Knowledge Transfer for Continual Federated Learning

An Bian
Hosted by Lukas Zilka and Jacek Jurewicz

October 12, 2018

Abstract

Federated learning (FL) enables learning a global model based on the participation of many clients (e.g., mobile phones), without the need to collect and store the data in the cloud. However, two practical problems naturally arise: 1, The data distributions on the clients are usually non-IID, which renders one single global model insufficient to capture the diversity of clients; 2, The data distribution on one specific client may change along the time in a continual setting, which requires new techniques to deal with the resulted distribution drift.

To tackle these issues, we firstly formulate it to be a new problem setting, termed "Continual Federated Learning (CFL)". In order to investigate new algorithms for CFL, we build a simulation system using low-level Tensorflow APIs. It has a great flexibility for testing new federated algorithms and plugging in different models. Then we design algorithms for the CFL setting by incorporating modern knowledge transfer techniques.

We customize the Reddit dataset to simulate the CFL setting and investigate performances of these algorithms on a LSTM next word prediction model. Extensive experimental results show that the proposed algorithms outperform the baseline. Furthermore, we observe intriguing behaviors of the proposed algorithms under different model architectures and algorithmic hyperparameters.

1. Introduction

1.1 Knowledge transfer for language models.

RNN-LSTM language model. RNN-LSTM (Hochreiter & Schmidhuber, 1997) is a versatile model that has been extensively used for next-word prediction (Mikolov et al., 2010), speech recognition (Graves et al., 2013), machine translation and so on.

Knowledge transfer. Knowledge transfer for RNN-LSTM has only been considered in the centralized setting (Mou et al., 2016; Yoon et al., 2017). The setting of transfer learning (aka. domain adaption) supposes there are two domains: the source domain $\mathcal{S}$ and the target domain $\mathcal{T}$, it aims to transfer knowledge from the source

domain to the target domain. One usually learns a model $M_{\mathcal{S}}$ from the $\mathcal{S}$, then: 1, Relearn the target model $M_{\mathcal{T}}$ using $M_{\mathcal{S}}$ as the initialization; 2, Adding "surplus" layers to $M_{\mathcal{S}}$, then train the surplus layers in order to obtain the target model; 3, Retrain part of the $M_{\mathcal{S}}$ in order to generate the target model.

1.2 Federated/Decentralized Learning

Decentralized data in federated learning. Federated learning (McMahan et al., 2017) is a new learning paragradim motivated by "on-device intelligence". It tries to learn a global model based on the participation of a massive number of clients, which generate *decentralized* data.

However, federated optimization is still a centralized algorithm (like FedAvg), each round it needs to communicate with the central server. This could make the whole system very sensitive to: 1) the reliability of the central server 2) the traffic jam in communicating with the central parameter server.

So a fully decentralized algorithm (like decentralized CoCoA) could be a much better fit. A decentralized algorithm basically does the following in each round (take first-order algorithm for example): i) each client calculates the gradient locally ii) each client updates its local model iii) each client fetches models of its neighbors and takes an average.

1.3 Problem Setup

Notation. We use boldface letters, e.g., $\mathbf{x}$ to represent a vector, boldface captical letters, e.g., $\mathbf{A}$ to denote a matrix. x_i is the i^{th} entry of $\mathbf{x}$, A_{ij} is the $(ij)^{\text{th}}$ entry of $\mathbf{A}$. We use $\mathbf{e}_i$ to denote the standard i^{th} basis vector. $[n] := \{1, ..., n\}$ for an integer $n \geq 1$. $\|\cdot\|$ means the Euclidean norm by default.

Suppose there are K clients, each one has n_k data points/samples, $k = 1, ..., K$. So there are in total $n = \sum_{k=1}^{K} n_k$ data points. The overall loss function is

$$f(\mathbf{w}) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(\mathbf{w}), \; F_k(\mathbf{w}) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(\mathbf{w}). \tag{1}$$

For a LSTM next word prediction model, $\mathbf{w}$ contains the word embeddings, parameters for the LSTM hidden layers and parameters for the LSTM final layer (softmax layer).

Assume there exists a split of model parameters $\mathbf{w} \in \mathbb{R}^d$,

$$\mathbf{w} = [\mathbf{w}^{\text{shared}}; \mathbf{w}^{\text{personal}}] \tag{2}$$

where $\mathbf{w}^{\text{shared}}$ can capture the shared knowledge and $\mathbf{w}^{\text{personal}}$ can capture the personal knowledge for a specific client. To make it more clear, we explicitly write $\mathbf{w}^{\text{all}}$ to denote all parameters in the seqel.

