Optimization Theory and Algorithm **Lecture 12 - 10/27/2021**

Lecture 12

Lecturer: Xiangyu Chang *Scribe: Xiangyu Chang*

Edited by: Junbo Hao

1 Federated Optimization

Federated learning (FL) enables a large amount of edge computing devices to jointly optimize (learn) a model without data sharing. FL has three unique characters that distinguish it from the standard parallel optimization.

- The training data are massively distributed over an incredibly large number of devices, and the connection between the central server and a device is slow.
- The FL system does not have control over user's device (stragglers).
- The training data are non-i.i.d.

Problem Formulation:

$$\min_{\mathbf{x}} \left\{ f(\mathbf{x}) = \sum_{k=1}^{K} p_k f_k(\mathbf{x}) \right\} \tag{1}$$

where K is the number of devices, and p_k is the weight of the kth device such that $p_k \geqslant 0$ and $\sum_k p_k = 1$. Suppose that kth device hold s m_k training data: $\mathbf{z}_{k,1}, \dots, \mathbf{z}_{k,m_k}$, then

$$f_k(\mathbf{x}) = \frac{1}{m_k} \sum_{j=1}^{m_k} \ell(\mathbf{x}; \mathbf{z}_{k,j}).$$

Example 1.1. (Federated Least Squares Problem) Suppose that we have K banks, they would like to jointly to train a model to predict the customer's income for "user profile" or to train a score system to estimate their financial credit (see Figure 1). They adopt a linear regression model, then

$$\min_{\mathbf{x}} \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|^2 = \frac{1}{2} \sum_{k=1}^{K} \|A_k\mathbf{x} - \mathbf{b}_k\|^2,$$

where

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_K \end{bmatrix} \in \mathbb{R}^{m\times n}, \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_K \end{bmatrix} \in \mathbb{R}^{m}.$$

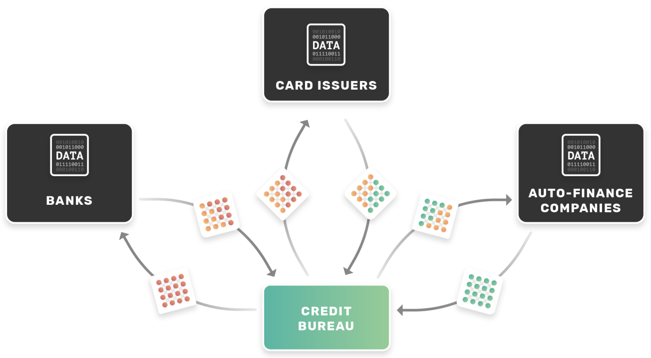

Figure 1: Federated Learning for Credit Scoring

However, we cannot combine the personal data set together due to the sensitive information and law regulations (E.g., GDPR). Then the idea is to transmit *some information* to a central server without sharing any dataset.

For the kth bank, it considers

$$\min_{\mathbf{x}} \frac{1}{2}\|A_k\mathbf{x} - \mathbf{b}_k\|^2.$$

Denote an operator $G_k(\mathbf{x}) = \mathbf{x} - s\nabla_{\mathbf{x}}(\frac{1}{2}\|A_k\mathbf{x} - \mathbf{b}_k\|^2) = (I - sA_k^\top A_k)\mathbf{x} + sA_k^\top \mathbf{b}_k$. The federated gradient descent algorithm is

$$\text{Step 1: } \mathbf{x}_k^{t+1/2} := G_k^E(\mathbf{x}_k^t), \tag{2}$$

$$\text{Step 2: } \mathbf{x}_k^{t+1} := \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k^{t+1/2}, \tag{3}$$

where $G_k^E(\mathbf{x})$ means that runs GD on the kth device E times.

