ISCAS 中国科学院软件研究所
Institute of Software Chinese Academy of Sciences

TablePuppet: Towards a Generic Framework for Learning over Relational Tables

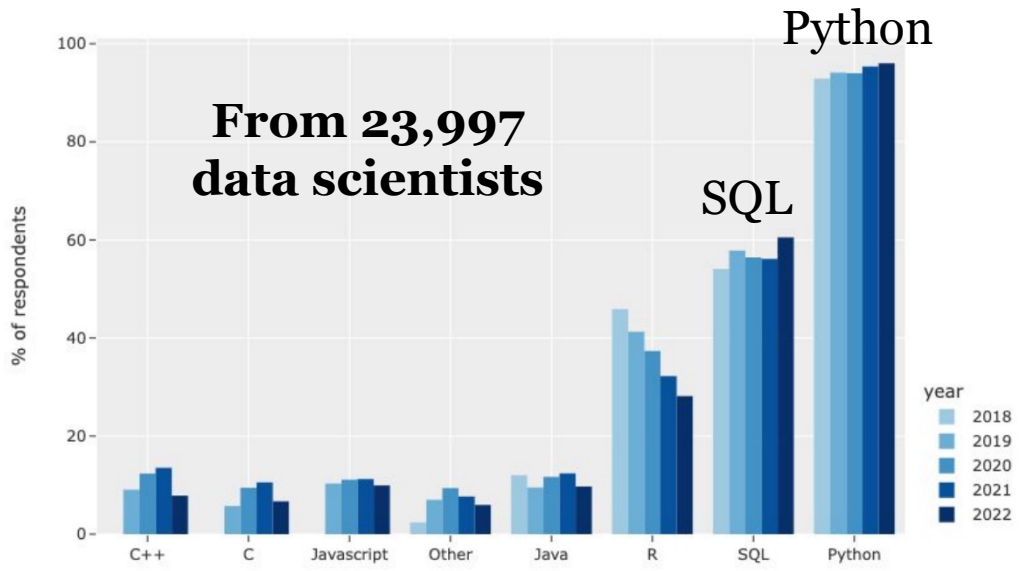
Lijie Xu[†], Chulin Xie, Yiran Guo, Haiyang Lu, Gustavo Alonso, Bo Li, Guoliang Li, Wei Wang, Wentao Wu, Ce Zhang

[†]Institute of Software, Chinese Academy of Sciences (ISCAS)
with others (UIUC, ETH, Tsinghua, Microsoft, UChicago)
xulijie@iscas.ac.cn

Learning over relational data

Kaggle DS & ML Survey 2022

Python and SQL remain the two most common programming skills for data scientists



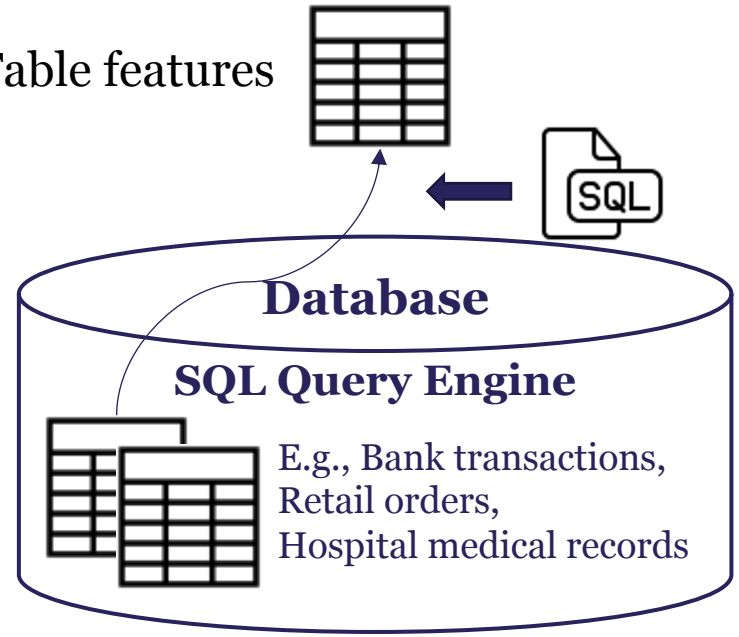
Source: https://www.kaggle.com/kaggle-survey-2022

Regression, classifiction, clustering, etc.

Train ML models on table features

scikit learn Spark MLlib PyTorch TensorFlow

Table features



Learning over relational data

In-Database Machine Learning

Apache MADlib

GaussML

Oracle Machine Learning

...

...

In-Database ML using SQL+ML query

```
CREATE MODEL m using SVM
FEATURES feature_cols
TARGET label_col
FROM table_name
WITH params = args;
```

Database

SQL + ML Query Engine

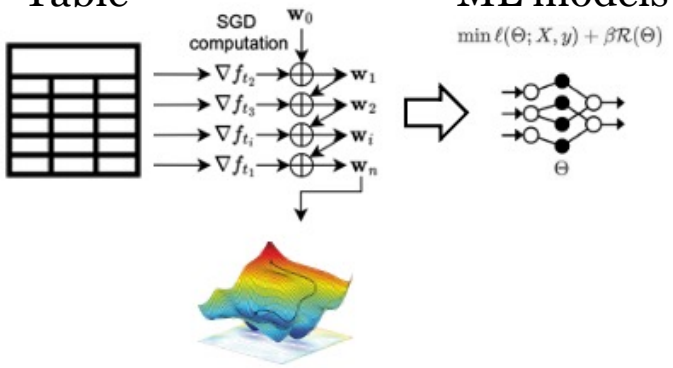
E.g., Bank transactions,
Retail orders,
Hospital medical records

Learning over relational data

How to implement ML training over relational data?

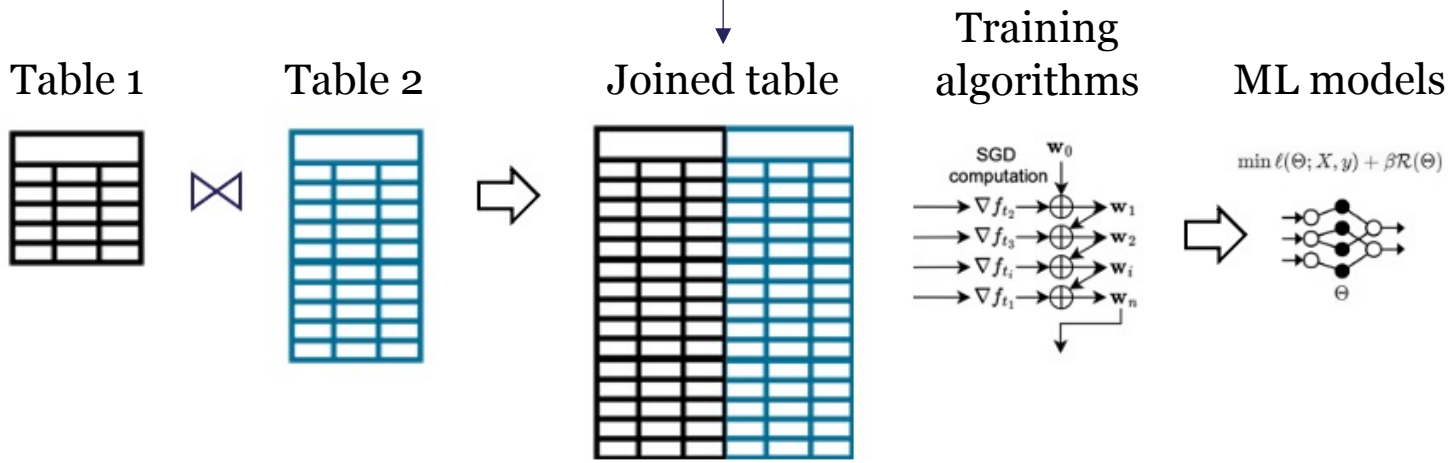
Learning over a single table

ML training algorithms
(e.g., Stochastic Gradient Descent)



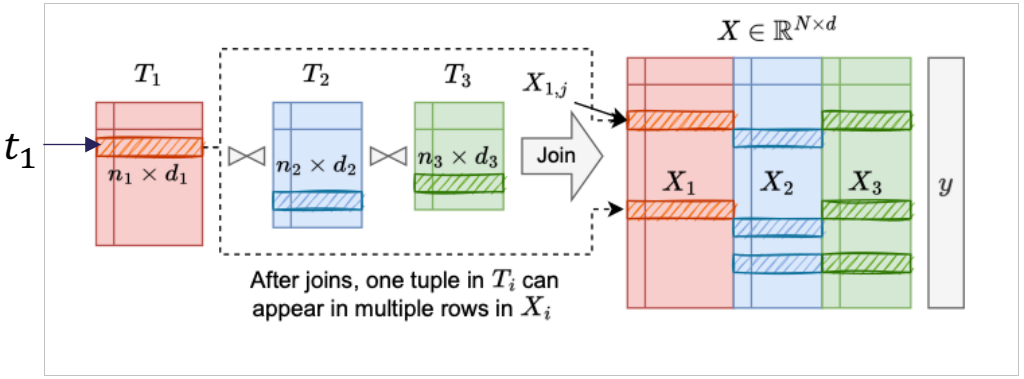
Learning over joins

Problem: The joined table could be much larger than individual tables → Storage and training overhead



Learning over joins - Example

- Example: Learning over joins on six tables



Favorita dataset

Transactions

date	store_nbr	transactions
2013-01-01	25	770
...	...	...
2017-08-15	54	802

Holidays

date	type	locale	locale_name	description	transferred
2013-01-01	Holiday	Local	Manta	Fundacion de Manta	False
...	...	...	...	...	...
2012-04-01	Holiday	Regional	Cotopaxi	Provincializaction de Cotopaxi	False

Sales

id	date	store_nbr	item_nbr	unit_sales	onpromotion
0	2013-01-01	25	103665	7.0	NaN
125497036	2017-08-15	54	2106464	1.0	True
...	...	...	...	...	...
125497032	2017-08-15	54	2087978	8.0	False

~4 GB

Items

item_nbr	family	class	perishable
96995	GROCERY	1093	0
...	...	...	...
103665	BREAD/BAKERY	2712	1

