Gradient Disaggregation: Breaking Privacy in Federated Learning by Reconstructing the User Participant Matrix

Maximilian Lam, Gu-Yeon Wei, David Brooks, Vijay Janapa Reddi, Michael Mitzenmacher

HARVARD UNIVERSITY

- Background
- Threat Model & Attack
- Results
- Conclusion

Background: Federated Learning

- What is Federated Learning?
  - Collaboratively learn a shared model across clients without sending raw data to central server

Background: Federated Learning

- What is Federated Learning?
  - Collaboratively learn a shared model across clients without sending raw data to central server
- Where is Federated Learning Used?
  - Next word prediction, sentiment learning, health monitoring, content suggestion
  - Google[1], Apple[2], Facebook[3]

[1] https://ai.googleblog.com/2017/04/federated-learning-collaborative.html
[2] https://www.technologyreview.com/2019/12/11/131629/apple-ai-personalizes-siri-federated-learning/
[3] https://ai.facebook.com/research/publications/fair-resource-allocation-in-federated-learning/

Background: Federated Learning

- What is Federated Learning?
  - Collaboratively learn a shared model across clients without sending raw data to central server
- Where is Federated Learning Used?
  - Next word prediction, sentiment learning, health monitoring, content suggestion
  - Google[1], Apple[2], Facebook[3]
- Why Federated Learning?
  - Privacy! Training data is kept on device, not revealed to central server

[1] https://ai.googleblog.com/2017/04/federated-learning-collaborative.html
[2] https://www.technologyreview.com/2019/12/11/131629/apple-ai-personalizes-siri-federated-learning/
[3] https://ai.facebook.com/research/publications/fair-resource-allocation-in-federated-learning/

Background: Federated Learning

- What is Federated Learning?
  - Collaboratively learn a shared model across clients without sending raw data to central server
- Where is Federated Learning Used?
  - Next word prediction, sentiment learning, health monitoring, content suggestion
  - Google[1], Apple[2], Facebook[3]
- Why Federated Learning?
  - Privacy! Training data is kept on device, not revealed to central server
- How does Federated Learning Work?
  - ⟶

Background: Federated Learning

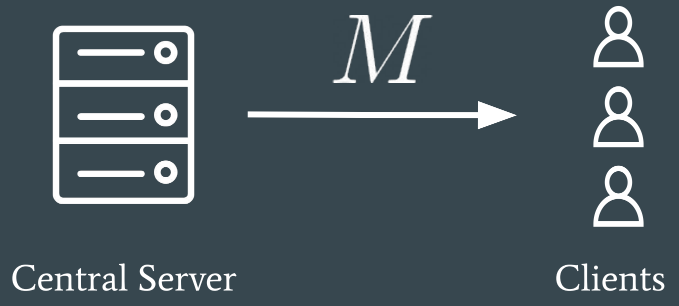
- How does Federated Learning Work?

Central Server

Clients

Background: Federated Learning

- How does Federated Learning Work?
    - 1. Central server broadcasts global model → clients



Background: Federated Learning

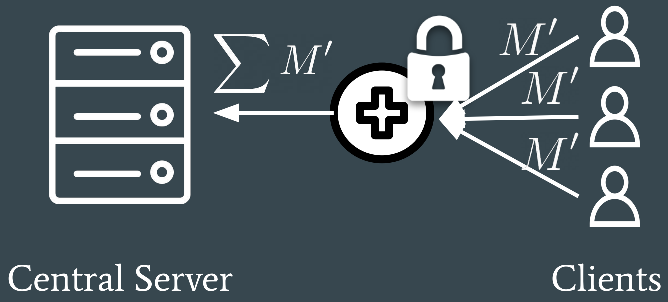
- How does Federated Learning Work?
  - 1. Central server broadcasts global model → clients
  - 2. Clients compute model update on local training data

Central Server

Clients

Background: Federated Learning

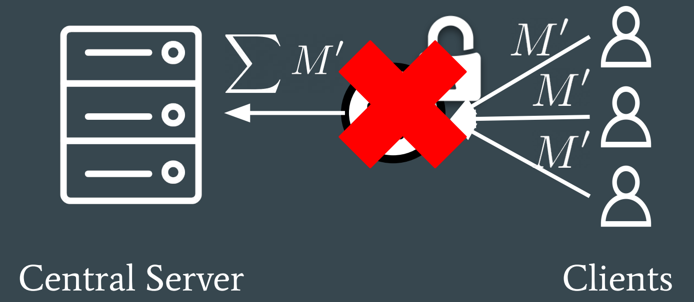
- How does Federated Learning Work?
  - 1. Central server broadcasts global model → clients
  - 2. Clients compute model update on local training data
  - 3. Server securely aggregates model updates via **Secure Aggregation**[1] → central server



1. Bonawitz et al., Practical Secure Aggregation for Privacy-Preserving Machine Learning, CCS 2017

Background: What is the Gradient Disaggregation Attack?

