Federated Machine Learning

Monday 9 September 2019 15:54

Idea:

- Example of medical images
- People may not want to share their imagery to make overall systems better
- So n number of people train a model on their side, using their images (identical models)
- And then do the following
    a. Google:
        - They send their models weights and biases to a central server, which averages it, sends it back and the end users train it again and this goes on for a few epochs
    b. Dereks idea
        - The people still send it to a central server, where weighted averages are done
        - Weights are based on similarities between people

More of a research approach

We will start with a dataset of sensor data about hand gestures and, as a way of beginning our ML journey, we will learn a single model (in traditional fashion) from all the data to predict which hand gesture is depicted. Then we will implement Google's federated learning, as above, on this data. Then, there are a number of possible extensions to the work. One is to implement some new ideas that I have had for averaging the models. Another is to move to more challenging datasets from the domain of human activity recognition, which will need use of more complex neural networks for the learning.

Paper

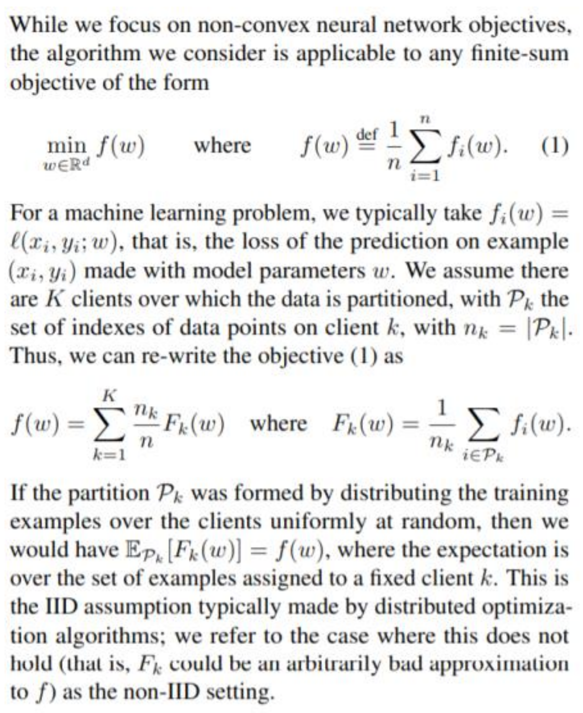
Sunday 29 September 2019 14:21

Don't need to know the sender to use their data, use tor
Optimisation of federated learning is related to optimisation in the distributed context

- How is it different/
  - Non iid
    - The training data on a given client is typically based on the usage of the mobile device by a particular user, and hence any particular user's local dataset will not be representative of the population distribution.
  - Unbalanced
    - Similarly, some users will make much heavier use of the service or app than others, leading to varying amounts of local training data
  - Massively distributed
    - We expect the number of clients participating in an optimization to be much larger than the average number of examples per client.
  - Limited communication M
    - Mobile devices are frequently offline or on slow or expensive connections.

Server first sends global weights and then users train

- Need to change this in my model



While we focus on non-convex neural network objectives, the algorithm we consider is applicable to any finite-sum objective of the form

$$\min_{w \in \mathbb{R}^d} f(w) \qquad \text{where} \qquad f(w) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w). \qquad (1)$$

For a machine learning problem, we typically take $f_i(w) = \ell(x_i, y_i; w)$, that is, the loss of the prediction on example (x_i, y_i) made with model parameters w. We assume there are K clients over which the data is partitioned, with $\mathcal{P}_k$ the set of indexes of data points on client k, with $n_k = |\mathcal{P}_k|$. Thus, we can re-write the objective (1) as

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w).$$

If the partition $\mathcal{P}_k$ was formed by distributing the training examples over the clients uniformly at random, then we would have $\mathbb{E}_{\mathcal{P}_k}[F_k(w)] = f(w)$, where the expectation is over the set of examples assigned to a fixed client k. This is the IID assumption typically made by distributed optimization algorithms; we refer to the case where this does not hold (that is, F_k could be an arbitrarily bad approximation to f) as the non-IID setting.

