Report

Project	**Convergence guarantees for gradient descent methods on optimization for non-convex function approximation**
Student	**Hoang Trung Hieu**
Supervisor	**Horváth Tomáš**
Program	MSc, Applied Mathematics
Semester	2020/2021, Autumn

Summary

Process and Results

-

Future work

References

[1] F. Sattler, S. Wiedemann, K. Müller, W. Samek, (2019), *Robust and communication-efficient federated learning from non-iid data*,IEEE transactions on neural networks and learning systems,

[2] H. Brendan McMahan, Eider Moore, Daniel Ramage, (2017) *Communication-Efficient Learning of Deep Networks from Decentralized Data*, https://arxiv.org/abs/1602.05629

[3] *TensorFlow Federated*,https://github.com/tensorflow/federated

[4] *Advanced Machine Learning Systems Course* , (2017), https://www.cs.cornell.edu/courses/cs6787/2017fa/

[5] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, (2019) *On the Convergence of FedAvg on Non-IID Data*, https://arxiv.org/abs/1907.02189

Non-convex optimization in federated learning

Hoang Trung Hieu

2020

Abstract

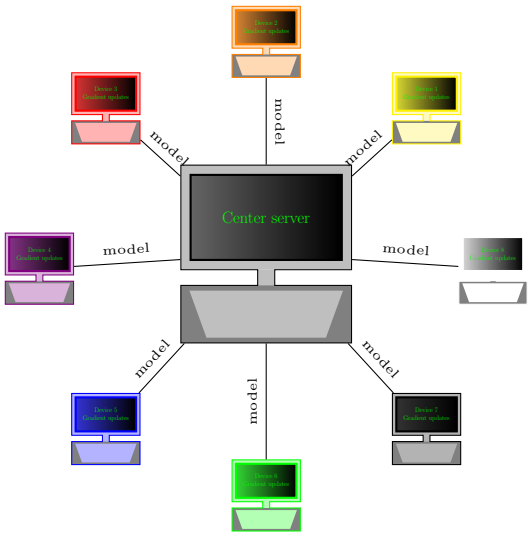
abstract-text

1 Introduction

1.1 Federated Learning

A new approach to train the model without access to the user's data is embedded machine learning. It can be described by two criteria: data never leaves the local devices and updates are communicated instead model. There are a few underlying distributed structure: peer-to-peer learning, distributed training, on-device inference and federated learning ([1]). In this project, we consider the new paradigm in Machine Learning-Federated Learning which is a machine learning setting where multiple entities collaborate in solving a learning problem, without directly exchanging data. The Federated training process is coordinated by a central server. It works like this: your device downloads the current model, improves it by learning from data on your phone, and then summarizes the changes as a small focused update. Only this update to the model is sent to the cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model. All the training data remains on your device, and no individual updates are stored in the cloud.
On the other hand, we face up to new challenges:

- *Communication.* Total communication can be calculated by the multiplication of the number of communication rounds by the number of parameters by average codeword length.It can be up to hundreds terabyte excluding the upload and download progress, but we do not mention it in this project.
- *Massively distributed.* The number of mobile device owners is massively bigger than average of the number of training samples on each device.
- *Unbalanced.* Some users produce significantly more data than others.
- *Non-IID.* The data generated by each user are quite different.



Figure[1]. Federated Leaning structure

1.2 Problem description

Let us consider the following distributed optimization model in which N clients cooperatively conquer

$$\min_w f(w) = \min_w \sum_{k=1}^{N} p_k f_k(w)$$

where p_k be the weight of $k-$th device and typically it is the proportion of the local data volume in global data volume.i.e $p_k \geq 0, \quad \sum_{i=1}^{N} p_k = 1$. The $k-$th client has n_k training data which is denoted $x_{k,1}, \cdots, x_{k,n_k}$. The local generalization error function are defined by

$$f_k(w) = \frac{1}{n_k} \sum_{j=1}^{n_k} \ell(w; x_{k,j})$$

where ℓ is a user loss function of the prediction.
The local generalization also can be defined by taking the expected value of training error of model parameters w over local data $\xi_k \sim \mathcal{D}_k$

$$f_k(w) = \mathbb{E}_{\xi_k \sim \mathcal{D}_k} F(w; \xi_k)$$

In case of the data is balanced (the volume of each device's data are equal)

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} \tilde{f}_k(w) \quad \text{where} \quad \tilde{f}_k(w) = p_k N f_k(w)$$

2 Algorithms

There are two most widely-used FL mechanisms, FedSGD and FedAvg. In general, FedSGD is barely influenced by the none independent and identically distributed (non-IID) data problem, but FedAvg suffers from a decline in accuracy. On the other hand, FedAvg is more efficient than FedSGD regarding time consumption and communication.

