Communication-Efficient Learning of Deep Networks from Decentralized Data

H. Brendan McMahan Eider Moore Daniel Ramage Seth Hampson Blaise Ag uera y Arcas

Google, Inc., 651 N 34th St., Seattle, WA 98103 USA

INTRODUCTION

1. Introduction of FederatedAveraging(or FedAvg) methodology for the federated learning environment.

2. To show the number of communication rounds needed by FedAvg to reach a test accuracy of 97 % and 99% on the MNIST dataset using two different models is less in comparison to done using a non-distributed set up.

MLP MODEL

2-hidden layers with 200 units each using ReLu activations (199,210 total parameters) this is referred as the MNIST 2NN.

Our initial study includes three model families on two datasets. The first two are for the MNIST digit recognition task [26]: 1) A simple multilayer-perceptron with 2-hidden layers with 200 units each using ReLu activations (199,210 total parameters), which we refer to as the MNIST 2NN.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 200)               157000

dense_1 (Dense)              (None, 200)               40200

dense_2 (Dense)              (None, 10)                2010

=================================================================
Total params: 199,210
Trainable params: 199,210
Non-trainable params: 0
_________________________________________________________________
```

Conclusion : Parameters matched

CNN model

1. A CNN with two 5x5 convolution layers
   first with 32 channels,
   second with 64 channels
   each followed with 2x2 max pooling
   fully-connected layer with 512 units and ReLu activation, and a softmax output layer
   (1,663,370 total parameters)

2) A CNN with two 5x5 convolution layers (the first with 32 channels, the second with 64, each followed with 2x2 max pooling), a fully connected layer with 512 units and ReLu activation, and a final softmax output layer (1,663,370 total parameters). To study federated optimization, we also

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        832

max_pooling2d (MaxPooling2D  (None, 14, 14, 32)        0
)

conv2d_1 (Conv2D)            (None, 14, 14, 64)        51264

max_pooling2d_1 (MaxPooling  (None, 7, 7, 64)          0
2D)

flatten (Flatten)            (None, 3136)              0

dense (Dense)                (None, 512)               1606144

dense_1 (Dense)              (None, 10)                5130

=================================================================
Total params: 1,663,370
Trainable params: 1,663,370
Non-trainable params: 0
_________________________________________________________________
```

Conclusion : Parameters matched

Hyperparameters

1. C: Fraction of clients that perform computation on each round
2. E: Number of training passes each client makes over its local dataset on each round
3. B: Local minibatch size used for the client updates.
4. Learning rate.

Hyperparameter values taken for the experiment :

1. C : 0 (all the clients taken into account)
2. E : 1 for 2NN model and 5 for CNN model
3. B : 10
4. Learning rate = 0.01

IID DISTRIBUTION

Total number of training samples: 60,000
Total number of classes in the dataset: 10
Total number of clients: 100
Shuffle the dataset
Number of datapoints each client = data points /total number of c
= 60,000/100
= 600

clients: **IID**, where the data is shuffled, and then partitioned into 100 clients each receiving 600 examples, and **Non-IID**, where we first *sort* the data by digit label, divide it into 200 shards of size 300, and assign each of 100 clients 2 shards. This is a pathological non-IID partition of the data, as most clients will only have examples of two digits. Thus, this lets us explore the degree to which our algorithms will break on highly non-IID data. Both of these partitions are balanced,

```
def get_iid_dataset(trainX_id,trainY_id,model_name):
  client_count=100
  e=600
  s=0
  trainY_id=to_categorical(trainY_id)
  local_xy_id=[]
  for i in range(0,client_count):
   # print(i)
    local_x=trainX_id[s:e,:]
    local_y=trainY_id[s:e,:]
    #local_y= to_categorical(local_y)
    if(model_name=="CNN"):
       local_x = local_x.reshape((local_x.shape[0], 28, 28, 1))

    local_xy_id.append((local_x,local_y))
    s=e
    e+=600
  print("iid distribution of data done")
  return local_xy_id
```

