - iv. $m_t \leftarrow \sum_{k \in S_t} n_k$
   - v. $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_t}{m_t} w_{t+1}^k$

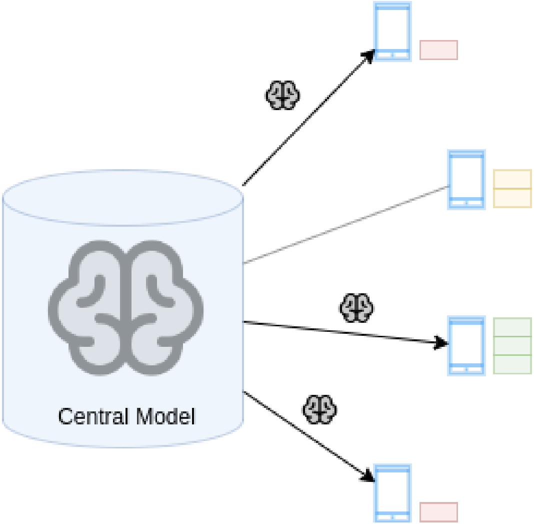
ClientUpdate(k, w):

1. $\mathcal{B} \leftarrow$ (split D_k into B-batches)
2. For $i = 1, ..., E$:
   - i. For batch $b \in B$:
     - a. $w \leftarrow w - \mu \nabla f_k(w, b)$
3. Return w

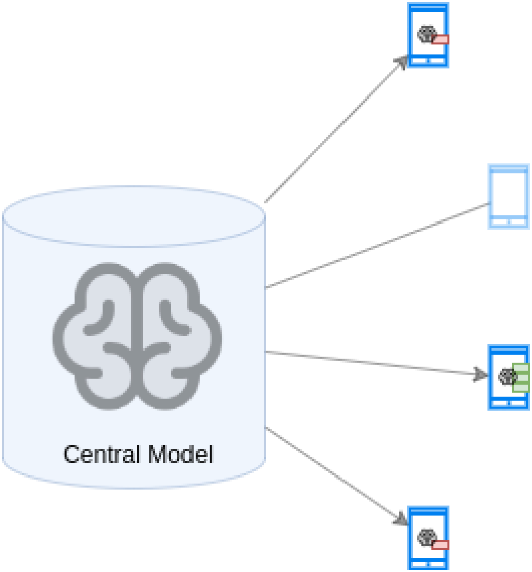
Visualization



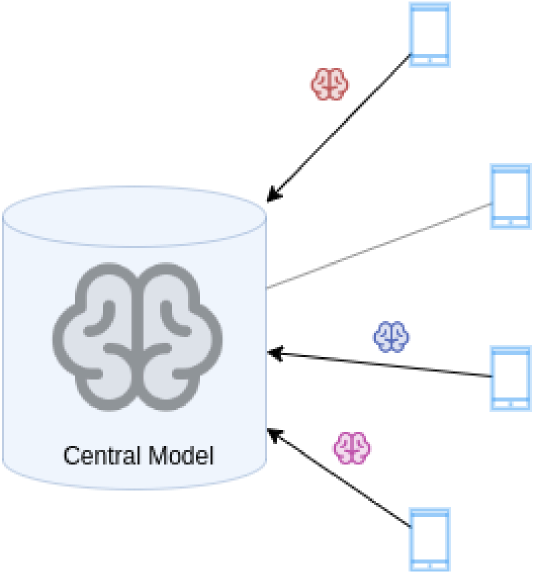
Visualization



Visualization



Visualization



Visualization

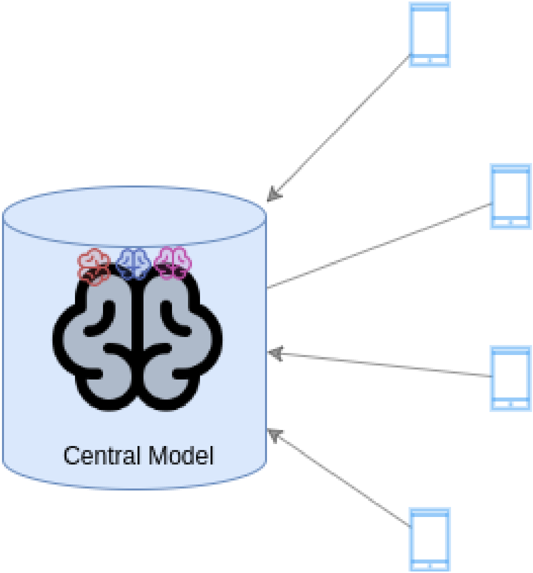


Table of Contents

Experiment #1: Increasing parallelism

They test the effect of varying C in the amount of needed rounds to achieve a certain accuracy.

