The principle, communication efficiency and privacy issues of federated learning

Hakan Uzuner
University of Passau
Passau, Germany
uzuner01@gw.uni-passau.de

***Abstract*—Standard machine learning approaches require a huge amount of training data to be stored centralized in order to feed the learning algorithms. Keeping and using data centralized brings many negative aspects with it. Those aspects can be inefficient communication between the centralized data center and the clients producing the data, privacy issues and quick usability of the profits and results of the training. Google's new approach, federated learning, on the other hand tackles all these problems. The training data is kept decentralized at the client's devices while communicating only with small updates of the common model. This method allows for optimizations of communication, keeping the privacy of users involved in the process and providing quick usability of the model's process. In this paper I will explain how the federated learning principle works. Further on, I will give a small insight on optimization possibilities of communication efficiency as well as on privacy issues involved in machine learning processes and how those can be solved using federated learning principles. Additionally, I will show the connection between the federated learning concept and organic computing.**

Keywords—federated learning, machine learning, organic computing, optimization, communication, efficiency, privacy

I. INTRODUCTION

This paper is based on already published research in the fields of federated learning, distributed optimization and machine learning in general. It is intended to be an overview of the principle, capabilities and problems of federated learning. Therefore, I am not claiming any of this work for myself. A brief summary of some of the current available literature in this field will be given in Chapter II.

Since the beginning of machine learning, developers must deal with one major problem: how to manage the huge amount of training data? Traditional approaches gather data and keep it in a central database. On this centralized data one can run a learning algorithm using a single, powerful computer. In contrary, federated learning decentralizes the training data by leaving it distributed on the devices it is gathered from. Instead of running an algorithm on one single huge database of training data, federated learning uses a system consisting of a model and updates of that model. The common model is delivered to each client, every single one of which calculates its own update. Those updates are collected by the server or cloud, where a new common model is calculated and propagated to the clients. Federated learning was first introduced by research scientists working for google, but in the meantime much more literature is available on this topic. In Chapter II. I will give an overview of federated learning related publications. In Chapter III. the model of federated learning will be shown. After the basic operating principle, some problems regarding communication optimizations and update methods must be addressed as they are crucial for the whole process to function efficiently enough to be profitable. Afterwards, I will give a short insight on more complex communication efficiency problems and privacy issues in Chapter IV. of this paper. Finally, I will show the connection of federated learning with organic computation.

II. LITERATURE REVIEW

Federated learning is a field in which various research was already published. But not only papers on federated learning, but also optimization in machine learning processes in general are applicable to this field. Here I will shortly review literature used by me in the following Chapters.

A. Model

The basic principle of federated learning is well explained by McMahan et al. in [1][2]. They explain what federated learning is and what concepts and methods are used, like Stochastic Gradient Descent (SGD). The finite-sum problem, which is the general form of the optimization problem, Gradient Descend and Stochastic Gradient Descent can be very well studied in [1][3]. The modification of the SGD, the FederatedSGD, and the Federated Average method are defined and described in [1].

B. Communication Efficiency

Communication Efficiency, in detail structured and sketched updates, are well explained in [6]. The basic principles used here are mostly from other sources, such as the effect of random rotation on quantization in [7].

C. Privacy Issues

Privacy issues are mostly addressed in a machine learning context in general. There are many methods and concepts available. Research as in [8][9][10] shows that there are many aspects and possibilities to keep training data and communication private. Some security aspects are also to be addressed, as in [11].

D. Organic Computing

A quick overview of Organic Computing is given in [13]. A much more detailed and deep view can be obtained by reading [12].

III. MODEL

A. Principle of federated learning

Federated learning brings advantages in three major aspects: the **value** and **privacy of the collected data** and the **communication and calculation efficiency of the training**.

The **value** of the gathered data depends on the use case. As federated learning is designed for many clients such as mobile phones, use cases like image classification or language models are common. McMahan et al. [1] state that real-world data is to be preferred over proxy data of a data center as it gives a more realistic picture of the real environment. For example, instead of using standard and uniform language corpora like Wikipedia or images from Flickr, it is much better to train with actual words typed by users in their everyday life.

