HUMBOLDT-UNIVERSITÄT ZU BERLIN

Make Some ROOM for the Zeros

Data Sparsity in Secure Distributed Machine Learning

Phillipp Schoppmann, Adrià Gascón, Mariana Raykova, Benny Pinkas

June 20, 2019

Data Sparsity

Data Sparsity

- Netflix dataset: 480k users, 17k movies, but only 100M out of 8.5B potential ratings.
  $< 1.2\%$
- Genomics: 3.2B base pairs, but a typical genome differs only at 5M sites.
  $< 0.2\%$
- 20 Newsgroups dataset: 9k vectors, 10^5 features, but only 100 non-zeros per vector.
  $< 0.1\%$

Sparse Storage Formats

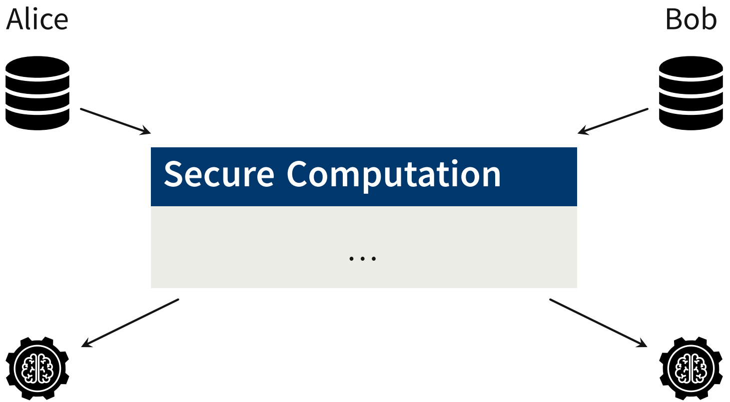
Let R be an arbitrary ring, $\mathbf{b} \in R^d$ be a vector, $\mathbf{A} \in R^{n \times d}$ a matrix.

- Sparse vector: $\textsc{Sparse}(\mathbf{b}) := \big((i, b_i)\big)_{b_i \neq 0}$
- Sparse matrix: $\textsc{Sparse}(\mathbf{A}) := \big(\textsc{Sparse}(\mathbf{a}_i)\big)_{i \in [n]}$

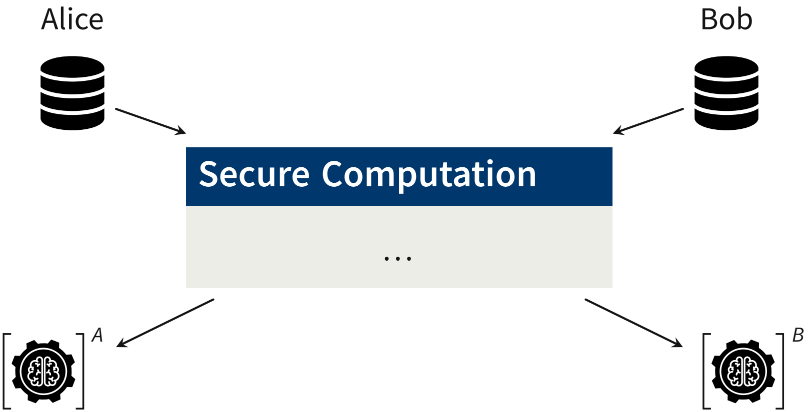
Related to the *Compressed Sparse Row (CSR)* or *Yale format* that is used in scientific computing libraries such as Eigen, SciPy, …