- Gradient Disaggregation
  - Undermines the **Secure Aggregation** protocol
  - Allows a central server to **recover individual model updates from sums + client participation counts**
    - **Individual model updates** reveal training data, violating clients' data privacy[1]



1. Zhu et al., Deap Leakage from Gradients, NeurIPS 2019

Method: Threat Model

- Core assumptions:
    1. Central server is adversarial and can fix the global model across rounds
    2. Client selection is somewhat random and a fraction is selected to participate
    3. Server has access to side channel information: client participation frequency

Method: Threat Model

- Core assumptions:

    1. Central server is adversarial and can fix the global model across rounds
    2. Client selection is somewhat random and a fraction is selected to participate
    3. Server has access to side channel information: client participation frequency

Method: Threat Model

- Core assumptions:
    1. Central server is adversarial and can fix the global model across rounds
    2. Client selection is somewhat random and a fraction is selected to participate
    3. Server has access to side channel information: client participation frequency

Method: Threat Model

- Core assumptions:
    1. Central server is adversarial and can fix the global model across rounds
    2. Client selection is somewhat random and a fraction is selected to participate
    3. Server has access to side channel information: client participation frequency[1]

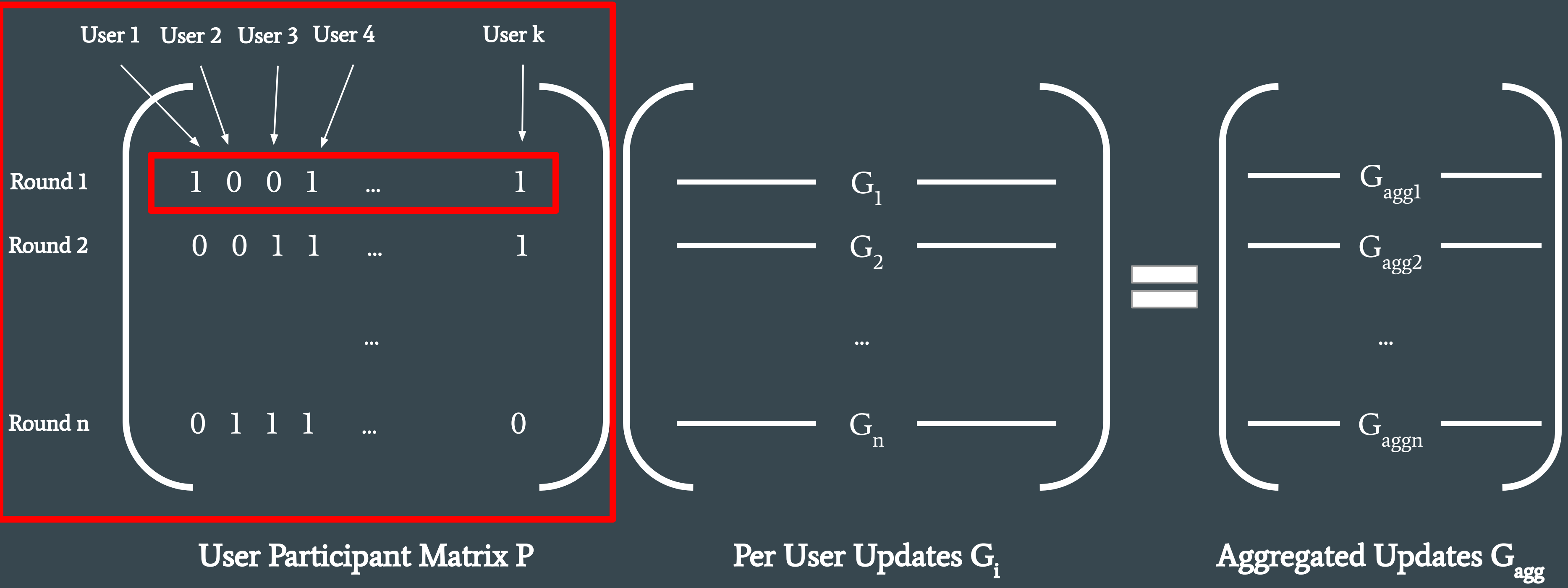
1. Bonawitz et al., Towards Federated Learning at Scale: System Design, SysML 2019

Method: Gradient Disaggregation by Reconstructing the User Participant Matrix

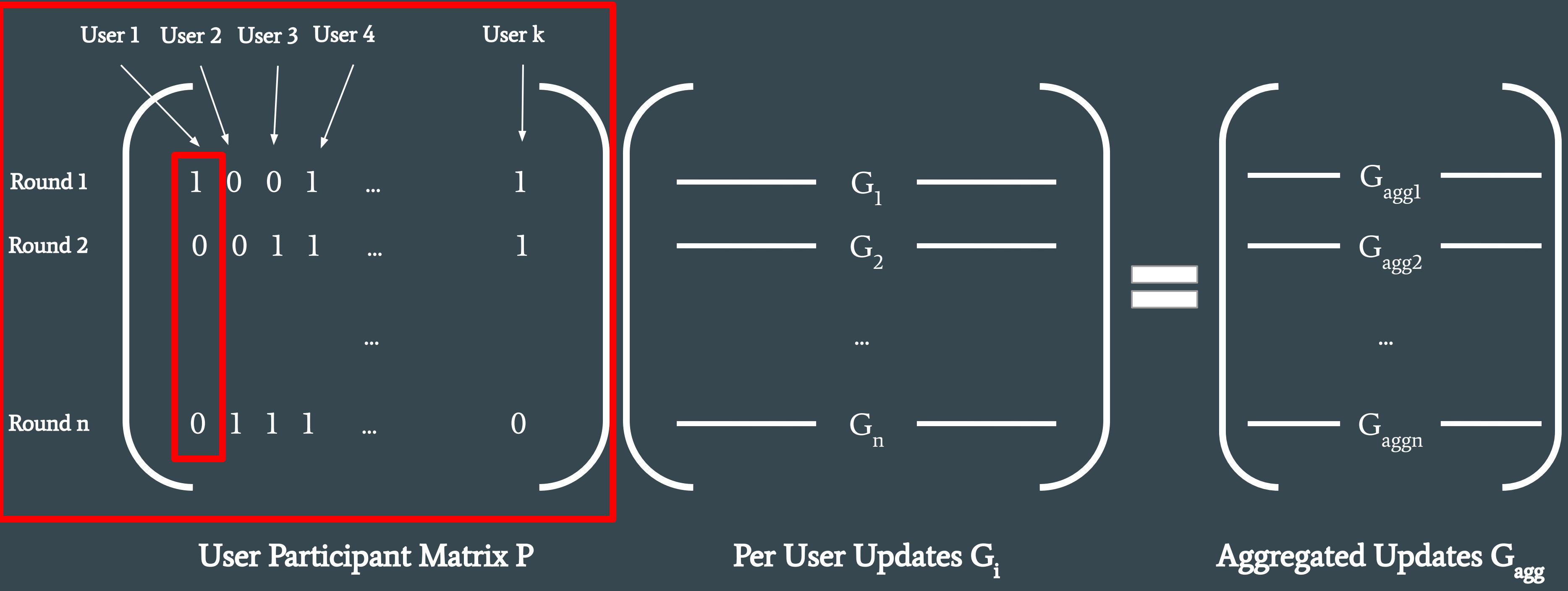
$$
\begin{pmatrix}
1 & 0 & 0 & 1 & \dots & 1 \\
0 & 0 & 1 & 1 & \dots & 1 \\
 & & & \dots & & \\
0 & 1 & 1 & 1 & \dots & 0
\end{pmatrix}
\begin{pmatrix}
\text{———} & G_1 & \text{———} \\
\text{———} & G_2 & \text{———} \\
 & \dots & \\
\text{———} & G_n & \text{———}
\end{pmatrix}
=
\begin{pmatrix}
\text{———} & G_{agg1} & \text{———} \\
\text{———} & G_{agg2} & \text{———} \\
 & \dots & \\
\text{———} & G_{aggn} & \text{———}
\end{pmatrix}
$$

