An Overview of Federated Learning

A Report
Submitted to Dr. Vaezi of
The Department of Electrical and Computer Engineering
Villanova University
by

John Fattore

5G Wireless Networks

VILLANOVA
UNIVERSITY

5/10/2023

Contents

1 Abstract

Since Google coined the term federated learning in 2016, the training method has blossomed in popularity as a way to train a centralized model using decentralized data. A centralized model collects data and trains the model all in one place. Federated learning provides privacy because rather than sending data to a central server, a model is sent to local devices to be trained. Communication cost are the largest road block for federated learning, but advances in wireless networks has reduces this burden and made federated learning possible. FedAvg is the first realization of this federated learning algorithm, but other techniques such as FedProx, q-FedAvg, and per-FedAvg have improved on the basic method. The last section of the paper explores a basic federated learning simulation using readily available NBA stats.

2 Background

2.1 Distributed Learning

Distributed learning features training a machine learning model across multiple nodes. The main motivation for distributed learning is typically increase parallelization and decrease training time. Large data sets require powerful resources to efficiently process and use to train a model. Distributed learning can also serve to reduce communication costs and provide security. If data is kept on multiple servers, it may not be practical to transfer it all to a central server for training. This communication also opens up the possibility for privacy concerns. However, with all the benefits there are certainly drawbacks. Centralized models have the benefit of all being in the same physical location and so do not run into the same communication issues as decentralized learning systems. [1]

2.2 Federated Learning vs Distributed Learning

Federated Learning is a type of distributed Learning aimed at gathering data from edge devices, while maintaining privacy. Both methods allocate training to various nodes, and benefit from a network of model trainers. Distributed learning aims to train the model on multiple nodes because of the efficiency of parallel computing. This method of learning's biggest advantage is the speed at which a large data set can be used to train a model, which is faster than a single centralized server could. These nodes are connected together with fast connections such as physical cable. In contrast, Federated learning can be slower at training models than a centralized server could because of the large communication costs. Federated learning allocated training to users' edge devices so that sensitive data never has to leave the device. Data is not collected in a central location and only weights are communicated between nodes. Magnitudes more nodes participate in federal learning compared to a typical distributed learning method. These nodes are not connected together by high speed networks, but rather through more expensive and slower networks such as WiFi or wireless 5G cellular networks.

2.3 5G/6G and Federated Learning

Federated learning has become a practical training algorithm thanks to ever advancing wireless networks that connect all our devices. Modern day requirements for wireless networks include enhanced mobile broadband, ultra-reliable and low latency communications and massive machine type communications. [2] This emphasis on massive machine type communications is

what has strengthened the practical use of federated learning. Before these advances, devices collecting relevant data were not connected enough to communicate to train a ML model. Wireless networks are now designed to connect everything including smart phones, IoT devices, and other devices all collecting real world data. Federated learning thrives on this prime information that is now accessible. These performance results have been made possible with the incorporation of technologies such as Massive MIMO, orthogonal frequency division modulation (OFDM), and the use of millimeter waves (mmWaves). Massive MIMO enables the utilization of large antenna arrays that are is infinitely scalable. Massive MIMO also allows for precise beam forming that points the signal right at the users to provide strong power and a high SNR. OFDM was standardized in 4G, but is still the modulation scheme of modern networks and WiFi. OFDM makes efficient use of the frequency spectrum by splitting the signal into 64 different frequencies that do not interfere with each other during physical transmission and provide a constant gain over the channel. Finally, the increase in use of wireless networks has made use of mmWaves practical. These signals have high loss, but bay stations placed close enough together allows high volumes of data to be transmitted. These mmWaves can be utilized in cities where the massive amount of data collecting sensors can be tapped for training data. Increases in device connection will be a prime concern in the adoption of 6G, which will enable even more communication between devices and easier implementations of federated learning. In the future, federated models may rely solely on the 5G/6G wireless network to connect devices.

2.4 IID vs Non-IID

A data set that is independent and identical distributed (IID), contains data samples that all share the same underlying distribution as well as being statistically independent. IID data sets are easier to handle and produces more accurate models when compared to non-IID data sets. Non-IID data sets contain heterogeneous data samples which vary in size, dependency, and distribution. Unfortunately, many data sets we find in the world are non-IID and so methods to deal with this issue are essential to develop. This is contrary to the data set in the NBA case study, which is particularly IID, greatly aiding model accuracy.