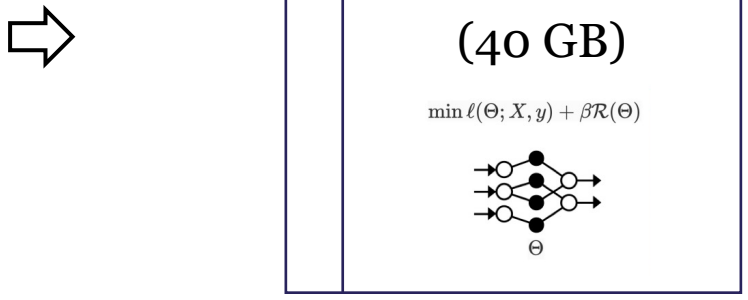
Oil

date	dcoilwtico
2013-01-02	93.14
...	...
2017-08-31	47.26

Stores

store_nbr	city	state	type	cluster
1	Quito	Pichincha	D	13
...	...	...	...	...
51	Guayaquil	Guayas	A	17

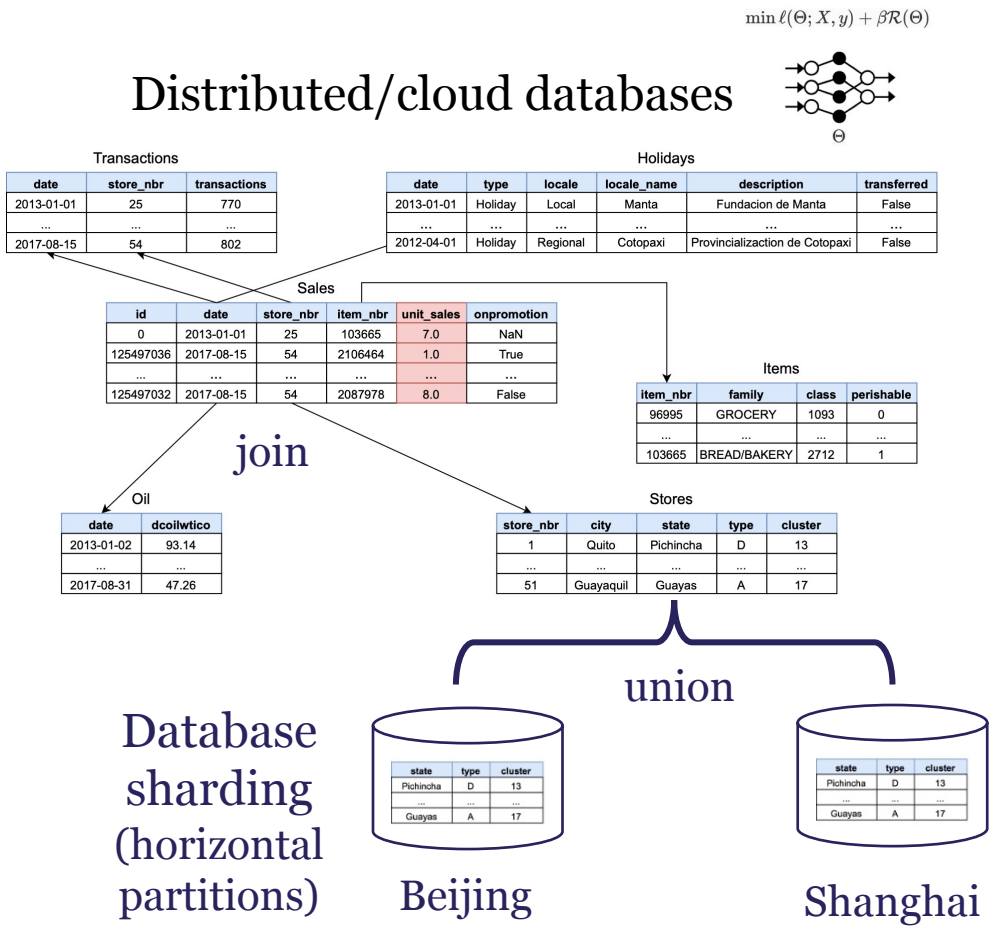
There are duplicate tuples after joins



It is impractical to store the joined table

More general case: Learning over joins and unions

- Learning over joins and unions in a distributed environment



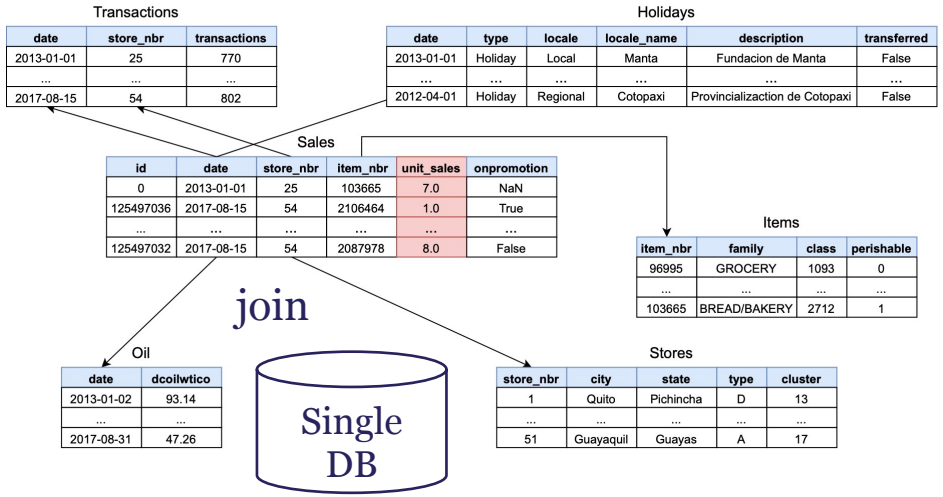
Challenges

1. High I/O communications to perform joins and unions
2. High storage and computation overhead for learning over the large joined table
3. Privacy problem due to distributed communications

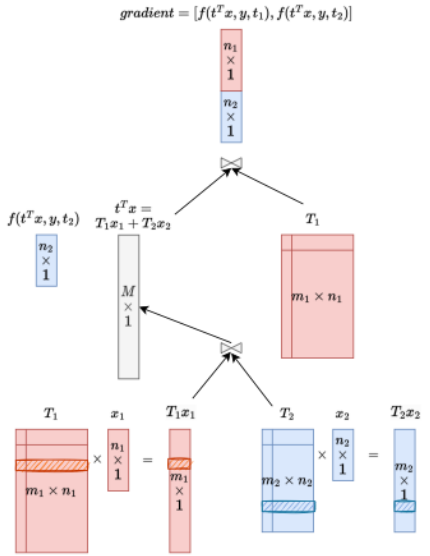
Research problem: how to train ML models on distributed tables requiring joins and unions without table sharing?

Related work - Factorized learning

- Learning over joins *in a single database* without materializing the joined table

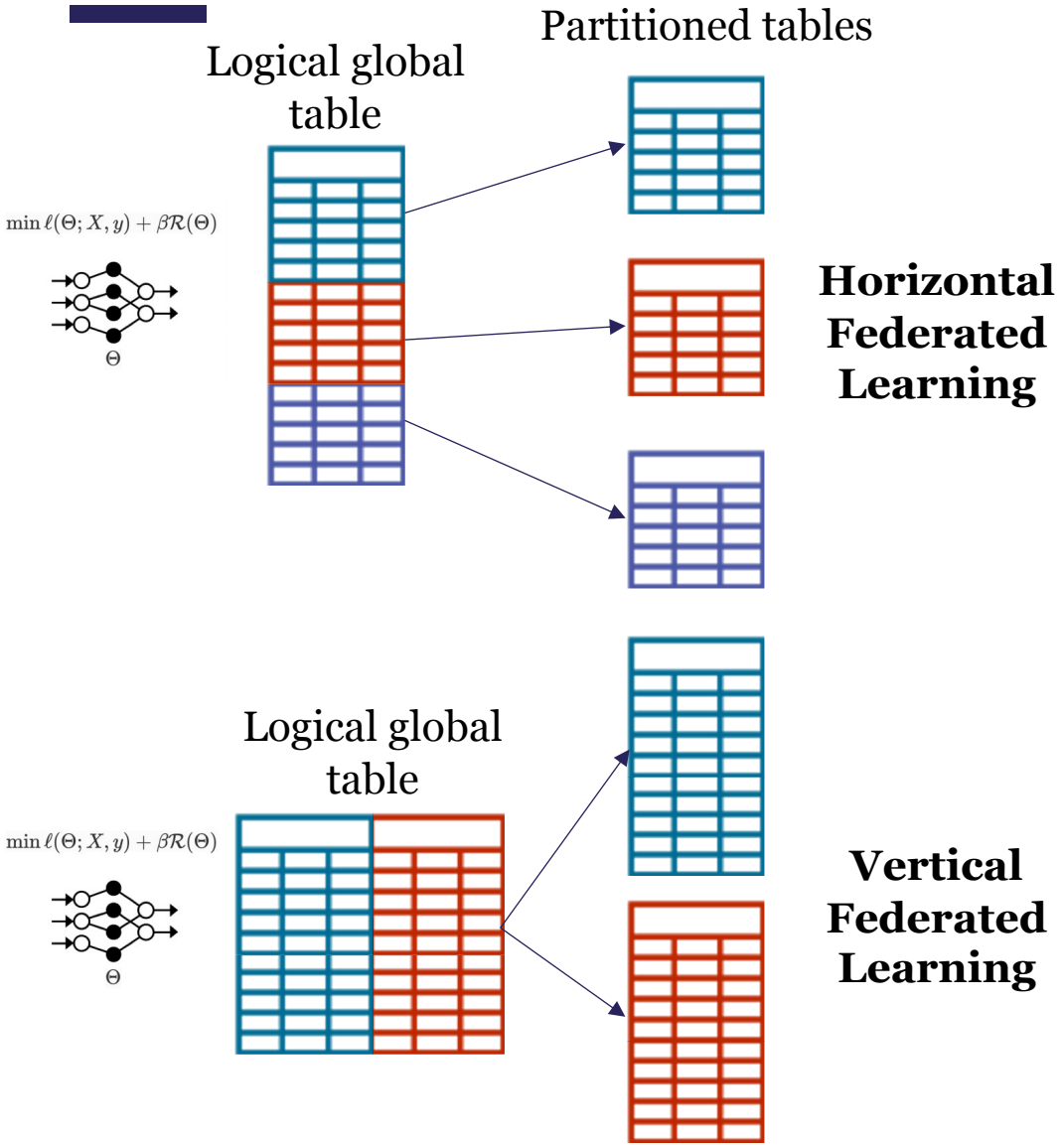


- Factorized learning approaches [1, 2, 3, 4]
  - Decouple the ML computation, distribute partial computation to each table, and aggregate/join the computation results.
  - Only support specific ML training algorithm (Gradient Descent), but do not support more efficient (mini-batch) SGD.
  - Linear models ☺, Neural network models ☹.