Continual Federated Learning (CFL). Two main features define the CFL setting: 1, The data distributions on the clients are highly non-IID,

2, The data distribution on one specific client may change along the time in a continual setting,

which requires new techniques to deal with the resulted distribution drift.

1.4 The Building Block: FedAvg

We use $\rho \in [0, 1]$ to be the fraction of clients that are selected in each round of FedAvg. The peusocode is summarized in line 1.

0: FedAvg

Server:
initialize server parameter to be $\mathbf{w}_0$;
for *each round* $r = 0, 1, 2, ...$ **do**
  $S_r \leftarrow$ a random subset of clients ;
  for *each client* $k \in S_r$ *in parallel* **do**
    $\mathbf{w}_{r+1}^k \leftarrow \text{ClientUpdate}(k, \mathbf{w}_r)$
  $\mathbf{w}_{r+1} \leftarrow \sum_{k \in \mathcal{S}_r} \frac{n_k}{\sum_{k \in \mathcal{S}_r} n_k} \mathbf{w}_{r+1}^k$
ClientUpdate$(k, \mathbf{w})$:
  Train client k for E local epochs.

1.5 Related Work

Decentralized optimization. decentralized SGD in Lian et al. (2017).
decentralized SAG: Mokhtari & Ribeiro (2016)
Decentralized Gradient descent : Yuan et al. (2016)

Federated optimization. This is a new setting first studied by Google, mainly motivated by the setting of "on-device machine learning", c.f. McMahan et al. (2017).

Centralized optimization. This class of algorithms needs a centralized parameter server, some representatives are centralized CoCoA-style algorithms (Smith et al., 2016; Jaggi et al., 2014), centralized parallel SGD such as (Agarwal & Duchi, 2011; Zinkevich et al., 2010) and a lot of others.

2. Models and Algorithms

2.1 Baseline: FedAvg($\mathbf{w}^{\text{all}}$)

In each cycle, run FedAvg for several passes of all clients statistically.

1: BASELINE: FEDAVG($\mathbf{w}^{\text{all}}$)

Cycle 0:
  Run FEDAVG on all parameters $\mathbf{w}^{\text{all}}$;
Cycle t, $t = 1, 2, ...$:
  (Optional) Update server parameters to be the average of clients' parameters;
  Run FEDAVG on all parameters $\mathbf{w}^{\text{all}}$;

2.2 Algorithm 0: FEDAVG($\mathbf{w}^{\text{all}}$) + RETRAINING($\mathbf{w}^{\text{all}}$)

2: ALG0: FEDAVG($\mathbf{w}^{\text{all}}$) + RETRAINING($\mathbf{w}^{\text{all}}$)

Cycle 0:
  Run FEDAVG on all parameters $\mathbf{w}^{\text{all}}$;
  Retrain each client on all parameters $\mathbf{w}^{\text{all}}$;
Cycle t, $t = 1, 2, ...$:
  Update server parameters to be the average of clients' parameters;
  Run FEDAVG on all parameters $\mathbf{w}^{\text{all}}$;
  Retrain each client on all parameters $\mathbf{w}^{\text{all}}$;

2.3 Algorithm 1: FEDAVG($\mathbf{w}^{\text{all}}$) + RETRAINING($\mathbf{w}^{\text{personal}}$)

3: ALG1: FEDAVG($\mathbf{w}^{\text{all}}$) + RETRAINING($\mathbf{w}^{\text{personal}}$)

Cycle 0:
  Run FEDAVG on all parameters $\mathbf{w}^{\text{all}}$;
  Retrain each client on personal parameters $\mathbf{w}^{\text{personal}}$;
Cycle t, $t = 1, 2, ...$:
  Update server parameters to be the average of clients' parameters;
  Run FEDAVG on all parameters $\mathbf{w}^{\text{all}}$;
  Retrain each client on *personal* parameters $\mathbf{w}^{\text{personal}}$;

2.4 Algorithm 2: FEDAVG($\mathbf{w}^{\text{shared}}$) + RETRAINING($\mathbf{w}^{\text{personal}}$)

Advantanges:

less computation for FEDAVG
less communication, since only sending the shared parameters $\mathbf{w}^{\text{shared}}$
promising performance on real-world datasets.

