Federated Learning

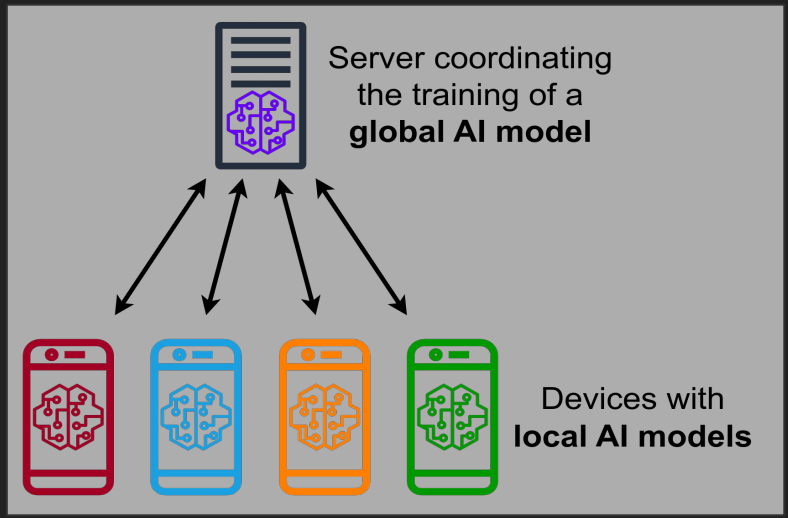
John Fattore

Distributed Learning vs Federated Learning

- Federated Learning is a type distributed learning
- Both delegate learning, but different motives
- Privacy and communication cost vs computational costs
- 10,000s of edge devices vs 10s of supercomputers

Motivations for Federated Learning

- Enabled by the decrease of communication costs, particularly for sensors / edge devices
- Train models on sensitive data on edge devices without compromising privacy
- Apple can train a health prediction model using client data, while maintaining privacy



Communication Costs

- Only eligible clients are chosen each training round
- Communication over unmetered WiFi connections are much cheaper than 5G wireless networks
- Training can happen at night when devices are plug in and connected to their home WiFi

FedAvg and FedSGD

- 7000+ citations FedAvg!
- FedSGD allows clients to do one round of training before sending updated weights
- FedAvg allows clients to do multiple rounds of training per update

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**

$w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$

$m_t \leftarrow \sum_{k \in S_t} n_k$

$w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$ *// Erratum*[4]

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**

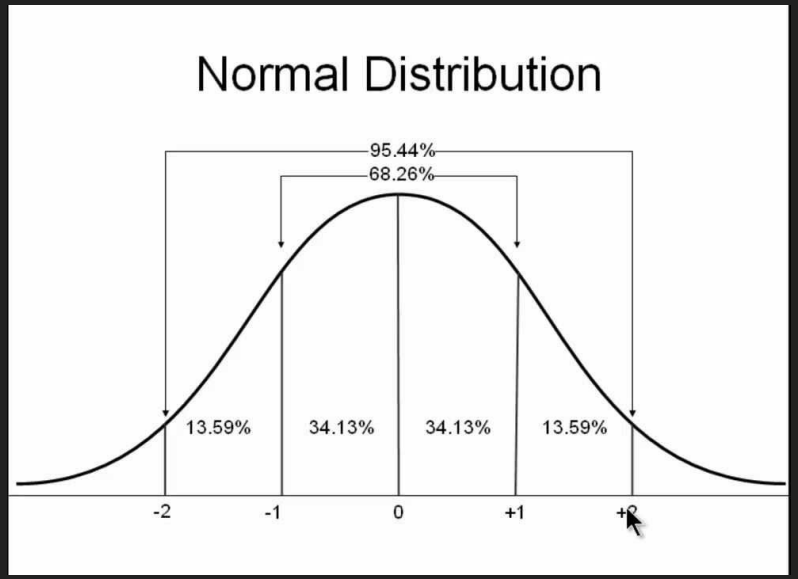
for batch $b \in \mathcal{B}$ **do**

$w \leftarrow w - \eta \nabla \ell(w; b)$

return w to server

IID vs Non-IID

- IID is uniform and less complicated to handle
- Non-IID are data samples that do not encompass the overall distribution
- Devices with outlier data will produce large gradients that will skew the global model