1. Arun Kumar et al. **Learning Generalized Linear Models Over Normalized Data.** SIGMOD 2015
2. Maximilian Schleich et al. **Learning Linear Regression Models over Factorized Joins**. SIGMOD 2016
3. Lingjiao Chen et al. **Towards Linear Algebra over Normalized Data**. VLDB 2017
4. Maximilian Schleich et al. **A Layered Aggregate Engine for Analytics Workloads**. SIGMOD 2019

Related work - Federated learning



- Federated learning approaches [1, 2, 3, 4]
  - Push ML computation (decouple gradient computation) to each table and aggregates the computation results.
  - Suppose a table is either horizontally or vertically partitioned.
  - SQL joins and hybrid scenarios (joins and unions) ☹

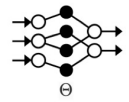
1. Yang Liu et al. **Vertical Federated Learning: Concepts, Advances, and Challenges**. IEEE TKDE 2024
2. Qiang Yang et al. **Federated Machine Learning: Concept and Applications**. ACM TIST 2019
3. Qinbin Li et al. **Federated Learning on Non-IID Data Silos: An Experimental Study**. ICDE 2022
4. Afsana Khan et al. **Vertical federated learning: a structured literature review**. Knowl. Inf. Syst. 2025

Our approach: Two-level decomposition (TablePuppet)

Problem:

Train ML models on the global joined table without table sharing

$$\min \ell(\Theta; X, y) + \beta\mathcal{R}(\Theta)$$



Transactions

date	store_nbr	transactions
2013-01-01	25	770
...	...	...
2017-08-15	54	802

Holidays

date	type	locale	locale_name	description	transferred
2013-01-01	Holiday	Local	Manta	Fundacion de Manta	False
...	...	...	...	...	...
2012-04-01	Holiday	Regional	Cotopaxi	Provincializaction de Cotopaxi	False

Sales

id	date	store_nbr	item_nbr	unit_sales	onpromotion
0	2013-01-01	25	103665	7.0	NaN
125497036	2017-08-15	54	2106464	1.0	True
...	...	...	...	...	...
125497032	2017-08-15	54	2087978	8.0	False

Items

item_nbr	family	class	perishable
96995	GROCERY	1093	0
...	...	...	...
103665	BREAD/BAKERY	2712	1

join

Oil

date	dcoilwtico
2013-01-02	93.14
...	...
2017-08-31	47.26

Stores

store_nbr	city	state	type	cluster
1	Quito	Pichincha	D	13
...	...	...	...	...
51	Guayaquil	Guayas	A	17

union

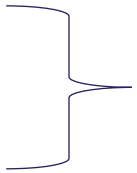
Database sharding (horizontal partitions)

Beijing

Shanghai

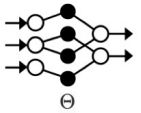
Two-level decomposition:

Decompose the learning process into two steps:
(1) learning over joins, followed by (2) learning over unions



Logical joined table

$$\min \ell(\Theta; X, y) + \beta\mathcal{R}(\Theta)$$



Learning over Joins

Pushes ML computation down to the individual tables being joined

Learning over Unions

Further pushes learning down to the horizontal partitions of each table

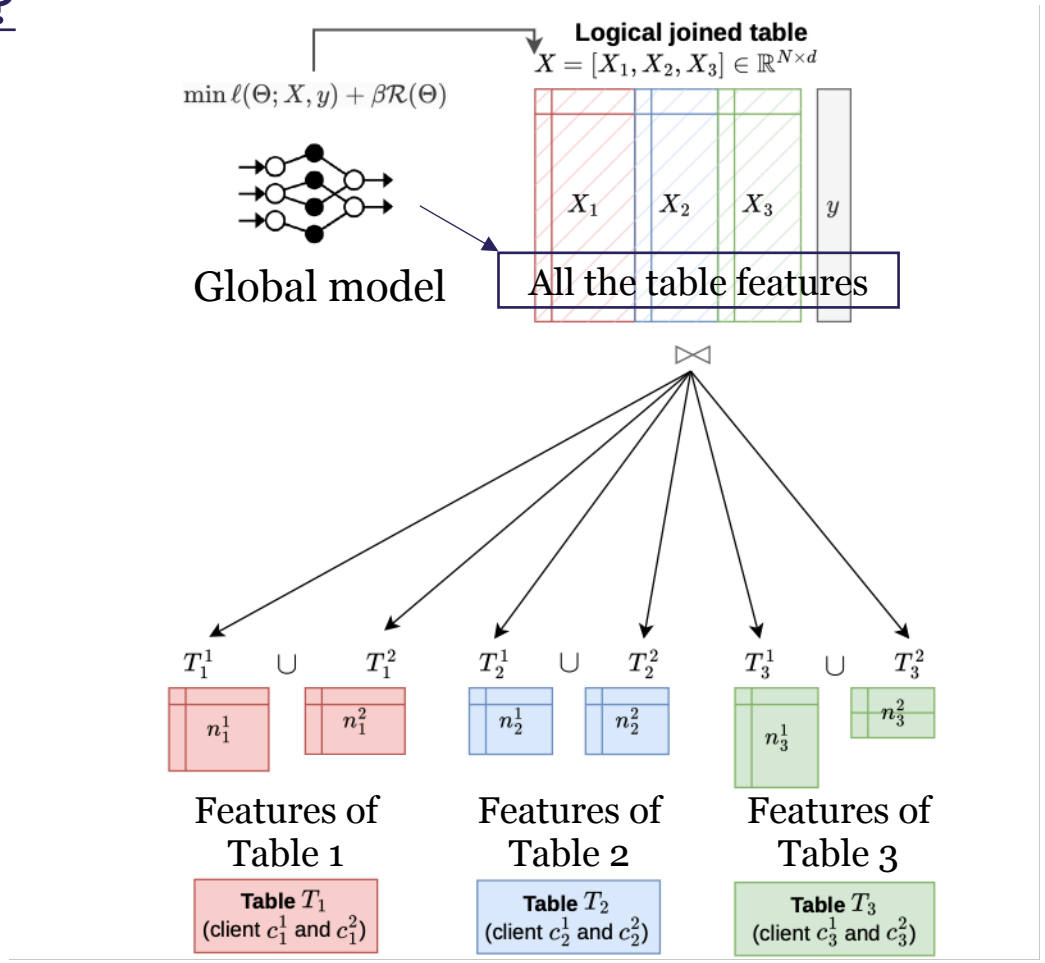
Our approach: Two-level decomposition

Problem 1: How to store/organize the ML models?

Global model needs to cover all the table features, but these features are distributed in different machines.

- **Model decomposition and distribution**
  - Do not store the global model

TablePuppet framework



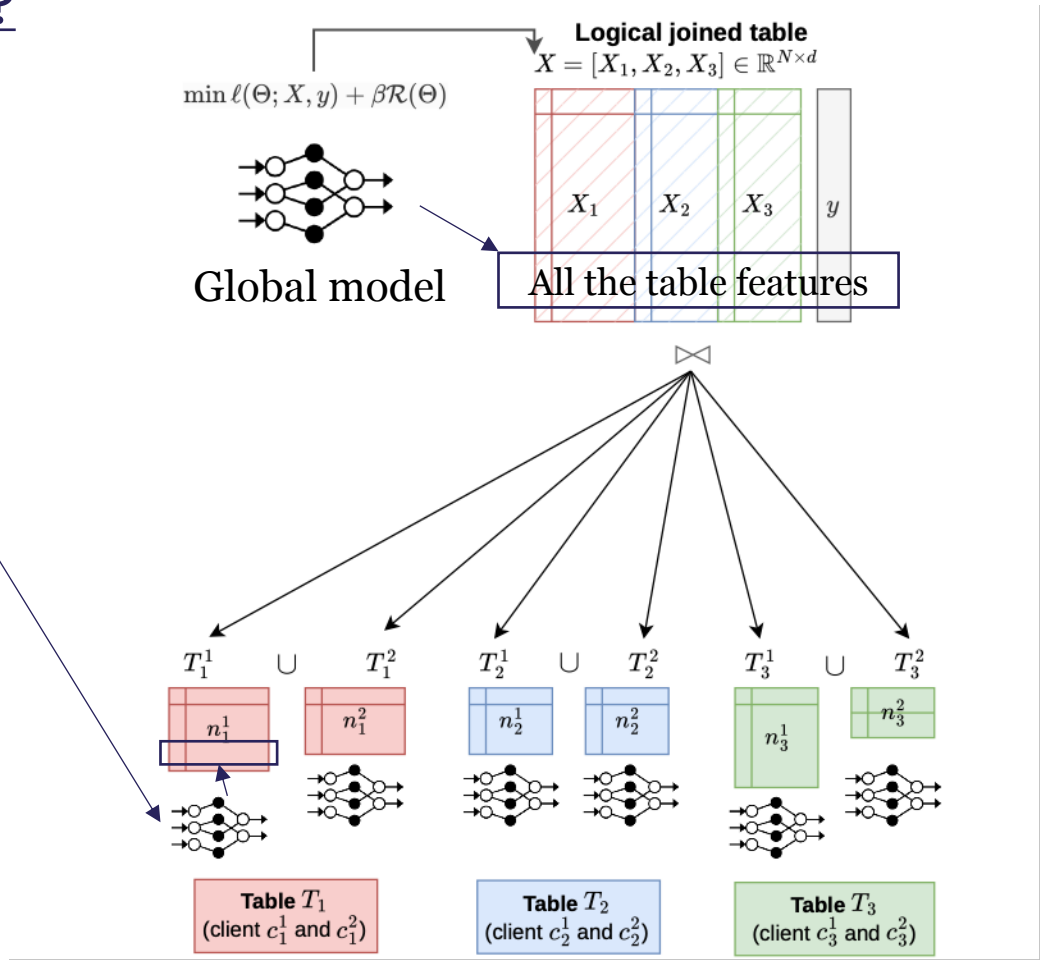
Our approach: Two-level decomposition

Problem 1: How to store/organize the ML models?