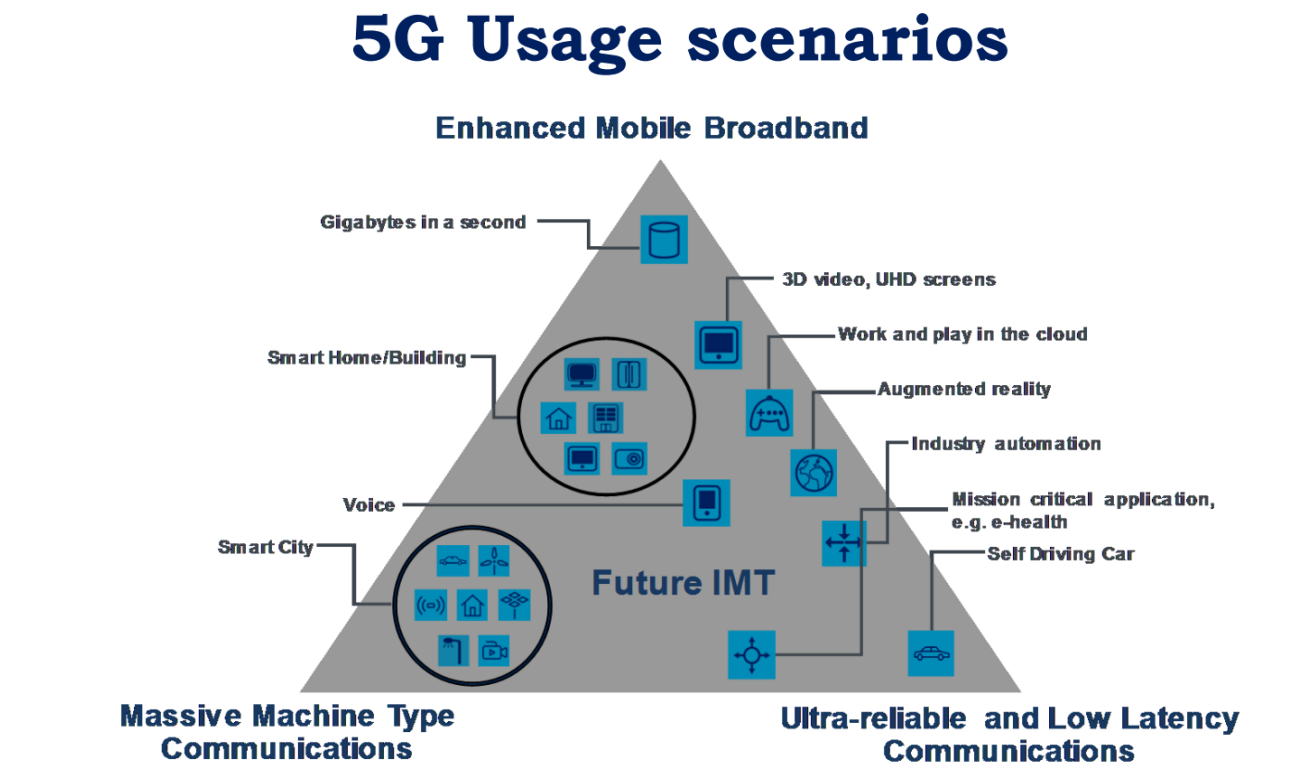

Figure 1: ITU Requirements for 5G Networks, Courtesy of [2]

3 Federated Learning

Federated learning is training a centralized model from decentralized data. A primary motivation to use this learning method is to preserve privacy, while accessing the large amounts of data scatter across various devices. Federated learning models are powered by the sensitive data of personal cellular devices. Even anonymous personal data can lead to security concerns when combined with other data. This method eliminates any worries of privacy leaks by never communicating raw data, but rather just the bare minimum to update the central model. In a basic federated learning model, a central server will hold a global model which is updated by selected nodes after each round. At the beginning of each round the current model is sent to all clients. A subsection of these clients are chosen to train the model and return training data back to the central server to be averaged together. A single round can take a long time to process as the central server waiting for clients to return training data. Low participation keeps rounds quick. Client updates are usually delayed because of communication costs. Clients will typically only participate in the training if the device is charged, plugged in, and connected to their home WiFi. Some key issues that federated learning is able to overcome is Non-IID data sets, massively distributed data, and limited communication among nodes.

