CS-E4740 - Federated Learning

L1 - From ML to FL

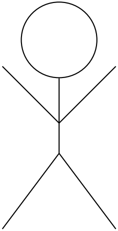
Assoc. Prof. Alexander Jung

Spring 2026

Calendar **Glossary** **Book** **GitHub**

Table of Contents

The Right Song Can Save the Day



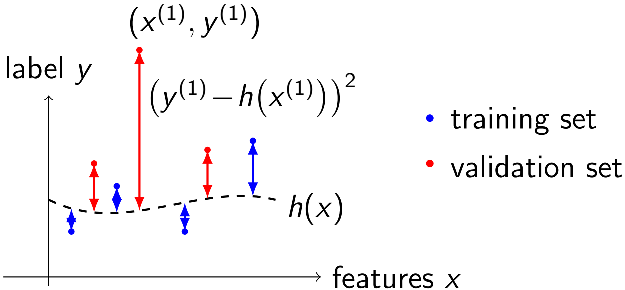
user: Alex Jung
mood: tired
evening

$\xrightarrow{h(x)}$ **Bohemian Rhapsody**

How do we get a good hypothesis map $h(x)$?

Wang, M., Wu, J., Yan, H. (2023). "Effect of music therapy on older adults with depression: A systematic review and meta-analysis." *Complementary Therapies in Clinical Practice* https://doi.org/10.1016/j.ctcp.2023.101809

Empirical risk minimization (ERM)



Learn $h \in \mathcal{H}$ by min. average loss (empirical risk),

$$\min_{h \in \mathcal{H}} \frac{1}{m} \sum_{r=1}^{m} L\left(\left(\mathbf{x}^{(r)}, y^{(r)}\right), h\right).$$

Different choices for $\mathcal{H}$ and loss L yield different ML methods.

see Chapters 3,4 of AJ, "Machine Learning: The Basics," Springer, 2022.
https://mlbook.cs.aalto.fi

ML with Python

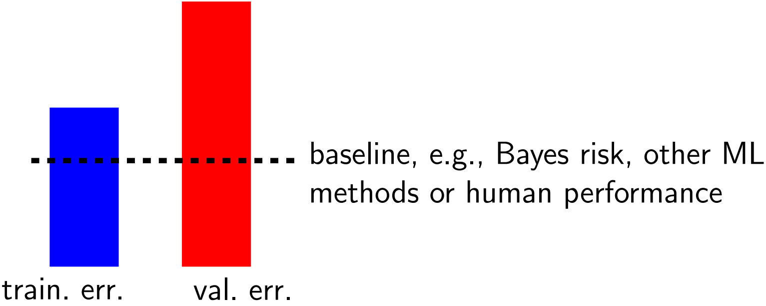
```
X, y = read_data()

# split data
Xtr, Xval, ytr, yval = train_test_split(X, y)

# train model
model = SGDRegressor()
model.fit(Xtr, ytr)

# compute errors
train_err = mean_squared_error(ytr, model.predict(Xtr))
val_err   = mean_squared_error(yval, model.predict(Xval))
```

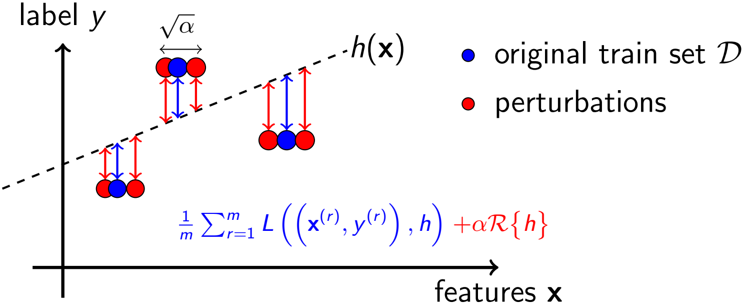
Applied ML - Diagnosis



compare training error with validation error and a baseline

see Chapter 6 of AJ, "Machine Learning: The Basics," Springer, 2022. https://mlbook.cs.aalto.fi