Take advantage of fast mobiles

1. More parallelism
2. increased computation on each client, this is better

Iid = independent and identically distributed

- Independent
  - No change due to other data

- Identically
  - Same data generating process

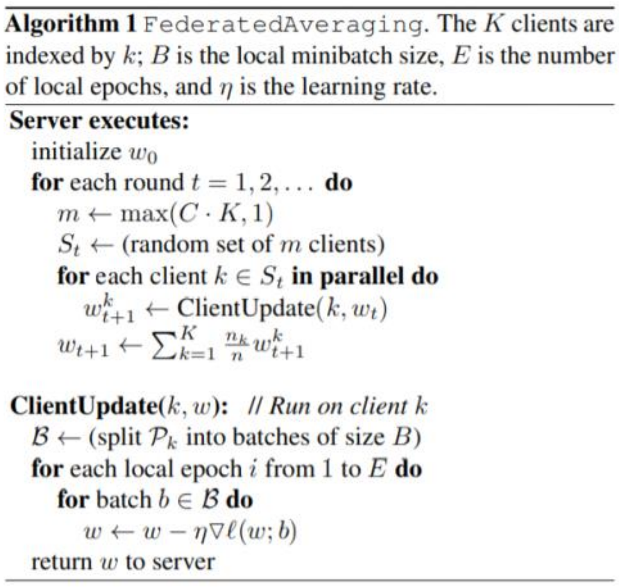
In the federated setting, there is little cost in wall-clock time to involving more clients, and so for our baseline we use large-batch synchronous SGD; experiments by Chen et al. [8] show this approach is state-of-the-art in the data center setting, where it outperforms asynchronous approaches. To apply this approach in the federated setting, we select a Cfraction of clients on each round, and compute the gradient of the loss over all the data held by these clients. Thus, C controls the global batch size, with C = 1 corresponding to full-batch (non-stochastic) gradient descent.2 We refer to this baseline algorithm as FederatedSGD (or FedSGD)

A typical implementation of FedSGD with C = 1 and a fixed learning rate η has each client k compute gk = OFk(wt), the average gradient on its local data at the current model wt, and the central server aggregates these gradients and applies the update wt+1 ← wt − η PK k=1 nk n gk, since PK k=1 nk n gk = Of(wt). An equivalent update is given by ∀k, wk t+1 ← wt − ηgk and then wt+1 ← PK k=1 nk n w k t+1. That is, each client locally takes one step of gradient descent on the current model using its local data, and the server then takes a weighted average of the resulting models. Once the algorithm is written this way, we can add more computation to each client by iterating the local update w k ← w k − ηOFk(w k) multiple times before the averaging step. We term this approach FederatedAveraging (or FedAvg). The amount of computation is controlled by three key parameters: C, the fraction of clients that perform computation on each round; E, then number of training passes each client makes over its local dataset on each round; and B, the local minibatch size used for the client updates. We write B = ∞ to indicate that the full local dataset is treated as a single minibatch. Thus, at one endpoint of this algorithm family, we can take B = ∞ and E = 1 which corresponds exactly to FedSGD. For a client with nk local examples, the number of local updates per round is given by uk = E nk B ; Complete pseudo-code is given in Algorithm 1.

What im doing

- That is, each client locally takes one step of gradient descent on the current model using its local data, and the server then takes a weighted average of the resulting models. Once the algorithm is written this way, we can add more computation to each client by iterating the local update w k ← w k − ηOFk(w k) multiple times before the averaging step. We term this approach FederatedAveraging (or FedAvg).
- Important params :
    - C, the fraction of clients that perform computation on each round;
    - E, then number of training passes each client makes over its local dataset on each round;
    - B, the local minibatch size used for the client updates
- We write B = ∞ to indicate that the full local dataset is treated as a single minibatch. Thus, at one endpoint of this algorithm family, we can take B = ∞ and E = 1 which corresponds exactly to FedSGD
- . For a client with nk local examples, the number of local updates per round is given by uk = E nk B ; Complete pseudo-code is given in Algorithm 1

when we start two models from the same random initialization and then again train each independently on a different subset of the data (as described above), we find that naive parameter averaging works surprisingly well (Figure 1, right): the average of these two models, 1 2w+ 1 2w 0 , achieves significantly lower loss on the full MNIST training set than the best model achieved by training on either of the small datasets independently