4: ALG2: FEDAVG($\mathbf{w}^{\text{shared}}$) + RETRAINING($\mathbf{w}^{\text{personal}}$)

Cycle 0:
  Run FEDAVG on all parameters $\mathbf{w}^{\text{all}}$;
  Retrain each client on personal parameters $\mathbf{w}^{\text{personal}}$;
Cycle t, $t = 1, 2, \ldots$:
  Update server *shared* parameters to be the average of clients' shared parameters;
  Run ALTERNATIVEMINIMIZATION for m rounds, in each round:
    Run FEDAVG on shared parameters $\mathbf{w}^{\text{shared}}$;
    Retrain each client on persoal parameters $\mathbf{w}^{\text{personal}}$;

3. Experiments

3.1 Dataset

In order to simulate the environment of continual federated learning, we made a dataset from the public reddit dataset. It contains the comments from reddit users.

To simulate the timeline, we took 10 months' data ranging from January 2015 to October 2015. We were trying to take comments from one reddit user to be the data of one client, however, the text posted by a normal user is usually so little that one cannot use it to train a reasonable LSTM model.

We turned to use the text from one "subreddit" to simulate a "super user". According to redditmetrics[1], there are 1,209,738 subreddits on reddit up to October 2018. One benefit of this way is that each subreddit has a unique data distribution, which is also interpretable, e.g., we know that the text from subreddit "science" is very possibly different from that from the subreddit "joke".

We used comments from 20 subreddits to simulate 20 super users, they are: science, funny, sports, worldnews, pics, gaming, videos, movies, Music, blog, gifs, explainlikeimfive, books, television, EarthPorn, DIY, food, Documentaries, history and InternetIsBeautiful.

3.2 Model and Algorithm Configurations

We implemented LSTM models with one hidden layer. We tried with three model size: small, medium and large. Details of them are summarized in Table 1.

4. Discusions and Conclusions

1. http://redditmetrics.com/

Table 1: Architectures of tested LSTM models.

Model Size	Embedding size	Hidden layer size	# unrolling	# parameters
small	100	100	20	
medium	200	200	30	
large	600	600	35	

References

Agarwal, Alekh and Duchi, John C. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pp. 873–881, 2011.

Graves, Alex, Mohamed, Abdel-rahman, and Hinton, Geoffrey. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649. IEEE, 2013.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jaggi, Martin, Smith, Virginia, Takác, Martin, Terhorst, Jonathan, Krishnan, Sanjay, Hofmann, Thomas, and Jordan, Michael I. Communication-efficient distributed dual coordinate ascent. In *Advances in Neural Information Processing Systems*, pp. 3068–3076, 2014.

Lian, Xiangru, Zhang, Ce, Zhang, Huan, Hsieh, Cho-Jui, Zhang, Wei, and Liu, Ji. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 5336–5346, 2017.

McMahan, Brendan, Moore, Eider, Ramage, Daniel, Hampson, Seth, and y Arcas, Blaise Aguera. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pp. 1273–1282, 2017.

Mikolov, Tomaš, Karafiat, Martin, Burget, Lukaš, Černockỳ, Jan, and Khudanpur, Sanjeev. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

Mokhtari, Aryan and Ribeiro, Alejandro. Dsa: Decentralized double stochastic averaging gradient algorithm. *Journal of Machine Learning Research*, 17(61):1–35, 2016.

Mou, Lili, Meng, Zhao, Yan, Rui, Li, Ge, Xu, Yan, Zhang, Lu, and Jin, Zhi. How transferable are neural networks in nlp applications? *arXiv preprint arXiv:1603.06111*, 2016.

Smith, Virginia, Forte, Simone, Ma, Chenxin, Takác, Martin, Jordan, Michael I, and Jaggi, Martin. Cocoa: A general framework for communication-efficient distributed optimization. *arXiv preprint arXiv:1611.02189*, 2016.

Yoon, Seunghyun, Yun, Hyeongu, Kim, Yuna, Park, Gyu-tae, and Jung, Kyomin. Efficient transfer learning schemes for personalized language modeling using recurrent neural network. *arXiv preprint arXiv:1701.03578*, 2017.

Yuan, Kun, Ling, Qing, and Yin, Wotao. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

Zinkevich, Martin, Weimer, Markus, Li, Lihong, and Smola, Alex J. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pp. 2595–2603, 2010.