User Participant Matrix P | **Per User Updates G_i** | **Aggregated Updates G_{agg}**

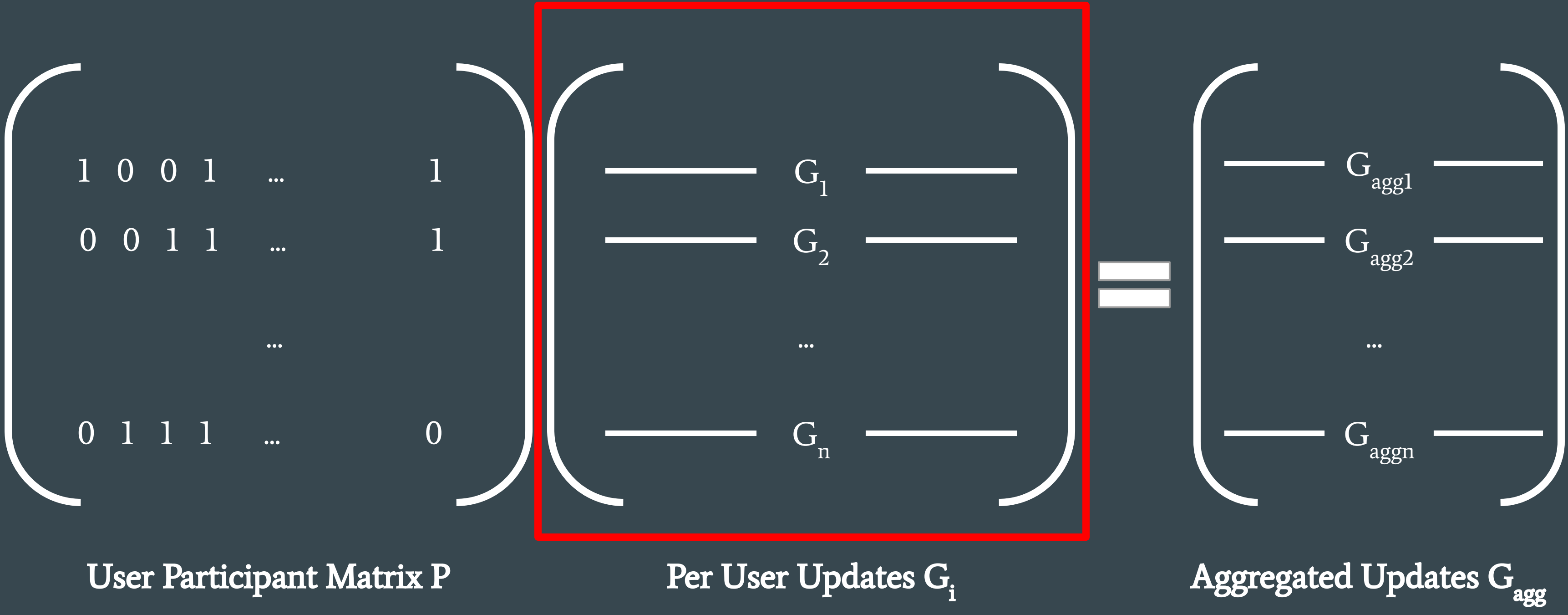
Method: Gradient Disaggregation by Reconstructing the User Participant Matrix



Method: Gradient Disaggregation by Reconstructing the User Participant Matrix



Method: Gradient Disaggregation by Reconstructing the User Participant Matrix



Method: Gradient Disaggregation by Reconstructing the User Participant Matrix

$$\begin{pmatrix} 1 & 0 & 0 & 1 & \dots & 1 \\ 0 & 0 & 1 & 1 & \dots & 1 \\ & & & \dots & & \\ 0 & 1 & 1 & 1 & \dots & 0 \end{pmatrix} \begin{pmatrix} \text{---} \; G_1 \; \text{---} \\ \text{---} \; G_2 \; \text{---} \\ \dots \\ \text{---} \; G_n \; \text{---} \end{pmatrix} = \begin{pmatrix} \text{---} \; G_{agg1} \; \text{---} \\ \text{---} \; G_{agg2} \; \text{---} \\ \dots \\ \text{---} \; G_{aggn} \; \text{---} \end{pmatrix}$$

