Helen: Maliciously Secure Coopetitive Learning for Linear Models

CS294 AI-Sys
Presented by Dylan Dreyer
April 10, 2019

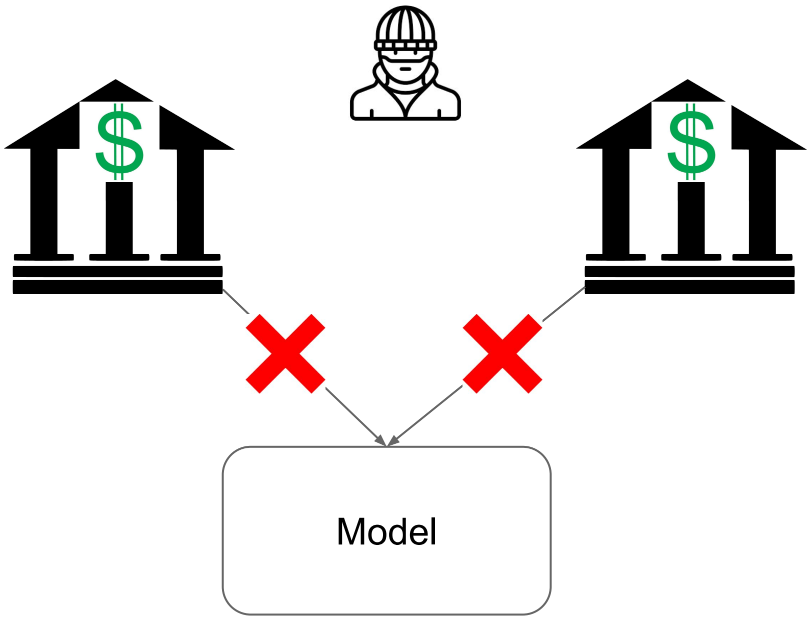
Outline

- Motivation + Problem Statement
- Background + Threat Model
- Overview of Helen + Key Features
- Results
- Discussion

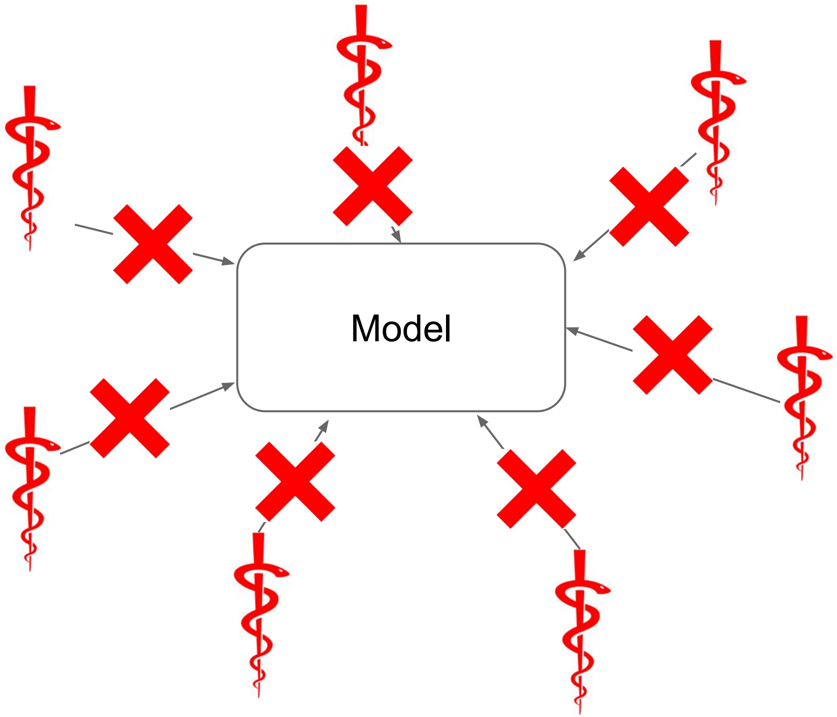
Outline

- Motivation + Problem Statement
- Background + Threat Model
- Overview of Helen + Key Features
- Results
- Discussion

Motivation #1



Motivation #2



Problem

“collaboratively train machine learning models on combined datasets for a common benefit”

“organizations cannot share their sensitive data in plaintext due to privacy policies and regulations or due to business competition”

Outline

- Motivation + Problem
- Background + Threat Model
- Overview of Helen + Key Features
- Results
- Discussion