Global model needs to cover all the table features, but these features are distributed in different machines.

- **Model decomposition and distribution**
  - Do not store the global model
  - Split global model to small local models for each table
  - Each model covers the parameters associated with the local table features.
  - Push ML over the global model (global joined table) down to each local model (horizontal table)

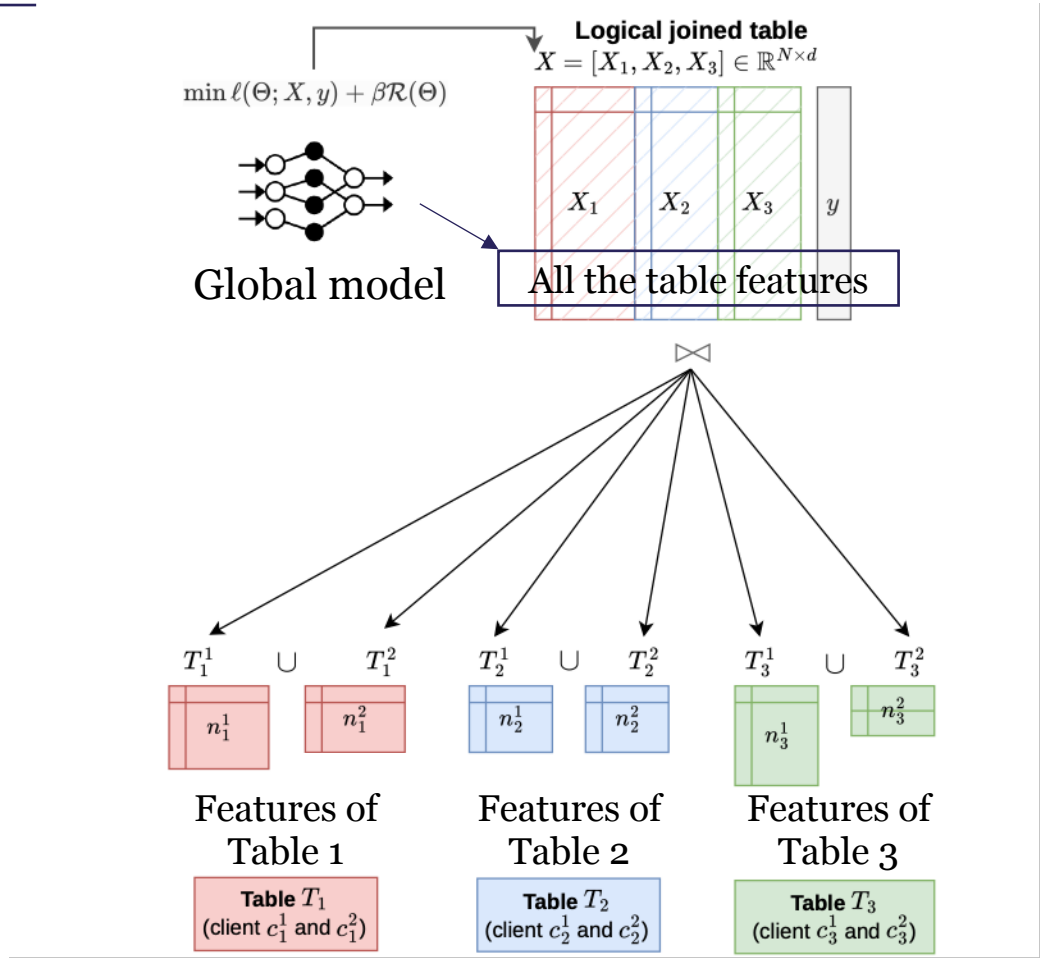
TablePuppet framework



Our approach: Two-level decomposition

Problem 2: How to generate the logical joined table?

TablePuppet framework

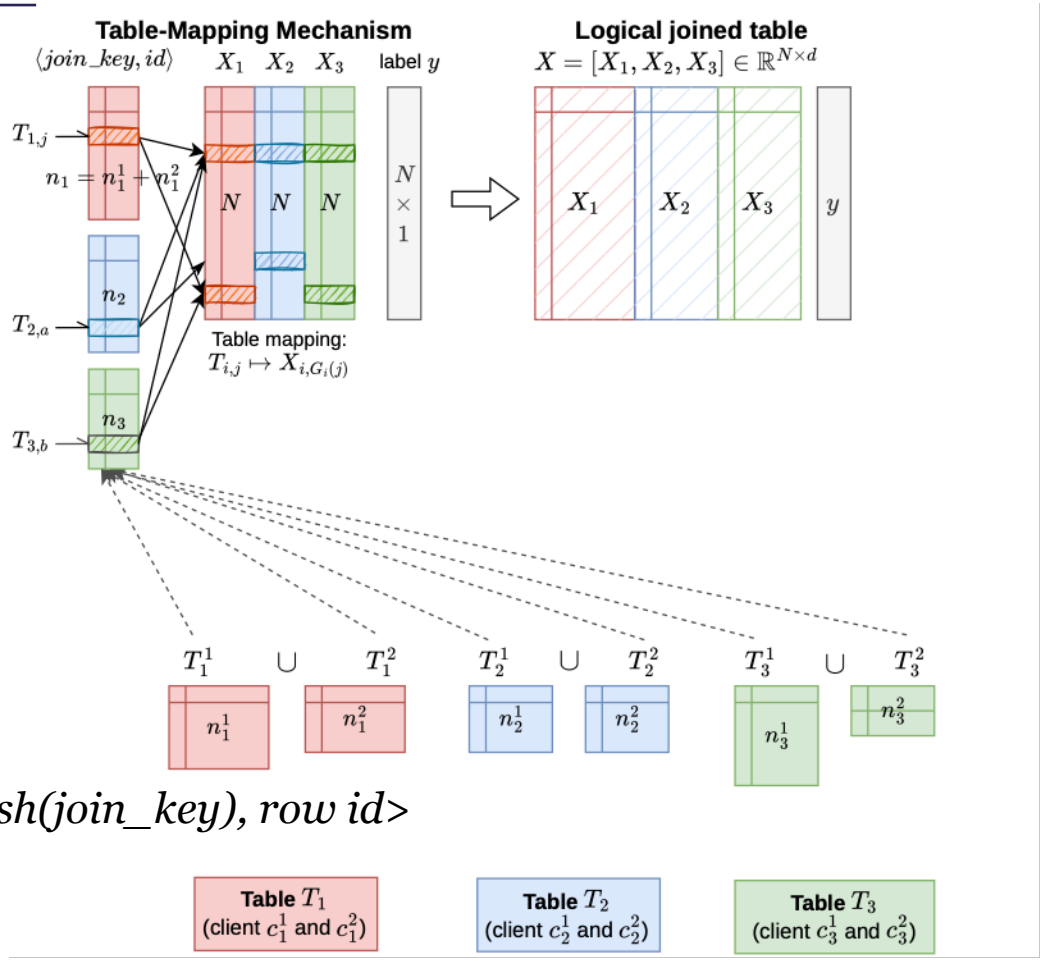


Our approach: Two-level decomposition

Problem 2: How to generate the logical joined table?

- **Table-mapping mechanism**
  - Aggregate the <*hash(join_key), row id*> of each horizontal table to form a logical joined table

TablePuppet framework

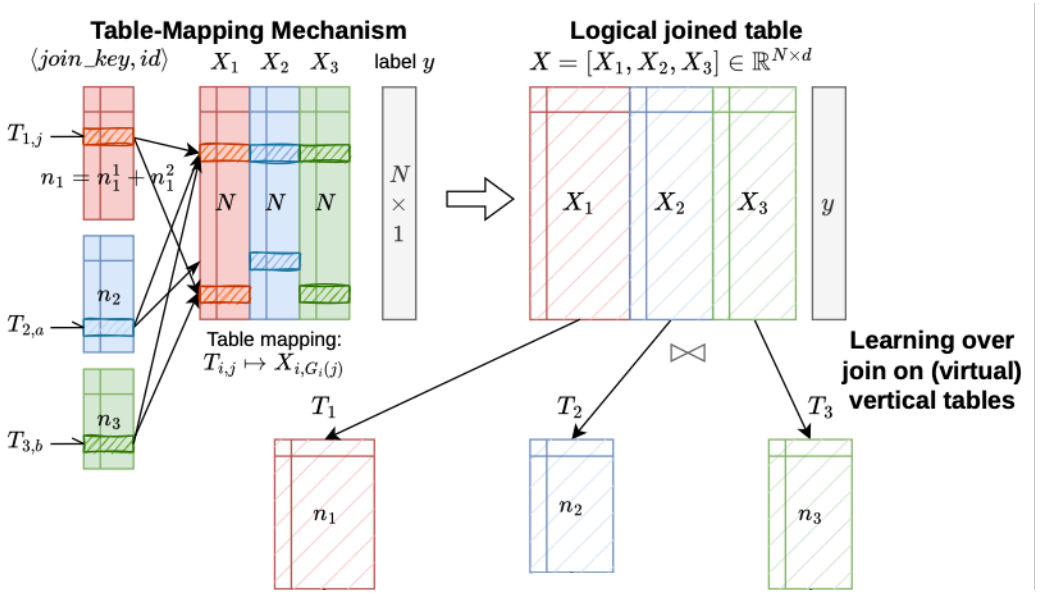


Our approach: Two-level decomposition

Problem 3: How to perform learning over unions?

- **Learning over Joins (LoJ)**
  - Push ML on joined table to (virtual) vertical tables

TablePuppet framework



Our approach: Two-level decomposition

Problem 3: How to perform learning over unions?