First, let us try to compute $G_k^2(\mathbf{x})$ as

$$\begin{aligned} G_k^2(\mathbf{x}) &= G_k(G_k(\mathbf{x})) = G_k((I - sA_k^\top A_k)\mathbf{x} + sA_k^\top \mathbf{b}_k) \\ &= (I - sA_k^\top A_k)((I - sA_k^\top A_k)\mathbf{x} + sA_k^\top \mathbf{b}_k) + sA_k^\top \mathbf{b}_k \\ &= (I - sA_k^\top A_k)^2\mathbf{x} + s[I + (I - sA_k^\top A_k)]A_k^\top \mathbf{b}_k. \end{aligned}$$

By induction, you can obtain that

$$G_k^E(\mathbf{x}) = (I - sA_k^\top A_k)^E\mathbf{x} + s[\sum_{e=0}^{E-1}(I - sA_k^\top A_k)^e]A_k^\top \mathbf{b}_k. \tag{4}$$

Thus,

$$\begin{aligned} \mathbf{x}^{t+1} &= \bar{\mathbf{x}}^{t+1/2} = \frac{1}{K}\sum_k \mathbf{x}_k^{t+1/2} = \frac{1}{K}\sum_k G_k^E(\mathbf{x}_k^t) \\ &= \frac{1}{K}\sum_k G_k^E(\mathbf{x}^t) = \frac{1}{K}[\sum_{k=1}^{K}(I - sA_k^\top A_k)^E]\mathbf{x}^t + \frac{s}{K}\sum_{k=1}^{K}\{[\sum_{e=0}^{E-1}(I - sA_k^\top A_k)^e]A_k^\top \mathbf{b}_k\}. \end{aligned}$$

That is $\mathbf{x}^{t+1} = B\mathbf{x}^t + C$, where

$$B = \frac{1}{K}\sum_k G_k^E(\mathbf{x}^t) = \frac{1}{K}[\sum_{k=1}^K (I - sA_k^\top A_k)^E]$$

and

$$C = \frac{s}{K}\sum_{k=1}^K \{[\sum_{e=0}^{E-1}(I - sA_k^\top A_k)^e]A_k^\top \mathbf{b}_k\}.$$

We konw that

$$\begin{aligned}\mathbf{x}^{t+1} &= B^{t+1}\mathbf{x}^0 + (I + B + \cdots + B^t)C \\ &= B^{t+1}\mathbf{x}^0 + (I - B)^{-1}(I - B^{t+1})C.\end{aligned}$$

So,

$$\mathbf{x}^*_{FGD} = \lim_{t\to\infty} \mathbf{x}^t = (I - B)^{-1}C.$$

1.1 FedAvg and Local SGD

FedAvg algorithm is proposed by [MMR+17] for training deep models distributed and efficiently.

Algorithm 1 Local Stochastic Gradient Descent

```
1: Input: Assumes that $K$ clients index by $k$, $B$ is the local mini-batch number, $E$ is the number of local epochs, $\eta$ is the learning rate $\mathbf{x}^0 \in \mathbb{R}^n$, and $t = 0$.
2: for $t = 0, 1, \dots, T$ do
3:     for $k \in 1, \dots, K$ do
4:         In parallel
               $$\mathbf{x}_k^{t+1} \leftarrow \mathbf{x}_k^t - \eta\nabla f_k(\mathbf{x}_k^t; \mathbf{z}_B),$$
           where $\mathbf{z}_B$ is the batch samples with size $B$.
5:     end for
6:     $\mathbf{x}^{t+1} \leftarrow \sum_{k=1}^K \frac{m_k}{m}\mathbf{x}_k^{t+1}$ and $t := t + 1$.
7: end for
8: Output: $\mathbf{x}^T$.
```

Let us summary the local SGD algorithm as follows:

- Local Update:

$$\mathbf{x}_k^{t+i+1} \leftarrow \mathbf{x}_k^{t+i} - s_{t+i}\nabla f_k(\mathbf{x}_k^{t+i}, \xi_k^{t+i}), i = 0, \dots, E-1,$$

  where ξ_k^{t+i} is a sample uniformly chosen from the local data and s_{t+i} is the learning rate.

- Server Update by Aggregation:

$$\mathbf{x}^{t+E} \leftarrow \sum_{k=1}^K p_k\mathbf{x}_k^{t+E}.$$

- Update Local Parameter:
$$\mathbf{x}_k^{t+E} \leftarrow \mathbf{x}^{t+E}, \forall k = 1, \ldots, K.$$

Let T be the total interactions, then $[2T/E]$ is the communication number.