User Participant Matrix P

Per User Updates G_i

Aggregated Updates G_{agg}

Method: Gradient Disaggregation by Reconstructing the User Participant Matrix

- Matrix factorization problem

$$PG = G_{agg}$$

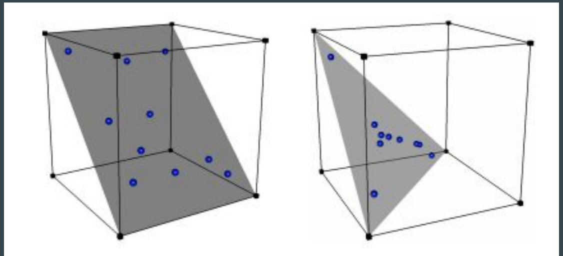
Method: Gradient Disaggregation by Reconstructing the User Participant Matrix

- Matrix factorization problem

$$PG = G_{agg}$$

- To solve we recover **columns** of P **individually**

$$\text{Find } p_k \text{ s.t.}$$
$$Nul(G_{agg}^T)p_k = 0$$
$$p_k \in \{0,1\}^n$$



Slawski et al., Matrix factorization with Binary Components, NeurIPS 2013

Method: Gradient Disaggregation by Reconstructing the User Participant Matrix

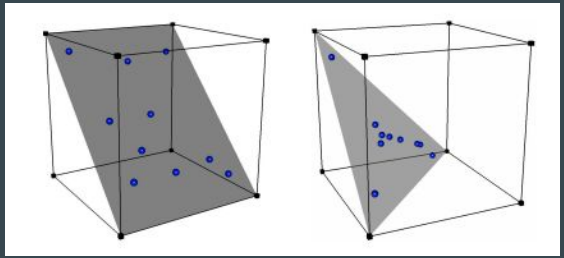
- Matrix factorization problem

$$PG = G_{agg}$$

- To solve we recover **columns** of P **individually**

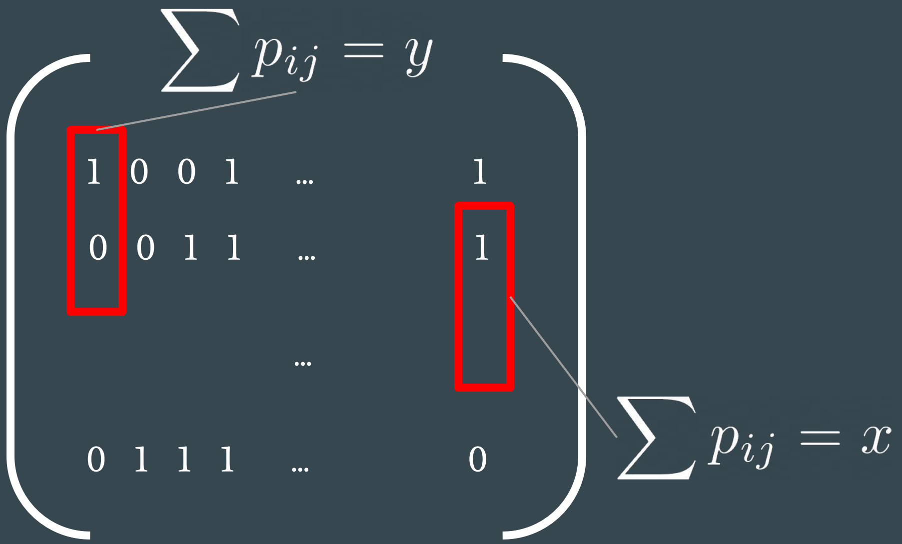
$$\text{Find } p_k \text{ s.t.}$$
$$Null(G_{agg}^T)p_k = 0$$
$$p_k \in \{0,1\}^n$$

Intractable



Slawski et al., Matrix factorization with Binary Components, NeurIPS 2013

Method: Gradient Disaggregation by Reconstructing the User Participant Matrix



User Participant Matrix P

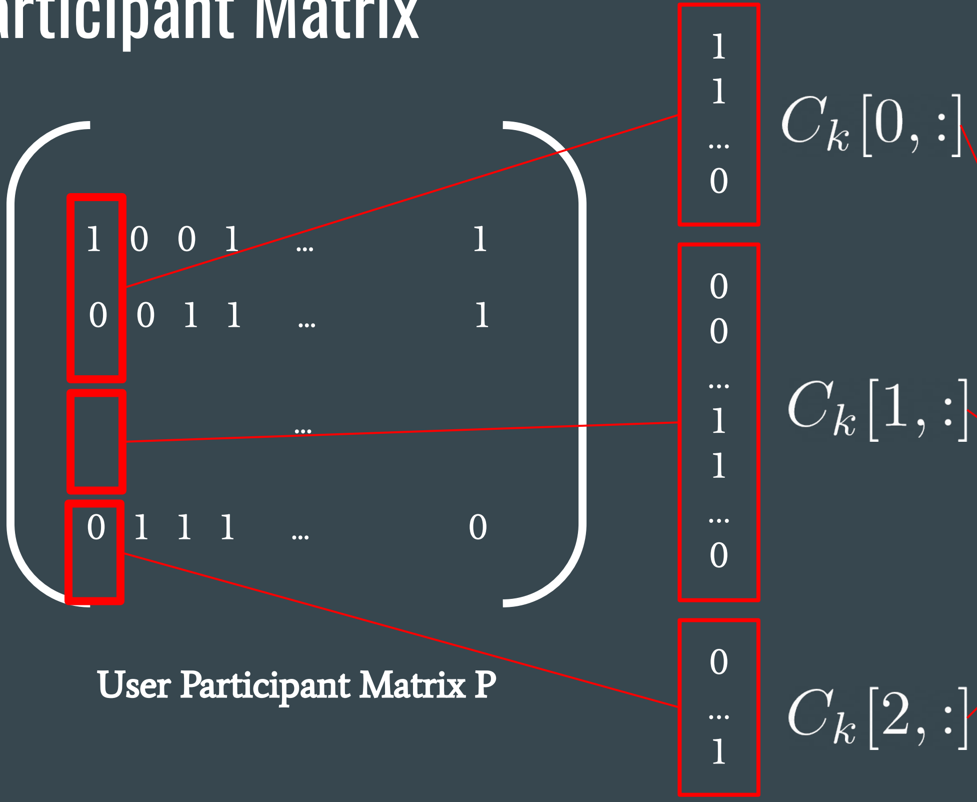
Find p_k s.t.

$$Nul(G_{agg}^T)p_k = 0$$

$$p_k \in \{0,1\}^n$$

$$C_k p_k - c_k = 0$$

Method: Gradient Disaggregation by Reconstructing the User Participant Matrix



User Participant Matrix P

Find p_k s.t.

$$Nul(G_{agg}^T)p_k = 0$$

$$p_k \in \{0,1\}^n$$

$$C_k p_k - c_k = 0$$

Method: Gradient Disaggregation by Reconstructing the User Participant Matrix

$$\text{Find } p_k \text{ s.t.}$$

$$Nul(G_{agg}^T)p_k = 0$$
$$p_k \in \{0,1\}^n$$
$$C_k p_k - c_k = 0$$

→

$$\text{Find } p_k \text{ s.t.}$$

$$\boxed{\min ||Nul(G_{aggregated}^T)p_k||^2}$$
$$p_k \in \{0,1\}^n$$
$$C_k p_k - c_k = 0$$

Method: Gradient Disaggregation by Reconstructing the User Participant Matrix

- Gradient Disaggregation Optimization
  - Framed as matrix factorization problem
    - Utilize user participation frequency to enable recovering P
  - Formulate as an integer linear programming problem
    - Leverage powerful integer linear programming solvers
  - Recover columns of P **individually**
    - Parallelize across users
  - Modify the optimization formulation to account for
    - Missing / partial / inexact constraints, etc

$$\text{Find } p_k \text{ s.t.}$$

$$\min ||Nul(G^T_{aggregated})p_k||^2$$

$$p_k \in \{0,1\}^n$$

$$C_k p_k - c_k = 0$$

Results: Success Rate of Recovering P

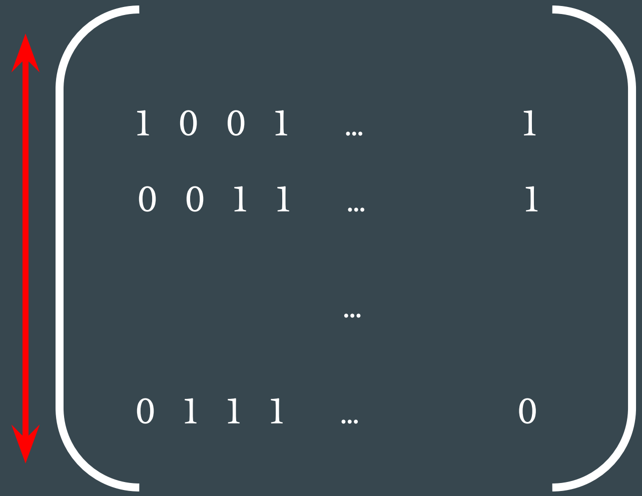
- Number of Rounds
- Number of Users
- Constraint Granularity
- Noise

$$\begin{pmatrix} 1 & 0 & 0 & 1 & \dots & 1 \\ 0 & 0 & 1 & 1 & \dots & 1 \\ & & & & \dots & \\ 0 & 1 & 1 & 1 & \dots & 0 \end{pmatrix}$$

User Participant Matrix P

Results: Success Rate of Recovering P

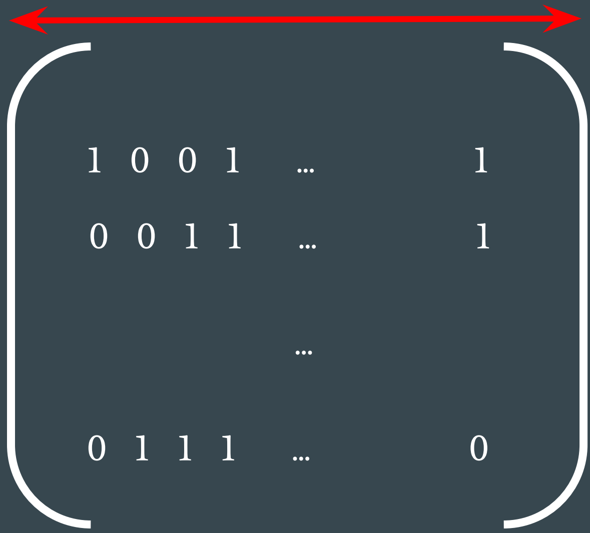
- Number of Rounds
- Number of Users
- Constraint Granularity
- Noise