FedProx

- Looks to increase accuracy of model when dataset is Non-IID and use edge device resources efficiently
- Allows devices to do variable amounts of work
- Proxy value restricts local updates, particularly on devices with more statistical heterogeneity

$$\min_w h_k(w;\ w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2 .$$

q-FedAvg

- Inspired by observations that while the global model may have acceptable performance, this accuracy does not always carry over to each device
- Aims to achieve device-level fairness
- Emphasizes devices with higher local loss
- Improves local accuracy with little effect on global accuracy

$$\min_w \; f_q(w) = \sum_{k=1}^{m} \frac{p_k}{q+1} F_k^{q+1}(w)\,,$$

per-FedAvg

- Aimed to provide global model that is beneficial to all users, like q-FedAvg
- Takes personalization one step further, providing a global model that must be “broken in” before use
- Global model is trained on local data to provide user with an accurate and personalized model

NBA Linear Regression FedSGD

- Model Objective: Predict NBA Team's Win Percentage Based on Season Stats
- 6 Features - [3-Point %, 3-Point Attempts, Field Goal %, Turnovers, Assists, Offensive Rebounds]
- Data - Season stats from the 2000s obtained from NBA.com
- Average Percent Error (MAPE): 29.36%
- Average Error (MAE): 11.387

Linear Regression FedSGD Training

1. Random selection of NBA teams chosen to participate in training
2. The global model is sent to each participating NBA team
3. NBA team trains model using their local data, finding gradients for each parameter
4. Gradients are averaged and global model makes a “step”, establishing a new model

```
# initialze parameters
weights=[-0.01, 0.01, -0.01, 0.01, 0.01, -0.01]
intercept=0
learning_rate = 0.0000001
MSEresults = []

epoch = 2500
for i in range(epoch):
  federatedGradients = [0, 0, 0, 0, 0, 0]
  federatedInterceptGradient = 0
  teamsTraining = 0
  # for each team in the nba
  for team in nba:
    LocalGradients = [0, 0, 0, 0, 0, 0]
    localIntercept = 0
    # random teams are sampled to train
    sampled = random.randrange(3)
    if (sampled == 1):
      teamsTraining = teamsTraining + 1
      y_predicted = predictY(weights,nba[team],intercept)
      scaledGradients = (multiplyListNum(WeightsGradients(nba[team],nbaWins[team],y_predicted), (learning_rate)))
      # local gradients calculated
      localGradients = (multiplyListNum(scaledGradients, sampled))
      scaledInterceptGradient = ((learning_rate) * InterceptGradient(nbaWins[team], y_predicted))
      localInterceptGradient = scaledInterceptGradient * sampled
      # gradients accululated
      federatedGradients = np.add(federatedGradients, localGradients)
      federatedInterceptGradient = federatedInterceptGradient + localInterceptGradient

  # gradients averaged
  federatedGradients = multiplyListNum(federatedGradients, (1/teamsTraining))
  federatedInterceptGradient = federatedInterceptGradient / teamsTraining
  # once local gradients are averaged together, make a gradient step
  weights = np.subtract(weights, federatedGradients)
  intercept = intercept - federatedInterceptGradient
  MSEresults.append(MSE(yTest, predictY(weights, xTest, intercept)))
print(weights, intercept)
```

```
[0.38758917 0.25064006 0.49326437 0.15252636 0.25464938 0.09917967] 0.010554256003817522
```

Sources

[1] Verbraeken, J., Rellermeyer, J. S., Verbelen, T., Kloppenburg, J., Katzy, J., & Wolting, M. (2020, March). A survey on Distributed Machine Learning. ACM Digital Library. Retrieved April 25, 2023, from https://dl.acm.org/doi/pdf/10.1145/3377454

[2] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Arcas, B.A. y. (2023, January 26). Communication-efficient learning of Deep Networks from Decentralized Data. arXiv.org. Retrieved April 25, 2023, from https://arxiv.org/abs/1602.05629

[3] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., & Smith, V. (2020, April 21). Federated optimization in Heterogeneous Networks. arXiv.org. Retrieved April 27, 2023, from https://arxiv.org/abs/1812.06127

[4] Li, T., Sanjabi, M., Beirami, A., & Smith, V. (2020, February 14). Fair Resource Allocation in Federated Learning. arXiv.org. Retrieved May 4, 2023, from https://arxiv.org/abs/1905.10497

[5] Fallah, A., Mokhtari, A., & Ozdaglar, A. (2020). Personalized federated learning with theoretical guarantees: A model ... NeurIPS 2020. Retrieved May 4, 2023, from https://proceedings.neurips.cc/paper/2020/file/24389bfe4fe2eba8bf9aa9203a44cdad-Paper.pdf

[6] Bellwood, L., & McCloud, S. (2016). Google Federated Learning. Google. Retrieved May 4, 2023, from https://federated.withgoogle.com/