Applied ML - Regularization



Start with large $\mathcal{H}$, then shrink it via (combinations of)

- data augmentation, e.g., $\mathbf{x} \mapsto \mathbf{x} + \mathcal{N}(0, \alpha)$,
- adding penalty term to loss function, e.g., $\ldots + \alpha \|\mathbf{w}\|_2^2$,
- constraining model parameters, e.g., $\|\mathbf{w}\|_2 \leq 1$.

see Chapter 7 of AJ, "Machine Learning: The Basics," Springer, 2022.
https://mlbook.cs.aalto.fi

Table of Contents

From ML to FL

Basic ML. train a single model $\mathcal{H}$ by minimizing average loss on a single dataset

FL. train several models $\mathcal{H}^{(i)}$ using interconnected devices

a device is anything that can

- access data,
- train a model, and
- communicate with other devices

From ML to FL

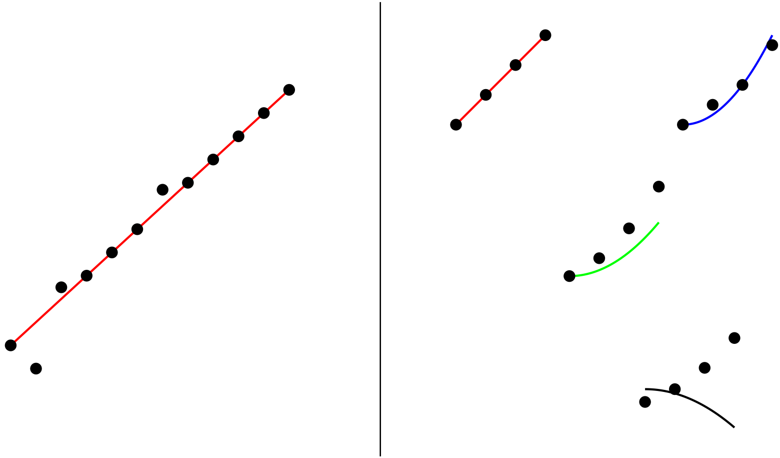


Figure: Left: A ML method uses a single dataset to train a single model. Right: FL methods train ML models from decentralized data.

ML with Python

```
X, y = read_data()
model = SGDRegressor()
model.fit(X, y)
```

FL with Python

IP: 192.168.0.1

```
model = SGDRegressor()
y_hat = recv_preds(192.168.0.3)
X, y = read_data()
Xa,ya = augment_data(X, y, y_hat)
model.fit(Xa,ya)
```

IP: 192.168.0.2

```
X,y = read_data()
model = LinearRegression()
model.fit(X, y)
```

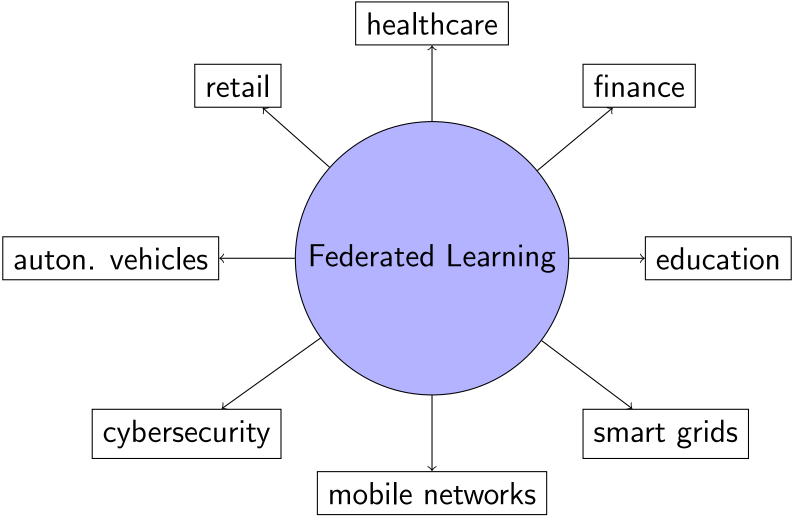
IP: 192.168.0.3

```
model = DecisionTree()
y_hat = recv_preds(192.168.0.2)
X, y = read_data()
Xa,ya = augment_data(X, y, y_hat)
model.fit(Xa,ya)
```