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
      $w_{t+1}^k \leftarrow$ 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
```

- As long as B is large enough to take full advantage of available parallelism on the client hardware, there is essentially no cost in computation time for lowering it, and so in practice this should be the first parameter tuned.

  For all three model classes, FedAvg converges to a higher level of test-set accuracy than the baseline FedSGD models. This trend continues even if the lines are extended beyond the plotted ranges.

Secure aggrigation

Saturday 4 January 2020 17:07

https://storage.googleapis.com/pub-tools-public-publication-data/pdf/ae87385258d90b9e48377ed49d83d467b45d5776.pdf

When Secure Aggregation is added to Federated Learning, the aggregation of
model updates is logically performed by the virtual, incorruptible third party induced by the secure multiparty
communication, so that the cloud provider learns only the aggregated model update

Masking with One-Time Pads. The first observation is that P u∈U xu can be computed with perfect secrecy if xu is masked in a particular way. Assume a total order on users, and suppose each pair of users (u, v), u < v agree on some random vector su,v. If u adds this to xu and v subtracts it from xv, then the mask will be canceled when their vectors are added, but their actual inputs will not be revealed. In other words, each user u computes
There are two shortcomings to this approach. The first is that the users must exchange the random vectors su,v, which, if done naively, would require quadratic communication overhead (|U|×|x|). The second is that there is no tolerance for a party failing to complete the protocol: if a user u drops out after exchanging vectors with other users, but before submitting yu to the server, the vector masks associated with u would not be canceled in the sum z.

Masking
- Pairs know masks for each other, add and subtract it from their values
- In the end their 2 values on the server will be the same

Seed sent to all nodes, and can be recoved if threshold number of nodes are alive
- If delayed response from x, and you know its share of mask, then you know x fully (as you know its mask)

FYP Notes

Tuesday 17 September 2019 14:02

- Academic supervisor
- Independent work
- Regular meetings
- Multiple deliverables
  - Project report is the most important
  - 2 new ones

- Open day
  - Mandatory
  - First half is presenting it to the academic supervisor
    - Second reader as well
    - Possibly some more academics
    - Prepare it
  - Second half
    - Everyone and company?
  - Its fun

Key dates

- October 7, 2019
  - All students must have an FYP assigned
- November 1, 2019
  - Outline document due
    - To see if you have understood the project
- March 9, 2019
  - Extended abstract due
    - Kickstarter of the report
    - Can copy paste into analysis section of the report

- March 25, 2020
  - TBC
  - Open day - You will remember this event
- April 2, 2020
  - TBC
  - Project report due
    - Electronic
      - A folder with code
    - 2 printed (binded) copies

Grading

- Module specifications for CS4501
- If more than 14 days late, 0 marks
- 300 marks
- No repeats
- Primary deliverable which will be graded is the project report
- Supervisor, second reader, exam board and external examiner are involved in the grading process
  - Independent grading

FYP Notes 2 (from Tim Creedon)

Tuesday 1 October 2019 17:45

Thanks blade

In mardown format

You should create an account and browse interesting projects - on the website.
The flow is the website which I gave is just for helping you.
What is confusing about the website at the moment, is that colleagues of mine haven't updated their space.
You approach your supervisor "I would like to do that project".

There's a shortcut to the presentation for last time - was in an email.

Then website.
You should create an account if you don't have one already.
Some lecturers do not expose their projects - only public.
You cannot filter projects.

{{ Login }}

Bsc, that's important.
Fill in your student number - dont worry your information is never exposed
What you want to do is here - Relevant (here you browse your projects). These are the ones that are not taken.

Students are usually better than lecturers to figure out how software works.
Academics including me are difficult to guide and to lead.

You can also search here - I take pride in my search implementation.

Look here for "Frees". Atm there's no sort, or no filter on search results.

WORDS WORDS WORDS WORDS
WORDS WORDS WORDS WORDS
WORDS WORDS BIRDS WORDS
WORDS WORDS WORDS WORDS

Report template structure example structure example structure example report JA
1. Analysis (no code)
2. Design (no code)
3. Implementation (3rd party packages...code...)
4. Testing
