CS-E4740 - Federated Learning

FL Algorithms

Assoc. Prof. Alexander Jung

Spring 2025

Playlist

Glossary

Course Site

Outline

Recap and Learning Goals

Applying Gradient Methods to GTVMin

Federated Learning Algorithms

Asynchronous FL Algorithms

Table of Contents

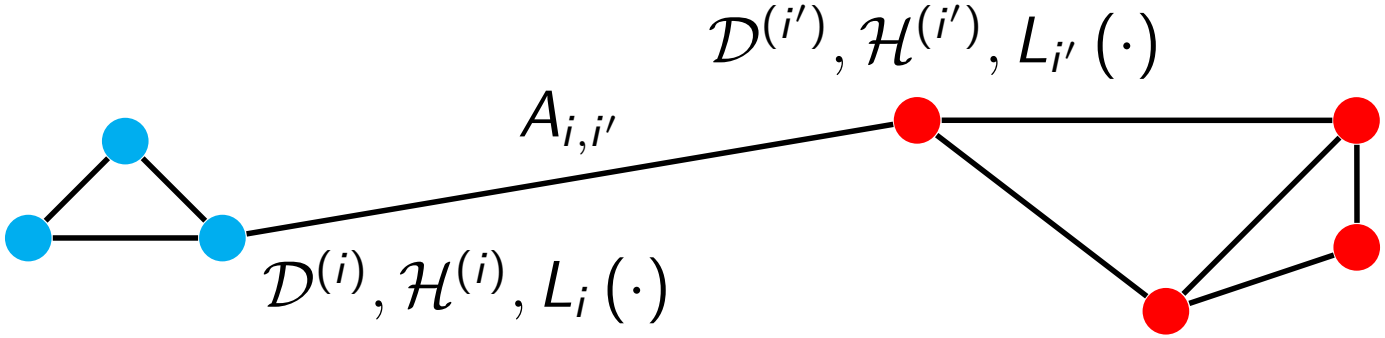
Recap and Learning Goals

Applying Gradient Methods to GTVMin

Federated Learning Algorithms

Asynchronous FL Algorithms

FL Network as Mathematical Model for FL



- An FL network consists of devices $i = 1, \ldots, n$.
- Some i, i' connected by edge with the weight $A_{i,i'} > 0$.
- Device i **generates data** $\mathcal{D}^{(i)}$ and **trains model** $\mathcal{H}^{(i)}$.
- Data $\mathcal{D}^{(i)}$ used to construct loss func. $L_i(\cdot)$.

GTV Minimization (for Parametric Models)

We train local models in a collaborative fashion by solving

$$\min_{\mathbf{w}^{(1)},\ldots,\mathbf{w}^{(n)}} \sum_{i=1}^{n} L_i\left(\mathbf{w}^{(i)}\right) + \alpha \sum_{\{i,i'\}\in\mathcal{E}} A_{i,i'} \left\|\mathbf{w}^{(i)} - \mathbf{w}^{(i')}\right\|_2^2 \quad \text{(GTVMin)}.$$

- Solution consists of learnt modelparams. $\widehat{\mathbf{w}}^{(i)}$.
- Tuning parameter $\alpha \geq 0$ controls clustering of $\widehat{\mathbf{w}}^{(i)}$.
- For $\alpha = 0$, GTVMin reduces to separate ERM for each i.
- Increasing α makes $\widehat{\mathbf{w}}^{(i)}$ more similar across nodes i.

Learning Goals

After completing this module, you know how

- FL alg. can be obtained from **gradient descent**,
- to implement GD as **message passing**,
- to generalize GD to handle **non-parametric models**,
- to implement **asynchronous FL algos.**

Table of Contents

Recap and Learning Goals

Applying Gradient Methods to GTVMin

Federated Learning Algorithms

Asynchronous FL Algorithms

Gradient Step for GTVMin

Starting from initial local params. $\mathbf{w}^{(i,0)}$, repeat grad. steps

$$\mathbf{w}^{(i,k+1)} = \mathbf{w}^{(i,k)} - \eta_{k,i} \left[\nabla L_i \left(\mathbf{w}^{(i,k)} \right) + 2\alpha \sum_{i' \in \mathcal{N}^{(i)}} A_{i,i'} \left(\mathbf{w}^{(i,k)} - \mathbf{w}^{(i',k)} \right) \right]$$

- The learn. rate $\eta_{k,i}$ determines extent of update.
- $\nabla L_i \left(\mathbf{w}^{(i,k)} \right)$ **steers** update towards **min. local loss**.
- $\left(\mathbf{w}^{(i,k)} - \mathbf{w}^{(i',k)} \right)$ **steers** update to **agree with neigh.**
- $\alpha A_{i,i''}$ balances those two steering effects.