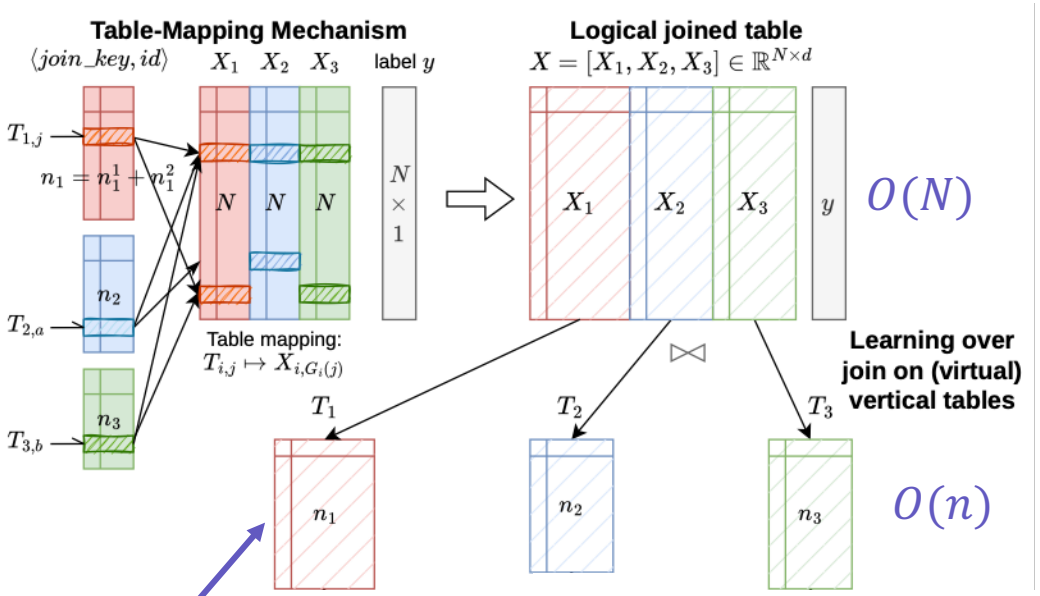
- **Learning over Joins (LoJ)**
  - Push ML on joined table to (virtual) vertical tables
  - Support both SGD and ADMM training algorithms
  - Optimize the computation on duplicate tuples

Table 2: The problem decomposition for LoJ, using SGD and ADMM with computation/communication reduction.

Algorithm	Server	Client with T_i
SGD	$\frac{\partial \ell_j}{\partial h_{i,j}}, \quad \forall j \in [N]. (4)$	$\nabla_{\theta_i} F(T_i) = \frac{1}{N}\sum_{j=1}^{N}\left(\frac{\partial \ell_j}{\partial \theta_i} + \beta \frac{\partial \mathcal{R}_i}{\partial \theta_i}\right), (5)$ $\theta_i := \theta_i - \eta \nabla_{\theta_i} F(T_i). (6)$
ADMM	$z_j^t = \underset{z_j}{\text{argmin}}\left(\ell(z_j; y_j) - \left(\lambda_j^{t-1}\right)^\top z_j + \frac{\rho}{2}\left\|\sum_{i=1}^{M} f_i(\theta_i^t; T_{i,p_i(j)}) - z_j\right\|^2\right), (7)$ $\lambda_j^t = \lambda_j^{t-1} + \rho\left(\sum_{i=1}^{M} f_i(\theta_i^t; T_{i,p_i(j)}) - z_j^t\right). (8)$	$\theta_i^{t+1} = \underset{\theta_i}{\text{argmin}}\left(\beta \mathcal{R}_i(\theta_i) + \frac{1}{N}\sum_{j=1}^{N} \lambda_j^{t\top} f_i(\theta_i; T_{i,p_i(j)}) + \frac{1}{N}\sum_{j=1}^{N}\frac{\rho}{2}\left\|s_{i,j}^t + f_i(\theta_i; T_{i,p_i(j)})\right\|^2\right). (9)$
SGD-Opt	$Y_{i,j} = \sum_{g \in G_i(j)} \frac{\partial \ell_g}{\partial h_{i,g}}, \quad \forall j \in [n_i], (10)$ $G_{i,j} = \lvert G_i(j)\rvert, \quad \forall j \in [n_i]. (11)$	$\nabla_{\theta_i} F(T_i) = \frac{1}{N}\sum_{j=1}^{n_i}\left(Y_{i,j}\frac{\partial h_{i,j}}{\partial \theta_i} + \beta G_{i,j}\frac{\partial \mathcal{R}_i}{\partial \theta_i}\right), (12)$ $\theta_i := \theta_i - \eta \nabla_{\theta_i} F(T_i). (13)$
ADMM-Opt	Apart from Eq. 7 and 8, the server also performs: $Y_{i,j}^t = \sum_{g \in G_i(j)} (\lambda_g^t + \rho s_{i,g}^t), \quad \forall j \in [n_i], (14)$ $G_{i,j} = \lvert G_i(j)\rvert, \quad \forall j \in [n_i]. (15)$	$\theta_i^{t+1} = \underset{\theta_i}{\text{argmin}}\left(\beta \mathcal{R}_i(\theta_i) + \frac{1}{N}\sum_{j=1}^{n_i}(Y_{i,j}^t)^\top f_i(\theta_i; T_{i,j}) + \frac{1}{N}\sum_{j=1}^{n_i}\frac{\rho G_{i,j}}{2}\left\|f_i(\theta_i; T_{i,j})\right\|^2\right). (16)$

ADMM: Alternating Direction Method of Multipliers

TablePuppet framework



Computation complexity from $O(N)$ to $O(n)$

Our approach: Two-level decomposition

Problem 4: How to perform learning over unions?

- **Learning over Unions (LoU)**
  - Push ML computation on the vertical table to local horizontal tables
  - Perform local model updates

TablePuppet framework

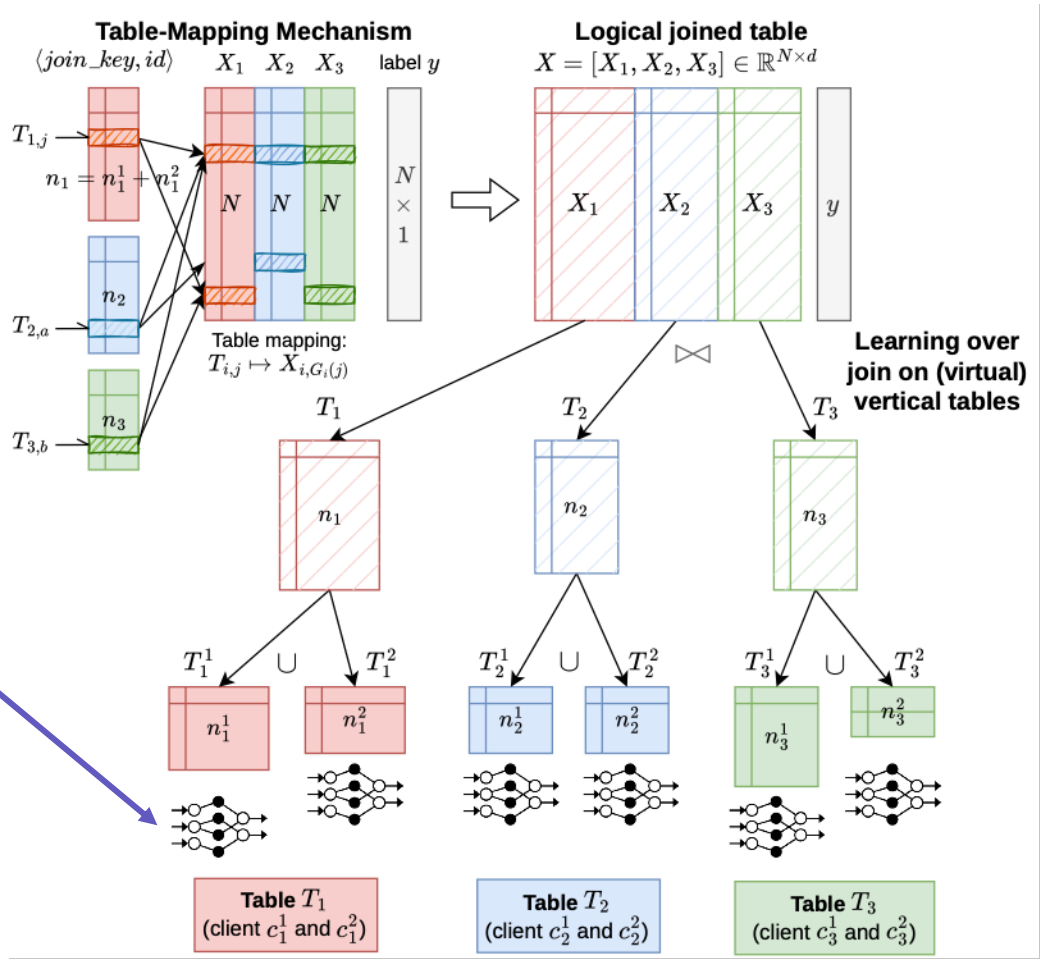


Table 3: The problem decomposition for LoU, using optimized SGD and ADMM.

Algorithm	Coordinator (Server)	Client with T_i^q
SGD-Opt	$\nabla_{\theta_i} F(T_i) = \frac{1}{N} \sum_{q=1}^{Q_i} \nabla_{\theta_i} F(T_i^q)$. (17)	$\nabla_{\theta_i} F(T_i^q) = \sum_{j=1}^{n_i^q} \left(Y_{i,j} \frac{\partial h_{i,j}}{\partial \theta_i} + \beta G_{i,j} \frac{\partial \mathcal{R}_i}{\partial \theta_i} \right)$, (18) $\theta_i := \theta_i - \eta \nabla_{\theta_i} F(T_i)$. (19)
ADMM-Opt	$w_i^\tau = \underset{w_i}{\text{argmin}} \left(\beta \mathcal{R}_i(w_i) + \frac{M\rho}{2} \left\| w_i - \overline{\theta_i^\tau} - \overline{u_i^{\tau-1}} \right\|^2 \right)$, (20) $u_i^{q,\tau} = u_i^{q,\tau-1} + \theta_i^{q,\tau} - w_i^\tau$. (21)	$\theta_i^{q,\tau+1} = \underset{\theta_i^q}{\text{argmin}} \left(\frac{1}{N} l\left(\theta_i^q; T_i^q\right) + \frac{\rho}{2} \left\| \theta_i^q - w_i^\tau + u_i^{q,\tau} \right\|^2 \right)$. (22)

