Machine Learning in Communications
Lecture 8: Gradient Compression for Federated Learning: A Wireless Communication Perspective

Harpreet S. Dhillon

Wireless@VT, Bradley Department of Electrical & Computer Engineering
Virginia Tech, Blacksburg, VA

https://www.dhillon.ece.vt.edu
hdhillon@vt.edu

Joint work with Keerthana Bhogi and Chiranjib Saha

JTG/IEEE Information Theory Society Summer School
IIT Kanpur

Role of Distributed Learning in Wireless Networks

- Conventional ML approaches rely on the assumption of having the data and processing heads in a single central entity.
- With the increasing number of mobile devices and sensors, the amount of data is increasing but much of it is of private and distributed in nature.
- Even if we ignore privacy for the sake of argument, it is very inefficient (and in some cases impossible) to transmit the local data at all the users to the central server (say a base station) to perform centralized learning.
- Because of these reasons, decentralized learning solutions are especially relevant for wireless applications.
  - Our focus will be on a specific decentralized learning technique, called Federated Learning.

Federated Learning (FL)

- Federated Learning: An emerging decentralized approach that allows the central server to collectively reap the benefits of the rich distributed data without the need to centrally store it where the central model is trained using the aggregated locally-computed updates.
- How does this aggregation of locally trained models at the central server occur?
- FL has unique properties that differentiate it from the other distributed learning techniques: Data is non-IID, unbalanced, and massively distributed, limited communication resources.

B. McMahan, *et al.*, "Communication-efficient learning of deep networks from decentralized data," in Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273-1282.

S. Niknam, H. S. Dhillon and J. H. Reed, "Federated Learning for Wireless Communications: Motivation, Opportunities, and Challenges", *IEEE Communications Magazine*, vol. 58, no. 6, pp. 46-51, Jun. 2020.

Gradient Descent (GD) and Stochastic Gradient Descent (SGD)

- Let $\mathbf{w}$, $\mathcal{B}$ be the model parameters to be optimized and dataset, $f(\cdot)$ be the loss function.

$$F(\mathbf{w}) = \frac{1}{\mathcal{B}} \sum_{\mathbf{u} \in \mathcal{B}} f(\mathbf{w}, \mathbf{u})$$

- The minimization of above loss is typically carried out through iterative gradient descent (GD), in which the model parameters at the t-th iteration, are updated as

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \nabla F(\mathbf{w}), \quad \nabla F(\mathbf{w}) = \frac{1}{\mathcal{B}} \sum_{\mathbf{u} \in \mathcal{B}} \nabla f(\mathbf{w}, \mathbf{u})$$

- We need to calculate the gradient $\nabla F(\mathbf{w})$ for the whole dataset to perform just one update of model parameters $\mathbf{w}$, GD can be very slow and intractable for large datasets.
- Unlike GD, SGD calculates gradients for every element of the dataset and updates the model for each gradient. One update of the model parameters is obtained as

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \mathbf{g}(t), \quad \mathbf{g}(t) = \nabla f(\mathbf{w}, \mathbf{u}), \mathbf{u} \in \mathcal{B}$$

  - Since the model is updated for every point in the dataset $\mathcal{B}$, the loss function has higher fluctuations and hence higher variance.

Gradient Descent (GD) and Stochastic Gradient Descent (SGD)

- There exists another variant of GD technique which is the mini-batch SGD which takes the best of the worlds of GD and SGD. The model parameters are updated as below where $\mathcal{M}$ is a mini-batch i.e. $\mathcal{M} \subset \mathcal{B}$.

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \mathbf{g}(t), \quad \mathbf{g}(t) = \frac{1}{|\mathcal{M}|} \sum_{\mathbf{u} \in \mathcal{M}} \nabla f(\mathbf{w}, \mathbf{u})$$

- Similar to SGD, If the mini-batch $\mathcal{M}$ at each iteration is sampled randomly, then the gradient estimates $\mathbf{g}(t)$ are unbiased: $\mathbb{E}[\mathbf{g}(t)] = \nabla F(\mathbf{w})$.
- If $\mathcal{M} = \mathcal{B}$, then mini-batch SGD = GD.
- If $|\mathcal{M}| = 1$, then mini-batch SGD = SGD.
- The mini-batch SGD is sometimes referred to as just SGD.