Secure Distributed Machine Learning

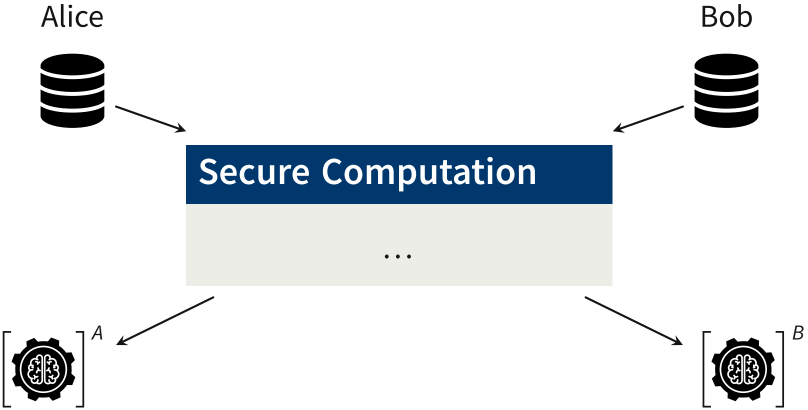
Two-Party Machine Learning



Two-Party Machine Learning

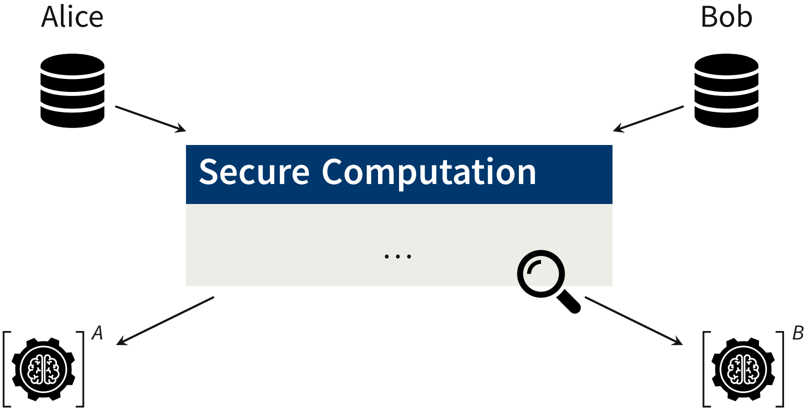


Two-Party Machine Learning



Throughout this talk: Two parties, semi-honest security

Two-Party Machine Learning



Throughout this talk: Two parties, semi-honest security

Building Block: Matrix-Vector Multiplication

$\mathbf{A} \in R^{n \times d}$

$\mathbf{b} \in R^{d}$

Secure Computation

Choose random $[\mathbf{c}]^A$, $[\mathbf{c}]^B$, such that

$$[\mathbf{c}]^A + [\mathbf{c}]^B = \mathbf{Ab}$$

$[\mathbf{c}]^A$

$[\mathbf{c}]^B$

P. Mohassel and Y. Zhang. 'SecureML: A System for Scalable Privacy-Preserving Machine Learning'. In: *IEEE Symposium on Security and Privacy*. 2017, pp. 19–38

Sparse Matrix-Vector Multiplication

☺ We don’t have to multiply elements if one of the factors is zero.

Sparse Matrix-Vector Multiplication

☺ We don’t have to multiply elements if one of the factors is zero.

☹ We can’t simply reveal which elements are zero.

Sparse Matrix-Vector Multiplication

☺ We don’t have to multiply elements if one of the factors is zero.

☹ We can’t simply reveal which elements are zero.

☺ In many settings, an upper bound on the *number* of non-zero elements is public.

Sparse Matrix-Vector Multiplication

☺ We don’t have to multiply elements if one of the factors is zero.

☹ We can’t simply reveal which elements are zero.

☺ In many settings, an upper bound on the *number* of non-zero elements is public.

Our Approach

1. Encode sparse vector in a *Read-Only Oblivious Map (ROOM)* data structure.
2. Implement matrix-vector multiplication as a batched oblivious map access.

Basic Primitive: ROOM

Read-Only Oblivious Maps

Query key
q

Database D
(k_1, v_1)
(k_2, v_2)
$\vdots$
(k_l, v_l)

Default value
$\bar{v}$

ROOM Protocol

Compute w as

$$w = \begin{cases} v & \text{if } (q, v) \in D \\ \bar{v} & \text{otherwise.} \end{cases}$$

$[w]^C$

Secret-shared
value

$[w]^S$

Read-Only Oblivious Maps (2)

Query keys

q_1
q_2
$\vdots$
q_k

Database D

(k_1, v_1)
(k_2, v_2)
$\vdots$
(k_l, v_l)