2 Block Coordinate Descent

2.1 Motivation

Let us recall the SGD, SVRG, FedAvg. They all consider a big data setting. For example, the ERM problem (finite-summation optimization):

$$\min_{\mathbf{x}} \frac{1}{m} \sum_{i=1}^{m} f(\mathbf{x}; \mathbf{z}_i).$$

Using GD,

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \frac{s_t}{m} \sum_{i=1}^{m} \nabla f(\mathbf{x}^t, \mathbf{z}_i).$$

The summation is so big due to the big data setting. The basic idea is the “sample decomposition”. SGD and FedAvg are the main instants.

Another type problem is sightly different, which involves so many decision variables in stead of the sample size called “high-dimensional problems”.

Example 2.1. For the least squares problem,

$$\mathbf{x}^* \in \arg\min_{\mathbf{x}} \frac{1}{2} \|A\mathbf{x} - \mathbf{b}\|^2,$$

where $\mathbf{x}^* = (A^\top A)^{-1} A^\top \mathbf{b}$. Computing $(A^\top A)^{-1}$, we need $O(n^3)$ which is determined by the number of decision. variables. Even if we can use GD,

$$\mathbf{x}^{t+1} = (I - s_t A^\top A)\mathbf{x}^t + s_t A^\top \mathbf{b}.$$

Computing $A^\top A$ of each iterative step need $O(n^2 m)$ which is prohibited for large n.

Example 2.2. Let us consider LASSO problem again.

$$\min_{\mathbf{x}} \frac{1}{2} \|A\mathbf{x} - \mathbf{b}\|^2 + \lambda \|\mathbf{x}\|_1 = \frac{1}{2} \|A\mathbf{x} - \mathbf{b}\|^2 + \sum_{j=1}^{n} |x_j|.$$

In this model, maybe n is so large.

General Formulation:

$$\min_{\mathbf{x}} f(\mathbf{x}) = f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K) + \sum_{k=1}^{K} r_k(\mathbf{x}_k), \tag{5}$$

where $\mathbf{x}_k \in \mathbb{R}^{n_k}$ and $\sum_k n_k = n$ which means the decision variables are decomposed into K groups. If $K = n$, then $\mathbf{x}_k \in \mathbb{R}$. In this part, we assume that $f \in C^1$ and $r_k, k \in [K]$ are convex.

2.2 BCD

Suppose that we have an iterative algorithm to obtain $\mathbf{x}^t$, then define that

$$f_k^t(\mathbf{x}_k) := f(\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_{k-1}^t, \mathbf{x}_k, \mathbf{x}_{k+1}^t, \ldots, \mathbf{x}_K^t). \tag{6}$$

Solve a sub-optimization problem of Eq.(5).

(i) $\mathbf{x}_k^{t+1} = \arg\min_{\mathbf{x}_k} \{ f_k^t(\mathbf{x}_k) + r_k(\mathbf{x}_k) \}$.

(ii) $\mathbf{x}_k^{t+1} = \arg\min_{\mathbf{x}_k} \{ f_k^t(\mathbf{x}_k) + \frac{1}{2}\|\mathbf{x}_k - \mathbf{x}_k^t\|^2 + r_k(\mathbf{x}_k) \}$.

(iii) $\mathbf{x}_k^{t+1} = \arg\min_{\mathbf{x}_k} \{ \frac{1}{2}\|\mathbf{x}_k - \mathbf{x}_k^t\|^2 + \langle \nabla f_k^t(\mathbf{x}_k), \mathbf{x}_k - \mathbf{x}_k^t \rangle + r_k(\mathbf{x}_k) \}$.

Using item $\frac{1}{2}\|\mathbf{x}_k - \mathbf{x}_k^t\|^2$ is to control the $\mathbf{x}^{t+1}$ is not far away from $\mathbf{x}_k^t$ in a certain sense.

References

[MMR$^+$17] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.