Federated Machine Learning With Python

Training models without looking at data

Dhanshree Arora
MLOps Engineer
Agenda

1. The Privacy Cost of Machine Learning
2. Privacy Preserving Machine Learning
3. Federated Learning
4. Building a Minimal FL System
5. Opportunities in FL
6. Conclusion
7. Questions
The Privacy Cost in Machine Learning
I just don't think everyone should have access to the ambient nano-cameras currently inside my anus.

God, you're old-fashioned.
Data breaches through the ages
Privacy Preserving Machine Learning (PPML)
Federated Learning

Training models on data at its source
Differential Privacy
Adding noise to de-identify data while preserving the distribution and relationships within the data

q(data + noise) → q(data) + noise
Homomorphic Encryption

Mathematical operations on encrypted data

Without HE

With HE
Federated Learning
Federated Learning vs Centralized Learning
Federated Learning Use-Cases

EDGE DEVICES:
- Recommendation
- Routine device storage maintenance
- Health Monitoring
- Predictive Typing
- Facial Unlocking

ENTERPRISE:
- Credit Card Fraud Detection
- Credit Lending
- Disease Prediction
- Sentiment analysis
- Autonomous vehicles
- Precision Medicine
## Horizontal Federated Learning

<table>
<thead>
<tr>
<th>sex</th>
<th>age</th>
<th>spo2</th>
<th>comorbidities</th>
<th>symptoms</th>
<th>oxygen_req</th>
<th>icu_num_days</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>52</td>
<td>74</td>
<td>0</td>
<td>...</td>
<td>200</td>
<td>7</td>
</tr>
<tr>
<td>f</td>
<td>23</td>
<td>89</td>
<td>1</td>
<td>...</td>
<td>150</td>
<td>13</td>
</tr>
<tr>
<td>f</td>
<td>42</td>
<td>90</td>
<td>0</td>
<td>...</td>
<td>190</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sex</th>
<th>age</th>
<th>spo2</th>
<th>comorbidities</th>
<th>symptoms</th>
<th>oxygen_req</th>
<th>icu_num_days</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>25</td>
<td>70</td>
<td>0</td>
<td>...</td>
<td>120</td>
<td>19</td>
</tr>
<tr>
<td>f</td>
<td>32</td>
<td>85</td>
<td>1</td>
<td>...</td>
<td>120</td>
<td>4</td>
</tr>
<tr>
<td>m</td>
<td>68</td>
<td>65</td>
<td>0</td>
<td>...</td>
<td>210</td>
<td>11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sex</th>
<th>age</th>
<th>spo2</th>
<th>comorbidities</th>
<th>symptoms</th>
<th>oxygen_req</th>
<th>icu_num_days</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>46</td>
<td>74</td>
<td>0</td>
<td>...</td>
<td>150</td>
<td>7</td>
</tr>
<tr>
<td>m</td>
<td>84</td>
<td>89</td>
<td>1</td>
<td>...</td>
<td>300</td>
<td>20</td>
</tr>
<tr>
<td>f</td>
<td>39</td>
<td>90</td>
<td>0</td>
<td>...</td>
<td>200</td>
<td>10</td>
</tr>
</tbody>
</table>

Hospital A

Hospital B

Hospital C
## Vertical Federated Learning

### Hospital A

<table>
<thead>
<tr>
<th>name</th>
<th>sex</th>
<th>age</th>
<th>spo2</th>
<th>comorbidities</th>
<th>symptoms</th>
<th>oxygen_req</th>
<th>icu_num_days</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person A</td>
<td>m</td>
<td>52</td>
<td>74</td>
<td>0</td>
<td>...</td>
<td>200</td>
<td>7</td>
</tr>
<tr>
<td>Person B</td>
<td>f</td>
<td>23</td>
<td>89</td>
<td>1</td>
<td>...</td>
<td>150</td>
<td>13</td>
</tr>
<tr>
<td>Person C</td>
<td>f</td>
<td>42</td>
<td>90</td>
<td>0</td>
<td>...</td>
<td>190</td>
<td>5</td>
</tr>
</tbody>
</table>