After explaining the general principle of how federated learning works, we will focus on the **communication and calculation efficiency of the training method** and the **privacy of the collected data**.

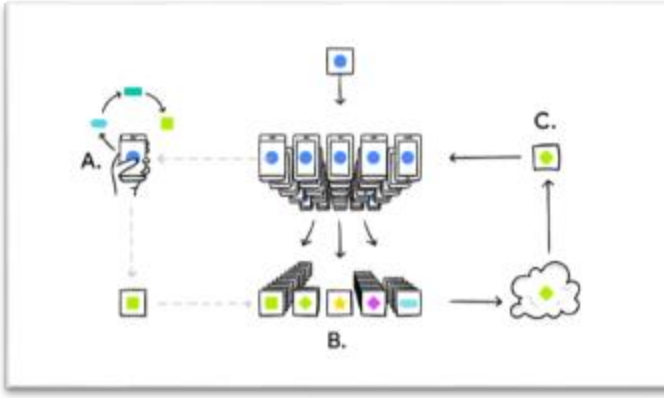

Fig. 1. Updating the model in federated learning. (Source: [2])

Using Fig. 1, the basic working principle of federated learning is explained as follows: Clients such as mobile phones download the common model and calculate their own specific update based on their local training data (A). Those updates are then sent to the cloud or server, where they immediately get averaged (B). Afterwards, the averaged update is applied to the common model, generating a new, updated common model (C). This model is again sent to the clients, which closes the circle and completes one round. After that a new round can start, repeating the steps. This procedure inherently preserves privacy of the training data because it keeps the data local at the clients while contributing only small updates to the learning system. Because of the immediate averaging on the server, even the single updates of the clients are not stored locally on the cloud. Even though this is a good starting point, the privacy aspect still is not quite done and has to be considered in some other points we will come to later in this paper. Also, both communication and calculation efficiency profit vastly from this principle. Many traditional approaches need as much processing power as possible to run on one huge pile of data. Whereas in federated learning the training data is already processed by the collecting nodes, for example mobile phones. Modern devices possess powerful enough hardware to cope with the calculations needed for most of the learning algorithms. There are even many situations where those devices stand idle with practically unlimited energy and Wi-Fi sources for a long time, e. g. while charging at night. Those cases provide an excellent time for resolving the problem of calculation expenditure by delegating it to the many clients instead of running it all at once on one single computer. There is also a communication efficiency aspect inherent in federated learning due to the principle of communicating updates instead of training data. However, this alone is not a certainty for efficient communication. Many more measures can be considered to improve and adapt the updates to ensure efficient communication.

B. Federated Optimization

Federated optimization describes four key properties to differentiate it from other distributed optimization problems. The key properties are as follows: **Non-IID, Unbalanced, Massively distributed** and **Limited communication**. In the following, I will explain those keys briefly.

Non-IID In distributed optimization algorithms, an IID assumption is made. This means that the centralized training data is evenly distributed to the clients. This way all clients have roughly the same data. A more realistic approach is the Non-IID assumption. The gathered data on a client is usually highly dependent on the underlying user. Users mostly act in their unique and very own way, so we cannot just assume that their data is equal. This means that the data of one client, with a very high probability, cannot be representative of the overall distribution [1][3].

Unbalanced The method of gathering training data in federated learning usually consists of collecting it from many clients which act independently. It is obvious that those clients work at unregulated and individual intervals. Therefore, they will provide different amounts of data with utmost certainty [1][3].

Massively distributed It has to be assumed that the number of participating clients is much bigger than the average amount of training data they contribute [1][3].

Limited communication Similarly to having an unbalanced amount of local training data because of uncontrolled environments, different clients also tend to be offline or on slow connections [1]. This aspect gets particularly interesting when it comes to upload updates efficiently on slow connections.

Those key properties are very important for the federating learning setting. A lot of optimization and efficiency can be done by handling those. However, this shall not be the further topic on this paper. Instead, in the following I will describe the mathematical background to minimize the amount of data used in updates delivered from the clients to the server.