Background on Coopetitive Learning

- Coopetitive -> cooperative and competitive
- Secure multi-party computation (MPC)
  - inefficient
- Previous works are limited
  - unrealistic threat models
  - limited to two parties

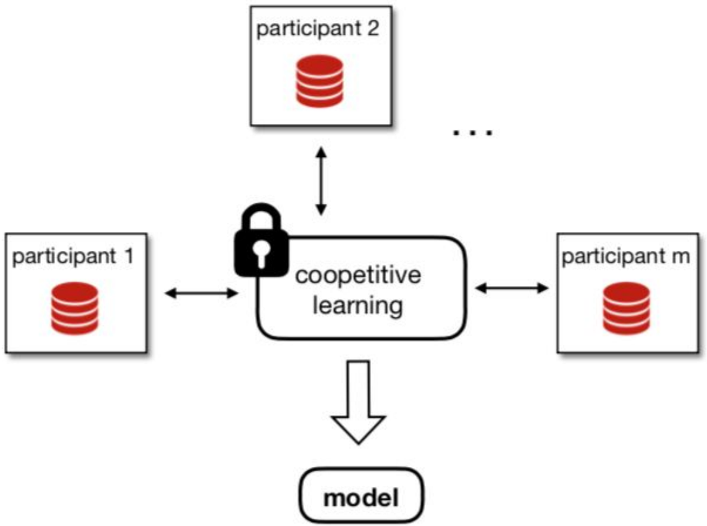


Fig. 1: The setting of coopetitive learning.

Threat Model

- malicious setting - only trust yourself!
- all other parties can misbehave/be malicious during protocol
- all parties agree on a functionality to compute
- confidentiality of final model not protected

Background on Crypto Building Blocks

- threshold partially homomorphic encryption
  - partially homomorphic
    - ex. Paillier -> Enc(X) * Enc(Y) = Enc(X+Y)
  - threshold
    - need enough shares of secret key to decrypt
- zero knowledge proofs
  - prove that a certain statement is true without revealing the prover’s secret
- secure multi party computation
  - jointly compute a function over inputs while keeping inputs private
  - SPDZ chosen over garbled circuits because matrix operations are more efficient

Outline

- Motivation + Problem
- Background + Threat Model
- Overview of Helen + Key Features
- Results
- Discussion

Overview of Helen

- platform for maliciously secure coopetitive learning
- supports regularized linear models
    - paper notes that these types of models are widely used
- few organizations, lots of data, smaller number of features

Key Features of Helen

- Overarching goal: Make expensive cryptographic computation independent of number of training samples
- Make all parties commit to input dataset and prove it
- Use ADMM (Alternating Direction Method of Multipliers)/LASSO
- use partially homomorphic encryption to encrypt global weights such that each party can compute in a decentralized manner
- 5 phases
    - Agreement Phase
    - Initialization Phase
    - **Input Preparation Phase**
    - **Model Compute Phase**
    - **Model Release Phase**

Input Preparation Phase

- Goal: broadcast encrypted summaries of data and commit
- Why? Malicious parties could use inconsistent data during protocol
- How? Encrypt data and attach various proofs of knowledge
- Naive method: commit on input dataset
    - crypto computation scales linearly
    - requires complex matrix inversions in MPC

$$\mathbf{A}_i = (\mathbf{X}_i^T \mathbf{X}_i + \rho \mathbf{I})^{-1}$$

$$\mathbf{b}_i = \mathbf{X}_i^T \mathbf{y}_i$$

Input Preparation Phase

- Goal: broadcast encrypted summaries of data and commit
- Why? Malicious parties could use inconsistent data during protocol
- How? Encrypt data and attach various proofs of knowledge
- **Better method: Decompose A and b via SVD**
  - all of these matrices are dimension *d*, no longer *n*
  - Each party broadcasts encrypted A, b, y*, V, Σ, Θ along with proofs of knowledge