Federated Learning Procedure

- Step 1: The distributed users $n = (1, \cdots, K)$ update local learning model $\mathbf{w}_n(t)$ based on the local data $\mathcal{B}_n, \mathcal{B} = \bigcup_{n=1}^{K} \mathcal{B}_n$, with the stochastic gradients $\mathbf{g}_n(t)$ obtained using SGD.
- Step 2: The users send the local gradients $\mathbf{g}_n(t)$ of the updated local models to the central server.
- Step 3: The server learns a global model $\mathbf{w}(t)$ by aggregating the received local updates corresponding to all the users.
- Step 4: The parameters of the updated global model at the server $\mathbf{w}(t)$ are sent back to all the users to update the local models for the next iteration $t+1$ i.e. $\mathbf{w}_n(t+1) = \mathbf{w}(t+1)$.

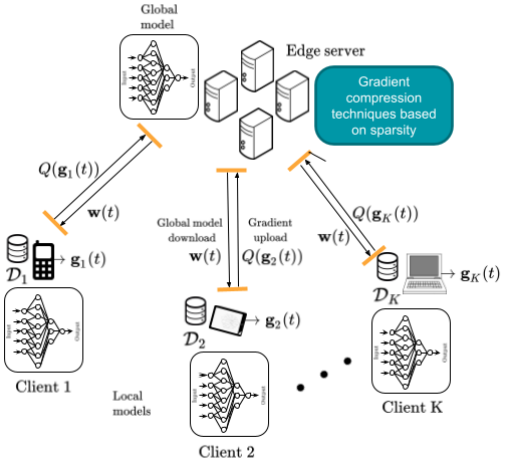

Figure: Federated learning procedure.

Summary of the Federated Learning Procedure

$$\mathbf{w}_n(t+1) := \mathbf{w}_n(t) - \eta\ \mathbf{g}_n(t), \tag{1}$$

$$\mathbf{g}(t) = \frac{1}{K}\sum_{n=1}^{K}\mathbf{g}_n(t), \quad \mathbf{g}^s(t) = \sum_{n=1}^{K}\mathbf{g}_n(t) \tag{2}$$

$$\mathbf{w}(t+1) := \mathbf{w}(t) - \eta\ \mathbf{g}(t) \tag{3}$$

- In Distributed SGD (DSGD), $\mathbf{g}(t)$ is updated as $\mathbf{g}(t) = \frac{1}{K}\sum_{n=1}^{K}\mathbf{g}_n(t)$.
- Take-away: For DSGD in FL, it is enough for the server to have the knowledge of gradient-sum i.e. $\mathbf{g}^s(t) = \sum_{n=1}^{K}\mathbf{g}_n(t)$ at every training iteration t.

Connection of FL with Wireless Communication

- We model the medium between the users and the central server as a shared wireless medium (specifically, a noisy wireless Gaussian MAC channel).
- Let $\mathbf{x}_n(t) \in \mathbb{R}^d$ be the signal transmitted by n-th user, $\mathbf{n}(t)$ be the AWGN noise at iteration t, then the signal received at the server is $\mathbf{y}(t)$.

$$\begin{aligned}\mathbf{y}(t) &= \sum_{n=1}^{K} \mathbf{x}_n(t) + \mathbf{n}(t) \\ &= \mathbf{x}(t) + \mathbf{n}(t)\end{aligned}$$

M. M. Amiri and D. Gunduz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," IEEE Trans. on Signal Process., vol. 68, pp. 2155-2169, 2020.

Our Focus

- Sending gradients and weights back and forth require a lot of wireless resources.
- One solution is to compress these gradients at the clients and reconstruct them at the server. This will be our focus.
  - Our proposed approach will be cognizant of the properties of the gradients being compressed.
- Notation and Assumptions:
  - Let $f_c(\cdot)$ be the gradient compressor, then $\mathbf{x}_n(t) = f_c(\mathbf{g}_n(t))$ is the compressed gradient for the n-th user.
  - We use a linear compressor $f_c(\cdot)$. Hence, $\mathbf{x}_n(t) = f_c(\mathbf{g}_n(t)) \implies \mathbf{x}(t) = f_c(\mathbf{g}(t))$.
  - With a linear compressor, the considered gaussian MAC channel model automatically provides the server with the noisy gradient-sum without needing to transmit gradient of each user separately.
  - Reconstructed gradient at the central server to update the global model: $\hat{\mathbf{g}}(t)$.

Revisiting the Federated Learning Procedure

Algorithm 1: Pseudo algorithm for FL procedure with DSGD

Initialize: $\mathbf{w}(0), \mathbf{\Delta}_n(0) = 0$

for $t = 0, \cdots, T-1$ **do**

- **devices/users do:**

  for $n = [K]$

  Compute $\mathbf{g}_n(t)$ using $\mathcal{B}_n$ using SGD.