5. Appendix (glossary babyyyy)

2 bad things in reports:
- colloquial lang (2nd person)
- excessively mentioning 3rd party tools, blah blah, tensorflow, blah blah facebook, blah blah react,

blah blah

(bonus round)
+ Bitmap files for diagrams (JPEG no no no)   (PDF Embedded, SVG, EPS)
+ Cpations included in image or diagram files (do not put it in the image itself)
+ Rephrased text for the WWW
+ Excessive screenshots

Log what you do ===> easier, more natural flow.

Do testing.

MICROSOFT OFFICE ==> NO WASSSIIIIWIIGGGGGGG (useful for shopping lists)

You need lAtEX brrrrroooooooooooooooo

(tex) is a compiler btw (low level stooof, latex is a nicer wrapper).

PDF-Latex compiler exists. There's various online tools.

Cloud-based document at the start for easy access, easy backup.

DOOOOOOOOCCCCCUUUUUUUUMENNNNNNNTTT EVERY STEP

Technical writing
* Proper citations (identify all 3rd party sources)
* Be clear and concise
* Charts, tables, figures etc must have captins and be properly referenced in the text. Use proper labels.

"I have a very good writing guidelines, I will make them available to you soon. If I forget to do so - remind me,
by next week, email".

List of figures of all your graphics - usually a good idea.
Code code, where should code appear.
Additionally code zipped electronic submit.

Appendix of code? no.
Crucial code...in the report...if.you.really.want

Extended abstract deliverable is ~~~~~ 5 pages.

Notes

Saturday 14 September 2019 15:13

Reshape data into 3d data

- No because the sensors are not ordered
- Each record is a set. The i-th marker of a given record does not necessarily correspond to the i-th marker of a different record. One may randomly permute the visible (not missing) markers of a given record without changing the set that the record represents. For the sake of convenience, all visible markers of a given record are given a lower index than any missing marker. A class is not guaranteed to have even a single record with all markers visible.

Information on the Class enumeration, ranges from 1 to 5 with

- 1 = Fist (with thumb out),
- 2 = Stop (hand flat),
- 3 = Point1 (point with pointer finger),
- 4 = Point2 (point with pointer and middle fingers),
- 5 = Grab (fingers curled as if to grab).

Dataset location

- https://archive.ics.uci.edu/ml/datasets/MoCap+Hand+Postures

To do

Saturday 28 September 2019 12:36

Things to keep in mind

- Reference need to be collected
- maybe bad to have too much accuracy. hard to see if the federated approach makes a diff

Derek notes

- Preprocess to scale it
- K fold validation
- Add model picture using tensorboard
- what about user 3 who has no data
  - He is like a new user into the federated system.
    - Can see how a user with no training data benefits from just getting averages of other people
  - see if there is test data, then they are useless, get them out

- Some people might benefit with using the federated approach, and some wont
  - Summarize this, see who is and who isnt based on accuracy
  - How many get better
    - summarise this

- We may optimise the model
  - Try playing with the hyperparameters
  - Maybe the federated model works better with a different architecture than the architecture that suits non federated approach

- delay in keras
  - Research this

- Instead of having a server taking decisions about biased averages, send the model data to users instead like a p2p approach
  - They have their own policies and will therefore result in a different average than others
- this
  - Important params : C, the fraction of clients that perform computation on each round;
  - E, then number of training passes each client makes over its local dataset on each round; and
  - B, the local minibatch size used for the client updates
    - As long as B is large enough to take full advantage of available parallelism on the client hardware, there is essentially no cost in computation time for lowering it, and so in practice this should be the first parameter tuned.

  - Have a central model which gives out weights
    - This will be useful for getting averages too
- Next week
  - see googles one in the tensorflow thing
    - Understand
    - See if we are missing anything

3 diff models

1. Train on local data and test on local
2. Train on all data nad test on local
3. Exclude terrible
   a. Currently I do it as

            i. One value, if you do not match it, you get kicked. But what if no one matches it? No one gets averages
    b. Better to do it relatively. If you are 2 std less of everyones means, you get kicked
4. The one I made, send data to server and it averages it
5. Everyone sends me data and I average it
6. Weighted average
    My weihgts times my accuracy and so on for all

For fair ness
    All same epochs
    Or stop by the time they are overfitting

======