$$\mathbf{A}_i = (\mathbf{X}_i^T \mathbf{X}_i + \rho \mathbf{I})^{-1}$$

$$\mathbf{b}_i = \mathbf{X}_i^T \mathbf{y}_i$$

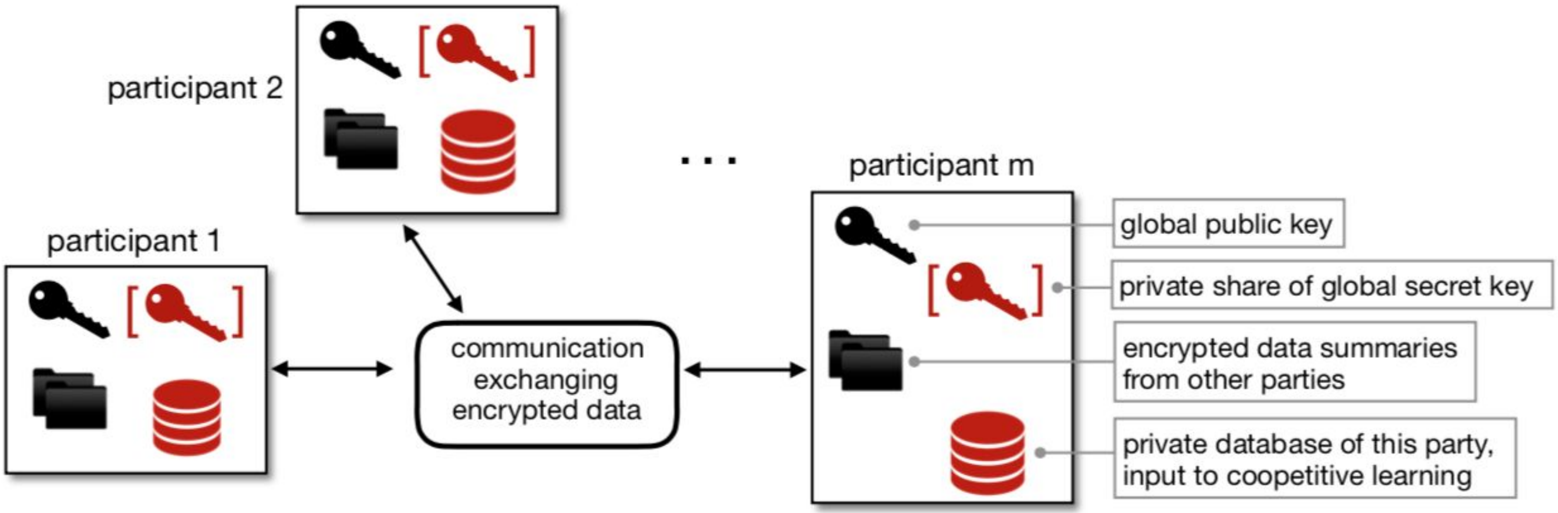
$\longrightarrow$

$$\mathbf{A} = \mathbf{V} \mathbf{\Theta} \mathbf{V}^T,$$

$$\mathbf{b} = \mathbf{V} \mathbf{\Sigma}^T \mathbf{y}^*,$$

Input Preparation Phase

- End of input preparation phase



Model Compute Phase

- Goal: run ADMM algorithm iteratively and update encrypted global weights
- Why ADMM?
  - efficient for linear models
  - converges in few iterations (10)
  - supports decentralized computation
  - reduces number of expensive MPC syncs
  - thus, efficient for cryptographic training

Model Compute Phase

- Goal: run ADMM iteratively to update encrypted global weights

- 1. Local optimization
    - Each party calculates Enc(w_i^{k+1})
    - also generate a proof of this
- 2. Coordination using MPC
    - Parties use input summaries to verify Enc(w_i^{k+1})
    - Convert weights to MPC
    - Compute softmax via MPC
    - Convert z back into encrypted form

The coopetitive learning task for LASSO

Input of party P_i: $\mathbf{X}_i, \mathbf{y}_i$

1) $\mathbf{A}_i \leftarrow \left(\mathbf{X}_i^T \mathbf{X}_i + \rho \mathbf{I}\right)^{-1}$
2) $\mathbf{b}_i \leftarrow \mathbf{X}_i^T \mathbf{y}_i$
3) $\mathbf{u}^{\mathbf{0}}, \mathbf{z}^{\mathbf{0}}, \mathbf{w}^{\mathbf{0}} \leftarrow \mathbf{0}$
4) `For` $k = 0$, `ADMMIterations`-1:
    a) $\mathbf{w}_i^{k+1} \leftarrow \mathbf{A}_i(\mathbf{b}_i + \rho\left(\mathbf{z}^k - \mathbf{u}_i^k\right))$
    b) $\mathbf{z}^{k+1} \leftarrow S_{\lambda/m\rho}\left(\frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{w}_i^{k+1} + \mathbf{u}_i^k\right)\right)$
    c) $\mathbf{u}_i^{k+1} \leftarrow \mathbf{u}_i^k + \mathbf{w}_i^{k+1} - \mathbf{z}^{k+1}$