2NN	IID		Non-IID	
C	$B = \infty$	$B = 10$	$B = \infty$	$B = 10$
0.0	1455	316	4278	3275
0.1	1474 (1.0x)	87 (3.6x)	1796 (2.4x)	664 (4.9x)
0.2	1658 (0.9x)	77 (4.1x)	1528 (2.8x)	619 (5.3x)
0.5	—	75 (4.2x)	—	443 (7.4x)
1.0	—	70 (4.5x)	—	380 (8.6x)

Conclusions:

- Increasing parallelism $\Longrightarrow$ Faster convergence.
- More noticeable in the non-IID case.

Experiment #2: Increasing computation per client

For this experiment, they fix $C = 0.1$, and increased the computation per client, by increasing E and decreasing B. Then, they measure the number of rounds needed to reach a certain accuracy.

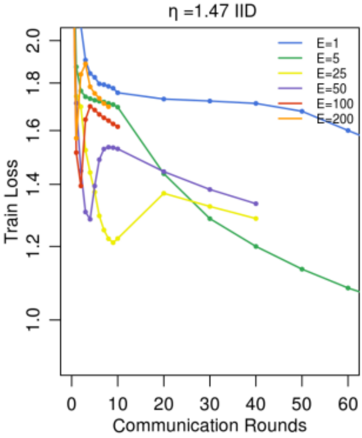
SHAKESPEARE LSTM, 54% ACCURACY			
E	**B**	**IID**	**Non-IID**
1	∞	2488	3906
1	50	1635 (1.5x)	549 (7.1x)
5	∞	613 (4.1x)	597 (6.5x)
1	10	460 (5.4x)	164 (23.8x)
5	50	401 (6.2x)	152 (25.7x)
5	10	192 (13.0x)	41 (95.3x)

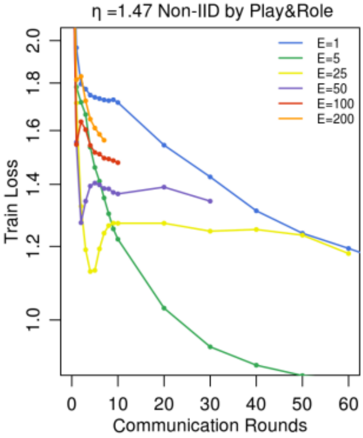
Conclusions:

- Increasing computation per client $\xRightarrow{\textit{Generally}}$ Faster convergence.
- More pronounced when data is non-IID and unbalanced.

Experiment #3: Can we overoptimize on the client dataset?

The fixed $C = 0.1$ and $B = 10$, and varied E, measuring the overall accuracy of the model.





Experiment #3: Can we overoptimize on the client dataset?

Conclusions:

- Increasing E leads to better performance, but only up to a certain point.
- After this point, the performance can get stuck or even decrease.
- Overfitting is therefore possible.

Table of Contents

Conclusion

In this paper, the authors defined the subfield of Federated Learning, showing that:

- It is possible to train models in a decentralized way.
- Without centralizing the data.
- With non-IID and unbalanced distributed data.
- In an efficient manner.

Table of Contents

The state of Federated Learning Today

- Adaptive FL [9]: adapted several optimization methods to the FL setting, like Adam, Adagrad,...
- FL with Differential Privacy [10]: enhanced the security of the approach, by adding differential privacy in the training process.
- Sparse Ternary Compression [11]: enhances the approach with a new compression approach that highly reduces communication costs.
- In general, FL shows primising results in the fields of IoT and Edge Computing.

Table of Contents

Bibliography I

[1] Brendan McMahan et al. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial Intelligence and Statistics*. PMLR. 2017, pp. 1273–1282.

[2] Ryan McDonald, Keith Hall, and Gideon Mann. "Distributed training strategies for the structured perceptron". In: *NAACL HLT*. 2010.

[3] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. "Parallel training of deep neural networks with natural gradient and parameter averaging". In: *ICLR Workshop Track*. 2015.

[4] Sixin Zhang, Anna E Choromanska, and Yann LeCun. "Deep learning with elastic averaging sgd". In: *NIPS*. 2015.

[5] Natalia Neverova et al. "Learning human identity from motion patterns". In: *IEEE Access* 4 (2016), pp. 1810–1820.

[6] Maria-Florina Balcan et al. "Distributed learning, communication complexity and privacy". In: *arXiv preprint arXiv:1204.3514* (2012).

[7] Yuchen Zhang et al. "Information-theoretic lower bounds for distributed statistical estimation with communication constraints". In: *Advances in Neural Information Processing Systems*. 2013.

[8] Jeffrey Dean et al. "Large scale distributed deep networks". In: *NIPS*. 2012.

[9] Sashank Reddi et al. "Adaptive Federated Optimization". In: *arXiv preprint arXiv:2003.00295* (2020).

[10] Kang Wei et al. "Federated Learning With Differential Privacy: Algorithms and Performance Analysis". In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 3454–3469.

Bibliography II

[11] Felix Sattler et al. "Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data". In: *IEEE Transactions on Neural Networks and Learning Systems* 31 (2020), pp. 3400–3413.