Synchronous Operation

The gradient step

$$\mathbf{w}^{(i,k+1)} = \mathbf{w}^{(i,k)} - \eta_{k,i} \left[\nabla L_i \left(\mathbf{w}^{(i,k)} \right) + 2\alpha \sum_{i' \in \mathcal{N}^{(i)}} A_{i,i'} \left(\mathbf{w}^{(i,k)} - \mathbf{w}^{(i',k)} \right) \right]$$

has to be carried out by all nodes $i = 1, \ldots, n$.

When these (local) gradient steps are completed, each node shares its new model params. with its neighbours.

After sharing the model params., start new iteration $k := k+1$.

Message Passing Implementation

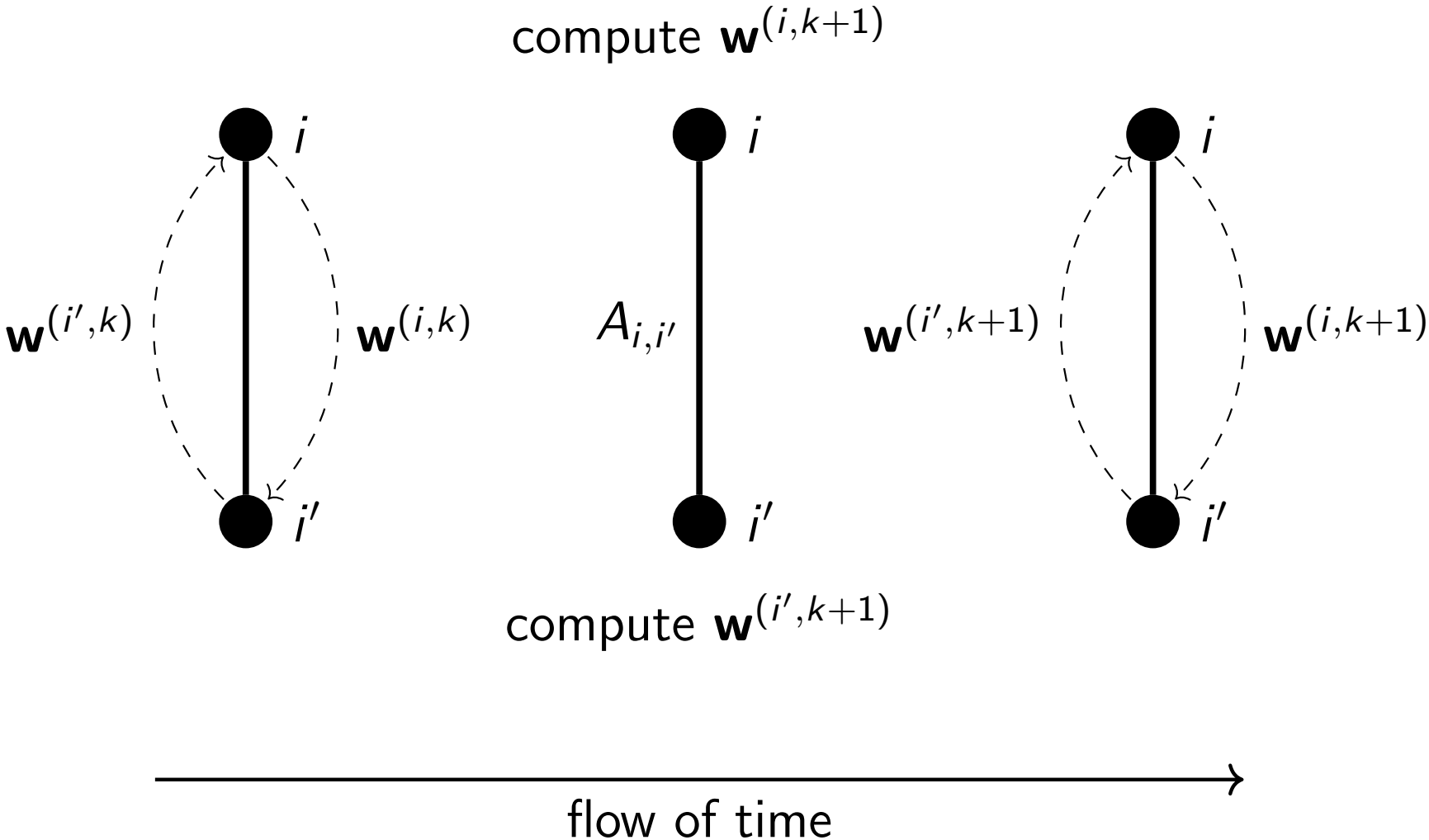


Table of Contents

Recap and Learning Goals

Applying Gradient Methods to GTVMin

Federated Learning Algorithms

Asynchronous FL Algorithms

Federated Gradient Descent (FedGD)