### Fitness Tracking App

<table>
<thead>
<tr>
<th>name</th>
<th>avg_active_mins</th>
<th>avg_rhr</th>
<th>avg_sleep</th>
<th>avg_emot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person A</td>
<td>120</td>
<td>65</td>
<td>8h30m</td>
<td>happy</td>
</tr>
<tr>
<td>Person B</td>
<td>40</td>
<td>70</td>
<td>5h30m</td>
<td>uninterested</td>
</tr>
<tr>
<td>Person S</td>
<td>300</td>
<td>52</td>
<td>6h30m</td>
<td>uninterested</td>
</tr>
</tbody>
</table>
Cross Device Federated Learning

Credits: Google AI Blog - Collaborative Machine Learning without Centralizing Data
Cross Silo Federated Learning
Centralized Federated Learning
Decentralized Federated Learning
Federated Aggregation

- Area of active research
- Simple approach: aggregate weights or gradients from local models.
- Secure aggregation: aggregate encrypted local updates and decrypt the result.
- Several caveats, discussed in the Challenges section.
Building a Minimal FL System
The Ingredients

- Centrally Coordinating Server
- A modelling and data processing utility
- A communication channel - we use websockets
- A medium to transfer local updates - we use Kafka
- Naive model averaging
- Tracking History
A Small Note

- Socketio - enables real-time bidirectional event-based communication between clients and a server.
- Kafka - a distributed event or message streaming platform that allows you to work with a Producer Consumer pattern.
class Server:
    def __init__(self):
        pass

    async def connect(self, sid, environ):
        pass
class Server:
    def __init__(self):
        pass

async def connect(self, sid, environ):
    # connect with nodes, start training on min nodes

async def start_round(self):
    # start a training round and send global model
class Server:
    def __init__(self):
        pass

    async def connect(self, sid, environ):
        # connect with nodes, start training on min nodes

    async def start_round(self):
        # start a training round and send global model

    async def fl_update(self, sid, data):
        # receive ack for updates

    def consume_updates(self):
        # consume updates when all updates are received
class Server:
    ....

    async def fl_update(self, sid, data):
        # receive ack for updates

    def consume_updates(self):
        # consume updates when all updates are received

    def aggregate(self, client_mapped_weights):
        # aggregate weights for layers with trainable weights

    def evaluate(self, aggregated_weights):
        # Evaluate on a holdout set

    def store_history(self):
        # Store federated losses across rounds
The Recipe - Client

class Node:
    def __init__(self):
        pass

    async def connect(self, sid, environ):
        # Connect to the server
class Node:
    def __init__(self, address, partition, client, epochs):
        pass

    def connect(self):
        # Connect to server

    def connection_received(self):
        # Get ack from server
class Node:

    def start_training(self, _model):
        # get model from json
        # compile model
        # fit
        # evaluate
        # send updates
        pass

    def fit(self, model):
        pass

    def send_updates(self, loss):
        # encode individual layers as b64 strings
        # produce over the updates topic on a Kafka server
        pass
class Node:

def end_session(self, data):
    # get latest model weights
    # update local model
    # Clean up as necessary
    pass

def disconnect(self):
    # Disconnect signal from server
Opportunities in FL
● The Non IID data conundrum
● Collaborations and Partnerships are difficult
  ○ No control over data collection stack
  ○ How do you do EDA?
● Technical failures:
  ○ Network latency
  ○ Connection dropouts
  ○ Corrupted local updates
Not a standalone solution to privacy:
  - Add noise by way of Differential Privacy

Larger attack surface with multiple nodes, if not implemented correctly.

LONG LIVE THE REVOLUTION.
OUR NEXT MEETING WILL BE AT THE DOCKS AT MIDNIGHT ON JUNE 29.

AHA, FOUND THEM!

 WHEN YOU TRAIN PREDICTIVE MODELS ON INPUT FROM YOUR USERS, IT CAN LEAK INFORMATION IN UNEXPECTED WAYS.
Conclusion

Build Privacy into solutions proactively.
Thanks!

Additional Resources:
- Code
- Advances and Open Problems in FL
- Google Federated Web comic

Find me here: