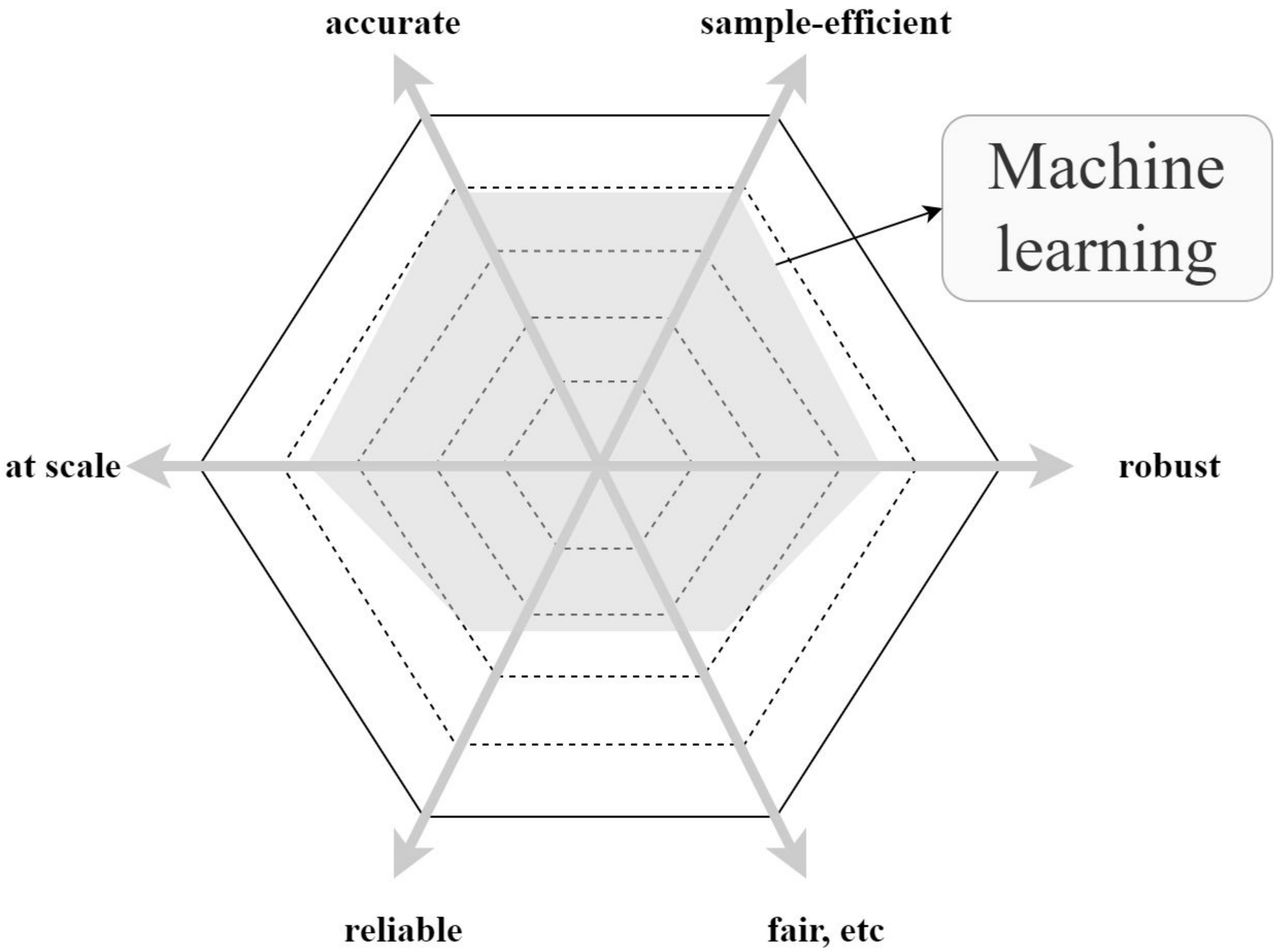
Reconstructing Training Data from Model Gradient, Provably

Qi Lei, Courant Math and CDS

With Zihan Wang, Jason Lee

https://arxiv.org/abs/2212.03714


accurate
sample-efficient
Machine learning
at scale
robust
reliable
fair, etc

Privacy！

Privacy attacks: infer information about user data/protected model

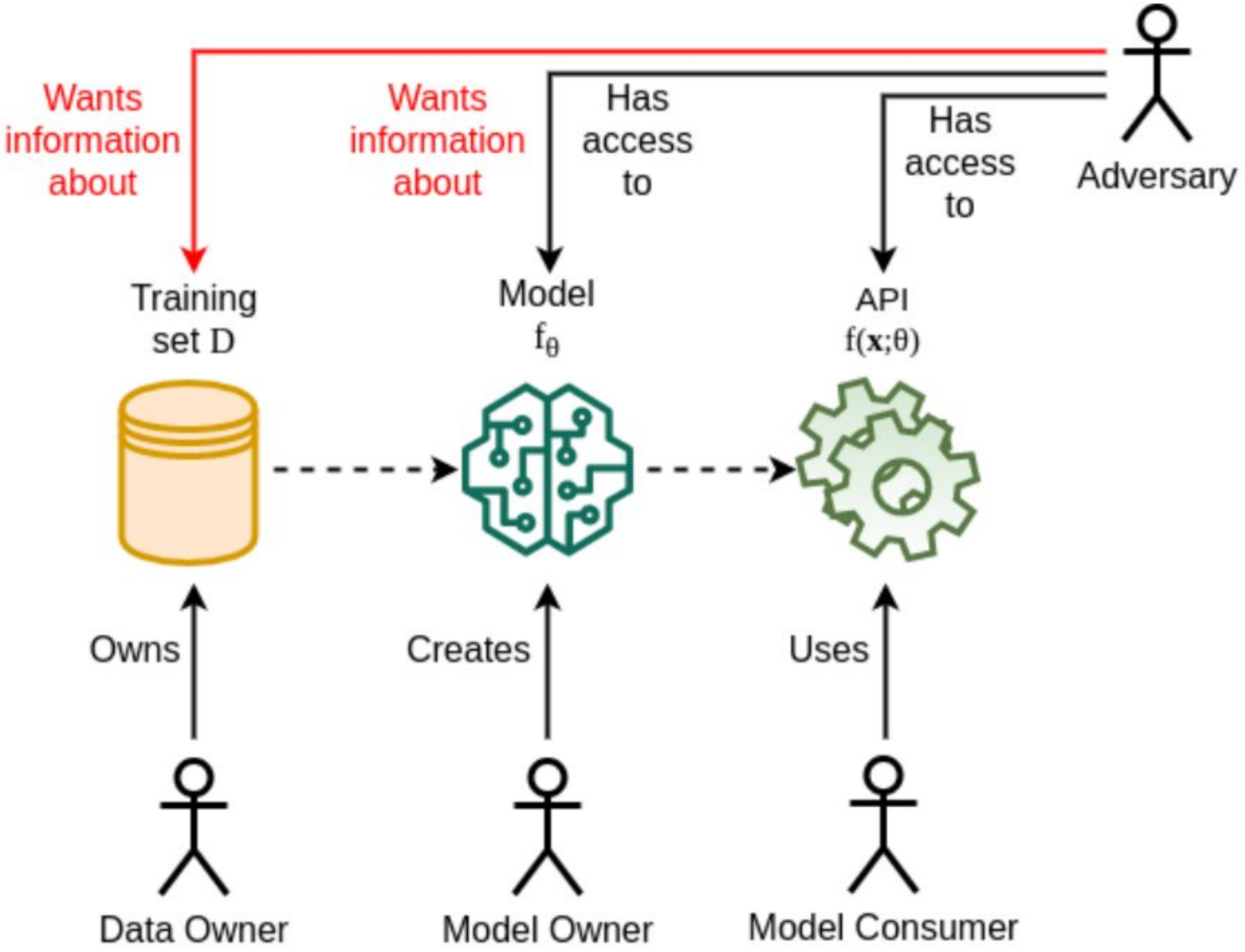
- Data curation stage
  - Data linkage attack (reidentification)

- Model training phase
  - Data reconstruction attack

- Inference/prediction time
  - Membership inference attack (tracing attack)
  - Data reconstruction attack

Priva ıser data

- Data (
  - Dat
- Mode
  - Dat
- Infere
  - Me
  - Dat



Resemblance to inverse problems

Different types of privacy attacks

- Membership Inference Attacks (differential privacy) [Shokri et al 2017]
- Reconstruction Attacks (our target) [Gong&Niu,2016]

- Property Inference Attacks [Ateniese et al 2015]
- Model Extraction Attacks (knowledge distillation) [Papernot et al 2018]

Different types of privacy attacks

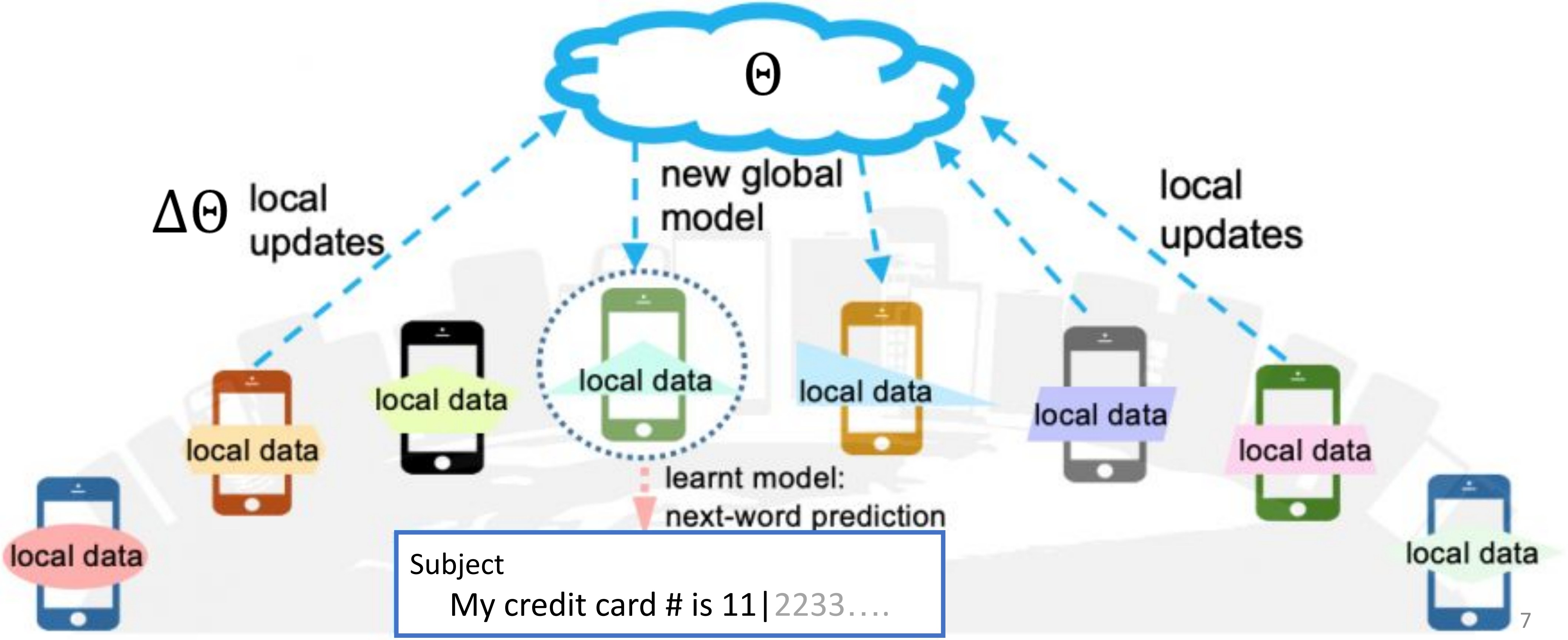
- Membership Inference Attacks (differential privacy) [Shokri et al 2017]
- **Reconstruction Attacks (our target) [Gong&Niu,2016]**

- Property Inference Attacks [Ateniese et al 2015]
- Model Extraction Attacks (knowledge distillation) [Papernot et al 2018]

Federated learning

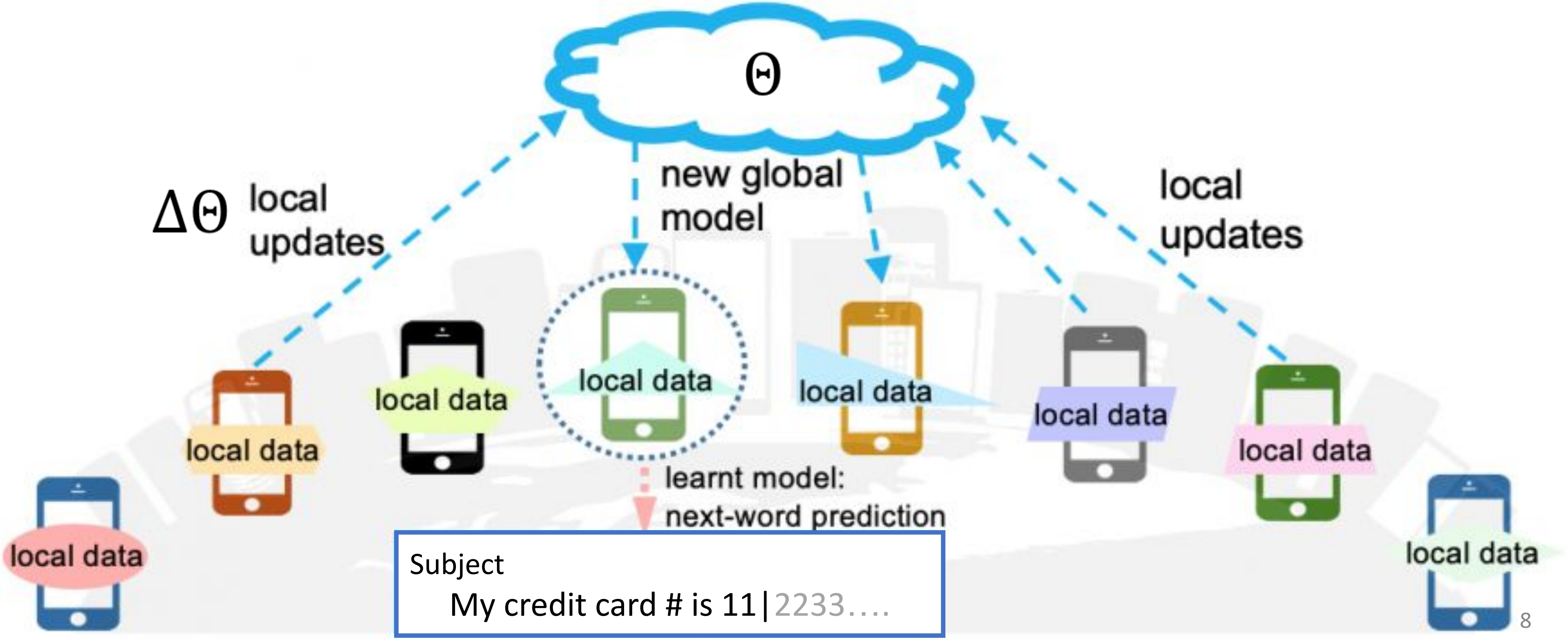
[Konečný et al. 2016, McMahan et al. 2017]

- Privacy leakage in distributed learning - Data and model not co-located



Privacy leakage in distributed learning

- Does local update reveal the training data?



Federated learning

Gautam Kamath
@thegautamkamath

Federated learning, wherein training data never leaves the user's device (only gradients or model parameters), is an effective way to protect the privacy of individual training points.

True	19.4%
False	**46.1%**
I don't know/show me	34.5%

956 votes · Final results

11:32 PM · Dec 11, 2022

More formal statement

- Local batch of data: $S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_B, y_B)\}$
- Prediction function: $x \rightarrow f(x; \Theta)$
- Local update: G : = $\frac{1}{B} \nabla_\Theta \sum_{i=1}^{B} \ell(f(x_i, \Theta), y_i)$

Fundamental questions

- Is the model gradient G sufficient to identify the training samples?
- If so, is there efficient algorithm to recover the samples?

Prior work

- Attacking methods
  - Learn to generate the training samples from a local user
  - Match the gradient: $\min_{S=\{(x_i,y_i)\}} \left|\left|G - \sum_{i=1}^{B} \nabla \ell(f(x_i;\Theta), y_i)\right|\right|^2$
  - Model inversion at inference time

- Defending methods
  - Quantizing the gradient
  - Add noise

Some folklore from empirical findings

What parameters to query at?

- Moderately trained model
  - Random network hasn't memorized the data,
  - Well-trained model makes gradient vanish

Wrong impressions!

Is gradient alone enough to identify the images?

- Prior work believed not.
  - Introduce prior information of the training data (model by generative models)

Setting

- Warm-up: two-layer neural network

$$f(x; \{W, a\}) = \sum_{j=1}^{m} a_j \sigma\left(w_j^\top x\right) = a^\top \sigma(W^\top x)$$

- Choose proper w_j, a_j to query the gradient at

$$\nabla_{a_j} L = \sum_{i=1}^{B} l_i' \sigma\left(w_j^\top x_i\right)$$

Caveat on linear and quadratic activations:

- Linear setting:
- $\nabla_a L = W\left(\sum_{i=1}^{B} l_i' x_i\right); \nabla_W L = a\left(\sum_{i=1}^{B} l_i' x_i\right)^{\top}$
- Can only identify a linear combination of X

Caveat on linear and quadratic activations:

- Linear setting:
- $\nabla_a L = W\left(\sum_{i=1}^{B} l_i' x_i\right); \nabla_W L = a\left(\sum_{i=1}^{B} l_i' x_i\right)^\top$
- Can only identify a linear combination of X

- Quadratic setting:
- $\nabla_{a_j} L = w_j^\top \bar{\Sigma} w_j; \nabla_{w_j} L = 2\bar{\Sigma} w_j$, here $\bar{\Sigma} = \sum_{i=1}^{B} l_i' x_i x_i^\top$
- Can only identify the span of X

Our algorithm: using Stein's lemma

- Stein's lemma: $E_{w\sim N(\mathbb{0},I)}\left[g(a^\top w)H_p(w)\right] = E\left[g^{(p)}a^{\otimes p}\right]$.
- Hermite function: $H_2(w) = ww^\top - I, H_3(w) = w^{\otimes 3} - w \widetilde{\otimes} I$.

- $\widehat{T_p} := \frac{1}{m}\sum_{j=1}^m g(w_j)H_p(w_j) \approx E_{w\sim N(\mathbb{0},I)}\left[g(w)H_p(w)\right]$

$$\equiv \sum_{i=1}^{B} E\left[\sigma^{(p)}(w^\top x_i)x_i^{\otimes p}\right] =: T_p$$

- $g(w_j) := \nabla_{a_j}L = \sum_{i=1}^{B} l_i'\sigma(w_j^\top x_i)$ is our observation from the model gradient

Tensor decomposition

- Now we can estimate $T_p := \sum_{i=1}^{B} E\left[\sigma^{(p)}(w^\top x_i) x_i^{\otimes p}\right]$
- Uniquely identify $\{x_1, x_2, \cdots, x_B\}$ through tensor decomposition when data is linearly independent for p>=3. [Kuleshov et al. 2015]

- Our strategy: choose $a_j = \frac{1}{m}, w_j \sim N(0, I)$, estimate T by $\widehat{T_3} \coloneqq \frac{1}{m}\sum_{j=1}^{m} g\left(w_j\right) H_3\left(w_j\right)$, $g\left(w_j\right) \coloneqq \nabla_{a_j} L$

Improving the dimension dependence

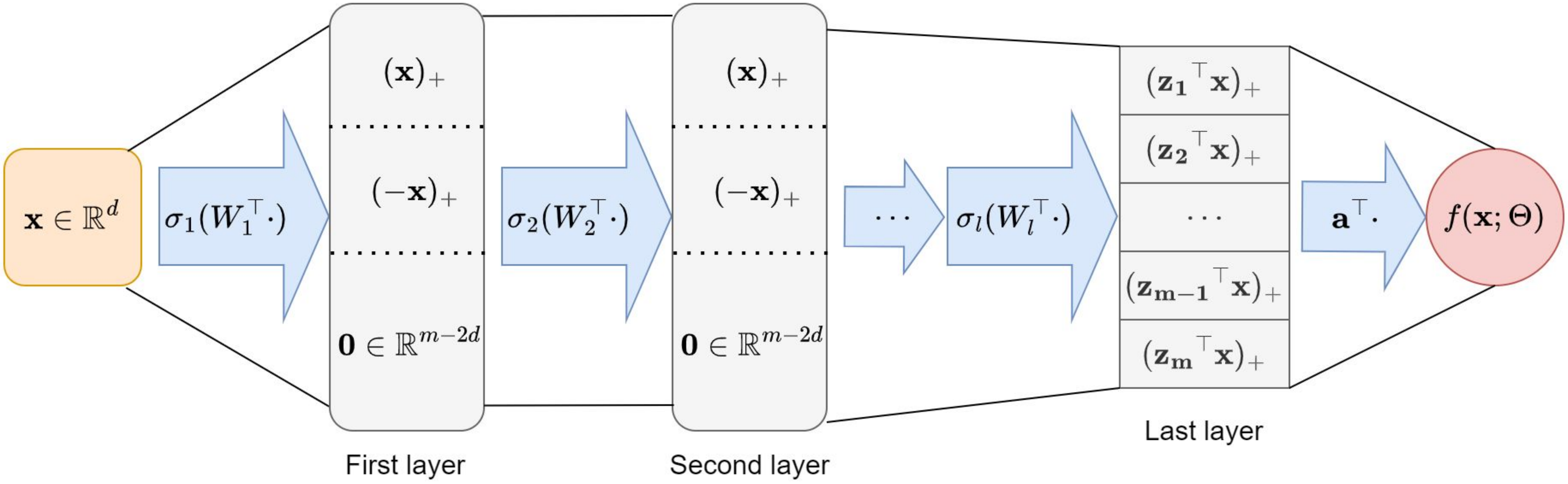
- First estimate the span of $\{x_1, x_2, \cdots, x_B\}$ by decomposing $T_2 = VV^\top$. $V \in R^{d\times B}$.
- Find $T_3(V, V, V) \in R^{B\times B\times B}$ and conduct tensor decomposition $\{u_1, u_2, \cdots, u_B]$.
- Project back to the original space $\widehat{x_j} = Vu_j$.
- Can also use the estimated x as initialization and do gradient matching.

-relevant strategy appeared in [Zhong et al 2017] for optimizing over 2-layer neural network (dual problem)

Theoretical analysis

- Applies when $E\left[\sigma^{(3)}(w)\right]$ or $E\left[\sigma^{(4)}(w)\right] \neq 0$. Applies to sigmoid, tanh, ReLU, leaky ReLU.
- Reconstruction error $\leq \tilde{O}\left(\sqrt{d/m}\right)$.

Extension to deeper neural networks



- Previous findings: if last two layers are fully connected, can recover the features from the $(l-2)$-th layer
- Other structured data modalities: recover the embeddings first

Discussions

- Identifiability
  - $E\left[\sigma^{(3)}(w)\right]$ or $E\left[\sigma^{(4)}(w)\right] \neq 0$ (work for most activation functions)
  - # of hidden nodes m scales at least linearly with d
  - Deeper neural network does not help or hurt

- Distinction to linear case or convex optimization:
  - Linear and neural networks are fundamentally different on whether the gradient leaks data

- Inspiration on defending algorithms:
  - Adding noise does not help

Thank you