Key Characteristics of FL

- can be fully de-centralized (no single point of failure)
- each device trains a tailored model (high-precision)
- scalability: more devices yield more compute and data
- no raw data is shared (privacy-friendly)

FL Applications



FL for Pandemics

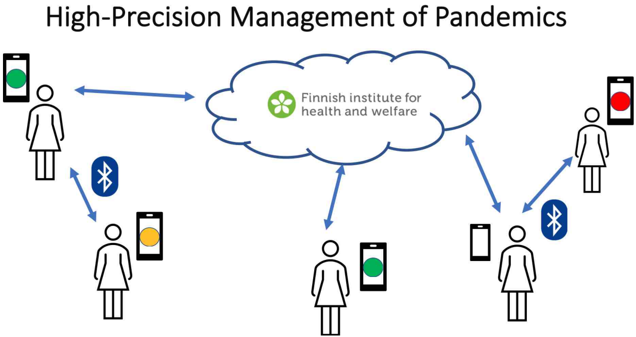


Figure: A hypothetical FL system for pandemic forecasting. Smartphones train personalized models based on their observations (e.g., audio recordings of coughing) as well as public health-care data.

FL in Healthcare

- turn smartphone into personal health-care advisor
- smartphone app uses FL to train personalized model.
- combine personal data with public health-care data.

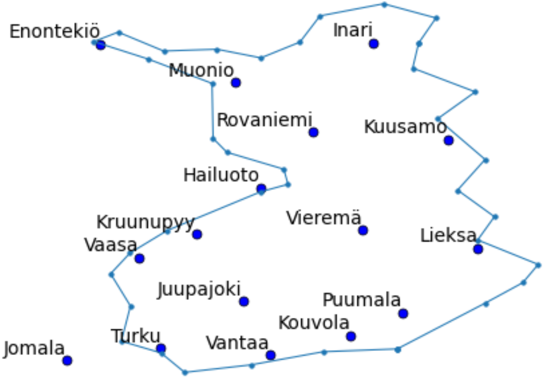
Rieke, N., et al. *The future of digital health with federated learning.* Nature Medicine, 2020.

FL in Finance

FL can help financial institutions to improve

- **Fraud detection.** N. F. Aurna, et.al., "Federated Learning-Based Credit Card Fraud Detection: Performance Analysis with Sampling Methods and Deep Learning Algorithms," 2023,
- **Risk assessment.** W. Li, et.al., "Personal Credit Evaluation Model Based on Federated Learning," 2024

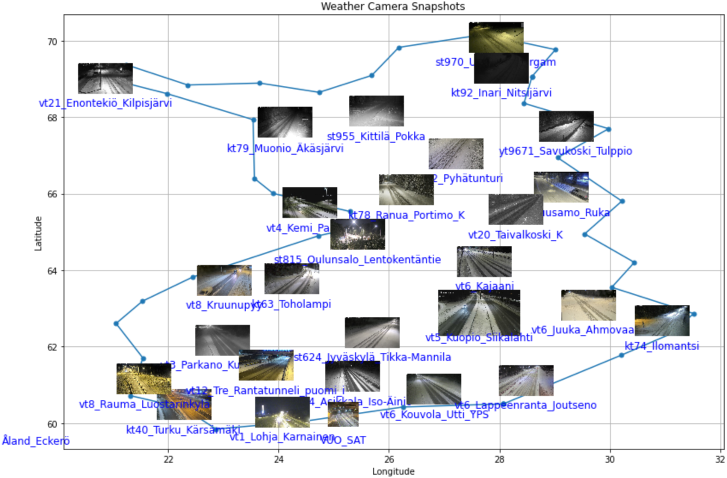
FL at FMI



Train a separate model for each Finnish Meteorological Institute (FMI) weather station