think about stopping training when you start over fitting

Refactor code to use describe instaed of stuff like loops in graps
final idea

- maybe give the uers the mothod of weighting
- send data to one userr
- they broadcase it to everyone else too
- use will initialise a model for everyone
    - run our test set on everyone elses model
- model
    - we throw models away based on our datas accuracy on their model
    - then we average based on our test datas accuracy on their model
- then we have our final model and everyone will have their

Post chrismast
    New dataset
    - Uci machine learning repo
    - Could have a simulated dataset
        - Imagenet
        - People have tastest and then that is theier data set
            - Like more pics of dogs for a person
    - Could have a voice oriented data set
        - Don't wanna share voice/spoeach data
    - Style of writing
        - Don't wanna share data
- Sanity check with tensorflow
- Piddle around with arhcitecutre
    - Convnet
    - Dense net
    - Max pooling layers?
- Recommender thing
    - People don't wanna share their models of rating to other people
    - Matrix facotrization

    Latex
Week 2
    Amazon not accept
    Ony ocll no sshaare

    Onlly llllllllllocatslll no sshare
    If not goodaaaass spost fffit

Not overffitting

Week 3

Make it probablitic

Equal distribution probablitistic
Evennmore skewed
Check the way the data is split for etsting and val

Early stopping?
How many epochs does google do

Ealy stoping for sharing
Early stopping for epoch
Or could just give max
    And give max and last values of the accuracy

Find the best modelf for the global approach and use that for fairness
- Training is union of training for users

Write some report
- Idea of problem
- Idea of wahts been done. Not my work, but other work and how it inspired me
    - Ml and deep learning
    - How to train
    - Google doing it
    - Privacy
        - Federated
            - Runing on edge too
        - Differential
- Building and designing this
    - More chapters for the models
        - One for tf and modily
    - Describe each

Sanity check
    Do this for images too?

Average all the results
    3 runs ofeach split and then average
Probabiliysitc model
Confusion matrix
Refactor to use train test split

Report

Work plan

Friday 27 September 2019 12:13

Week 1

- Basic model using the mnist model
- Started the book

Week 2

- Finished the book

Week 3

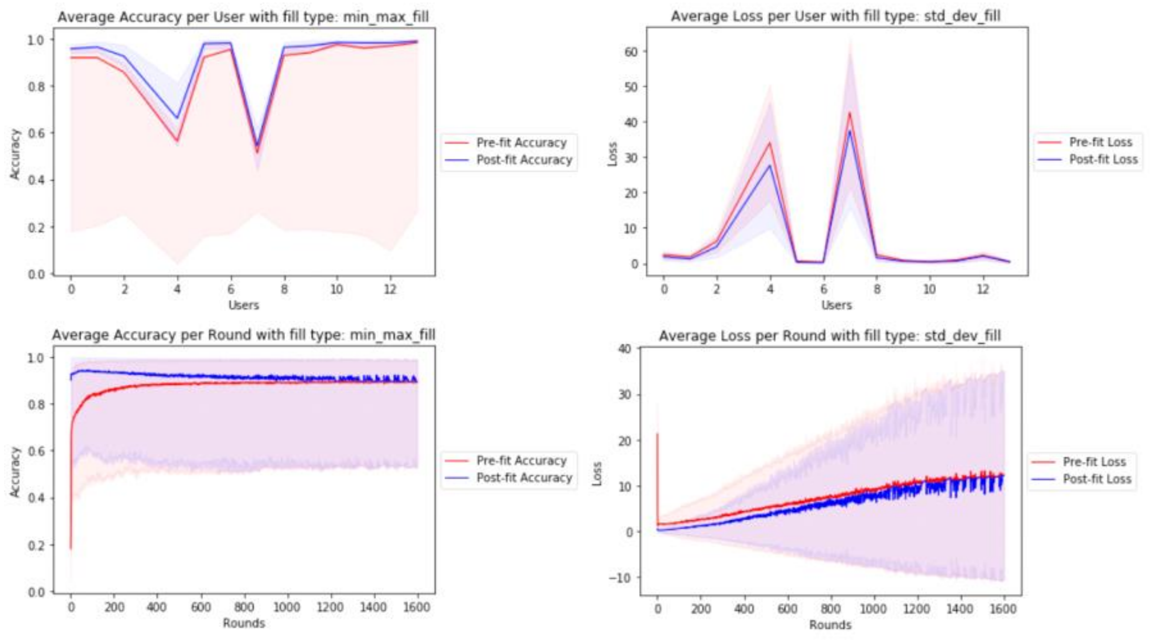
- Made the traditional model, all data trained on one model
- Work on implementing the google implementation of federated machine learning
    - # weights = [[user0] [user1] [] [list of numpy arrays of layer weights and biases] [] []]
    - Sensor data is a set, so sensor values dont line up. hence 2d instead of conv net

Week 4

- Get a powerful pc
- Read the paper and got the GPU executions working
    - Benchmarks say that simple model doesn't need gpu
- Fair playing field
    - Each user has 16 epochs
    - 64 iterations of back and forth
    - The model trained with everyones data must be with 16 times 64 epochs
- Graphs
    - Added box plot to see outliers
    - Plot to see how the accuracies and losses vary over the rounds in the federated learning process
    - Save them for easier viewing
- Skip some weights maybe theyre dodgy or bad results
    - I skip based on accuracy epsilon

Week 5

- Refactored the code to
    - store roundwise accuracy and loss values in the user instance
    - Use numpy arrays instead of lists for convenience and performance
        - 1600 rounds on 16 epochs took
    - Bad for space new code, better for speed
- The sausage graph
    - Made for
        - Average of every user over the rounds
        - Average of the round from every user



- ○ It has options of min_max_fill and std_dev fills
- ○ Not proud of the code, but it works

- Get onenumber for the system
  - Accuracy and std dev
  - Average of all usrs and them individually
- include accuracy in the report
  - talk about statisitc you see and the fact that 1 std less is bad, results in a slightly worse overall model

1. Exclude terrible
   a. Currently I do it as
      i. One value, if you do not match it, you get kicked. But what if no one matches it? No one gets averages
   b. Now it is
      i. Based on 1 std dev, if less than one std less than average, we exclude them
         1) This was slightly worse

Week 6

- weighted average is dividing by sum of accuracies
  - My weights times my accuracy, for all
  - Done, slightly better than others overall
- Tried implementing p2p
  - Issues

Week 7

- Models now are reproducible
  - Clearing keras session means that the graph of operation nodes are gone. Lose all data about the model, including weights. Not ideal
  - https://keras.io/getting-started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development