Conclusion : set up matched

NON-IID DISTRIBUTION

clients: **IID**, where the data is shuffled, and then partitioned into 100 clients each receiving 600 examples, and **Non-IID**, where we first *sort* the data by digit label, divide it into 200 shards of size 300, and assign each of 100 clients 2 shards. This is a pathological non-IID partition of the data, as most clients will only have examples of two digits. Thus, this lets us explore the degree to which our algorithms will break on highly non-IID data. Both of these partitions are balanced,

- First, sort by label
- Create 200 shards of size 300
- Number of shards per client =2
- Number of data points per client =600 (2x300)

```
# number of shrads =200
# arrange the shrads by label in sorted manner
#L[i] is the set of shrads having the ith label

for i in range(0, 10):
  L[i]=S[i*20]...S[i*20+19]# each label has 20 shrads

for c in range(0:99) :
  Client[c]=Data(S[c]+S[c+100])# pairwise knowledge for each client ; client with knowledge of label i will have knowledge of label i+5
```

Conclusion : Ambiguity on data distribution

NON-IID SETUP

Original approach 

```
def get_nid_dataset(trainX_nid,trainY_nid,model_name):
  sorted_label_list=[]
  local_xy_nid=[]
  local_shrad_nid=[]
  for i in range(np.size((np.unique([trainY_nid])))) :
    _,label = np.where([trainY_nid==i])

    sorted_label_list.append(label)
  sorted_label= np.hstack(sorted_label_list)
  trainY_nid=trainY_nid[sorted_label]
  trainX_nid=trainX_nid[sorted_label]
  trainY_nid= to_categorical(trainY_nid)
  s=0
  client_count=100
  e=300# shrad size
  num_shrads=200
  for i in range(0,num_shrads):
    local_x=trainX_nid[s:e,:]
    local_y=trainY_nid[s:e,:]
    if model_name=="CNN":
     local_x = local_x.reshape((local_x.shape[0], 28, 28, 1))
    local_shrad_nid.append((local_x,local_y))
    s=e
    e+=300
  j=0
 # rand_index=np.random.shuffle(np.arrange(num_shrads))
  for i in range(0,num_shrads):

    local_x,local_y=local_shrad_nid[i]

    if(j>=100):
      j=j%100
      local_xy_nid[j].append((local_x,local_y))
    else:
      local_xy_nid.append((local_x,local_y))
    j=+1

  print("non-iid data distribution done ")
  return local_xy_nid
```

ALTERNATE IDEA FOR NON-IID SET UP

```
shrad_index=np.random(np.arrange(0,200)) #arrange and shuffle all the shrads
# get index of the shrads i.e S[0]...S[19] has index 0 and so on
client_tag=[] # has the index tag of each client
i=0

for j in range(0,100): # allocating one label to each client
  shrad_index=shrad_index_array[j]
  C[j]=S[shrad_index]
  tag= index_of(S[shrad_index])
  client_tag.append(tag)

P=[True for k in range(100,200)]# place holder for shrad's status ; whether allocated to any client or not

for j in range(100,200):# assiighning the second label to each client
  temp_j=j
  shrad_index=shrad_index_array[j]
  if(client_tag[i]==index_of(S[shrad_index])and P[shrad_index]==True):# to ensure none of the client has same label/index  shrads
    temp_j=+1
    shrad_index=teamp_j
  C[i]=S[shrad_index]
  p[shrad_index]=False # shrad taken by a cleint
  i=+1
```

Remark : yet to implement (pending)

ALGORITHM

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**
$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
$w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

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

Input: K (number of clients) Clients indexed by k, B, the minibatch size, E, the number of local epochs, and the learning rate

Process
Server-side:

1. Number of rounds required = number of rounds to reach the accuracy of 97% on the test set by the central model
2. m: It is the number of clients chosen
3. Each of the selected clients receives the weights of the central model, which they update by using their dataset.
4. Once the server receives all the updated weights from all the clients, the weighted aggregation of updated weights is done, and the averaged weight obtained is sent to the central model.