Python script for reproducing the Fig.:

FL for Finnish Road Safety



Train separate model for each camera operated by FinTraffic

Python script for reproducing the Fig.:

The Internet of Things (IoT) is Growing

IoT connections (billion)

IoT	2023	2029	CAGR
Wide-area IoT	3.6	7.2	12%
Cellular IoT	3.4	6.7	12%
Short-range IoT	12.1	31.6	17%
Total	**15.7**	**38.8**	**16%**

Note: Based on rounded figures. Cellular IoT figures are also included in the figures for wide-area IoT.

Figure: Some IoT statistics from .

The IoT - A Global FL System

Table of Contents

A “Real-World” FL system

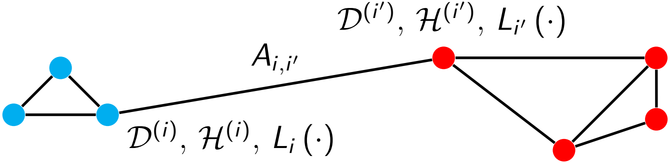
Abstracting Away System Details

to reason about an FL system, we deliberately ignore many implementation details such as

- properties of communication links (latency, bandwidth)
- communication protocols and message formats
- hardware and operating systems of devices
- the precise version numbers of `Python` packages

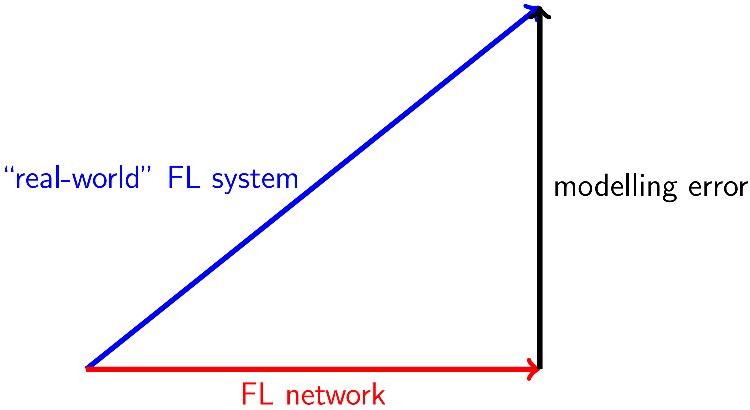
Goal: isolate the *essential structure* of a FL system

The FL network as an Abstraction



- an FL network is an undirected graph with nodes $i=1,\ldots,n$
- edge $\{i,i'\}$ with weight $A_{i,i'} > 0$ encodes collaboration
- each node i holds local dataset $\mathcal{D}^{(i)}$ and trains model $\mathcal{H}^{(i)}$
- a local dataset induces a local loss function $L_i(\cdot)$

FL network is an Approximation



A Precise Definition

An FL network consists of:

- a finite set of **nodes**, denoted as $\mathcal{V} := \{1, \ldots, n\}$
- a **local model** $\mathcal{H}^{(i)}$ at each node $i \in \mathcal{V}$
- a **local loss function** $L_i(\cdot)$ at each node $i \in \mathcal{V}$
- a set of undirected **edges**, denoted as $\mathcal{E}$
- a positive **edge weight** $A_{i,i'} > 0$ for each edge $\{i, i'\} \in \mathcal{E}$

"FL network $=$ undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ $+$ model and loss function attached to each node"

Building a FL network in Python

```
import networkx as nx
from sklearn.linear_model import
   LinearRegression

# Step 1: Create an undirected FL network
G = nx.Graph()
num_clients = 5
G.add_nodes_from(range(num_clients))
G.add_edges_from([(0,1), (1,2), (2,3), (3,4)])

# Step 2: Attach a local model to each node
for node in G.nodes:
  G.nodes[node]["model"] = LinearRegression(
   fit_intercept=False)

# Example: access local model at client 2
local_model = G.nodes[2]["model"]
```