  Compress $\mathbf{g}_n(t)$ for transmission: $\mathbf{x}_n(t) = f_c(\mathbf{g}_n(t))$ ← a step that is of current interest

  end for

- **Channel does:**

  $\mathbf{y}(t) = \sum_{n=1}^{K} \mathbf{x}_n(t) + \mathbf{n}(t), \quad \mathbf{x}_n(t) = f_c(\mathbf{g}_n(t))$

- **BS/Server performs:**

  $\hat{\mathbf{g}}(t) = \mathbf{g}(t)$ that is reconstructed from $\mathbf{y}(t)$ ← a step that is of current interest

  $\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \hat{\mathbf{g}}(t)$

end for

Setting up the Problem

- The objective here is to recover $\mathbf{g}^s(t) = \sum_{n=1}^{K} \mathbf{g}_n(t)$ from $\mathbf{x}(t) = \sum_{n=1}^{K} \mathbf{x}_n(t) = \sum_{n=1}^{K} f(\mathbf{g}_n(t)) = f_c(\mathbf{g}^s(t))$ at the server instead of recovering each $\mathbf{g}_n(t)$ from $\mathbf{x}_n(t)$, $\forall n = [K]$. Here, the compression ratio (CR) is given as

$$\text{CR} = \frac{\text{length}(\mathbf{g}_n(t))}{\text{length}(f_c(\mathbf{g}_n(t)))} = \frac{N}{d}$$

- We now present a technique that compresses and reconstructs the gradient efficiently (which will be inspired by the key properties of the gradients).

M. Yu, *et al.*, "GradiVeQ: Vector quantization for bandwidth-efficient gradient aggregation in distributed CNN training," in Advances in Neural Information Processing Systems, 2018, pp. 5123-5133.

Y. Lin, *et al.*, "Deep gradient compression: Reducing the communication bandwidth for distributed training," arXiv preprint arXiv:1712.01887, 2017.

Properties of Gradients in Neural Networks

The gradients generated during training of a neural network (NN) exhibit following properties:

- Sparsity
- Temporal correlation

Sparsity:

- An empirical observation is that the gradients of a NN are extremely skewed [2,3] with most of the values close to zero.
- This indicates that there are relatively fewer gradient elements than the actual number which are important.
- The gradients can be sparsified by removing the gradients that are close to zero (by absolute value) using a threshold and retaining only important/dominant gradients. This is referred to as gradient sparsification which has been used widely in the literature.
  - We will also be using gradient sparsification here.

Temporal correlation:

- The observed gradients evolve slowly over training iterations, which is due to the steady gradient directions and step size under reasonable learning rates.

Convolutional Neural Networks (CNN)

- The above two properties are satisfied for any NN in general.
- The gradients of a convolutional layer exhibit an additional property of spatial correlation along with the sparsity and temporal correlation.
- A convolutional layer applies filters on the input data to extract features that are relevant to the application. So, the weights of the filters are the quantities that are learnt during training of a CNN.
- The gradient corresponding to a convolutional layer with F filters as shown in the adjacent figure can be represented as $4D$-tensor $\mathcal{G} \in \mathbb{R}^{H \times W \times D \times F}$.

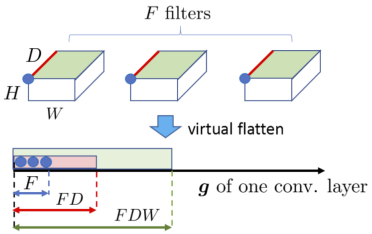

Figure: Each convolutional layer has F filters of dimension (H, W, D), where H, W, D is the height, width, depth of each filter which acts on the input producing $FDWH$ gradients in every iteration. We flatten the gradients into a vector $\mathbf{g}$ by placing every F collocated gradients from all the F filters next to each other in $\mathbf{g}$. The location selection traverses depth, width, then height. Picture taken from [4].

Spatial Correlation of Convolutional Gradients

Figure: Plot showing a sample gradient $\mathbf{G} \in \mathbb{R}^{25 \times 200}$ corresponding to convolutional layer of dimension $(5, 5, 10, 20)$

Figure: Plot showing the above gradient $\mathbf{G}$ sparsified with a threshold for sparsification at 95 percentile

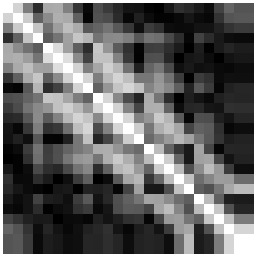

Figure: Correlation matrix $\mathbf{R} \in \mathbb{R}^{25 \times 25}$ of $\mathbf{G}$

- A matrix representation of $\mathcal{G}$ is $\mathbf{G} \in \mathbb{R}^{HW \times FD}$ and the corresponding correlation matrix $\mathbf{R}$ is an $HW \times HW$ matrix containing the pairwise correlation coefficient between each pair of rows in the $HW \times FD$ matrix $\mathbf{G}$.