User Participant Matrix P

Results: Success Rate of Recovering P

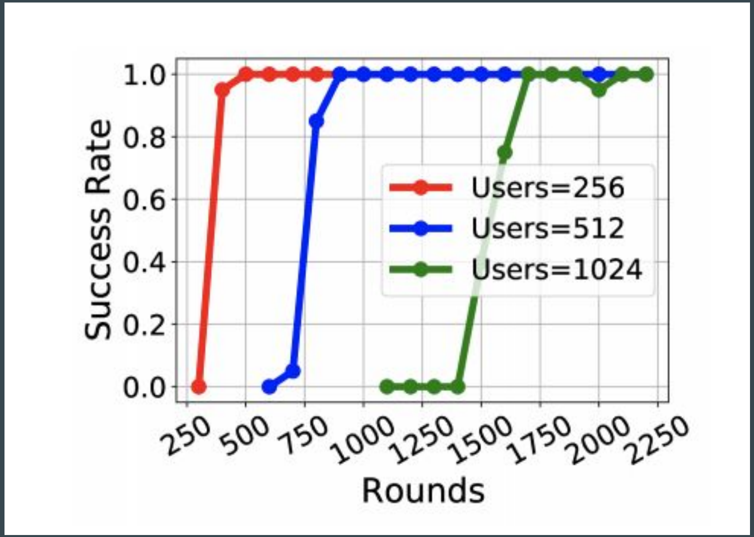
- Number of Rounds
- Number of Users
- Constraint Granularity
- Noise



User Participant Matrix P

Results: Success Rate of Recovering P

- Number of Rounds
- Number of Users



$$\begin{pmatrix} 1 & 0 & 0 & 1 & \dots & 1 \\ 0 & 0 & 1 & 1 & \dots & 1 \\ & & & & \dots & \\ 0 & 1 & 1 & 1 & \dots & 0 \end{pmatrix}$$

User Participant Matrix P

Results: Success Rate of Recovering P

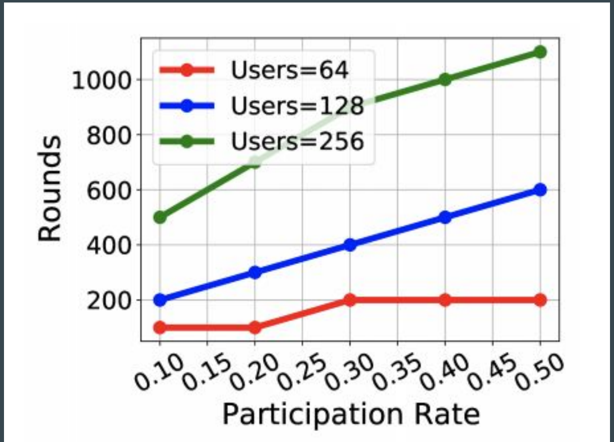
- Number of Rounds
- Number of Users
- Participation Rate
- Noise

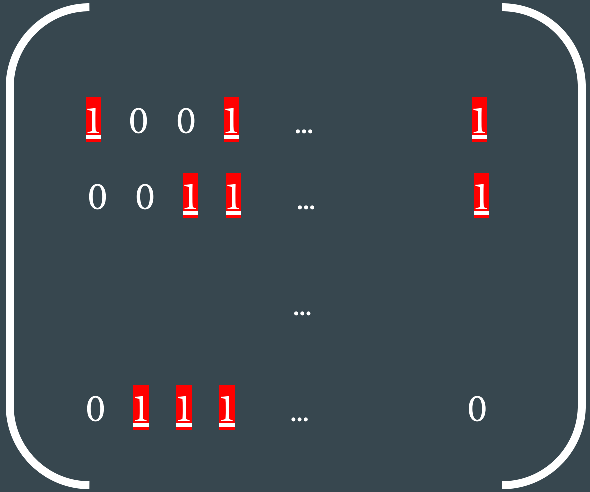
$$\begin{pmatrix} 1 & 0 & 0 & 1 & \dots & 1 \\ 0 & 0 & 1 & 1 & \dots & 1 \\ & & & \dots & & \\ 0 & 1 & 1 & 1 & \dots & 0 \end{pmatrix}$$

User Participant Matrix P

Results: Success Rate of Recovering P

- Participation Rate

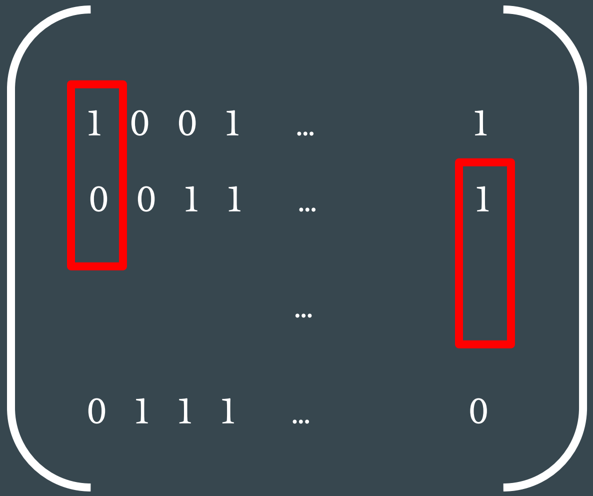




User Participant Matrix P

Results: Success Rate of Recovering P

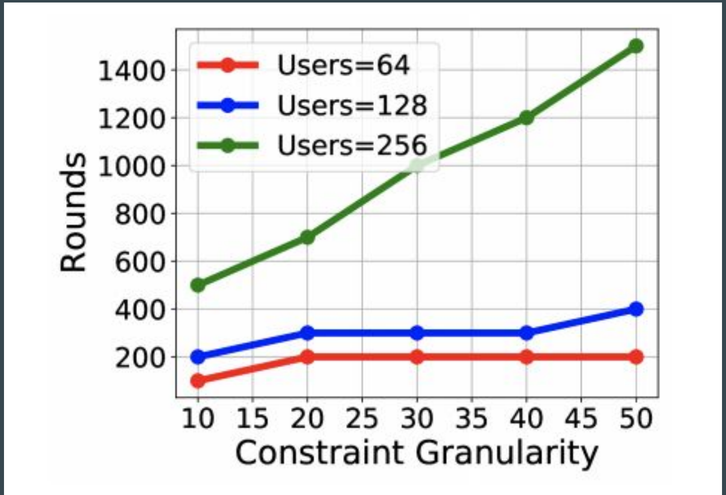
- Number of Rounds
- Number of Users
- Constraint Granularity
- Noise