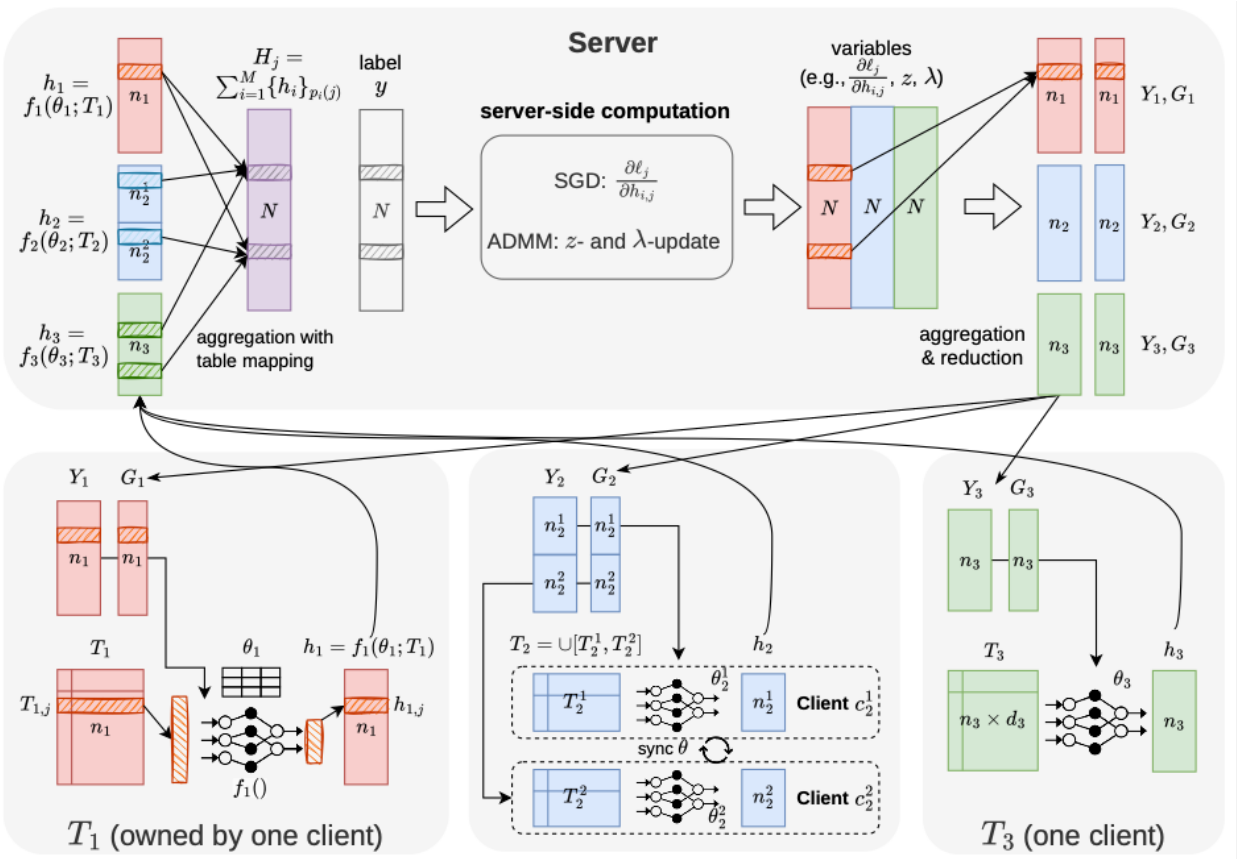
TablePuppet framework implementation

Server-Client architecture based on RAY

- **Server-side**
  - Model prediction aggregation
  - Server-side computation
  - Optimization from $O(N)$ to $O(n)$

- **Client-side**
  - local model update
  - Send model predictions to the server

TablePuppet training process



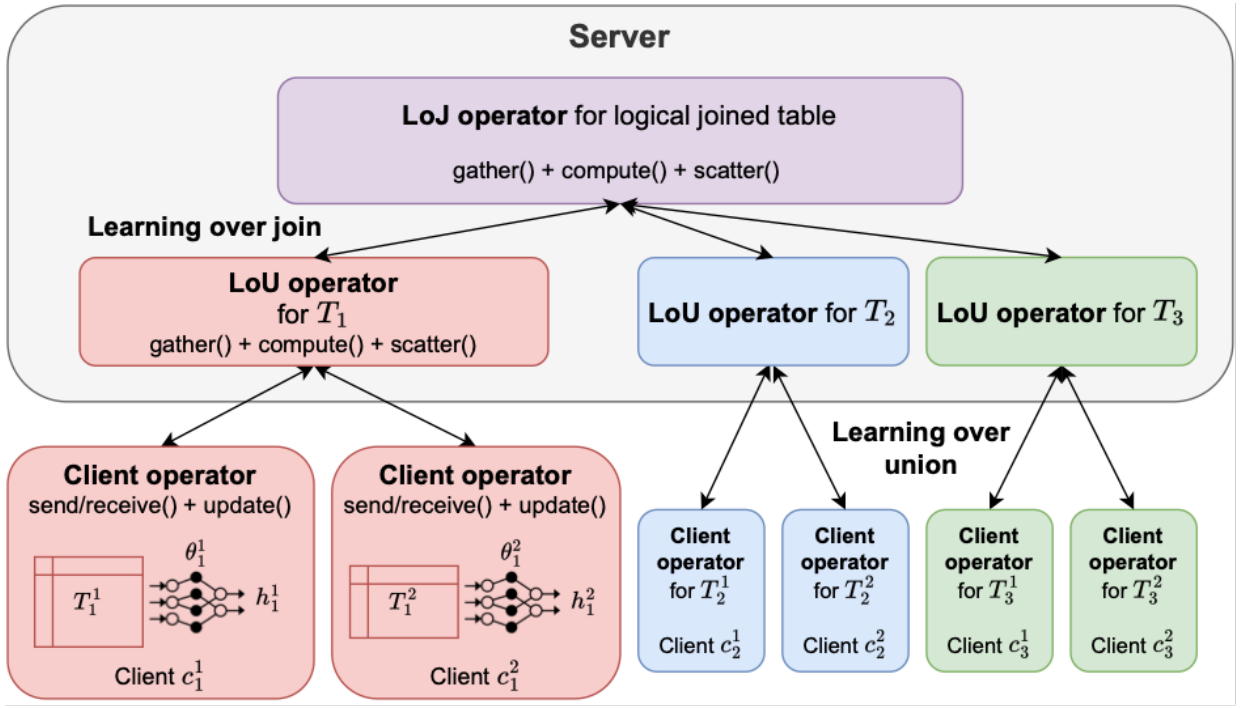
TablePuppet framework implementation

General operators to cover two ML training algorithms

Table 4: The physical operators with computation and communication functions, which cover both SGD and ADMM.

Operator	Function	SGD	ADMM
LoJ operator	*gather*()	model predictions	model predictions
	compute()	Eq. 10, 11	Eq. 7, 8, 36, 37
	scatter()	partial derivatives	auxiliary variables
LoU operator	*gather*()	model predictions, partial gradients	model predictions, model parameters
	compute()	Eq. 17	Eq. 20 and Eq. 21
	scatter()	aggregated gradients	auxiliary variables
Client operator	*send*()	model predictions, partial gradients	model predictions, model parameters
	receive()	aggregated gradients	auxiliary variables
	compute()	Eq. 18 and 19	Eq. 22

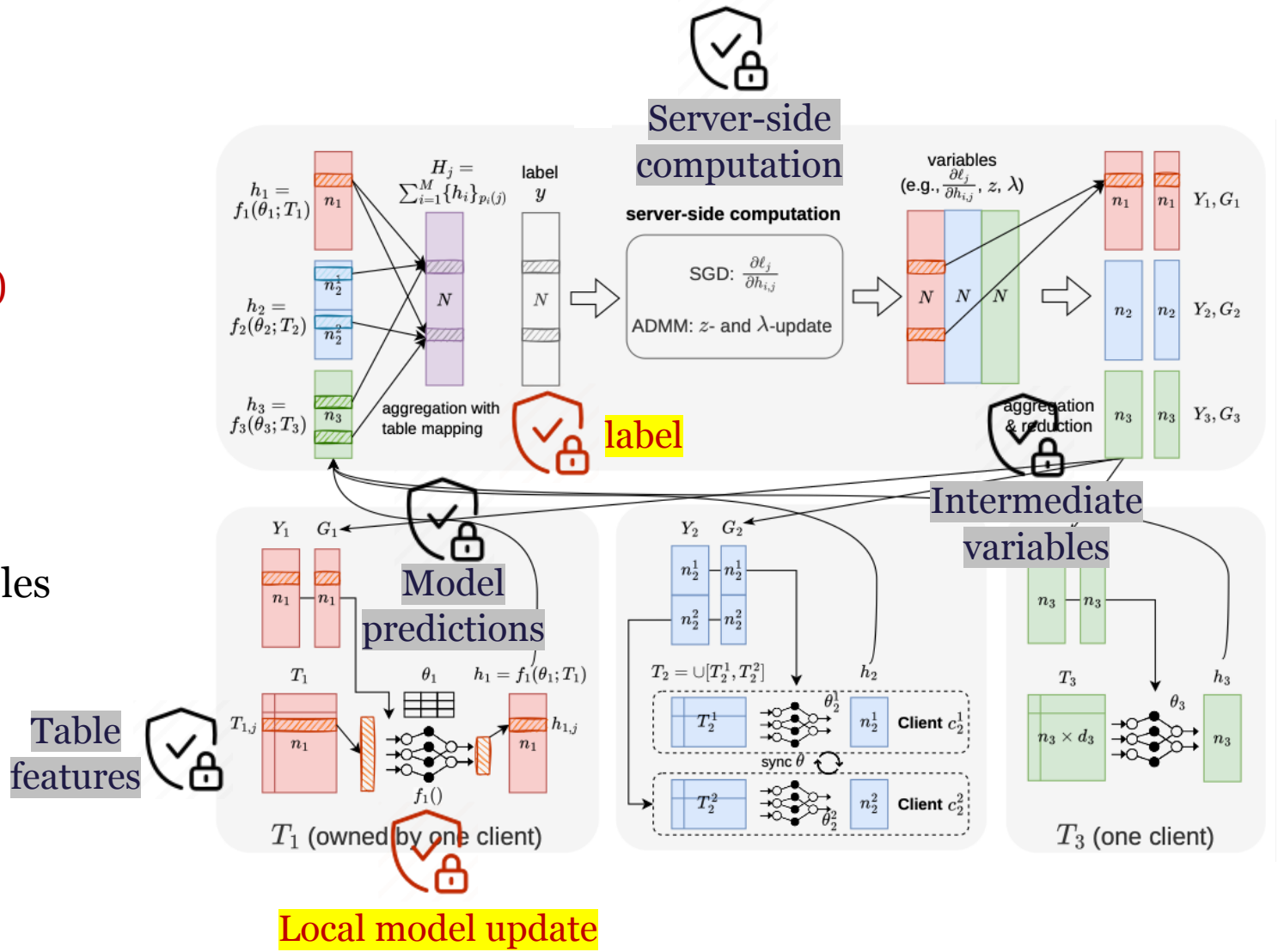
TablePuppet architecture (Three physical operators)