3.1 FedSGD and FedAvg

Gradient descent is a popular and powerful tool that is used to minimize cost functions and ultimately train models. A successful variation of gradient descent is stochastic gradient descent (SGD). This optimization method is the starting point for FederatedSGD (FedSGD). Specifically, large-batch synchronous SGD is used because there is little cost in involving lots of clients. The simplest implementation of federated learning is FedSGD, which trains the model on edge devices using stochastic gradient descent. A typical implementation of FedSGD will have each client k compute a gradient from the current model and its local data. These gradients are averaged together at the central server and used to make one gradient descent step to update the model. FederatedAveraging (FedAvg) takes this approach, but allows the edge devices to perform multiple rounds of training before reporting back its calculated gradient. The computations per round of the FedAvg algorithm are controlled by C, the fraction of clients training the model, E, the training passed each client makes per round, and B, the size of the mini-batch of data. FedAvg becomes FedSGD when C = 1, B = infinity, and E = 1, in other words when all client makes one training pass over the entirely of its local data per training round. Slow clients tpyically dropped for speed [3]

3.2 FedProx

FedAvg is an ideal federated training method if the model is being fed an IID data set. Non-IID introduces new factors that may cause inaccuracies in the basic FedAvg model. FedProx is a federated training method based on FedAvg that aims to make the algorithm better at handling heterogeneous data. Local devices with extreme data will provide extreme gradients that can skew the overall gradient step. FedProx fixes this problem by introducing a proxy term to the objective function being optimized. This proxy term provides two main benefits: extreme local gradients will be kept in check and clients are able to perform a variable amount of local iterations each round. Because FedAvg allows for multiple local training rounds per update, a device has time to produce an extreme gradient, and then compound this gradients after successive weight updates. Once the training round is finished and gradients averaged, this extreme gradient can throw off the overall average. The proxy term restricts the local updates and keeps the local weights close to the global weights. This also allows the safe incorporation of a variable amount of work from each device. This makes the process more efficient by letting fast working devices to continue working, instead of making them wait for the slower devices such as in FedAvg. This also ensures that the same type of slow devices are not always dropped,

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

(a) FedSGD and FedAvg Objective Function

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

initialize w_0

for each round $t = 1, 2, \dots$ **do**

$m \leftarrow \max(C \cdot K, 1)$

$S_t \leftarrow$ (random set of m clients)

for each client $k \in S_t$ **in parallel do**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$ *// Erratum*[4]

ClientUpdate(k, w): *// Run on client k*

$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)

for each local epoch i from 1 to E **do**

for batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return w to server

(b) FederatedAveraging (FedAvg) Training Method

Figure 2: FederatedAveraging (FedAvg) Algorithm, Courtesy of [3]

$$\min_w h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2 .$$

(a) FedProx Objective Function

Algorithm 2 `FedProx` (Proposed Framework)

Input: $K, T, \mu, \gamma, w^0, N, p_k, k = 1, \cdots, N$
for $t = 0, \cdots, T-1$ **do**
Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
Server sends w^t to all chosen devices
Each chosen device $k \in S_t$ finds a w_k^{t+1} which is a γ_k^t-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w\ h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$
Each device $k \in S_t$ sends w_k^{t+1} back to the server
Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
end for

(b) FedProx Training Method

Figure 3: FedProx Algorithm, Courtesy of [4]

causing an improperly trained model [4]

3.3 q-FedAvg

While the models trained using FedAvg or FedProx algorithms yield acceptable global accuracy, this performance does not always carry over to all local devices. Particularly with non-IID data, simply optimizing the cost function may disproportionately favor some devices over others. A model's performance between different devices can vary wildly because of local data varying in both size and distribution. However, global accuracy should not sacrificed in the name of device level fairness and a balance between the two must be found. q-FedSGD and q-FedAvg are algorithms that performs well for all devices without dropping global accuracy. Both algorithms optimize the same objective function in order to achieve this fairness. The objective function

is called q-Fair Federated Learning (q-FFL) and including a q term that empathizes devices with large loss / gradients in order to improve the performance on those devices. This q term can be scaled to vary device fairness and increases proportionally with the value of q. If q is 0, q-FedSGD becomes FedSGD and q-FedAvg becomes FedAvg. Like before, the q-FedSGD varient limits local training to one round while FedAvg allows for multiple local training rounds before global convergence. It is observed that the q-FedAvg algorithm can reduce the variance of accuracies across devices by 45% on average when compared to FedAvg while maintaining the same level of global accuracy. [5]