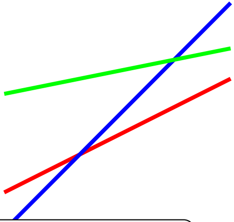
Nodes of an FL network

- consider an FL system with finite number n of devices
- we index devices as $i = 1, \ldots, n$
- indices form the set of nodes $\mathcal{V}$ in an FL network
- node $i \in \mathcal{V}$ **represents** a physical device
- we use "device i" and "node i" interchangeably

Local models

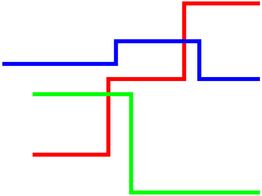
- consider an FL system with devices $i = 1, \ldots, n$
- each device trains local (personal) model $\mathcal{H}^{(i)}$
- devices might use (very) different local models
- we use local model parameters $\mathbf{w}^{(i)}$ for parametric $\mathcal{H}^{(i)}$

$\mathcal{H}^{(i)}$



`model=LinearRegression()`

$\mathcal{H}^{(i')}$



`model=DecisionTreeRegressor()`

Local Loss functions

- consider device i, training its local model $\mathcal{H}^{(i)}$.
- to train a model is to learn a useful hypothesis $h^{(i)} \in \mathcal{H}^{(i)}$.
- measure usefulness of $h^{(i)}$ by a local loss function

$$L_i\left(\cdot\right) : \mathcal{H}^{(i)} \rightarrow \mathbb{R} : h^{(i)} \mapsto L_i\left(h^{(i)}\right)$$

- different devices can use different loss functions.

Local Loss functions - ctd.

- FL methods use different constructions of loss functions
- for parametric models $\mathcal{H}^{(i)}$, with model parameters $\mathbf{w}^{(i)} \in \mathbb{R}^d$,

$$L_i\left(\cdot\right) : \mathbb{R}^d \rightarrow \mathbb{R} : \mathbf{w}^{(i)} \mapsto L_i\left(\mathbf{w}^{(i)}\right)$$

- can use average loss on local dataset

$$L_i\left(\mathbf{w}^{(i)}\right) := \frac{1}{m_i} \sum_{r=1}^{m_i} \left(y^{(i,r)} - \left(\mathbf{w}^{(i)}\right)^T \mathbf{x}^{(i,r)}\right)^2$$

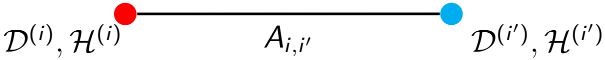
- loss can also be estimated from a reward signal

Edges of an FL network

- FL network consists of **undirected weighted** edges $\mathcal{E}$
- $\{i, i'\} \in \mathcal{E}$ means **collaboration** between devices i and i'
- **extent of collaboration is edge weight** $A_{i,i'} > 0$
- we view edges primarily as a **design choice**

Effect of Placing an Edge

FL algorithms are executed over an FL network



placing an edge $\{i, i'\} \in \mathcal{E}$ has two consequences:

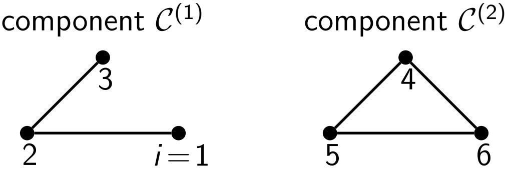
- requires communication channel between devices i, i' (edge weight $A_{i,i'} \approx$ channel capacity).
- model parameters at i, i' are forced to be similar.

Connectivity of an FL network

consider an FL network with graph $\mathcal{G}$.