TablePuppet framework - Privacy guarantee

Differential Privacy (DP)

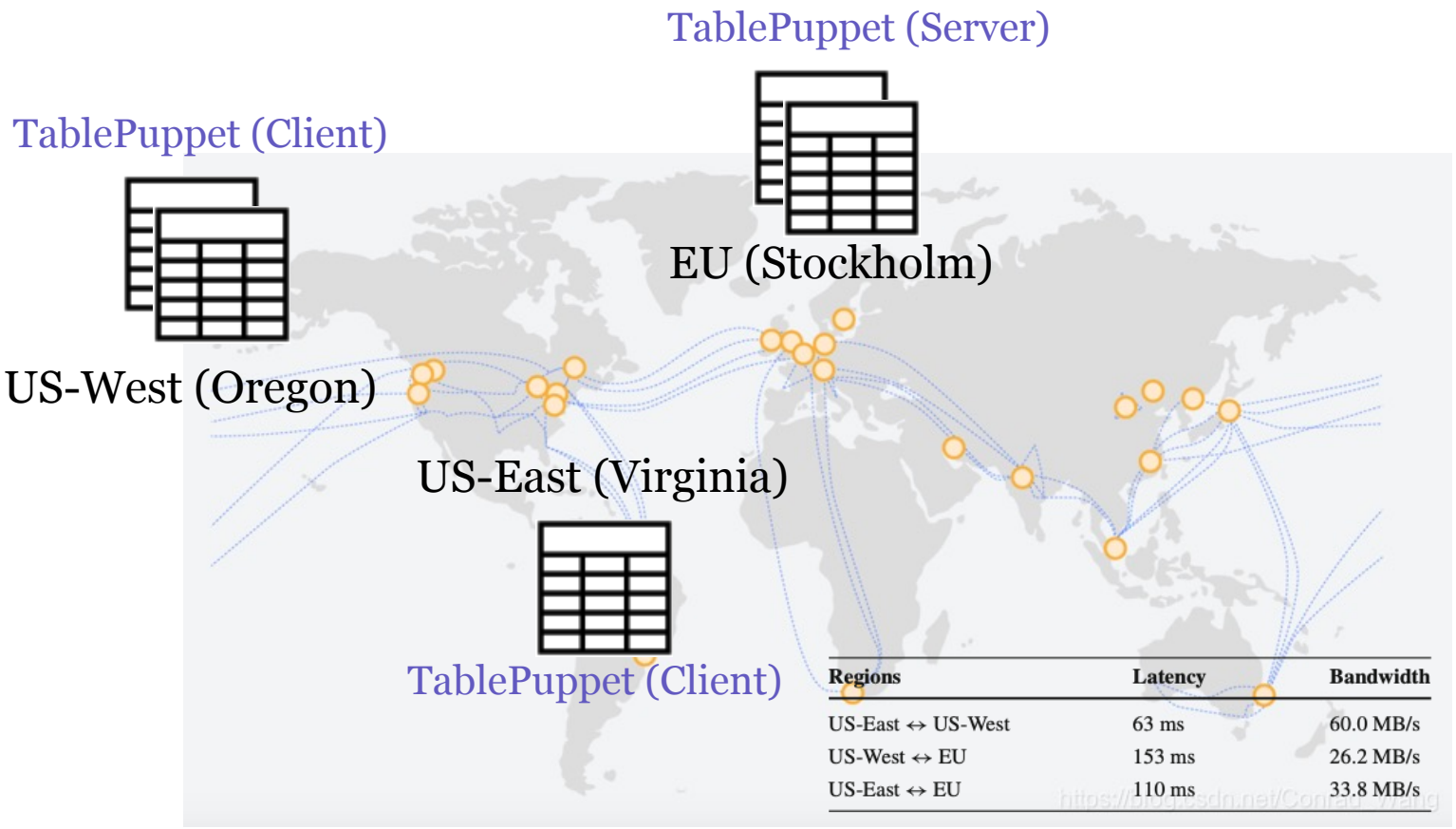
- Where to add DP?
  - Table features
  - Local model updates (DP-SGD)
  - Model predictions
  - Label (label-DP)
  - Server-side computation
  - Auxiliary (intermediate) variables



Evaluation – Experiments on AWS cloud instances

Train both linear models and neural network models
on three datasets with multiple tables

Dataset (ML Model)	#Table	#Tuple	#Feature
MIMIC-III (NN)	5	[35K, 2.9M]	[2, 717]
Yelp (BERT-Softmax)	3	[35K, 3.2M]	[3, 775]
MovieLens-1M (Linear)	3	[6K, 0.9M]	[4, 52]
MovieLens-1M (NN)	3	[6K, 0.9M]	[4, 52]



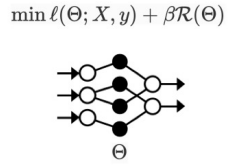
Regions	Latency	Bandwidth
US-East ↔ US-West	63 ms	60.0 MB/s
US-West ↔ EU	153 ms	26.2 MB/s
US-East ↔ EU	110 ms	33.8 MB/s

Accuracy evaluation: Model accuracy on TablePuppet vs. Model accuracy directly on the global joined table

Performace evalution: Training time and communication time (TablePuppet vs. Federated learning)

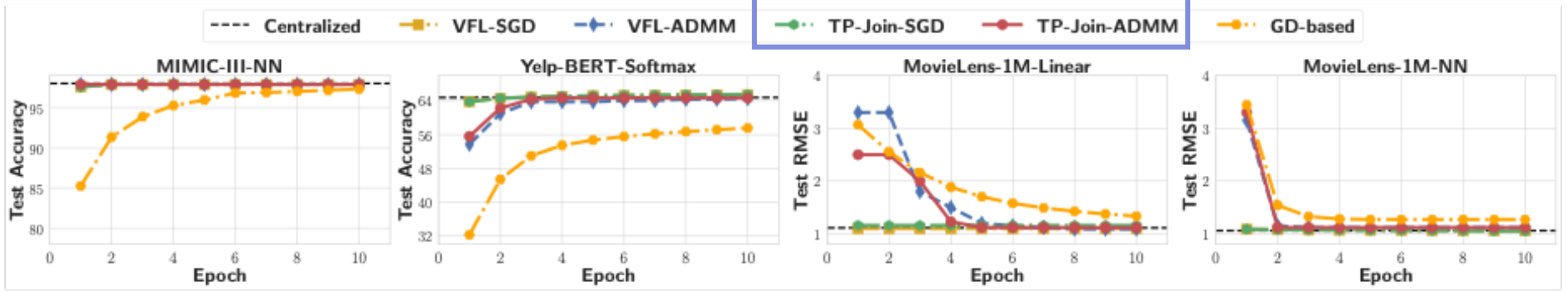
Evaluation - Accuracy results

Baseline: Directly running ML over the global joined table (dash line)

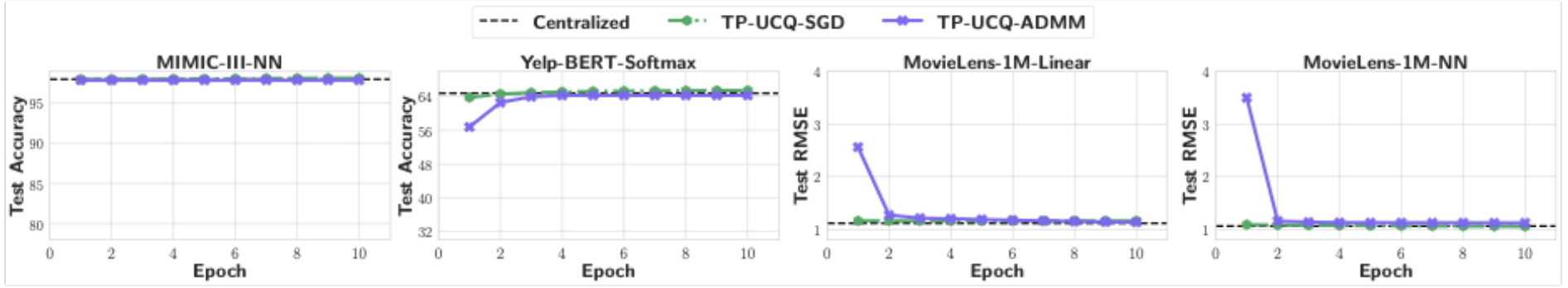


Result: SGD/ADMM algorithms atop TablePuppet can achieve comparable model accuracy to the baseline.

Join-only scenario



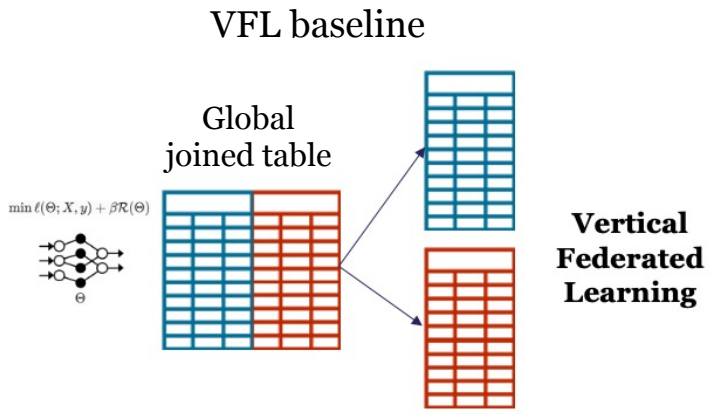
Join-Union scenario

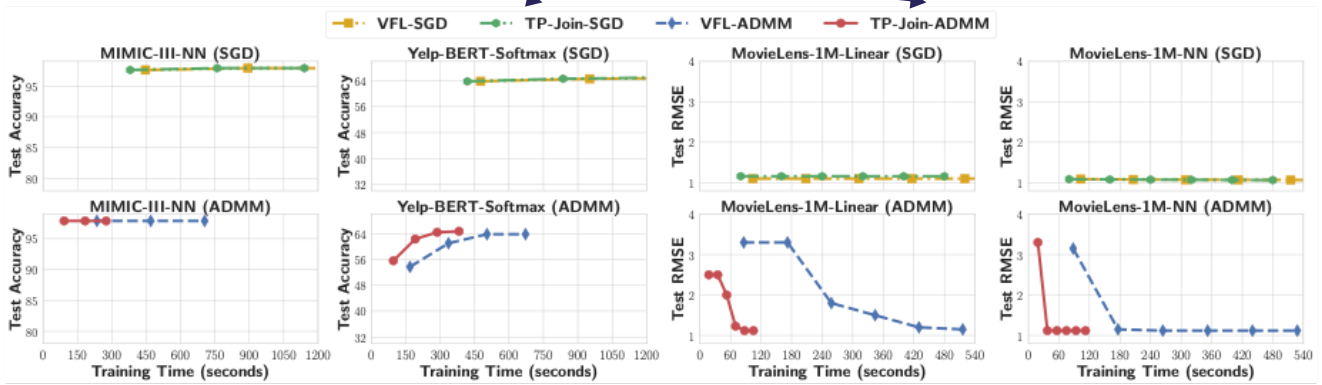


Implication: Our two-level decomposition approach with optimizations on duplicate tuples is effective.

