Distributed Learning and Decentralized learning

Zhen Qin

2025

Talk Overview

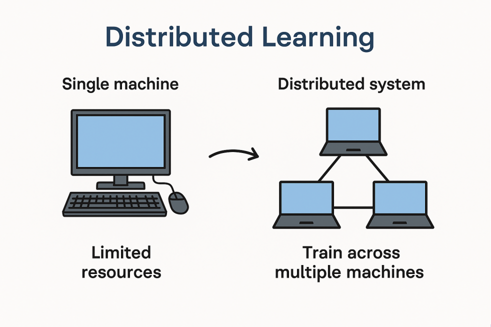
- Motivation for Distributed Learning
- Federated vs. Decentralized Paradigms
- Math Formulation of Decentralized Optimization
- Multi-Agent Image Regression Example
- Consensus Mechanism
- DSGD: Algorithm and Intuition
- Key Challenges and Summary

Motivation for Distributed Learning

- Modern datasets (e.g., images, video, logs) are massive.
- Deep learning models can have billions of parameters. (Chatgpt3: 175 billions)
- Training on a single machine faces:
  - Memory constraints
  - Computational bottlenecks
  - Long training time (days or weeks)

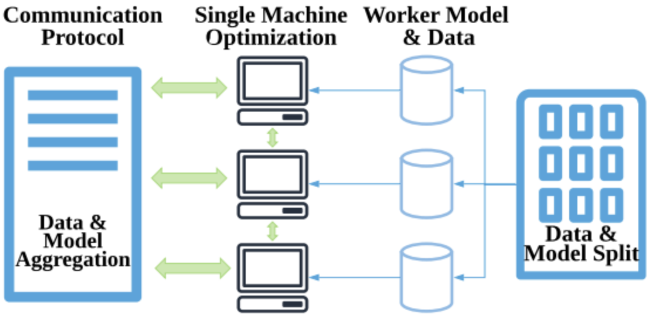
Motivation for Distributed Learning

- Modern datasets (e.g., images, video, logs) are massive.
- Deep learning models can have billions of parameters. (Chatgpt3: 175 billions)
- Training on a single machine faces:
  - Memory constraints
  - Computational bottlenecks
  - Long training time (days or weeks)
- Distributed learning splits both data and model to compute them across nodes.



What Is Distributed Learning?

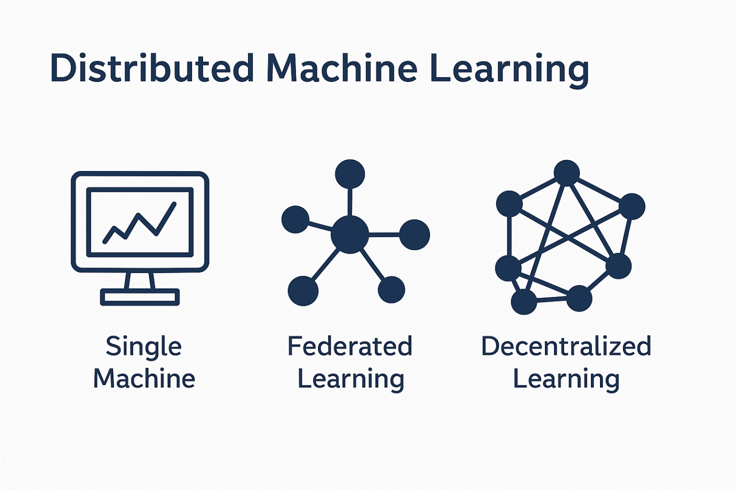
- Train one global model across multiple data holders.
- Each device performs local computation on its own data.
- Communication and coordination needed to combine insights.



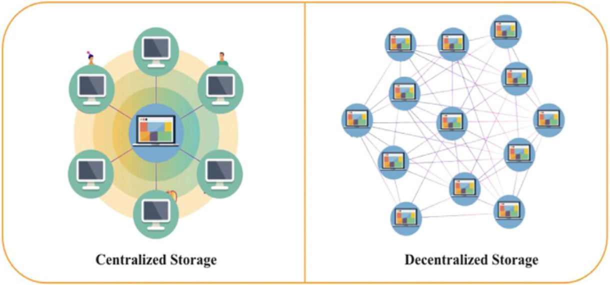
What Is Distributed Learning?