Each node $i = 1, \ldots, n$ initializes

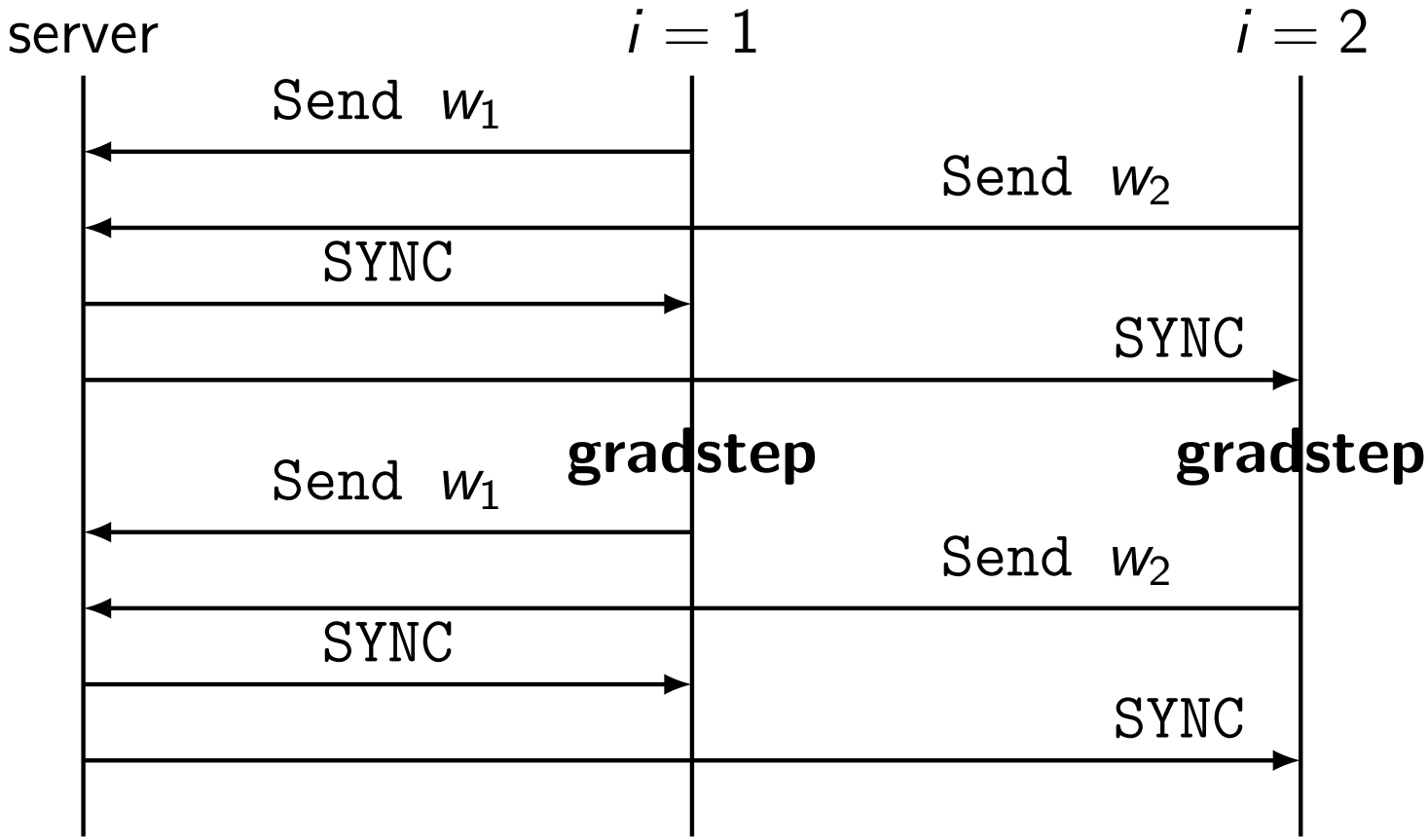
- local model params. $\widehat{\mathbf{w}}_0^{(i)} := \mathbf{0}$, and
- iteration counter $k := 0$.