C. Finite-sum problem

To solve those update optimization problems, we have to view them in a mathematical context. Most of the optimization problems can be seen as a finite-sum problem as described in [1][3] in the form of

$$\min_{w\in\mathbb{R}^d} f(w) \quad \text{where} \quad f(w) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(w). \qquad (1)$$

This is a general formula, which can solve many problems depending on $f_i(w)$. Suppose we have a set of input pairs $\{x_i, y_i\}_{i=1}^{n}$, we can use the general formula e.g. for *linear regression [3]:*

$$f_i(w) = \frac{1}{2}(x_i^T w - y_i)^2, y_i \in \mathbb{R}$$

D. (Stochastic) Gradient Descend and FederatedSGD

To find the minimum of a problem like (1), it is common to use the optimization algorithm Stochastic Gradient Descent (SGD) or variations of it [1][3]. In this section I will explain what the Gradient Descent (GD) and the SGD are, referring to the descriptions of Konečný et al. [3]. Then we will learn about the adaption of SGD by McMahan et al. [1] defining the FederatedSGD (FedSGD).

The **Gradient Descent** generally is used to find the minimum of a function. The algorithm is an iterative process which takes the direction of the negative Gradient to approach a local minimum. The iteration is as follows:

$$w^{t+1} = w^t - h_t \nabla f(w^t),$$

where w is a point in the graph, $h_t > 0$ is a stepsize parameter and t is the current iteration [3]. The Gradient, $\nabla f(w^t)$, is subtracted to reach the minimum or added to reach the maximum. As the finite-sum problem tries to find the minimum, we are subtracting here. The GD can be used for smooth functions, whereas the Subgradient Descend can be used for non-smooth functions [3]; however, this will not be a subject here. As already mentioned, in federated learning we have to deal with a vast amount of data, which would make the iteration process slow and inefficient. Although there are some ideas from Polyak and Nesterov to accelerate this process, a big amount of data still cannot be handled efficiently [3].

The **Stochastic Gradient Descent** is not that different from the GD:

$$w^{t+1} = w^t - h_t \nabla f_{i_t}(w^t),$$

where w is a point in the graph, $h_t > 0$ is a stepsize parameter and t is the current iteration [3]. In each iteration there will be a function chosen at random with $i_t \in \{1, 2, \dots, n\}$ in iteration t. This can be done because the update direction is an unbiased estimate of the gradient [3]. This method mostly benefits from great efficiency when it comes to a huge amount of data to process. There is a method called Random Reshuffling (RR), which has an even better performance than SGD, proven by Gürbüzbalaban et al. [4]. The SGD is designed to use only one dataset, whereas RR is basically an improved form of GD, which uses all data. In federated learning there are many clients and the main idea is to always use a specific fraction of those client's data. Therefore, as we want to stay flexible on the amount of data used in each round, SGD and RR are probably both not quite suitable for federated learning.

The **FederatedSGD** was introduced by McMahan et al. [1] and solves the problem of choosing the right number of clients. They use a large-batch synchronous SGD as it is proven by Chan et al. [5] to outperform asynchronous approaches in data centers. Further, they select a C-fraction of clients in each round to calculate the gradient over those clients. C = 1 means to use all clients, which is referred to as full-batch (non-stochastic) gradient descent [1].

E. Federated Averaging

The updates g_k have to be calculated by the clients using FedSGD, typically using C = 1 and the current model w_t [1]:

$$g_k = \nabla F_k(w_t).$$

After they were communicated to the server, some kind of averaging has to be done in order to apply the update to the current model. Defined by McMahan et al. [1] we will learn about the the definition of FederatedAveraging (FedAvg) in this section. First of all, McMahan et al. rewrite the objective of (1) as follows:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in P_k} f_i(w),$$

while P_k is a set of indexes of data points on client k and $n_k = |P_k|$.