Model Release Phase

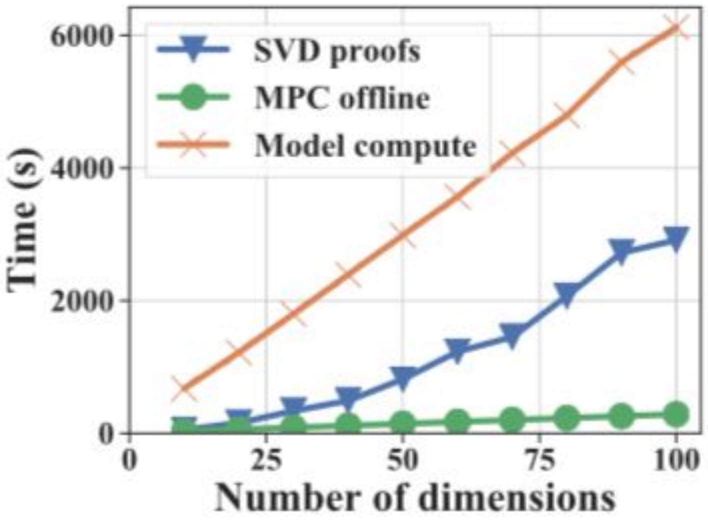
- Goal: jointly decrypt and release model parameters (z)
  - ciphertext to MPC conversion
  - verify this conversion
  - jointly decrypt model parameters (z)

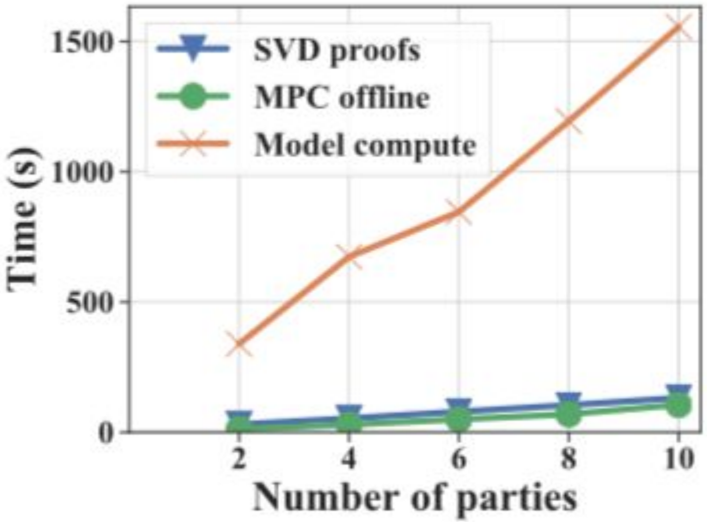
Outline

- Motivation + Problem
- Background + Threat Model
- Overview of Helen + Key Features
- Results
- Discussion

Results

- Evaluation of runtime of Helen's different phases using a synthetic dataset





Results

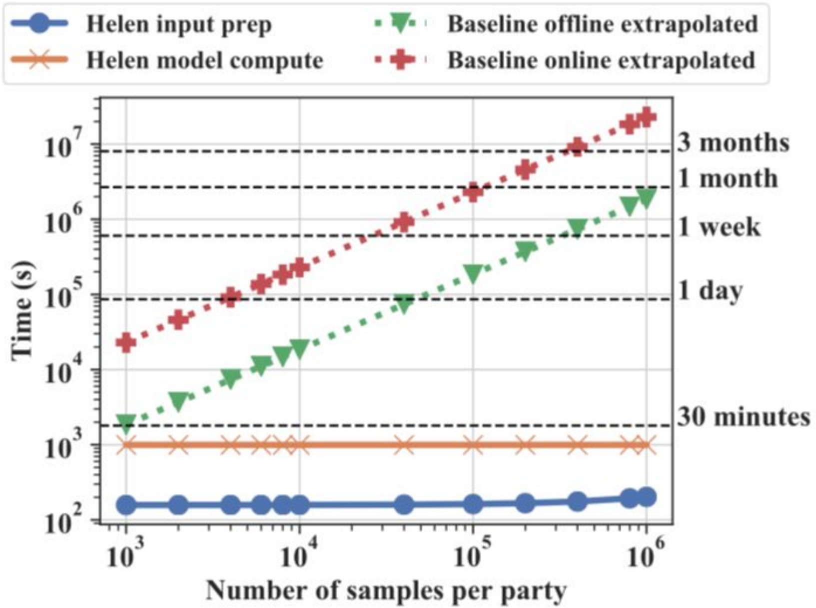
Samples per party	2000	4000	6000	8000	10K	40K	100K	200K	400K	800K	1M
sklearn L2 error	8937.01	8928.32	8933.64	8932.97	8929.10	8974.15	8981.24	8984.64	8982.88	8981.11	8980.35
Helen L2 error	8841.33	8839.96	8828.18	8839.56	8837.59	8844.31	8876.00	8901.84	8907.38	8904.11	8900.37
sklearn MAE	57.89	58.07	58.04	58.10	58.05	58.34	58.48	58.55	58.58	58.56	58.57
Helen MAE	57.23	57.44	57.46	57.44	57.47	57.63	58.25	58.38	58.36	58.37	58.40