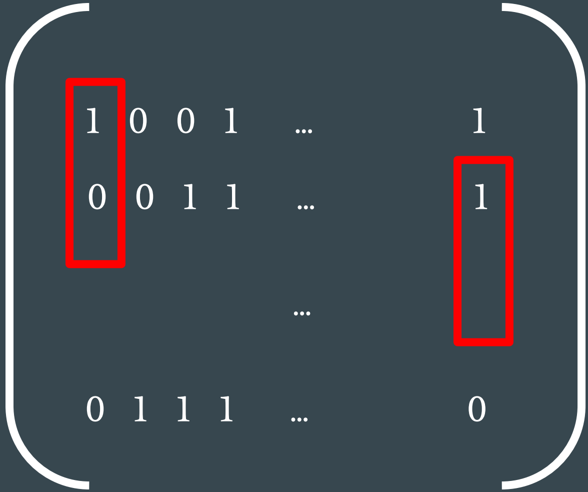


User Participant Matrix P

Results: Success Rate of Recovering P

- Constraint Granularity

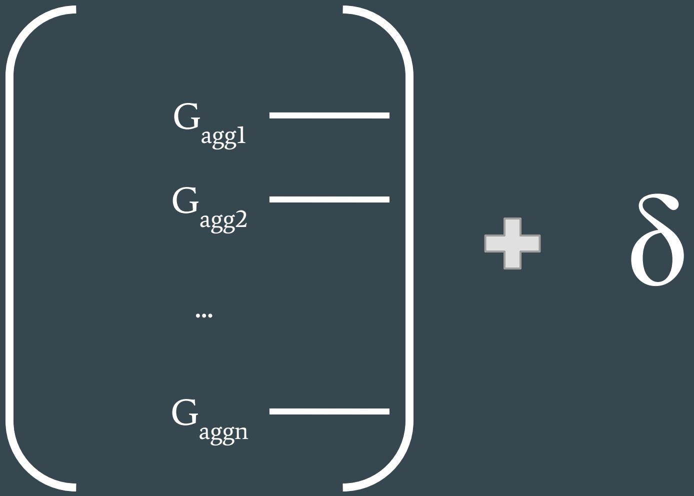




User Participant Matrix P

Results: Success Rate of Recovering P

- Number of Rounds
- Number of Users
- Constraint Granularity
- Noise
  - Federated averaging
  - Changes in training data
  - etc.



Aggregated Updates G_{agg}

Results: Success Rate of Recovering P

- Noise
  - Federated averaging
  - Changes in training data
  - etc.

Dataset Size D	Batch Size b	Local Epochs e						
		1	2	4	8	16	32	64
64	8	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	16	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0
128	8	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	16	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0
64 (momentum=.9)	8	.99	1.0	1.0	1.0	1.0	1.0	1.0
	16	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0
128 (momentum=.9)	8	1.0	1.0	1.0	1.0	1.0	1.0	.96
	16	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0
64 (fraction=.9)	8	.06	.66	.97	.99	.98	.99	.85
	16	1.0	1.0	1.0	1.0	1.0	1.0	.99
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0
128 (fraction=.9)	8	1.0	1.0	1.0	1.0	1.0	1.0	.90
	16	.59	.96	1.0	1.0	1.0	1.0	.99
	32	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Table 1. Fraction of P reconstructed with FedAvg model updates (users=100, rounds=200, Cifar10 LeNet, participant rate=.1, granularity=10, time limit per column=10 min). We exactly reconstruct P in the majority of FedAvg settings.