- $\mathcal{G}$ is **connected** if there is a path between any $i, i' \in \mathcal{V}$.
- a **component** $\mathcal{C} \subseteq \mathcal{V}$ is a connected subgraph with no edges between $\mathcal{C}$ and $\mathcal{V} \setminus \mathcal{C}$.
- the **neighborhood** of $i \in \mathcal{V}$ is $\mathcal{N}^{(i)} := \{i' \in \mathcal{V} : \{i, i'\} \in \mathcal{E}\}$.
- **weighted node degree** of i is $d^{(i)} := \sum_{i' \in \mathcal{N}^{(i)}} A_{i,i'}$.
- **maximum node degree** is $d_{\mathrm{max}} := \max_{i \in \mathcal{V}} d^{(i)}$.

Connectivity of an FL network- Example



- FL network containing $n=6$ nodes.
- uniform edge-weights, $A_{i,i'}=1$ for all $\{i,i'\}\in\mathcal{E}$.
- two components $\mathcal{C}^{(1)}=\{1,2,3\},\mathcal{C}^{(2)}=\{4,5,6\}$.
- $d^{(1)}=1$, $\mathcal{N}^{(2)}=\{1,3\}$, $d_{\max}=2$.

From FL network to FL system

each node $i \in \mathcal{V}$,

- can access local dataset $\mathcal{D}^{(i)}$,
- maintains model parameters $\mathbf{w}^{(i)}$
- sends/receives messages from neighbors $\mathcal{N}^{(i)}$.

an FL algorithm specifies *when* and *how* these model parameters are updated.

FL Algorithms

each node i uses some of current model parameters $\mathbf{w}^{(1,t)},\ldots,\mathbf{w}^{(n,t)}$ to compute new model parameters $\mathbf{w}^{(i,t+1)}$,

$$\mathbf{w}^{(i,t+1)} = \mathcal{F}^{(i)}\big(\mathbf{w}^{(1,t)},\ldots,\mathbf{w}^{(n,t)}\big) \text{ at time instants } t = 0, 1, \ldots.$$

the node-wise operator $\mathcal{F}^{(i)}$ includes

- local model updates (e.g., via gradient steps)
- sharing model parameters across edges of FL network

Table of Contents

Laplacian matrices

consider a FL system that is represented (or modelled) by a FL network with graph $\mathcal{G}$

the properties of a FL system depends crucially on the connectivity structure of the underlying FL network

the connectivity structure can be analyzed via the Laplacian matrix associated with $\mathcal{G}$

Definition of Laplacian matrix

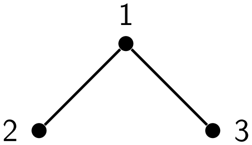
- consider FL network with a weighted, undirected graph $\mathcal{G}$
- associated Laplacian matrix $\mathbf{L}^{(\mathcal{G})} \in \mathbb{R}^{n \times n}$ is defined element-wise as:

$$L^{(\mathcal{G})}_{i,i'} := \begin{cases} -A_{i,i'} & \text{for } i \neq i', \{i,i'\} \in \mathcal{E} \\ \sum_{i'' \neq i} A_{i,i''} & \text{for } i = i' \\ 0 & \text{else.} \end{cases}$$

Note: the main diagonal entries are the node degrees $d^{(i)}$, for $i = 1, \ldots, n$

Laplacian matrix - Example

graph $\mathcal{G}$ with uniform edge weights $A_{i,i'} = 1$



$$\mathbf{L}^{(\mathcal{G})} = \begin{pmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

Properties of the Laplacian matrix

The Laplacian matrix $\mathbf{L}^{(\mathcal{G})}$ of an FL network is

- symmetric $\mathbf{L}^{(\mathcal{G})} = \left(\mathbf{L}^{(\mathcal{G})}\right)^T$ (since edges are undirected)
- and positive semi-definite (psd),

$$\mathbf{w}^T \mathbf{L}^{(\mathcal{G})} \mathbf{w} \geq 0 \text{ for every } \mathbf{w} \in \mathbb{R}^n. \quad (1)$$

The psd property (1) follows from the identity

$$\mathbf{w}^T \mathbf{L}^{(\mathcal{G})} \mathbf{w} = \underbrace{\sum_{\{i,i'\} \in \mathcal{E}} A_{i,i'} \left(w^{(i)} - w^{(i')}\right)^2}_{\text{total variation}}$$

which holds for every $\mathbf{w} = \left(w^{(1)}, \ldots, w^{(n)}\right)^T \in \mathbb{R}^n$.

