Federated Learning

Dr. Ram Prasad Reddy Sadi
Associate Professor
Overview

✓ Basics of Federated Learning
✓ Linear Models
✓ Decision Trees
✓ XGBoost
Enterprise Data is **Fragmented**

- Moving data across silos is costly, risky, and slow
- Duplication is expensive
- AI applications must be built and run across different clouds
- Data privacy restrictions prohibit Cross—Border Data Transfers
What is Federated Learning?

Federated Learning enables training models across disparate systems, keeping data in place.

**Objective**
Collaboratively build AI models with any data, no matter where it lives by combining outputs from different clouds across your applications

More Data = Better Models

---

**Small N Scenario**
- Robots in Mars, IoT & Edge

**Large N Scenario**
- Personal Data & Legislations
  - IoT, Smartphone, GDPR, HIPPA

**Connectivity Constraints**
- Competitors
  - Cable Companies, Banks

---

<table>
<thead>
<tr>
<th>Manufacturing Equipment</th>
<th>Healthcare</th>
<th>Mobile Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictive Maintenance at the Edge</td>
<td>Computer Vision Across Consortium of Data Sets Patient Analytics</td>
<td>Location Driven Predictions Auto Completion</td>
</tr>
</tbody>
</table>

Financial Services
Fraud Detection
Money Laundering
Basic Federated Framework

Aggregator (A)

Party 1 (P1)
Party 2 (P2)
...  
Party N (PN)
Basic Federated Framework
1. **Aggregator** queries each parties about information necessary for learning a predictive model. (e.g. Weights, Gradients, Samples Counts).
Basic Federated Framework

• **Aggregator** queries each **parties** about information necessary for learning a predictive model. (e.g. Weights, Gradients, Samples Counts).

• Given the **query (Q)**, each party computes a **reply (R)** based on their own **local data (D)**.
Basic Federated Framework

• **Aggregator** queries each **parties** about information necessary for learning a predictive model. (e.g. Weights, Gradients, Samples Counts).
• Given the **query** (**Q**), each party computes a **reply** (**R**) based on their own **local data** (**D**).

Each party then sends its computed **reply** (**R**) back to the aggregator, where the results are then fused together as a single **model** (**M**).
Basic Federated Framework

- **Aggregator** queries each **parties** about information necessary for learning a predictive model. (e.g. Weights, Gradients, Samples Counts).
- Given the **query** (Q), each party computes a reply (R) based on their own **local data** (D).

Each party then sends its computed reply (R) back to the aggregator, where the results are then fused together as a single **model** (M).

**Key Point:**
Raw data from each party is **never shared**, it **remains where it is stored**.
### Different Learning Paradigms in Federated Learning

FFL provides a method for how and where the learning occurs.

Every ML algorithm falls into one of the three following categories defined by:

1. **Model Structure**
2. **Learning Process (i.e. Loss)**
3. **Location of Model Updates**

<table>
<thead>
<tr>
<th>Static Model</th>
<th>Hybrid Model</th>
<th>Dynamic Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predefined fixed model architecture.</td>
<td>Fixed loss structure, with dynamic architecture.</td>
<td>Predefined dynamic model architecture.</td>
</tr>
<tr>
<td>Learning (i.e. model weight updates) occurs at party side.</td>
<td>Learning process occurs in both aggregator and party side.</td>
<td>Learning (i.e. tree growth) occurs at aggregator.</td>
</tr>
<tr>
<td>Fusion of the models occur at the aggregator.</td>
<td>Parties perform loss-based computation (i.e. compute gradients, hessians, etc).</td>
<td>Parties perform very simple queries (i.e. get simple count values).</td>
</tr>
</tbody>
</table>

**Examples:**
- **Linear Models**
- **Neural Networks**
- **XGBoost**
- **Decision Trees**
Linear Models in Federated Learning

Key Highlights

• Supports Popular Linear Models
• Supports Different Fusion Algorithms
• Fast and Easy to Train
• Compatible with Standard Scikit-Learn Libraries with Fast Production Ready Deployment
IBM FL Supports Linear Models