  - Models are initialsed with the same weihgts
    - Fixed bug where final layers were random, not anymore
  - Same seed used to split the dataframes
    - Fixed a bug where the dataframes for 2 sets of users were not split the same way
  - The fit method shuffled data before fitting, turned that to false in parameters for fit
  - Dataframe is now seeded shuffled

    using below line and not np.random.seed(SEED) as otherwise, that line needs to be called everytime before shuffling the dataframe cause it "moves"

- Changed user latest data lamba to a function
- Added p2p
  - Works for all and std_dev and weighted_avg

        - Doesn't work for std dev and weighted avg
            - Pending testing
- Outline doc
- Setup the github for the project
- Prefit models are better with std dev in p2p model
- All averaging method is the same in p2p and central model

Week 8
- Nothing
- Loss needs to hav ea different way instead of accuracy

Week 9
- To do
    - Do user specific imputation
    - Shuffle split
    - Check if dictionaries are being instantiated correctly when using from keys

    - Check tensorflow implementation
        - Look for what we are doing
        - What we are missing
        - The encryption thing
        - Run on our data and see what we get and compare our model and tensorflows model
    - Play around with networks and structures
        - Architectures
        - Hyperparameters
        - Stop training when we start overfitting
        - loses
    - Change dataset
        - Something based on users
- that said, as you can find out by studying the paper on the Federated Averaging algorithm, achieving convergence in a system with randomly sampled subsets of clients in each round can take a while, and it would be impractical to have to run hundreds of rounds in this interactive tutorial.

  From <https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification>

REPORT TIPS

Give contextual information

Don't start at the begininng

Blah blah dataaset

Ml chap 1

Convnet

Densely nods

Different ways

Security

Differential privicy

But theres federated instead chap 2

My implemetation chapter 3

Extensions to fml chap 4

The ideas

Exxperiemnets

For each dataset show the otuput we got

Christmas break
- Security
    - Its about aggregation protocols and using a way in which the server never sees anything

  - but the aggregate
  - Not to do with actual algorithms in my opinion
  - https://storage.googleapis.com/pub-tools-public-publication-data/pdf/ae87385258d90b9e48377ed49d83d467b45d5776.pdf
- Changed metrics to string type

Week 1
- Changed dataset split logic
- Implemented tensorflow federated for sanity check
  - Mine is significantly slower
  - And 4% worse accuracy on an 80-20 split

Week 2
- Implemented version where users only train on their local data
  - And as always, test on their local data too
- Need to find out
  - If local_only is better than post fit
    - If so then no point in federated
    - If close enough then post fit gives accuracy too
  - Compare with global model and see whats up

Epochs	Rounds	Model	Accuracy pre fit	Accuracy post fit	Loss pre fit	Loss post fit
512	1	Global server		0.77784465		0.98681176
16	32	Local training only	0.95385	0.95502	0.379573	0.374581
16	32	Federated (all)	0.753598	0.961534	2.14365	0.209885
16	32	Central Federated (avg - 1 *std_dev)	0.77012	0.957461	2.81633	0.211869
16	32	P2P Federated (avg - 1 *std_dev)	0.778403	0.956558	2.14895	0.224604
16	32	Central Federated (weighted average)	0.75585	0.955715	2.30846	0.201472
16	32	P2P Federated (weighted average)	0.798695	0.95583	1.7076	0.223523

Images

Epochs	Rounds	Model	Accuracy pre fit	Accuracy post fit	Loss pre fit	Loss post fit
400	1	Global server				
20	20	Local training only	0.725658	0.731091		
20	20	Federated (all)	0.621451	0.786981		
20	20	Central Federated (avg - 1 *std_dev)	0.583144	0.795676		
20	20	P2P Federated (avg - 1 *std_dev)	0.677205	0.812635		
20	20	Central	0.596836	0.796256		

		Federated (weighted average)				
20	20	P2P Federated (weighted average)	0.631476	0.703305		

Week 3

Implemented reproducibility

Not fully because of parallelisatiino

Found a model that is decent for for global usage (85-88%)

Implemental global model trainining

Memory optimisattion by using uint8 instead of float32

Majority split -> 1/8 (equally distributed)

Epochs	Rounds	Model	Accuracy pre fit	Accuracy post fit	Loss pre fit	Loss post fit
512	1	Global server		0.9067308		1.15681871541
16	32	Local training only	0.808654	0.809615		
16	32	Federated (all)	0.909615	0.906731		
16	32	Central Federated (avg - 1*std_dev)	0.903846	0.879808		
16	32	P2P Federated (avg - 1*std_dev)	0.9	0.893269		
16	32	Central Federated (weighted average)	0.908654	0.903846		
16	32	P2P Federated (weighted average)	0.906731	0.907692		

Majority split -> 0.5