Client-side :

1. The dataset of each client divided into batches
2. For each batch, SGD run
3. The process iterates for E number of epochs

Output: Updated central model
As per our experiment, we are trying to replicate one result.
As the dataset with each client is balanced thus, the weight factor can be neglected.

RESULTS

Table 1: Effect of the client fraction C on the MNIST 2NN with $E = 1$ and CNN with $E = 5$. Note $C = 0.0$ corresponds to one client per round; since we use 100 clients for the MNIST data, the rows correspond to 1, 10 20, 50, and 100 clients. Each table entry gives the number of rounds of communication necessary to achieve a test-set accuracy of 97% for the 2NN and 99% for the CNN, along with the speedup relative to the $C = 0$ baseline. Five runs with the large batch size did not reach the target accuracy in the allowed time.

2NN	IID		NON-IID	
C	$B = \infty$	$B = 10$	$B = \infty$	$B = 10$
0.0	1455	316	4278	3275
0.1	1474 (1.0×)	87 (3.6×)	1796 (2.4×)	664 (4.9×)
0.2	1658 (0.9×)	77 (4.1×)	1528 (2.8×)	619 (5.3×)
0.5	— (—)	75 (4.2×)	— (—)	443 (7.4×)
1.0	— (—)	70 (4.5×)	— (—)	380 (8.6×)
CNN, $E = 5$				
0.0	387	50	1181	956
0.1	339 (1.1×)	18 (2.8×)	1100 (1.1×)	206 (4.6×)
0.2	337 (1.1×)	18 (2.8×)	978 (1.2×)	200 (4.8×)
0.5	164 (2.4×)	18 (2.8×)	1067 (1.1×)	261 (3.7×)
1.0	246 (1.6×)	16 (3.1×)	— (—)	97 (9.9×)

REMARK : The results to be replicated are highlighted

RESULT COMPARISON

RESULTS STATED BY THE PAPER

For 2NN

C=1,B=10, E=1
The number of communication rounds:
Iid - 70
Nid - 380

For CNN
C=1,B=10,E=5
The number of communication rounds:
Iid - 16
nid-97
Baseline test-set accuracy to achieve = 97% for 2NN and 99% for CNN

2NN	IID		Non-IID	
C	$B = \infty$	$B = 10$	$B = \infty$	$B = 10$
0.0	1455	316	4278	3275
0.1	1474 (1.0×)	87 (3.6×)	1796 (2.4×)	664 (4.9×)
0.2	1658 (0.9×)	77 (4.1×)	1528 (2.8×)	619 (5.3×)
0.5	— (—)	75 (4.2×)	— (—)	443 (7.4×)
1.0	— (—)	70 (4.5×)	— (—)	380 (8.6×)
CNN, $E = 5$				
0.0	387	50	1181	956
0.1	339 (1.1×)	18 (2.8×)	1100 (1.1×)	206 (4.6×)
0.2	337 (1.1×)	18 (2.8×)	978 (1.2×)	200 (4.8×)
0.5	164 (2.4×)	18 (2.8×)	1067 (1.1×)	261 (3.7×)
1.0	246 (1.6×)	16 (3.1×)	— (—)	97 (9.9×)

RESULTS OBTAINED

For 2NN

C=1,B=10, E=1

Iid - Accuracy achieved is **95% in 70 communication** rounds

Nid - **50 % accuracy in 380 rounds** of communication rounds

For CNN
C=1,B=10,E=5

Iid - **98 % accuracy in 16 rounds** of communication rounds

nid- **50 % accuracy in 90 rounds** of communication rounds

Conclusion : Results for iid is nearly same to that recorded in the paper but large difference in result for non- iid

CIFAR-10 Experiments Result

Accuracy	80%	82%	85%
SGD	6000	10000	N/A
FEDSGD	2000	3300	N/A
FEDAVG	280	N/A	N/A