Repeat the following steps at each node i:

- Send $\widehat{\mathbf{w}}_k^{(i)}$ to all neighbours $\mathcal{N}^{(i)}$.
- Do a gradient step.
- Increment iteration counter $k := k + 1$.

CAUTION: Nodes must execute steps synchronously!

Implementing FedGD with a Sync-Server



Python demo

Federated Stochastic Gradient Descent (FedSGD)

- Consider FL network with node i carrying local dataset

$$\mathcal{D}^{(i)} = \left\{ \left(\mathbf{x}^{(i,1)}, y^{(i,1)}\right), \ldots, \left(\mathbf{x}^{(i,m_i)}, y^{(i,m_i)}\right)\right\}.$$

- Node i uses local loss function

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

- FedGD requires to compute gradient,

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

Stochastic Gradient Approximation

For some applications, the computation of

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

is intractable, e.g., too many data points or too slow access.

$\Rightarrow$ Use instead a sum over random subset $\mathcal{B} \subseteq \{1, \ldots, m_i\}$,

$$\sum_{r \in \mathcal{B}} \mathbf{x}^{(i,r)} \left(y^{(i,r)} - \left(\mathbf{w}^{(i)}\right)^T \mathbf{x}^{(i,r)} \right).$$

We refer to $\mathcal{B}$ as batch with batch size $|\mathcal{B}|$.

Federated Averaging (FedAvg)

- Some FL applications use common model at all nodes,

$$\mathbf{w}^{(i)} = \mathbf{w}^{(i')} \quad \forall\, i, i' \in \mathcal{V}.$$

- GTVMin becomes constrained optimization problem:

$$\min_{\mathbf{w}^{(1)},\ldots,\mathbf{w}^{(n)}} \sum_{i \in \mathcal{V}} L_i\left(\mathbf{w}^{(i)}\right) \text{ s.t. } \quad \mathbf{w}^{(i)} = \mathbf{w}^{(i')} \quad \forall\, i, i' \in \mathcal{V}.$$

- For diff. $L_i\left(\mathbf{w}^{(i)}\right)$ we can apply projected GD.

- Projection step amounts to averaging $(1/n)\sum_{i=1}^{n} \mathbf{w}^{(i)}$.

(Almost) FedAvg

Init. counter (clock) $k := 0$ and model params $\widehat{\mathbf{w}} := \mathbf{0}$.

1. **Broadcast.** Server sends $\widehat{\mathbf{w}}$ to all nodes $i \in \mathcal{V}$.
2. **Local Gradient Step.** Each node computes

$$\mathbf{w}^{(i,k)} = \widehat{\mathbf{w}} - \eta_{k,i} \nabla L_i \left(\widehat{\mathbf{w}} \right).$$

3. **Collect.** Nodes send $\mathbf{w}^{(i,k)}$ back to server.
4. **Aggregate.** Server computes $\widehat{\mathbf{w}} := (1/n) \sum_{i=1}^{n} \mathbf{w}^{(i,k)}$.
5. **Clock Tick.** Server increments $k := k + 1$. Go to step 1.

FedAvg

We obtain FedAvg via the following modifications:[1]

- Use (stochastic) approximations of gradients.
- Instead of single GD step, compute several GD steps.
- Only a subset of nodes compute local updates during each iteration.

[1] B. McMahan et.al., Communication-Efficient Learning of Deep Networks from Decentralized Data, PMLR, 2017

FedProx

FedProx replaces GD steps in FedAvg with[2]

$$\mathbf{w}^{(i)} := \underset{\mathbf{v}\in\mathbb{R}^d}{\operatorname{argmin}} \left[L_i(\mathbf{v}) + (1/\eta) \left\| \mathbf{v} - \widehat{\mathbf{w}}^{(\text{global})} \right\|_2^2 \right].$$

Empirical studies found FedProx to result in more robust FL systems compared to FedAvg.

FedProx seems to handle well varying computational power of devices.

[2] T. Li, et.al, Federated Optimization in Heterogeneous Networks, Proc. of Machine Learning and Systems 2, 2020.