Epochs	Rounds	Model	Accuracy pre fit	Accuracy post fit	Loss pre fit	Loss post fit
512	1	Global server		0.88291746		1.65815507575
16	32	Local training only	0.85673	0.855825		
16	32	Federated (all)	0.89123	0.925544		
16	32	Central Federated (avg - 1*std_dev)	0.890336	0.911184		
16	32	P2P Federated (avg - 1*std_dev)	0.890606	0.914792		
16	32	Central Federated (weighted average)	0.895508	0.922043		
16	32	P2P Federated (weighted average)	0.899096	0.91807		

Majority split -> 0.7

Epochs	Rounds	Model	Accuracy pre fit	Accuracy post fit	Loss pre fit	Loss post fit
512	1	Global server		0.8722382		2.13644868530
16	32	Local training only	0.875199	0.875409		
16	32	Federated (all)	0.819616	0.916871		
16	32	Central Federated (avg - 1*std_dev)	0.869484	0.929704		
16	32	P2P Federated (avg - 1*std_dev)	0.889362	0.927284		

16	32	Central Federated (weighted average)	0.639407 (went down)	0.865209 (went down)		
16	32	P2P Federated (weighted average)	0.888332	0.931224		

Majority split -> 0.99

Epochs	Rounds	Model	Accuracy pre fit	Accuracy post fit	Loss pre fit	Loss post fit
512	1	Global server		0.8333333		3.78403481614
16	32	Local training only	0.999132	0.999132		
16	32	Federated (all)	0.598921	1		
16	32	Central Federated (avg - 1*std_dev)	0.523298	1		
16	32	P2P Federated (avg - 1*std_dev)	0.505658	0.999		
16	32	Central Federated (weighted average)	0.618307	0.995835		
16	32	P2P Federated (weighted average)	0.997097	0.99359		

If extremely skewed
- Accuracy would be high because it would predict the majority class well, but the rest would be ignored completely. So metrics are not ideal for something like 90% skew

Probablistic model
- Probabilty to distribute between all the classes?
  - If that's the case then the model is not very useful imo
  - Cause its essentially a biased random with the chance of a user ending up with no images for a class
- Or majority vs rest split is probabiliyic with the rest being equally divided

When the global model is tested on weights from all bar local_only. The metrics are still good/comparable or better than normal global model weights

TFF

Friday 3 January 2020 22:00

Security

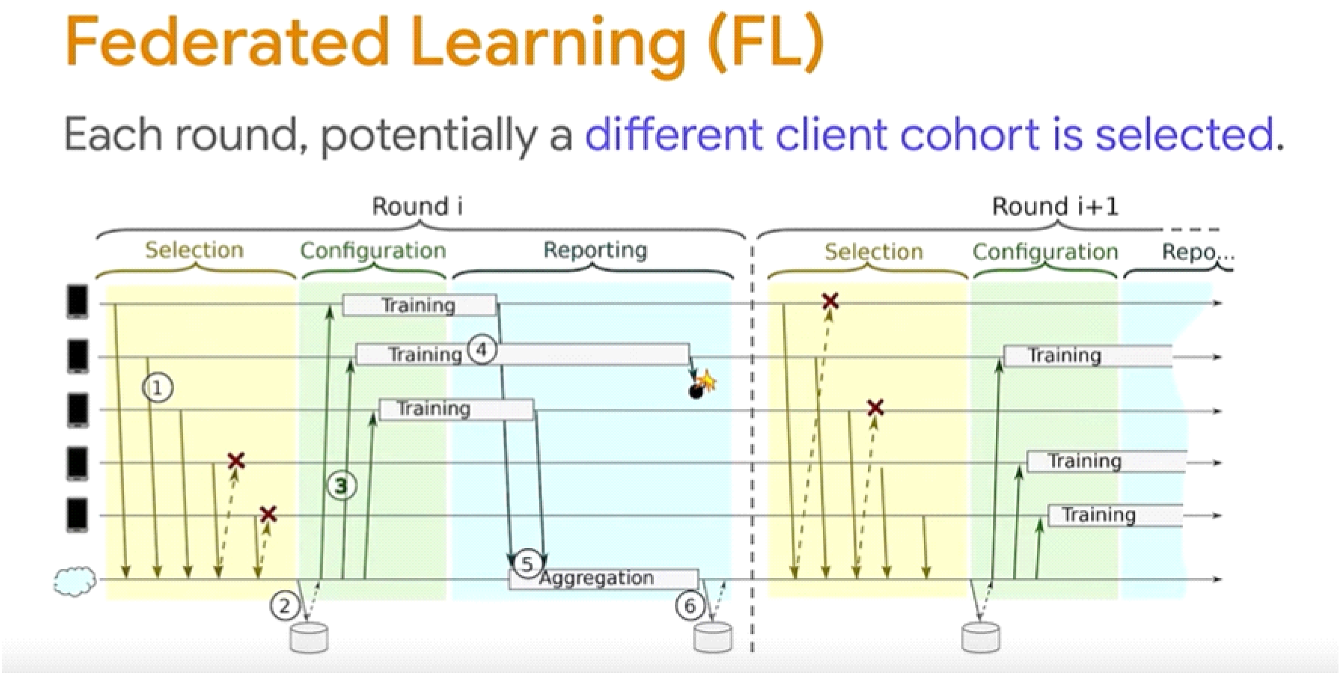
- Server doesn't have individual client contributions
  - Only final aggregate
- Clients add some noise in their data so the server cant retrace anything

Only a subset of devices take part in it, so set of clients per round can be diff

Cause not all are online at a given time

Only when being charged

Data ccharges



Federated Learning (FL)

Each round, potentially a different client cohort is selected.

[Federated learning with TensorFlow Federated (TF World '19)]()

Client characteristics

- Anonymous
- Limited availabliliy
- Unreliable can drop out
  - Wont matter in reality, grand scheme of matter. 1 machine is not big
- Stateless cause they can be missing later
- Non iid data on device

System wide chars

- Parallel
- Comms bound
- Not p2p

Why use it

- If local data is big
- We need it labelled and user interaction can give it

Tff

- Has an underlying language under python
  - Does the comms and tf
- Find the best modelf for the global approach and use that for fairness (found one with 85%)
  - Training is union of training for users

Report

Friday 14 February 2020 16:11

Abstract
Declaration
Table
Figures

Background
- What is the problem
- Intro
- Motivation
- Idea of wahts been done. Not my work, but other work and how it inspired me

Lit review
- Diff preivacy
- Federated

Core design
- Design
- Implementation
- Psuedocode
- Testing
  - Tensorlfow

Extensions

Testing with the data

Conclusion future work

Bibliography

Friday 21 February 2020 10:39

- Chapter 1 needs to be called introduction
- intro
  - Intro is good **DONE**
  - The "correct" line is too conversational. Use more formal **DONE**
  - Don't use too amny questions **DONE**
- Write a better motivation
  - Talk about how wonderful AI was going ot be **done**
    - Be better **DONE**
    - Never get tired **DONE**
    - Reach at least human levels **DONE**
    - Maybe better **DONE**
  - Oo problems secutiyu **DONE**
  - Explanations of the reasonsings **DONE**
  - But I am interesting in privacy **DONE**

  - Motivation about what this project is
    - Provacy **DONE**
    - Don't worry because im coming and im going to solve this problem **done**

  - Outlien of the report
    - Just say youll be doing fml and epxlain chapt 2 **DONE**
    - Proposed by google **DONE**
    - Ill extend it too **DONE**
- End section 1
- Move the ai section into introduction. Rename AI to state of the art/lit review **DONE**
  - Give the picture in there **DONE**
  - Loses what AI **DONE**

  - Decision trees and all **DONE**
  - We work with NN **DONE**
    - Because ehot struff **DONE**
  - Explain NN **DONE**

  - Talk about diff privacy
  - Fml

- Dataset and expereiments the same **DONE**