The Spectrum of the Laplacian matrix

- We can decompose any Laplacian matrix $\mathbf{L}^{(\mathcal{G})} \in \mathbb{R}^{n \times n}$ as

$$\mathbf{L}^{(\mathcal{G})} = \sum_{j=1}^{n} \lambda_j \mathbf{u}^{(j)} \left(\mathbf{u}^{(j)}\right)^T,$$

- with orthonormal eigenvecs. $\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(n)} \in \mathbb{R}^n$, i.e.,

$$\left(\mathbf{u}^{(j)}\right)^T \mathbf{u}^{(j')} = \begin{cases} 1 & \text{for } j = j' \\ 0 & \text{otherwise,} \end{cases}$$

- and non-neg. eigenvalues

$$0 = \lambda_1 \leq \ldots \leq \lambda_n \leq 2d_{\max}.$$

The spectrum of $\mathbf{L}^{(\mathcal{G})}$ is the set of distinct eigenvalues.

Spectral Characterization of FL Networks

FL network $\mathcal{G}$ with k connected components $\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(k)}$.

Then, the Laplacian matrix $\mathbf{L}^{(\mathcal{G})} = \sum_{j=1}^{n} \lambda_j \mathbf{u}^{(j)} \big(\mathbf{u}^{(j)}\big)^T$

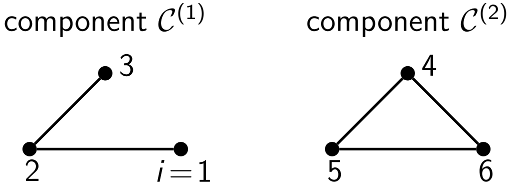
- has eigvals. $\lambda_c = 0$ for $c = 1, \ldots, k$, with
- corresponding eigvecs. $\mathbf{u}^{(c)}$, given entry-wise as

$$u_i^{(c)} = \begin{cases} \frac{1}{\sqrt{\left|\mathcal{C}^{(c)}\right|}} & \text{for } i \in \mathcal{C}^{(c)} \\ 0 & \text{otherwise.} \end{cases}$$

$\boxed{\mathcal{G} \text{ is connected } (k=1) \text{ if and only if } \lambda_2 > 0.}$

Spectral Clustering - Toy Example

Consider a FL network $\mathcal{G}$ with two components:



- The Laplacian matrix has two zero eigvals. $\lambda_1 = \lambda_2 = 0$.
- What are corresp. eigvecs. $\mathbf{u}^{(1)}, \mathbf{u}^{(2)}$? Are they unique?

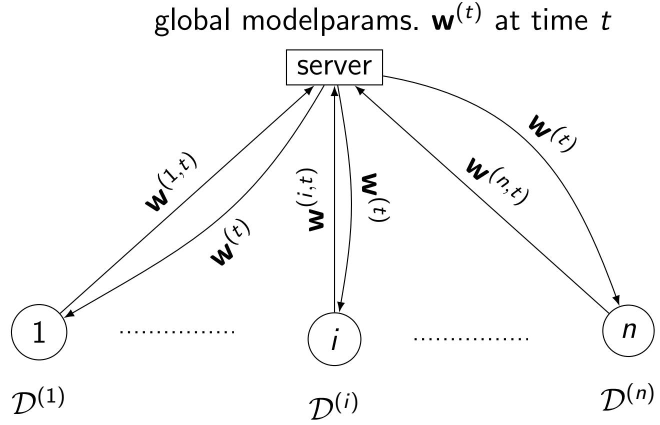
Table of Contents

Server-Client FL

at iteration (time instant) t,

- server holds global model parameters $\mathbf{w}^{(t)} \in \mathbb{R}^d$.
- clients $i = 1, \ldots, n$ carry local datasets $\mathcal{D}^{(i)}$
- use $\mathcal{D}^{(i)}$ to compute update $\mathbf{w}^{(t)} \mapsto \mathbf{w}^{(i,t)}$
- sever aggregates $\mathbf{w}^{(i,t)}$ to update $\mathbf{w}^{(t)} \mapsto \mathbf{w}^{(t+1)}$