Two paradigms:

- **Federated Learning (FL)**: with a server
- **Decentralized Learning (DL)**: no server



Federated vs Decentralized Learning



Federated Learning

- Clients train locally
- Server aggregates model updates
- Server failure = system failure

Decentralized Learning

- No central server
- Each node communicates with neighbors
- More robust to node or link failures

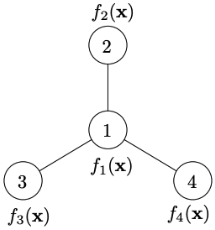
Key difference: Coordination architecture

Why No Server in Distributed Learning?

1. **Networks With No Infrastructure**
   - Ad hoc sensor networks for environmental monitoring
   - Multi-agent systems: autonomous vehicles, UAVs, robotics
   - Battlefield autonomous swarms
   - In-situ disaster recovery
   - Networks using random access (e.g., CSMA, ALOHA)
2. **Security, Robustness, and Privacy**
   - Avoid single point of failure
   - Reduce attack surface (no centralized target)
   - Prevent communication bottlenecks
   - Preserve information privacy
   - Prevent centralized control or manipulation
3. **Economic and Social Motivation**
   - Enable fair competition or cooperation between entities
   - Establish trust among autonomous parties
   - Support personalization and diversity
   - Avoid dominance by centralized infrastructure

Math Formulation of Decentralized Optimization

- The network is a connected undirected graph: $G = (\mathcal{N}, \mathcal{L})$
- $|\mathcal{N}| = N$: number of nodes
  $|\mathcal{L}| = L$: number of communication edges
- $x \in \mathbb{R}^d$: the global model to be learned

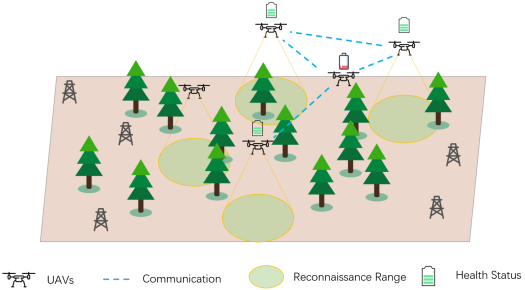


- Each node i can only evaluate a local loss: $f_i(x) = \mathbb{E}_{\xi_i \sim \mathcal{D}_i}[f_i(x, \xi_i)]$
- Global objective: $f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x)$
- Goal: collaboratively minimize $f(x)$ without a central server

Example: Decentralized Learning in Multi-UAV Systems

Scenario:

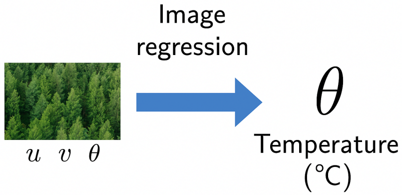
- A fleet of UAVs (Unmanned Aerial Vehicles, i.e., drones) explores a geographic region.
- Each UAV collects high-resolution, geo-tagged images of the environment.
- The learning objective is to predict a physical quantity such as ground temperature or elevation from the image.
- UAVs are connected in a communication graph and share model parameters with neighbors.



Example: Decentralized Learning in Multi-UAV Systems

Regression Model:

- Each UAV i has local dataset $\{u_{ij}, v_{ij}, \theta_{ij}\}_{j=1}^{N_i}$
- u_{ij}, v_{ij} are image feature vectors; θ_{ij} is the temperature or elevation label
- Agents aim to collaboratively solve a regression problem using a linear model:$x = \left[x_1^\top \;\; x_2^\top\right]^\top$
- Local objective: $f_i(x) = \frac{1}{N_i} \sum_{j=1}^{N_i} \left(\theta_{ij} - (u_{ij}^T x_1 + v_{ij}^T x_2)\right)^2$
- Global decentralized objective: $\min_x f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x)$



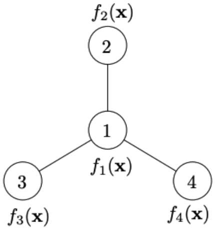
Consensus Mechanism: Reformulation

How to deal with the communications?

Consensus Mechanism: Reformulation

How to deal with the communications?
Goal: Solve the global optimization problem in a decentralized and collaborative way:

$$\min_{x \in \mathbb{R}^d} f(x) = \min_{x \in \mathbb{R}^d} \frac{1}{N} \sum_{i=1}^{N} f_i(x)$$



Consensus Reformulation:

$$\min_{\{x_i \in \mathbb{R}^d\}_{i=1}^{N}} \left\{ \frac{1}{N} \sum_{i=1}^{N} f_i(x_i) \quad \text{subject to} \quad x_i = x_j, \ \forall (i,j) \in \mathcal{L} \right\}$$

The variable x is replaced by local copies x_i, and consensus constraints ensure they agree over the network.