Once the server has the updates, it can calculate the update with

$$w_{t+1} \leftarrow w_t - \gamma \sum_{k=1}^{K} \frac{n_k}{n} g_k$$

since

$$\sum_{k=1}^{K} \frac{n_k}{n} g_k = \nabla f(w_t)$$

, while γ is the fixed learning rate. This update is also given by the following statement:

$$\forall k, w_{t+1}^k \leftarrow w_t - \gamma g_k , \qquad \text{then}$$

$$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k .$$

This far, each client performs one single gradient descent step and the server takes the weighted average of the resulting single updates. The local model of the clients can be iterated several times as follows:

$$w^k \leftarrow w^k - \gamma \nabla F_k(w^k).$$

This all together is called FederatedAveraging [1]. Furthermore, there are three parameters to control the computational expenditure: C, the number of clients active in one round; E, the number of passes each client makes on its local data; B, the local minibatch size. The values E = 1 and B = ∞ correspond to the basic FedSGD.

As a small summary, the FedAvg method is hereby completed. We now know how updates of the current model are generated locally at the client and how those clients are chosen as a subset of all clients. We also have learned how the server aggregates those updates and calculates an update for the common model. The computational effort can be adjusted with various parameters. For any further information on this I refer to the work of [1][3] and their references.

IV. COMMUNICATION EFFICIENCY

This Chapter is dedicated to give the reader a small insight into the complex concepts of communication efficiency. Even though federated learning inherently handles this aspect pretty well, there are still some methods that can be applied to improve the efficiency. I refer to the respective literature for further information, as it is intended to only mention the basic principles here.

The main aspect in communication efficiency is how the updates are handled. As we have many clients, preferably sending many updates, it is obvious that the communication traffic is a bottleneck of federated learning systems. As already mentioned above, in federated learning it is common to work with mobile devices, which almost certainly have not the best upstream capacities. This is usually handled by assuming that uploading updates is performed while the devices are plugged in and unused. This way the computational power as well as the availability of the internet connection is largely guaranteed. But this is not the primary source of traffic problems. At the time when the updates are calculated there are many possibilities to minimize the size of the data packages which need to be sent to the server. Described by Konečný et al. [6] there are structured and sketched updates, which will be the topic of this section. But first of all, they describe the general problem as follows:

$$W_{t+1} = W_t + \gamma_t H_t , \quad H_t := \frac{1}{n_t} \sum_{i \in S_t} H_t^i,$$

while $t \geq 0$ is the current round of updates and W_t is the current model delivered to the subset S_t of n_t clients. The learning rate is γ_t and H_t^i is the local update of client I at round t. As W_{t+1} and H_t are calculated at server side, we only have to worry about H_t^i in terms of communication cost. Minimizing the amount of data will be the goal of the following update methods.

A. Structured Updates

Structured Updates lower the data traffic by keeping the update in a pre-specified structure [6]. There are two types of Structured Updates described by Konečný et al.: Low Rank and Random Mask. In Low Rank updates the local update is split up into two matrices:

$$H_t^i = A_t^i B_t^i \quad \text{where} \quad A_t^i \in \mathbb{R}^{d_1 X k}, B_t^i \in \mathbb{R}^{k X d_2}$$

and k is a fixed number. The rank of the update matrix H_t^i should be at most k. Generating A_t^i as a random seed constant and only optimizing, training and sending B_t^i to the server saves a factor of d_1/k in communication. There were also attempts to fixing B_t^i and training A_t^i as well as training both, neither of which proved to be as effective as the first alternative. The Random Mask approach benefits from using a sparsity pattern to send only a sparse matrix update. As the sparsity pattern is a random seed, only the seed together with all non-zero entries of the update have to be sent to the server.