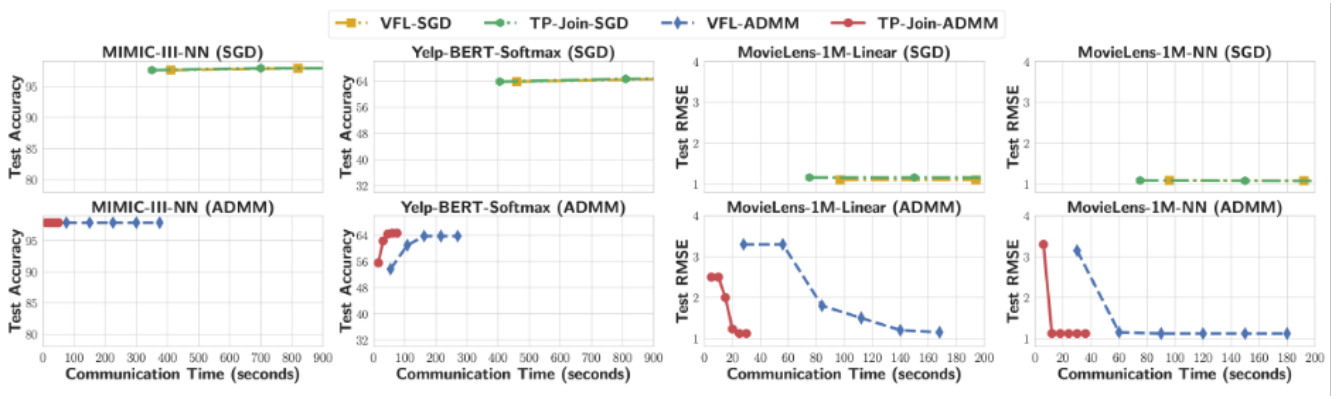
Evaluation - Performace reusults in Join-Only scenario

Baseline: Vertical Federated Learning (VFL) approaches on global joined table that are vertically partitioned.
Result: SGD/ADMM algorithms atop TablePuppet have less training & communication time than baseline.





Implication: Our optimizations on duplicated tuples are efficient (reduces the computation and communication overhead)



Evaluation - Performance results in Join-Union scenario

Result: ADMM atop TablePuppet takes shorter communication time to converge.

Join-Union scenario

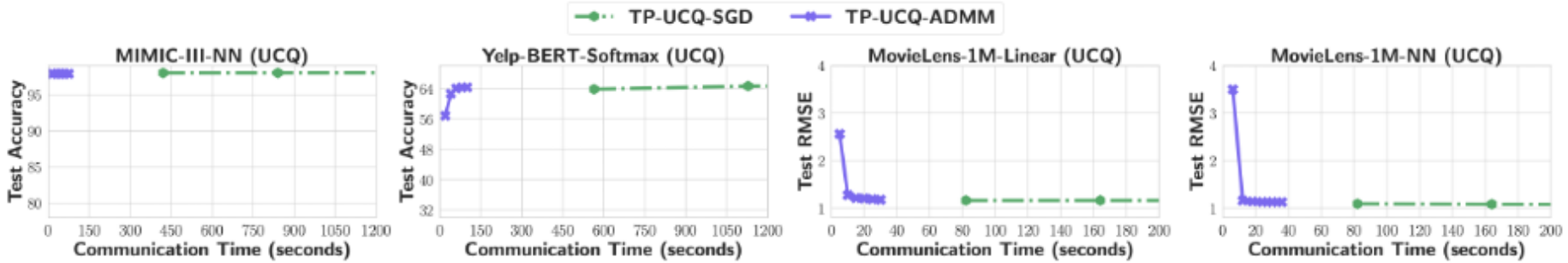


Fig. 10 The model accuracy vs. communication time for join-union scenario.

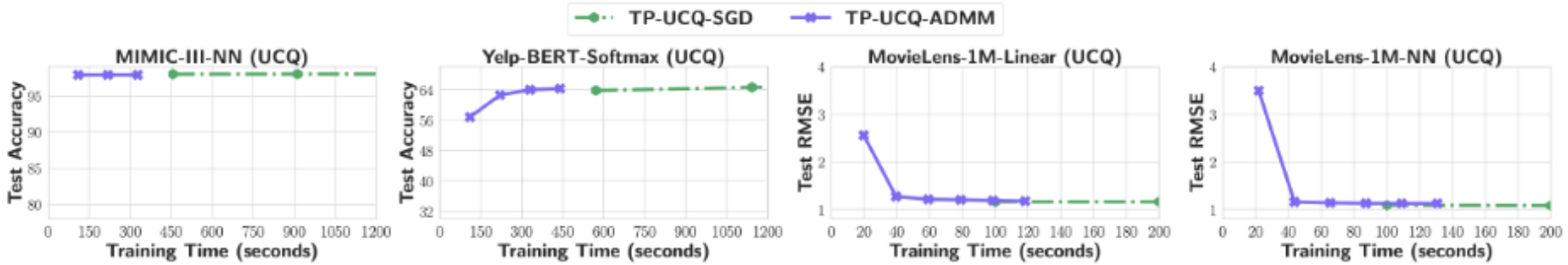
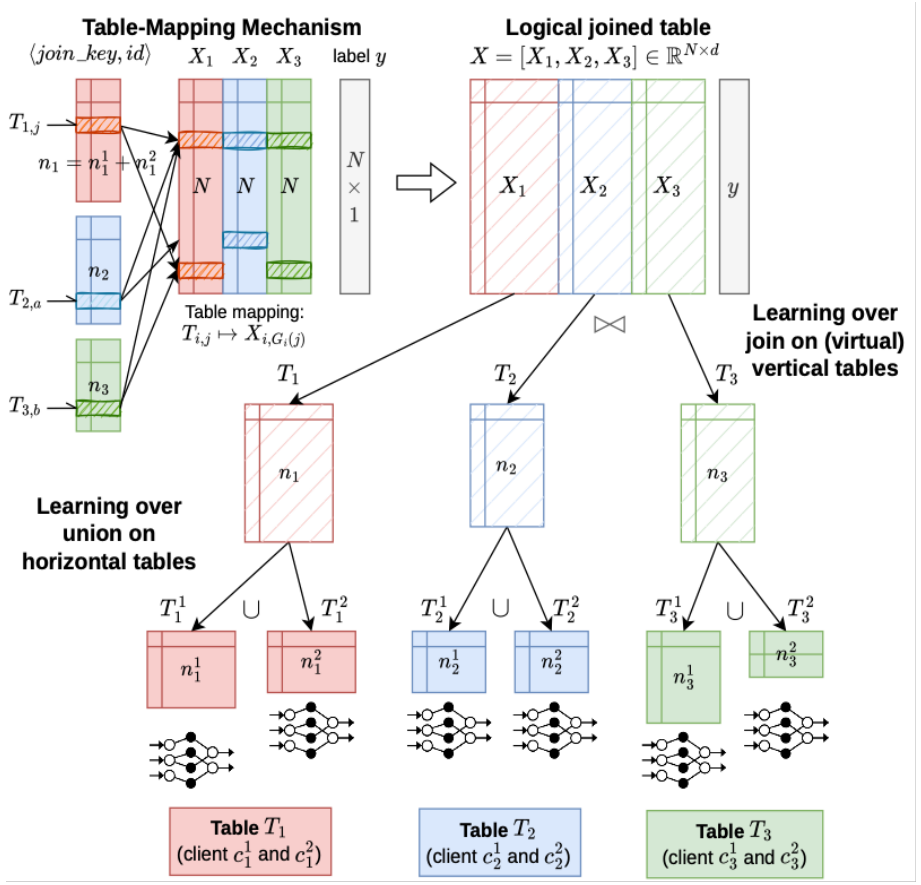


Fig. 11 The model accuracy vs. training time for join-union scenario.

Open source

TablePuppet framework



https://github.com/JerryLead/tablepuppet

Flower

Federated Baseline Contributions!

General

williamlm 11 Jan 19

Flower Labs is continuously looking to integrate baselines to our Flower framework. There's always an opportunity to make a difference and contribute to open-source code by adding another cool baseline to the repository.

Below you can find a list of baselines that can be implemented:

- FedDF - Ensemble Distillation for Robust Model Fusion in Federated Learning 15
- FedNTD - Preservation of the Global Knowledge by Not-True Distillation in Federated Learning 2
- FedSAM - Improving Generalization in Federated Learning by Seeking Flat Minima 4
- FedSMOO - Dynamic Regularized Sharpness Aware Minimization in Federated Learning: Approaching Global Consistency and Smooth Landscape 3
- q-FFL - Fair Resource Allocation in Federated Learning 3
- TablePuppet - A Generic Framework for Relational Federated Learning 3
- FedPG-BR - Fault-Tolerant Federated Reinforcement Learning with Theoretical Guarantee 2
- FedRS - Federated Learning with Restricted Softmax for Label Distribution Non-IID Data
- FedDefender - FedDefender: Backdoor Attack Defense in Federated Learning
- Ditto - Fair and Robust Federated Learning Through Personalization 1

Thank you! Q&A

TablePuppet framework