Our Objective

- The sparse nature of the gradients indicates that the actual dimension of the gradients is much smaller than its size. Hence, we can compress the (approximately) sparse gradients before transmitting.
- The objective of our gradient transmission method is to jointly exploit the sparsity of the gradients and the spatial correlation in the gradients of the convolutional layer.

Sparsity and Compressive Sensing (CS)

- CS is signal processing technique used for efficiently acquiring and reconstructing signals by finding solutions to underdetermined linear systems.
- It is based on the principle that the sparsity of a signal can be exploited to recover it from far fewer samples than required by the Nyquist-Shannon sampling theorem.
- Let $\mathbf{s} \in \mathbb{R}^N$ be a sparse signal which is measured through a matrix $\mathbf{A} \in \mathbb{R}^{d \times N}$ using the following noisy linear measurement where $\mathbf{n}$ is random noise.

$$\mathbf{y} = \mathbf{A}\mathbf{s} + \mathbf{n}$$

- If $d < N$, then the above system of equations is underdetermined. Although $d < N$, $\mathbf{s}$ can be reconstructed exactly or approximately if $\mathbf{s}$ is sparse.
- Let N_o be the number of non-zero entries of $\mathbf{s}$, then its sparse ratio (SR) is N_o/N.
- Sparse signal reconstruction (SSR): The goal of SSR is to recover $\mathbf{s}$ from $\mathbf{y}, \mathbf{A}$.

Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications.* Cambridge University Press, 2012.

Modeling Convolutional Gradients

- For the Gaussian MAC channel model, we have Gaussian likelihood $p(\mathbf{y}|\mathbf{s})$ with the conditional pdf $p(\mathbf{y}|\mathbf{s})$ where $\sigma^2\mathbf{I}$ is the co-variance matrix of the zero-mean gaussian noise vector $\mathbf{n}$. Therefore the likelihood is

$$p(\mathbf{y}|\mathbf{s}) = (2\pi\sigma^2)^{-\frac{N}{2}} \exp\left(-\frac{1}{2\sigma^2} \|\mathbf{y} - \mathbf{A}\mathbf{s}\|_2^2 \right)$$

- In order to model the sparsity of the gradients, we assume a spike and slab prior on $\mathbf{s} = [s_1, \cdots, s_N]$ (a sparsity-promoting prior) with the following joint pdf on $\mathbf{s}$.

$$p(\mathbf{s}) = \prod_{i=1}^{N} p(s_i) = \prod_{i=1}^{N} [(1-\lambda_i)\delta(s_i) + \lambda_i h(s_i)],$$

where $\lambda_i \in (0, 1)$ models sparsity, i.e., the probability of s_i being non-zero, $\delta(s_i)$ is the Dirac delta, and $h(s_i)$ is the distribution to model the non-zero entries of the sparse signal, which is assumed to be Gaussian with mean θ and variance ϕ.

$$h(s_i) = \mathcal{N}(s_i; \theta, \phi) \ \forall \ i = [N]$$

- Let $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_N]$.

Proposed Approach to Gradient Reconstruction: Bayesian SSR

- $\mathbf{q} = (\boldsymbol{\lambda}, \theta, \phi, \sigma^2)$: set of prior parameters required for the estimation of $\mathbf{s}$.
- We propose the SSR as the MMSE estimation of $\mathbf{s}$ where

$$\hat{\mathbf{s}}^{\mathrm{MMSE}} = \arg\min_{\hat{\mathbf{s}}} \mathbb{E}(\|\mathbf{s} - \hat{\mathbf{s}}\|_2^2) = \mathbb{E}_{\mathbf{s}|\mathbf{y}}\left[\mathbf{s}|\mathbf{y}; \mathbf{q}\right]$$

- Sparsity is incorporated through the spike-and-slab prior as we have already seen.
- Spatial correlation is incorporated in the prior $p(\mathbf{s})$ through sparse ratio $\boldsymbol{\lambda}$ where λ_i and λ_j are correlated through *local averaging* during the process of updating the parameters.
- In this FL setting, the sparse signal is nothing but the sparsified gradient-sum $\mathbf{g}(t)$.
- We adopt a Bayesian approach for the SSR to obtain $\mathbf{s}^{\mathrm{MMSE}}$ which combines two powerful inference frameworks: expectation maximization (EM) and approximate message passing (AMP).
  - We use a CS reconstruction algorithm from the class of AMP algorithms [6] to obtain $\mathbf{s}^{\mathrm{MMSE}}$.
  - The reconstruction algorithm requires $\mathbf{q}$ to estimate $\mathbf{s}^{\mathrm{MMSE}}$. For this purpose, we use EM [7] to estimate the prior parameters $\mathbf{q}$ from $\mathbf{y}$.