Recall What We Did When We Have a Server

Centralized (Server-Based) Learning:

- Each node (or client) i computes:

$$x_{i,k+1} = \bar{x}_k - \eta_k g_{i,k}$$

  where the global average is: $\bar{x}_k = \frac{1}{N}\sum_{i=1}^{N} x_{i,k}$

- This update relies on a central server to compute and broadcast $\bar{x}_k$

Decentralized Idea:

- How to approximate the average locally?

$$x_{i,k+1} = \text{"Some approximation of } \bar{x}_k\text{"} - \eta_k g_{i,k}$$

- This leads to the field of **Decentralized Consensus Optimization**

Consensus Mechanism: Computing Average

How to describe the network in math?

Consensus Mechanism: Computing Average

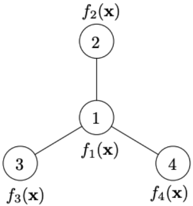
How to describe the network in math?

Consensus Matrix Setup

Let $W \in \mathbb{R}^{N \times N}$ be a consensus matrix satisfying:

- **Doubly stochastic:** $\sum_{i=1}^{N} W_{ij} = \sum_{j=1}^{N} W_{ij} = 1$
- **Sparsity pattern:** $W_{ij} > 0$ if $(i, j) \in \mathcal{L}$; $W_{ij} = 0$ otherwise
- **Symmetric:** $W_{ij} = W_{ji}$ if $(i, j) \in \mathcal{L}$

Example Network and Associated W:



$$W = \begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 3/4 & 0 & 0 \\ 1/4 & 0 & 3/4 & 0 \\ 1/4 & 0 & 0 & 3/4 \end{bmatrix}$$

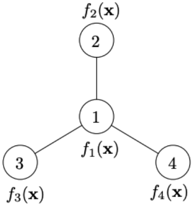
Consensus Mechanism: Computing Average

1. **Initialization:** Let $k = 0$. Each node i starts with an initial value $x_{i,0}$.
2. **Communication:** In iteration k, each node i sends $x_{i,k}$ to its neighbors $j \in \mathcal{N}(i)$.
3. **Consensus Update:** Upon receiving values from neighbors, each node updates:

   $$x_{i,k+1} = \sum_{j \in \mathcal{N}(i)} W_{ij} x_{j,k}$$

   where $W_{ij} > 0$ if $(i, j) \in \mathcal{L}$ and W is a doubly stochastic consensus matrix.
4. **Repeat:** Let $k \leftarrow k + 1$ and return to Step 2.

Consensus Mechanism: Computing Average



$$W = \begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \\ 1/4 & 3/4 & 0 & 0 \\ 1/4 & 0 & 3/4 & 0 \\ 1/4 & 0 & 0 & 3/4 \end{bmatrix}$$

Consensus Update: Upon receiving values from neighbors, each node updates:

$$x_{i,k+1} = \sum_{j \in \mathcal{N}(i)} W_{ij} x_{j,k}$$

where $W_{ij} > 0$ if $(i, j) \in \mathcal{L}$ and W is a doubly stochastic consensus matrix.

$$x_{1,k+1} = \left[0.25x_1 + 0.25x_2 + 0.25x_3 + 0.25x_4\right]$$

Decentralized Stochastic Gradient Descent (DSGD)

Steps:

1. **Initialization:** Let $k = 1$. Choose initial value $x_{i,1}$ and step size α_i for all i.
2. **Communication:** Each node i sends $x_{i,k}$ to all its neighbors $j \in \mathcal{N}(i)$.
3. **Local Update:** Upon receiving $x_{j,k}$ from all $j \in \mathcal{N}(i)$, node i performs:

$$x_{i,k+1} = \underbrace{\sum_{j \in \mathcal{N}(i)} W_{ij} x_{j,k}}_{\text{Consensus Step}} - \underbrace{\alpha_k \nabla F_i(x_{i,k}, \xi_{i,k})}_{\text{Local SGD Step}}$$

   where $\xi_{i,k}$ is a stochastic sample at node i.
4. **Iterate:** Let $k \leftarrow k + 1$ and repeat from Step 2.

Performance and Practical Challenges

- **Slower convergence** on sparse graphs
- **Data heterogeneity** causes divergence
- **Asynchrony** may cause inconsistency
- **Communication cost** limits frequency
- Gradient tracking and momentum can help

Summary and Takeaways

- Distributed learning enables parallel training.
- Decentralized learning eliminates central coordination.
- DSGD blends local SGD with peer-to-peer averaging.
- Key tradeoff: speed vs. communication cost.

Thank You