  - Or diff chapters for images **DONE**

- Conclusions and future works **DONE**

- Layer of NEURONS not perceptrons **DONE**
  - Perceoptron is a single layer of neurons **DONE**
    - Densly connected too **DONEDONE**
- Redo convnet **DONE**
  - Conv2d **DONE**
  - Maxpooling **DONE**
  - Kernel **DONE**
  - Why is it there **DONE**

- Featuremaps **DONE**
    - Connected to a stack of different neuraon in the layer below **DONE**
    - Weidht and height based on window **DONE**
    - Depth based on feature maps **DONE**
    - Shared weights **DONE**

- No seperateion between architecture and training **DONE**

- Supervised and unsueprvisoed **DONE**
- Backprop **DONE**

- NO CONVERSATINOAL **DONE**

- FML **DONE**
  - Weights are not explained before hand **DONE**
  - Needs to be moved **DONE**

- Comoing up with decent model **DONE**
- Traditional ml results **DONE**

- Redo abstract **DONE**

- Change the working with derek statement **DONE**

- Can just saw 6 other graphs were xyz for whatever reason
  - Give a reason to why im shoing somethin g

- Not written in the passice
  - Name the agent
  - Describe the system
    - Give it a name
    - User
    - Name parts of the system
    - Edge agent, central agent
  - Present and past

- Last thign can have I and stuff

Edges ei with data di
Central model mi

Weights for all w in M, the weights equal double the weights of I divided by the number of edge drevices

Show what happens on edge and central model

Explain it as if it is for a dum dum

Say formla 7 becomes formula 12
- This is how you connect difererrnet ways
- Average to weighetd

Sending weights and accuracy instead of weights

Something and symbols are important

Implemenation
- Heres the code using keras
- Conv and non conv
- Say how keras gets weight by making refernce to algorithms

Revist this

Monday 16 March 2020 12:29

Revist fed ml section 2.7 NO

Restyle classes to be a monospace? NO

Rename captions and functions DONE

Ccheck refererences and if theyre all caps or not. Standardize **CHECK**

Add parameter to averaging function **NO I DID THE COMMENT INSTEAD**

Change model section to say non reproducibe when gpu used **DONE**

Derek meeting talking points

Monday 16 March 2020 20:09

Revised notes

- Move the splitt stuff into experimentatation DONE
- Evaluation can be done in experiments
- Section 3 is supposed to be about just the algos
    - We assume training set and model somehow
- We can talk about the simulation of different uses in design but also in implementation.
- Extended ideas

        Swap averaging functions
        Everything stays the same
        implementation as a parargraph even
        Talk about averagin class
- In implementation talk specifics of how algo is done
    - Line xyz keras is used to get weights
- Tff
    - Why we didn't do it in this
    - No deisng
    - Just implementation
    - How did we check?
        - Experiemnts in section 5
        - Close enough
        - So we were good
- Data representation is into dfed ml implemneation
    - Asume U_i means this now
- Talk about keras functions used a lot
- Design and implementation for everything

Talk to him about server sending to the users the models

No api for this, magic method of sorts

How do I separate implementaiton from design

Ive no idea

Talk about implemnentaiton is the firt implementation

Line x says get weights nd line 4 says set wiegts
Get weights and set weights

Simulation can be design and impleenation

Tff

Why dindt od it in this
No design here
Ho did we check

Future chapter reference

Splitting put into experiments

Move data rep to fed ml

    No need to again refer to this

Talk about keras a lot in the implemmentiaon

Design and implementation for everything

In extended ideas we just do a last para for implementation

    Its given what I do in ml

    Just do blah blah

Averaging class

No need to touch on evaluation

Assume training set

Assume we mahav a moe

    Say we say in chapter 5 how we got it

Second derek meeting online

Friday 20 March 2020 15:51

Design
    How things are done and the algorithm

Implementation
    Strictly when talking about the code

Extended ideas
    How exactly to structure this as well?

Experiments

What about kieran

Third derek meeting online

Friday 3 April 2020 16:29

Comments on graphs

- What do I write
    - Talk a lil bit about anything important
    - Comprare to benchmarks
    - Talk about trends and progress over time
    - Compary privacy
    - Trade off
    - Conclusion
- Talk about the dataset
    - Issues with destures made used wise data weird
    - Bad
- 1 subsection for dataset discussion and one for compraing

Graphinh section

Half a page on implementation
Write design a lot

Email

User E_{x,y} instead of E_{y} DONE
Add parameter to averaging function

AI 2 is open book