B. Sketched Updates

Sketched Updates in general use the full update and compresses it before sending it to the server. There are several compressing methods, two of them described by Konečný et al., which I will sum up here: subsampling and probabilistic quantization. Subsampling consists of clients sending only a subset of the values of the local update to the server. The server can average the updates again, creating an unbiased estimator of the true average [6]. Probabilistic quantization compresses the update by reducing each scalar to one bit in the following form:

$$\tilde{h}_j \begin{cases} h_{max}, with\ probability\ \dfrac{h_j - h_{min}}{h_{max} - h_{min}} \\ h_{min}, with\ probability\ \dfrac{h_{max} - h_j}{h_{max} - h_{min}} \end{cases}$$

while $\tilde{h}$ is the compressed update of $h = (h_1, \dots, h_{d_1 x d_2}) = vec(H_t^i)$

and

$$h_{max} = max_j(h_j), h_{min} = \min_j(h_j).$$

This method brings an advantage of 32 times the compression in comparison to a 4-byte float. Errors can be prevented by improving it with structured random rotations [6].

This section was only to sketch an idea of the complexity of determining the right method for calculating updates. In practice, there is need of a complex combination of the mentioned and other methods to achieve an efficient communication. There is much more research and literature available on this topic than to be covered in the boundaries of this paper, such as Suresh et al. [7], who study efficient algorithms for distributed mean estimation.

V. Privacy Issues

This section is meant to give the reader a perspective on the privacy issues of federated learning. It is again only to sketch a general idea, as the field is very complex and has a depth that cannot be covered in this paper.

Federated learning always means to gather data from many different clients. In language prediction context for example there is a huge amount of data collected from users, i. e. words they typed in private conversations. These conversations can contain sensitive data that the users probably do not want to share with everyone. The concept of communicating with updates in federated learning inherently has a method of protecting privacy by omitting data through sending as little information with updates as possible. It is still pretty easy to derive one user's data with access to raw, unsecured updates. Therefore, it is necessary to take precautions, some of which I will introduce in the following.

Bonawitz et. al. [8] describe a protocol for secure aggregation of update data in machine learning. They do not use individual user's updates but the elementwise weighted averages of the update vectors. In combination with a secure aggregation protocol for averaging these updates, it is ensured that the server only can learn about what words were used, but not by which users.

The protocol described by Xie et al. [9] uses Crypto-Nets to limit the communication and calculation to encrypted data only. This means that the client sends an encrypted ciphertext and the server processes this without deciphering. Afterwards, the server sends the result back, again encrypted and only readable for the user. They achieve this by using homomorphic encryption.

Pathak et al. [10] show that using Hidden Markov Models, homomorphic cryptosystems and a protocol for secure forwarding works for probabilistic inference. They apply this in a use case of speech recognition.

All in all, advances are made in terms of research and publications when it comes to privacy. All of the mentioned and many more methods of privacy protection can be considered to be applied to federated learning, directly or in modified form. Therefore, there are and will be many possibilities to counteract the privacy concerns in federated learning. Since there is almost always a privacy sensitive area involved in working with training data gathered from end users, those methods of privacy preservation must be applied to guarantee a secure environment of the machine learning process. In terms of security there are also some aspects to be considered. As described by Bagdasaryan et al. [11], there is a way to manipulate the learning process. They explain how each user in federated learning can bring in some backdoor functionality, e. g. to have an image classifier assign an attacker-chosen label to images. This shows that the learning process is not only vulnerable to attacks from outside but also from inside the system.

VI. Relation to Organic Computing

Advances in technical systems have led to an increasing complexity. Organic Computing tries to tackle this topic by implementing concepts found in nature. As natural and social systems also need to deal with complex situations by using benefits of organisation and adaption, these observations can very well be studied and applied or imitated. Organic Computing aims to improve technical systems in classical capabilities such as robustness, flexibility and efficiency by applying such observations and concepts. An important aspect

of Organic Computing is autonomy. In this scope there are the self-* properties as described in [13].

Federated learning can also profit from advances in the fields of Organic Computing as efficiency, optimization and privacy are highly important issues in federated learning. There are some fields of Organic Computing described by Müller-Schloer and Tomforde in [12], who write about the possibilities and importance of keeping federated learning optimized and private. They explain an efficient and robust way of data communication which benefits the efficiency concerns we had earlier.

The self-* properties help designing a federated learning system. Self-configuration for example is seen in the local updates. Structured Updates often have different pre-specified structures, which can be adapted each round.