2.1 FedSGD

Deep learning training mainly relies on variants of stochastic gradient descent, where gradients are computed on a random subset of the total dataset and then used to make one step of the gradient descent.

Federated stochastic gradient descent[13] is the direct transposition of this algorithm to the federated setting, but by using a random fraction C of the nodes and using all the data on this node. The gradients are averaged by the server proportionally to the number of training samples on each node, and used to make a gradient descent step.

2.2 FedAvg

Algorithm 1: FederatedAveraging(FedAvg).

Server executes:
initialization w_0 ;
for *each round* $t = 1, 2, \cdots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m client)
  for *each client* $k \in S_t$ **do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$;
  end
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$;
end
ClientUpdate(k, w)**:**
$B \leftarrow$ (split P_k into batches of size B)
for *each local epoch* **do**
  for *batch* $b \in B$ **do**
    $w \leftarrow w - \eta \nabla l(w; b)$
  end
end

Federated averaging (FedAvg) is a generalization of FedSGD, which allows local nodes to perform more than one batch update on local data and exchanges the updated weights rather than the gradients. The rationale behind this generalization is that in FedSGD, if all local nodes start from the same initialization, averaging the gradients is strictly equivalent to averaging the weights them-

selves. Further, averaging tuned weights coming from the same initialization does not necessarily hurt the resulting averaged model's performance.

3 Non-Convex optimization

Machine learning algorithms use optimization all the time. We minimize loss, or error, or maximize some kind of score functions. Gradient descent is the "hello world" optimization algorithm covered on probably any machine learning course. It is obvious in the case of regression, or classification models, but even with tasks such as clustering we are looking for a solution that optimally fits our data (e.g. k-means minimizes the within-cluster sum of squares). So if you want to understand how the machine learning algorithms do work, learning more about optimization helps. Moreover, if you need to do things like hyperparameter tuning, then you are also directly using optimization.

One could argue that convex optimization shouldn't be that interesting for machine learning since instead of dealing with convex functions, we often face up to the composition of convex function which usually is not convex. For examples, matrix completition, principle component analysis, low-rank models and tensor decomposition, maximum likelihood estimation with hidden variables and the most important, deep neural networks.
In order to find convergence guarantees for non-convex function(even convex) is quite hard or unrealistic in deep learning. But there are a few assumptions that appears frequently in recent papers.

Assumption 1. $L-$ **Lipschitz gradient**: Local objective functions f_i are all $L-$ smooth:

$$\|\nabla f_i(u) - \nabla f_i(v)\| \leq L \|u - v\|$$

Or it can be rewritten:

$$f_i(u) \leq f_i(v) + (u - v)^T \nabla f_i(v) + \frac{L}{2}\|u - v\|^2$$

Assumption 2. Bounded variance

$$\mathbb{E}_{\xi_k \sim D_i}\left[\|\nabla\|^2\right] \leq^2$$

Assumption 3. Bounded gradient

$$\mathbb{E}_{\xi_k \sim D_i}\left[\|\nabla\|^2\right] \leq^2$$

Assumption 4. Bounded variance

$$\mathbb{E}$$

Besides, to analysing a limitation of FL, other limitations usually are not counted. In []When the gradients of the loss function over our function approximator (NN) are unbiased, there are already convergence analyses given. In FL however, local gradients are computed over different data distributions and the updates aggregated from them are biased, which makes this analysis very hard. (To the best of our knowledge, nobody gave any meaningful results for this problem)