Logistic Regression
• Classification Model
• Regression Model

Linear SVM
• Linear Support Vector Machine Model

Regularized Linear Models
• L2 Regularization: A standard regularizer for Linear SVM
• L1 Regularization: Lasso Model, Feature Selection, Sparse Model
• L1+L2 Regularization: Elasticnet Model, Feature Selection, Sparse Model
Supported Fusion Algorithms for Linear Models

Standard Fusion Algorithms
- Simple Average
- FedAvg (Weighted Average)

Robust Fusion Algorithms
- Coordinate Median

Query:
Local Party’s Model Weights

REPEAT PROCES UNTIL MAX ROUNDS.

WM = Aggregation over W1, W2, …, WN Based on Fusion Algorithm

Aggregator (A)

Party 1 (P1)
Party 2 (P2)
Party N (PN)

Supported Fusion Algorithms
Robust Fusion Algorithms
Example: Multiclass Classification

Dataset: MNIST* - Handwritten Digits

Experiment Setup:
• Logistic Classifier + l1 Regularizer (Feature Selection)

• 3 parties, each of them has 1000 training data samples and 3333 test samples, all randomly draw from the MNIST dataset.

Simple average fusion algorithm selected

3 global training rounds, each with 10 local epochs
Flask communication
Example (Continued)

### Global Model Performance at Each Party

<table>
<thead>
<tr>
<th>Party</th>
<th>Accuracy</th>
<th>F1</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party 1</td>
<td>0.896</td>
<td>0.896</td>
<td>0.895</td>
<td>0.896</td>
</tr>
<tr>
<td>Party 2</td>
<td>0.909</td>
<td>0.907</td>
<td>0.907</td>
<td>0.909</td>
</tr>
<tr>
<td>Party 3</td>
<td>0.896</td>
<td>0.895</td>
<td>0.895</td>
<td>0.894</td>
</tr>
</tbody>
</table>

### Party 1 Local Model Performance (Centralized)

<table>
<thead>
<tr>
<th>Party 1</th>
<th>Accuracy</th>
<th>F1</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Party 1</td>
<td>0.853</td>
<td>0.852</td>
<td>0.854</td>
<td>0.856</td>
</tr>
</tbody>
</table>
Decision Trees in Federated Learning

Key Highlights

• Good Explainability & Model Interpretability
• Handling Categorical Data Type
• High Accuracy for Binary and Multiclass Classification Problems
IBM FL Supports ID3 Decision Trees

- Decision Trees grows at the Aggregator Side
- Parties Won’t Reveal its Local Dataset to the Aggregator
- Handles Categorical Data Types

**Aggregator (A)**

**Query:** Provide the feature values, ask for label counts.

**WM** = Sum over C1, C2, ..., CN
Find the best feature value to split based on information gain formula
**Example: Binary Classification**

A glimpse of the adult dataset:

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>State-gov, 77516, Bachelors, 13, Never-married, Adm-clerical, Not-in-family, White, Male, 21-24, 0, 40, United-States, &lt;=50K</td>
</tr>
<tr>
<td>50</td>
<td>Self-emp-not-inc, 83311, Bachelors, 13, Married-civ-spouse, Exec-managerial, Husband, White, Male, 0, 0, 13, United-States, &lt;=50K</td>
</tr>
<tr>
<td>30</td>
<td>Private, 215640, HS-grad, 9, Divorced, Handlers-cleaners, Not-in-family, White, Male, 0, 0, 40, United-States, &lt;=50K</td>
</tr>
<tr>
<td>53</td>
<td>Private, 234731, 11th, 7, Married-civ-spouse, Handlers-cleaners, Husband, Black, Male, 0, 0, 40, United-States, &lt;=50K</td>
</tr>
</tbody>
</table>