Preliminary Results

- This is still an ongoing work but we will present some preliminary results in order to demonstrate the performance of the proposed algorithm.
- The benchmark technique [8] shown here uses a basic CS reconstruction technique (AMP algorithm [9]) for reconstruction of the gradients.
- The benchmark method does not take into consideration the spatial correlation property of the convolutional gradients during their reconstruction at the server.
- The parameters required for the FL procedure are
  - For simplicity, we assume a FL setting with 1 user i.e. $K = 1$, SR = 0.25, CR = 2, $\eta = 0.1$.
  - We trained a convolutional NN with 3 layers with dimensions $((H, W, D, F))$ $(5, 5, 5, 1)$, $(5, 5, 10, 5)$, $(5, 5, 20, 10)$ followed by a dropout layer, fully connected layer with 320 neurons and a softmax layer with 10 neurons using MNIST dataset for classification problem.

Preliminary Results

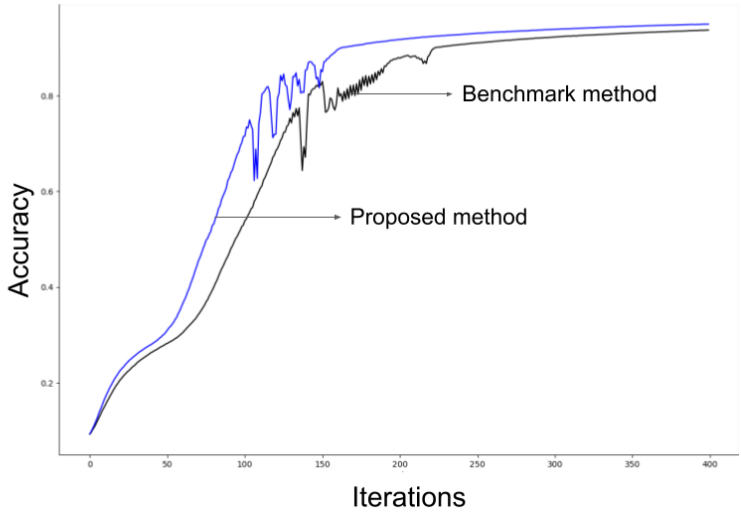

Figure: Test accuracy with iterations (MNIST dataset).

Summary

- In the last lecture of this course, we focused on a case study on distributed learning that is of specific interest to wireless networks.
- We specifically focused on the problem of gradient compression and reconstruction.
- We incorporated the sparsity and spatial correlation properties of the gradients through aprporpiately chosen priors.
- The introduction of this prior also meant that we need to learn its parameters, which is where the EM algorithm came into the picture.
- This is yet another example of how appropriate domain knowledge (in this case the properties of the gradients) could be exploited to develop efficient learning solutions.
- A preprint based on this work will be available on arXiv soon.

References

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

[2] A. F. Aji and K. Heafield, "Sparse communication for distributed gradient descent," arXiv preprint arXiv:1704.05021, 2017.

[3] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," in Sixteenth Annual Conference of the International Speech Communication Association, 2015.

[4] M. Yu, *et al.*, "GradiVeQ: Vector quantization for bandwidth-efficient gradient aggregation in distributed CNN training," in Advances in Neural Information Processing Systems, 2018, pp. 5123-5133.

[5] Y. C. Eldar and G. Kutyniok, "Compressed sensing: theory and applications, Cambridge University Press, 2012.

[6] J. P. Vila and P. Schniter, "Expectation-maximization gaussian-mixture approximate message passing," IEEE Trans. on Signal Process., vol. 61, no. 19, pp. 4658-4672, 2013.

References

[7] A. Dempster, N. M. Laird, and D. B. Rubin, "Maximum-likelihood from incomplete data via the EM algorithm," J. of the Roy. Stat. Soc., vol. 39, pp. 1-17, 1977.

[8] M. M. Amiri and D. Gunduz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," IEEE Trans. on Signal Process., vol. 68, pp. 2155-2169, 2020.

[9] D. L. Donoho, A. Maleki, and A. Montanari, "Message passing algorithms for compressed sensing I. Motivation and construction," in IEEE Info. Theory Workshop, 2010, pp. 1-5.

Thank You

Please contact Prof. Dhillon at hdhillon@vt.edu if you have any questions, comments or suggestions about any of the lectures of this short course.