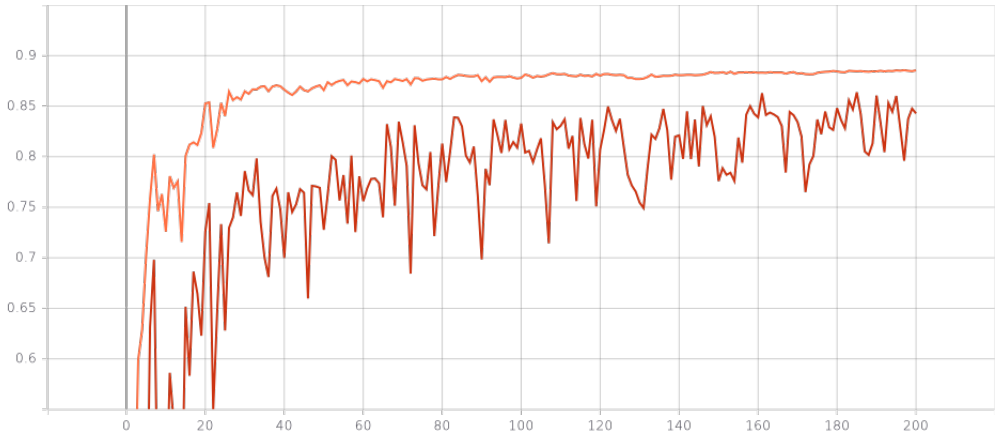
4 Experimental results

In most cases, the assumption of independent and identically distributed samples across local nodes does not hold for federated learning setups. Under this setting, the performances of the training process may vary significantly according to the unbalancedness of local data samples as well as the particular probability distribution of the training examples (i.e., features and labels) stored at the local nodes. To further investigate the effects of non-iid data, the following description considers the main categories presented in the by Peter Kiarouz and al. in 2019.

Federated learning parameters Once the topology of the node network is chosen, one can control different parameters of the federated learning process (in opposition to the machine learning model's own hyperparameters) to optimize learning:

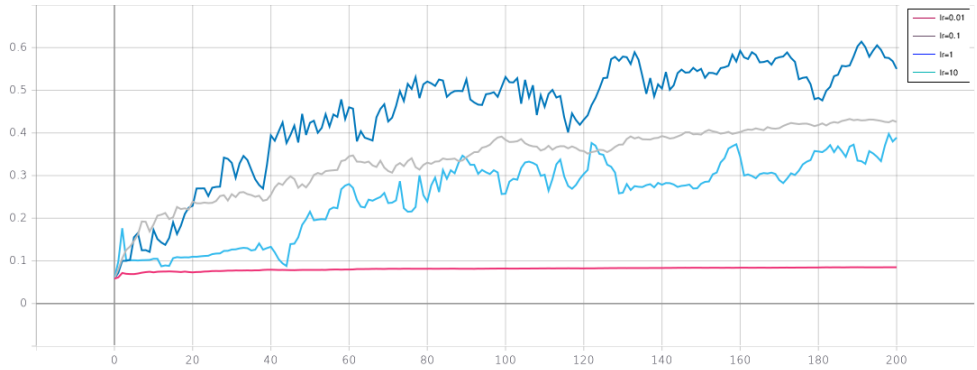
Number of federated learning rounds:T
Total number of nodes used in the process: K
Fraction of nodes used at each iteration for each node: C
Local batch size used at each learning iteration: B
Other model-dependent parameters can also be tinkered with, such as:
Number of iterations for local training before pooling: N Local learning rate: η
The dataset we choose to examine is MNIST dataset. To simulate the non-IID data distribution by setting each client to hold only digits from one certain class and the number of clients from the same class to be N=10. To simulate the data unbalance, we let the number of samples on each client roughly follow a normal distribution with mean and variance .

Logistic model and 2NN



Figure[2]. Logistic model and 2 hidden layers NN model

Impact of parameters



Figure[3]. Logistic model and 2 hidden layers NN model

Impact of balance

References

[1] F. Sattler, S. Wiedemann, K. Müller, W. Samek, (2019), *Robust and communication-efficient federated learning from non-iid data*,IEEE transactions on neural networks and learning systems,

[2] H. Brendan McMahan, Eider Moore, Daniel Ramage, (2017) *Communication-Efficient Learning of Deep Networks from Decentralized Data*, https://arxiv.org/abs/1602.05629

[3] *TensorFlow Federated*,https://github.com/tensorflow/federated

[4] *Advanced Machine Learning Systems Course* , (2017), https://www.cs.cornell.edu/courses/cs6787/2017fa/

[5] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, (2019) *On the Convergence of FedAvg on Non-IID Data*, https://arxiv.org/abs/1907.02189

Apendix

Which cases we get poor-quality results when performing the classical FedAvg?

Theorem 1.