A highly imbalanced dataset

**Dataset:** Adult* - "Census Income" dataset

**Experiment Setup:**
- 3 parties, each of them has 1000 training samples randomly draw from the Adult dataset. The whole Adult dataset (32561 samples) as a global test set.
- Additional pre-processing steps: binning on most features, and drop 'fnlwgt' feature

ID3 fusion algorithm
Maximum tree depth is set to 3
Flask communication
Example (Continued)

Tree structure trained via FL differs from that of Party 1:

Global model performance at the global test set:

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=50K</td>
<td>0.88</td>
<td>0.86</td>
<td>0.90</td>
</tr>
<tr>
<td>&gt;50K</td>
<td>0.58</td>
<td>0.54</td>
<td>0.63</td>
</tr>
</tbody>
</table>

Party 1 local model performance (centralized)

<table>
<thead>
<tr>
<th></th>
<th>F1</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=50K</td>
<td>0.87</td>
<td>0.86</td>
<td>0.89</td>
</tr>
<tr>
<td>&gt;50K</td>
<td>0.58</td>
<td>0.55</td>
<td>0.61</td>
</tr>
</tbody>
</table>
XGBoost in Federated Learning

Extreme Gradient Boosting (XGBoost)

• XGBoost is a decision tree-based ensemble method which utilizes a gradient-boosting based approach for optimizing against the loss function.

• Gradient Boosting methods have demonstrated state-of-the-art performance in various supervised tasks.

• Highly utilized in various settings such as classification, regression, and ranking based problems.

• Recently popular among the Kaggle community for its use in various machine learning competitions.
XGBoost in Framework for Federated Learning

XGBoost is now offered in IBM FL, supporting **classification** & **regression** models.

Classification Models:
- Binary (Binary Cross Entropy Loss)
- Multiclass (Categorical Entropy Loss)

Regression Models:
- Least Squares

**Key Advantages of XGBoost in Federated Learning**

1. Model Interpretability
2. Handles Both Categorical and Numerical Features
3. Less Preprocessing required & Handles Missing data
4. Joint Ensemble Modeling Method – Robust Performance Across All Parties’ Data *(Especially for Non-IID Data)*
Preliminary: XGBoost Basics

Given the following objective function defined previously:

$$Obj = \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k} \Omega(f_k)$$

Our optimization objective would entail learning the best CART function via an Additive Training method in an iterative manner.

1. Define an initial fixed function.
2. Learn a new tree which we append to the previous fixed function.
3. Repeat until some criterion (i.e. max tree, metrics, etc...).

For each leaf in the tree, to add a split we compute the resulting gain based on the split to find the best feature and value to split on:

$$Gain = \frac{1}{2} \left( \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right) - \gamma$$

- Score of Left Child
- Score of Right Child
- Score if We Do Not Split
- Complexity Cost Incurred by Adding Leaf
Participants (P)
X,G,H

---

STEP 1: Initialize Aggregator Model \( f_0^{(A)} \) and obtain data set size from each party.

STEP 2: Send current model and epsilon parameter from Aggregator.

STEP 3: Using given base aggregator model \( f_0^{(A)} \), compute predictions on local participant’s dataset. Compute the gradient and hessian statistics of the prediction. Perform quantile sketch on feature values using \( \epsilon \) parameter. Aggregate gradient statistics with respect to the binned split value proposals.

STEP 4: Send back corresponding (approximate) split value, gradient and hessian statistics from each candidate.

STEP 5: Merge aggregated values as single distribution and merge gradient statistics accordingly.

STEP 6: Send back split proposal, gradient, and hessian statistics. Perform split finding and build model recursively.

---

XGBoost in Federated Learning

REPEAT PROCES UNTIL MAX ROUNDS. CHOOSE BEST MODEL FROM ALL_rounds.

---

Aggregator (A)

---

Aggregated Split Finding Process

---

Participants (P)
Few Recent Works on Federated Learning

✓ Advances and Open Problems in Federated Learning (2019)
✓ Novel Privacy-preserved Recommender System Framework Based on Federated Learning (2020)
✓ The Future of Digital Health with Federated Learning (2020)
Thank You