VII. Summary

Federated learning uses a great concept of model and update. The adaptation of general machine learning approaches is a very important aspect. Even though SGD is a common way for optimizing, the use of modified methods like FedSGD is inevitable. Also, the use of FedAvg for the averaging process is an immense advantage. Certainly, any more evolvement in optimization in this rather new field can and should be done in the future.

In terms of communication efficiency, the concept of structured and sketched updates is a good basis for improvement. The combination of those should be beneficent for reducing communication cost and improving the overall efficiency. As far as privacy issues go, there are many starting points. Even though there isn't much done for federated learning in particular, there is much research done in machine learning or communication in general. Many of those concepts and methods can easily be adapted to be applied to federated learning. Of course, this should be considered as a secondary aim, as the primary should be to optimize the efficiency of the methods and algorithms used in federated learning. Nevertheless, it is an important point which must not to be neglected at last.

Federated learning is an Organic Computing system in many aspects. Not only the organisation of many single nodes but also the optimization, efficiency and privacy issues are very important aspects. These can profit immensely from concepts of Organic Computing by adapting real-world analogies, which can make the learning process more robust and efficient overall.

In conclusion, the field of federated learning promises an interesting field of research for modern and future technologies and use-cases, e.g. language prediction. Many aspects of this field are already researched, proven and optimized. Others still have to be looked into. Federated learning gives a possibility for many scenarios to improve through efficient and self-organized updates. The direct involvement of end users into the learning process is an innovative alternative to traditional, big data centers. But there are also still many stumbling blocks in the way of federated learning. Anyways, federated learning will get more and more attention in the future, as the concept behind it fits very well in our nowadays society. Almost everyone is a mobile phone user, thereby a potential client in a federated learning system. I am very eager to continue my own research on this field and am excited about the future improvements and achievements that will certainly be made.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. Agüera y Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data", arXiv:1602.05629v3, 28. Feb. 2017

[2] B. McMahan, D. Ramage, "Federated Learning: Collaborative Machine Learning without Centralized Data", Google AI Blog, April 6. 2017

[3] J. Konečný, B. McMahan, D. Ramage, P. Richtárik, "Federated Optimization: Distributed Machine Learning for On-Device Intelligence", arXiv:1610.02527v1, 6. Oct. 2016

[4] M. Gürbüzbalaban, A. Ozdaglar, P. Parillo, "Why Random Reshuffling Beats Stochastic Gradient Descent", arXiv:1510.08560, 29. Oct. 2015

[5] J. Chen, R. Monga, S. Bengio, R. Jozefowicz. "Revisiting Distributed Synchronous SGD." In ICLR Workshop Track, 2016

[6] J. Konečný, B. McMahan, F. Yu., A. Suresh, D. Bacon, P. Richtárik, "Federated Learning: Strategies for Improving Communication Efficiency", arXiv:1610.05492v2, 30. Oct. 2017

[7] A. Suresh, F. X. Yu, S. Kumar, B. McMahan, "Distributed Mean Estimation with Limited Communication", arXiv:1611.00429, 25. Sep. 2017.

[8] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth, "Practical Secure Aggregation for Privacy-Preserving Machine Learning", 2017

[9] P. Xie, M. Bilenko & T. Finley, R. Gilad- Bachrach & K. Lauter & M. Naehrig, "Crypto-Nets: Neural Networks over Encrypted Data", arXiv:1412.6181v2, 24. Dec. 2014

[10] M. Pathak, S. Rane, W. Sun, Bhiksha Raj, "Privacy Preserving Probabilistic Inference with Hidden Markov Models", May 2011

[11] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, V. Shmatikov, "How to Backdoor Federated Learning", arXiv:1807.00459v2, 1. Oct. 2018

[12] C. Müller-Schloer, S. Tomforde, "Organic Computing – Technical Systems for Survival in the Real World"

[13] S. Tomforde, B. Sick, C. Müller-Schloer, "Organic Computing in the Spotlight", arXiv:1701.08125v1, 27. Jan. 2017