TABLE II: Select errors for gas sensor (due to space), comparing Helen with a baseline that uses sklearn to train on all plaintext data. L2 error is the squared norm; MAE is the mean average error. Errors are calculated after post-processing.

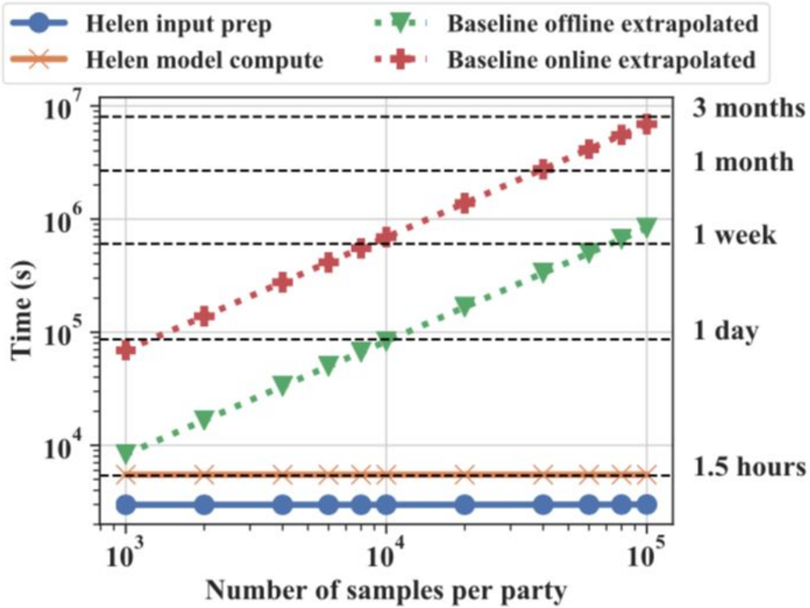
Samples per party	1000	2000	4000	6000	8000	10K	20K	40K	60K	80K	100K
sklearn L2 error	92.43	91.67	90.98	90.9	90.76	90.72	90.63	90.57	90.55	90.56	90.55
Helen L2 error	93.68	91.8	91.01	90.91	90.72	90.73	90.67	90.57	90.54	90.57	90.55
sklearn MAE	6.86	6.81	6.77	6.78	6.79	6.81	6.80	6.79	6.79	6.80	6.80
Helen MAE	6.92	6.82	6.77	6.78	6.79	6.81	6.80	6.79	6.80	6.80	6.80

TABLE III: Errors for song prediction, comparing Helen with a baseline that uses sklearn to train on all plaintext data. L2 error is the squared norm; MAE is the mean average error. Errors are calculated after post-processing.

Results



Helen input prep
Helen model compute
Baseline offline extrapolated
Baseline online extrapolated
Time (s)
Number of samples per party
3 months
1 month
1 week
1 day
30 minutes



Helen input prep
Helen model compute
Baseline offline extrapolated
Baseline online extrapolated
Time (s)
Number of samples per party
3 months
1 month
1 week
1 day
1.5 hours

Outline

- Motivation + Problem
- Background + Threat Model
- Overview of Helen + Key Features
- Results
- Discussion

Discussion

- Is there a need to extend to other types of models? Consequences of this?
- Trusted hardware (enclaves) is another popular approach to computing on sensitive data. Is it more viable?
- What happens when more parties get involved? Comparison vs. federated learning?
- Questions?