Results: Qualitative Results

- Gradient Inference Attack
  - Inference attack with and without disaggregation

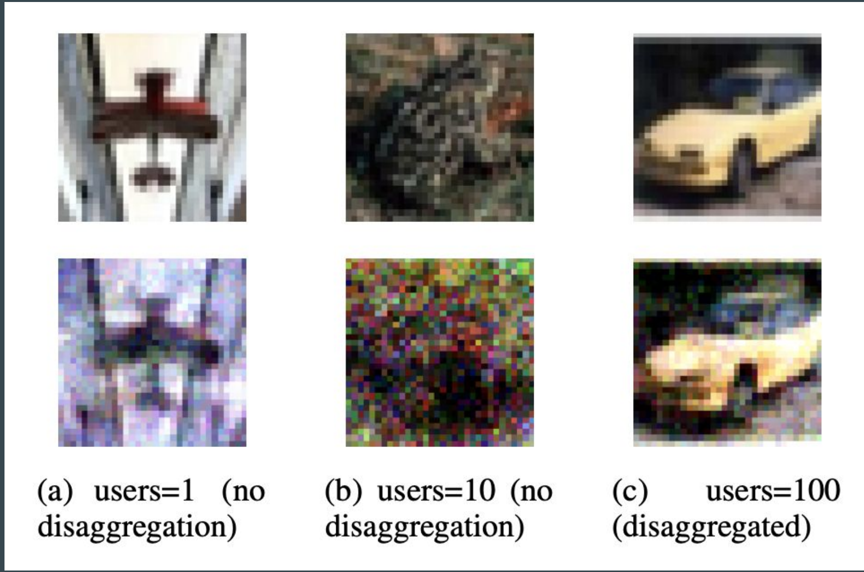
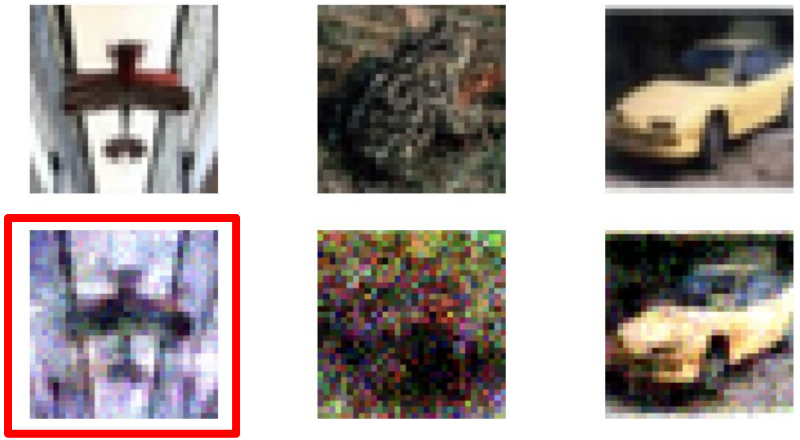


Figure 8. Recovered images from FedAvg updates across users (top image is closest ground truth). Gradient disaggregation enables high quality inversion on noisy FedAvg updates aggregated across many users; unlike disaggregation on exact gradients, disaggregation on noisy updates recovers the average update submitted across rounds, and we are able to reconstruct high quality images on noisy updates aggregated across many users. Without disaggregation, inversion on updates aggregated over multiple users (users=10) significantly degrades quality.

Results: Qualitative Results

- Gradient Inference Attack
    - Inference attack with and without disaggregation

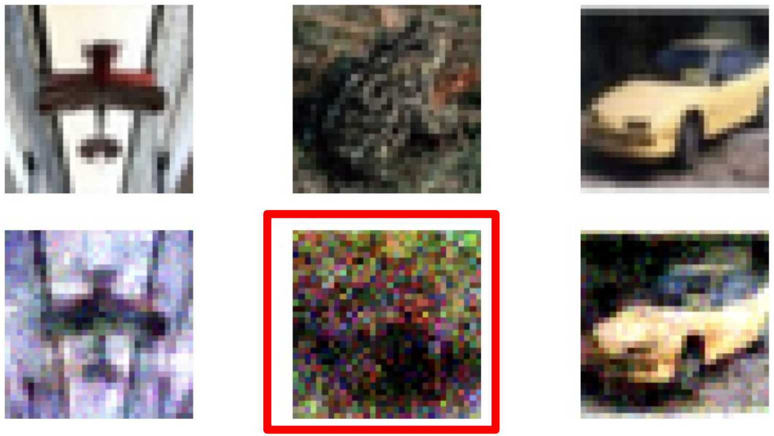


(a) users=1 (no disaggregation)   (b) users=10 (no disaggregation)   (c) users=100 (disaggregated)

Figure 8. Recovered images from FedAvg updates across users (top image is closest ground truth). Gradient disaggregation enables high quality inversion on noisy FedAvg updates aggregated across many users; unlike disaggregation on exact gradients, disaggregation on noisy updates recovers the average update submitted across rounds, and we are able to reconstruct high quality images on noisy updates aggregated across many users. Without disaggregation, inversion on updates aggregated over multiple users (users=10) significantly degrades quality.

Results: Qualitative Results

- Gradient Inference Attack
  - Inference attack with and without disaggregation



(a) users=1 (no disaggregation) (b) users=10 (no disaggregation) (c) users=100 (disaggregated)

Figure 8. Recovered images from FedAvg updates across users (top image is closest ground truth). Gradient disaggregation enables high quality inversion on noisy FedAvg updates aggregated across many users; unlike disaggregation on exact gradients, disaggregation on noisy updates recovers the average update submitted across rounds, and we are able to reconstruct high quality images on noisy updates aggregated across many users. Without disaggregation, inversion on updates aggregated over multiple users (users=10) significantly degrades quality.

Results: Qualitative Results

- Gradient Inference Attack
    - Inference attack with and without disaggregation

(a) users=1 (no disaggregation) (b) users=10 (no disaggregation) (c) users=100 (disaggregated)

Figure 8. Recovered images from FedAvg updates across users (top image is closest ground truth). Gradient disaggregation enables high quality inversion on noisy FedAvg updates aggregated across many users; unlike disaggregation on exact gradients, disaggregation on noisy updates recovers the average update submitted across rounds, and we are able to reconstruct high quality images on noisy updates aggregated across many users. Without disaggregation, inversion on updates aggregated over multiple users (users=10) significantly degrades quality.

Conclusion

- Gradient disaggregation attack on Federated Learning
  - Framed as matrix factorization & integer linear programming
  - Leverage user participation frequency to make intractable problem tractable
- Secure aggregation is not enough
  - Leveraging side channel information, a central server may uncover individual user gradients
  - Across multiple observations, sum reveal significant information about its terms
- Dangers of side channel information in Federated Learning systems
  - Seemingly benign metrics used to monitor device performance can be dangerous
- Possible defenses
  - Add even more noise through differential privacy
  - Eliminate side channel information (but must balance this w/ cost to utility!)
  - Homomorphic encryption