3.4 per-FedAvg

Per-FedAvg is similar to q-FedAvg in that both training algorithms are focused on increasing local device model accuracy. A model which is optimized by simply minimizing the global objective function is not personalized to specific local data. The per-FedAvg algorithm achieves local accuracy by requiring the global model to be trained by the local device before it is ready to use. Instead of a fully optimized model as as the global model, the global model is a initial point for training shared between all users. The global model is optimized using local data, thus a personalized model is produced. The objective function includes a gradient step so that once the function is optimized, a good model can be found on all devices after one local gradient step. It is analogous to breaking in a new pair of football cleats so that they fit properly. A single global model is not accurate enough for all users, but the right global model that can be easily tweaked by the local devices into their own personalized model.

4 Case Study: NBA FedSGD

A simple linear regression model that predicts NBA season win percentage is trained using the FedSGD algorithm. The study is intended to provide some context to the federated learning process. Data is obtained from NBA.com and is comprised of 10 seasons ranging from 2008 to 2019. The data is split up into training and testing data, with the two most recent seasons being the testing data. 6 parameters are chosen including 3-Point %, 3-Point Attempts, Field Goal %, Turnovers, Assists, and Offensive Rebounds. Gradient descent is used to optimize the MSE cost / optimization function of the linear regression mode. The simulated federated learning features a global model that is updated each epoch by the average gradient of a random selection of NBA teams.

$$\min_{w} \; f_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w)\,,$$

(a) q-FFL Objective Function

Algorithm 1 q-FedSGD

Input: $K, T, q, 1/L, w^0, p_k, k = 1, \cdots, m$
for $t = 0, \cdots, T-1$ **do**
  Server selects a subset S_t of K devices at random (each device k is chosen with prob. p_k)
  Server sends w^t to all selected devices
  Each selected device k computes:

$$\Delta_k^t = F_k^q(w^t)\nabla F_k(w^t)$$
$$h_k^t = qF_k^{q-1}(w^t)\|\nabla F_k(w^t)\|^2 + LF_k^q(w^t)$$

  Each selected device k sends Δ_k^t and h_k^t back to the server
  Server updates w^{t+1} as:

$$w^{t+1} = w^t - \frac{\sum_{k\in S_t}\Delta_k^t}{\sum_{k\in S_t} h_k^t}$$

end for

Algorithm 2 q-FedAvg

Input: $K, E, T, q, 1/L, \eta, w^0, p_k, k = 1, \cdots, m$
for $t = 0, \cdots, T-1$ **do**
  Server selects a subset S_t of K devices at random (each device k is chosen with prob. p_k)
  Server sends w^t to all selected devices
  Each selected device k updates w^t for E epochs of SGD on F_k with step-size η to obtain $\bar{w}_k^{t+1}$
  Each selected device k computes:

$$\Delta w_k^t = L(w^t - \bar{w}_k^{t+1})$$
$$\Delta_k^t = F_k^q(w^t)\Delta w_k^t$$
$$h_k^t = qF_k^{q-1}(w^t)\|\Delta w_k^t\|^2 + LF_k^q(w^t)$$

  Each selected device k sends Δ_k^t and h_k^t back to the server
  Server updates w^{t+1} as:

$$w^{t+1} = w^t - \frac{\sum_{k\in S_t}\Delta_k^t}{\sum_{k\in S_t} h_k^t}$$

end for

(b) q-FedAvg Training Method

Figure 4: q-FedAvg Algorithm, Courtesy of [5]

$$\min_{w\in\mathbb{R}^d} F(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w - \alpha\nabla f_i(w)),$$