Federated Relaxation (FedRelax)

- Consider GTVMin objective function

$$f\left(\mathbf{w}^{(1)},\ldots,\mathbf{w}^{(n)}\right)=\sum_{i=1}^{n} L_i\left(\mathbf{w}^{(i)}\right)+\alpha\sum_{\{i,i'\}\in\mathcal{E}} A_{i,i'}\left\|\mathbf{w}^{(i)}-\mathbf{w}^{(i')}\right\|_2^2.$$

- Complicated due to coupling terms $A_{i,i'}\left\|\mathbf{w}^{(i)}-\mathbf{w}^{(i')}\right\|_2^2$.

- Without coupling, GTVMin would be much easier.

- Optimize $f(\cdot)$ w.r.t. $\mathbf{w}^{(i)}$, holding $\left\{\mathbf{w}^{(i')}\right\}_{i'\in\mathcal{V}\setminus\{i\}}$ fixed![3]

[3] Similar idea is used in the Jacobi method for solving linear equations.

FedRelax for Parametric Models

- **Init.** Set counter $k := 0$, local model params. $\widehat{\mathbf{w}}_0^{(i)} := \mathbf{0}$.
- **Repeat until stopping criterion:**
  - Each node i shares $\widehat{\mathbf{w}}_k^{(i)}$ with neighbours $\mathcal{N}^{(i)}$.
  - **Local Update.** Each node i computes

$$\widehat{\mathbf{w}}_{k+1}^{(i)} := \underset{\mathbf{w}^{(i)} \in \mathbb{R}^d}{\operatorname{argmin}} \, L_i\left(\mathbf{w}^{(i)}\right) + \alpha \sum_{i' \in \mathcal{N}^{(i)}} A_{i,i'} \left\| \mathbf{w}^{(i)} - \widehat{\mathbf{w}}_k^{(i')} \right\|_2^2 .$$

  - **Clock Tick.** $k := k + 1$.

Note the similarity of local update with ridge regression!

FedRelax for Non-Parametric Models

- **Init.** $k := 0$, construct test-set $\mathcal{D}^{\{i,i'\}}$ for each $\{i, i'\} \in \mathcal{E}$
- **Repeat until stopping criterion:**
  - Each i shares $\widehat{h}_k^{(i)}(\mathbf{x})$ for each $\mathbf{x} \in \mathcal{D}^{\{i,i'\}}$ and $i' \in \mathcal{N}^{(i)}$.
  - **Local Update.** Each node i computes

$$\widehat{h}_{k+1}^{(i)} \in \underset{h^{(i)} \in \mathcal{H}^{(i)}}{\operatorname{argmin}} L_i\left(h^{(i)}\right) + \alpha \sum_{i' \in \mathcal{N}^{(i)}} A_{i,i'} D\left(h^{(i)}, \widehat{h}_k^{(i')}\right).$$

  - **Clock Tick.** $k := k + 1$.

Here, we use the discrepancy measure

$$D\left(h^{(i)}, h^{(i')}\right) := (1/m') \sum_{\mathbf{x} \in \mathcal{D}^{\{i,i'\}}} \left[h^{(i)}(\mathbf{x}) - h^{(i')}(\mathbf{x})\right]^2.$$

Table of Contents

Recap and Learning Goals

Applying Gradient Methods to GTVMin

Federated Learning Algorithms

Asynchronous FL Algorithms

FL Algorithms are Fixed-Point Iterations

- Consider FL net. with parametric local models.
- FL algorithms presented so far can be written as

$$\mathbf{w}^{(i,k+1)} = \mathcal{F}^{(i)}\left(\mathbf{w}^{(1,k)}, \ldots, \mathbf{w}^{(n,k)}\right).$$

- This is a synchronous fixed-point iteration with operators

$$\mathcal{F}^{(i)} : \mathbb{R}^{nd} \to \mathbb{R}^{d} \text{ , for each node } i = 1, \ldots, n.$$

Challenges of Synchronous FL

Implementing synchronous fixed-point iteration is challenging.

- Devices might have limited computational resources.
- The evaluation of local loss might require data collection.
- Message passing over unreliable wireless links..
- Devices might spontaneously join or drop out.

Asynchronous FL Algorithms

- Consider a FL algorithms with fixed point operators $\mathcal{F}^{(i)}$.
- We obtain an asynchronous variant by the update

$$\mathbf{w}^{(i,k+1)} = \mathcal{F}^{(i)}\left(\mathbf{w}^{(1,k_{i,1})}, \ldots, \mathbf{w}^{(n,k_{i,n})}\right).$$

- k is not a clock tick but only some event index.
- $k - k_{i,i'}$ represents communication delay from i' to i.