Server-Client Implementation



- client i computes $\mathbf{w}^{(i,t)}$ using $\mathbf{w}^{(t)}$ and $\mathcal{D}^{(i)}$
- server aggregates $\mathbf{w}^{(1,t)}, \ldots, \mathbf{w}^{(n,t)}$ to compute $\mathbf{w}^{(t+1)}$

Horizontal federated learning (HFL)

$$\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} & y^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} & y^{(2)} \\ x_1^{(3)} & x_2^{(3)} & \cdots & x_d^{(3)} & y^{(3)} \\ x_1^{(4)} & x_2^{(4)} & \cdots & x_d^{(4)} & y^{(4)} \\ x_1^{(5)} & x_2^{(5)} & \cdots & x_d^{(5)} & y^{(5)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_d^{(m)} & y^{(m)} \end{bmatrix} \begin{array}{l} \mathcal{D}^{(1)} \\ \\ \\ \mathcal{D}^{(i)} \\ \\ \\ \mathcal{D}^{(n)} \end{array}$$

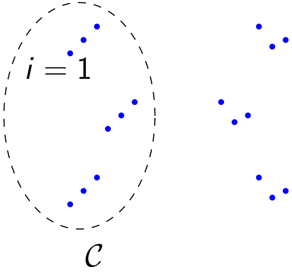
local datasets are (overlapping) subsets of a single underlying global dataset

Vertical federated learning (VFL)

$$\begin{array}{c} \mathcal{D}^{(1)} \qquad \mathcal{D}^{(i)} \\ \underbrace{\begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \cdots & x_d^{(1)} & y^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \cdots & x_d^{(2)} & y^{(2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_1^{(m)} & x_2^{(m)} & \cdots & x_d^{(m)} & y^{(m)} \end{bmatrix}}_{\mathcal{D}} \end{array}$$

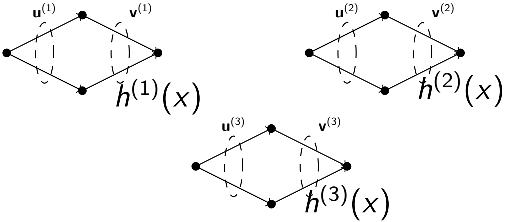
local datasets contain same data points but using different features

Clustered federated learning (CFL)



- devices form clusters
- devices in same cluster $\mathcal{C}$ have stat. similar datasets

Personalized FL



- partitioned model parameters $\mathbf{w}^{(i)} = \left(\left(\mathbf{u}^{(i)}\right)^T, \left(\mathbf{v}^{(i)}\right)^T \right)^T$
- collaborate only for learning $\mathbf{u}^{(i)}$ (input layer)
- no collaboration for $\mathbf{v}^{(i)}$ (output layer)

Table of Contents

Wrap Up

- basic ML trains single model from single dataset
- FL uses collection of collaborating devices
- each device has local dataset and a local model
- different FL flavours use different forms of collaboration between devices

What's Next?

L2-"FL Design Principle" introduces generalized total variation minimization (GTVMin) as our main design principle for FL algorithms.

We use GTVMin to guess useful choices for the node-wise update operator $\mathcal{F}^{(i)}$ that define an FL algorithm.

References

- AJ, "Machine Learning: The Basics," Springer, 2022. available via Aalto library.
- AJ, "Federated Learning: From Theory to Practice," Springer, 2026.
- AJ et.al., "The Aalto Dictionary of Machine Learning," 2026. https://aaltodictionaryofml.github.io/