(a) per-FedAvg Objective Function

Algorithm 1: The proposed Personalized FedAvg (Per-FedAvg) Algorithm

Input: Initial iterate w_0, fraction of active users r.
for $k : 0$ to $K-1$ **do**
  Server chooses a subset of users $\mathcal{A}_k$ uniformly at random and with size rn;
  Server sends w_k to all users in $\mathcal{A}_k$;
  for all $i \in \mathcal{A}_k$ **do**
    Set $w^i_{k+1,0} = w_k$;
    for $t : 1$ to τ **do**
      Compute the stochastic gradient $\tilde{\nabla} f_i(w^i_{k+1,t-1}, \mathcal{D}^i_t)$ using dataset $\mathcal{D}^i_t$;
      Set $\tilde{w}^i_{k+1,t} = w^i_{k+1,t-1} - \alpha\tilde{\nabla} f_i(w^i_{k+1,t-1}, \mathcal{D}^i_t)$;
      Set $w^i_{k+1,t} = w^i_{k+1,t-1} - \beta(I - \alpha\tilde{\nabla}^2 f_i(w^i_{k+1,t-1}, \mathcal{D}^{''i}_t))\tilde{\nabla} f_i(\tilde{w}^i_{k+1,t}, \mathcal{D}^{'i}_t)$;
    end for
    Agent i sends $w^i_{k+1,\tau}$ back to server;
  end for
  Server updates its model by averaging over received models: $w_{k+1} = \frac{1}{rn}\sum_{i\in\mathcal{A}_k} w^i_{k+1,\tau}$;
end for

(b) per-FedAvg Training Method

Figure 5: per-FedAvg Algorithm, Courtesy of [6]

4.1 Federated vs Centralized Learning

The federated model performed almost exactly the same to the none federated model. A homogeneous data set contributed to this results and a more heterogeneous data set may have had more trouble being trained with federated learning. However, this study promotes the use of practical federated learning. The model created is able to converge into a reliable predictor without the need to centralize data and access potentially private data. A central server can train this NBA win predictor without ever seeing any of the raw data possessed by the teams.

4.2 Model Improvements

More complicated versions of the federated learning are not utilized in this example for simplicity and practicality. Communication costs could be cut further with the implementation of FedAvg, which would require less overall training rounds. The data is quite homogeneous, which discourages the use of ProxFed, q-Fed, Per-FedAvg. Simplicity makes the study easier to analyze, but also increases efficiency and doesn't incorporate unneeded complexity.

4.3 Evaluation

The trained model was evaluated using mean absolute percent error (MAPE), mean absolute error (MAE), mean square error (MSE), and root mean square error (RMSE), scoring 29.36, 11.387, 185.29, and 13.6 respectively. The MAPE score of 29.36% means each prediction is off about 30% and the MAE means each prediction has an error of about 11%. For example, the Golden State Warriors in the 17-18 season had a winning percentage of 70.7%, but the model predicted 60.3%.

References

[1] Verbraeken, J., Rellermeyer, J. S., Verbelen, T., Kloppenburg, J., Katzy, J., & Wolting, M. (2020, March). A survey on Distributed Machine Learning. ACM Digital Library. Retrieved April 25, 2023, from https://dl.acm.org/doi/pdf/10.1145/3377454

[2] IMT Vision – Framework and overall objectives of the future development of IMT for 2020 and Beyond. International Telecommunication Union. (2015). Retrieved May 6, 2023, from https://www.itu.int/dmspubrec/itu-r/rec/m/R-REC-M.2083-0-201509-I!!PDF-E.pdf

[3] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Arcas, B. A. y. (2023, January 26). Communication-efficient learning of Deep Networks from Decentralized Data. arXiv.org. Retrieved April 25, 2023, from https://arxiv.org/abs/1602.05629

[4] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020, April 21). Federated optimization in Heterogeneous Networks. arXiv.org. Retrieved April 27, 2023, from https://arxiv.org/abs/1812.06127

[5] Li, T., Sanjabi, M., Beirami, A., & Smith, V. (2020, February 14). Fair Resource Allocation in Federated Learning. arXiv.org. Retrieved May 4, 2023, from https://arxiv.org/abs/1905.10497

[6] Fallah, A., Mokhtari, A., & Ozdaglar, A. (2020). Personalized federated learning with theoretical guarantees: A model ... NeurIPS 2020. Retrieved May 4, 2023, from https://proceedings.neurips.cc/paper/2020/file/24389bfe4fe2eba8bf9aa9203a44cdad-Paper.pdf

[7] Bellwood, L., & McCloud, S. (2016). Google Federated Learning. Google. Retrieved May 4, 2023, from https://federated.withgoogle.com/