Default values

$\bar{v}_1$
$\bar{v}_2$
$\vdots$
$\bar{v}_k$

ROOM Protocol

Compute $\mathbf{w}$ s. t. $\forall j \in [k]$:

$$w_j = \begin{cases} v & \text{if } (q_j, v) \in D \\ \bar{v}_j & \text{otherwise.} \end{cases}$$

$[\mathbf{w}]^C$

Secret-shared values

$[\mathbf{w}]^S$

Related Functionalities

	PIR [Cho+95]	Batched PIR [Ang+18]	Keyword PIR [CGN98]	Symmetric PIR [Ger+00]	Labeled PSI [Che+18]
Query privacy	✓	✓	✓	✓	✓
Database privacy	×	×	×	✓	✓
Sparse database	×	×	✓	×	✓
Batched queries	×	✓	×	×	✓
Shared Output	×	×	×	×	×

Related Functionalities

	PIR [Cho+95]	Batched PIR [Ang+18]	Keyword PIR [CGN98]	Symmetric PIR [Ger+00]	Labeled PSI [Che+18]
Query privacy	✓	✓	✓	✓	✓
Database privacy	×	×	×	✓	✓
Sparse database	×	×	✓	×	✓
Batched queries	×	✓	×	×	✓
Shared Output	×	×	×	×	×

ROOM is shorter than *batched symmetric keyword PIR with shared output.* ☺

Read-Only Oblivious Maps (2)

Query keys

q_1
q_2
$\vdots$
q_k

Database D

(k_1, v_1)
(k_2, v_2)
$\vdots$
(k_l, v_l)

Default values

$\bar{v}_1$
$\bar{v}_2$
$\vdots$
$\bar{v}_k$

ROOM Protocol

Compute $\mathbf{w}$ s. t. $\forall j \in [k]$:

$$w_j = \begin{cases} v & \text{if } (q_j, v) \in D \\ \bar{v}_j & \text{otherwise.} \end{cases}$$

$[\mathbf{w}]^C$

Secret-shared values

$[\mathbf{w}]^S$

Building a ROOM

Naive approach: Ignore database sparsity.

1. Server extends database with dummy elements to span entire key domain:

$$\mathbf{x} = (\perp, \ldots, \perp, \underset{\substack{\uparrow \\ \text{index } k_1}}{v_1}, \perp, \ldots, \perp, \underset{\substack{\uparrow \\ \text{index } k_2}}{v_2}, \ldots)$$

Building a ROOM

Naive approach: Ignore database sparsity.

1. Server extends database with dummy elements to span entire key domain:

$$\mathbf{x} = \big(\perp, \ldots, \perp, \underset{\substack{\uparrow \\ \text{index } k_1}}{v_1}, \perp, \ldots, \perp, \underset{\substack{\uparrow \\ \text{index } k_2}}{v_2}, \ldots\big)$$

2. Server encrypts $\mathbf{x}$ element-wise and sends it to client:

$$\tilde{\mathbf{x}} = (\mathrm{Enc}_K(x_1), \ldots, \mathrm{Enc}_K(x_d))$$

Building a ROOM

Naive approach: Ignore database sparsity.

1. Server extends database with dummy elements to span entire key domain:

$$\mathbf{x} = \big(\bot, \ldots, \bot, \underset{\substack{\uparrow \\ \text{index } k_1}}{v_1}, \bot, \ldots, \bot, \underset{\substack{\uparrow \\ \text{index } k_2}}{v_2}, \ldots\big)$$

2. Server encrypts **x** element-wise and sends it to client:

$$\tilde{\mathbf{x}} = (\mathrm{Enc}_K(x_1), \ldots, \mathrm{Enc}_K(x_d))$$

3. For each query q_i, Client selects $\tilde{x}_{q_i}$ and the parties perform an MPC with inputs $\tilde{x}_{q_i}$, K, $\bar{v}_i$. The MPC
   a) Decrypts $x_{q_i} = \mathrm{Dec}_K(\tilde{x_{q_i}})$,
   b) Secret-shares x_{q_i} if $x_{q_i} \neq \bot$, otherwise $\bar{v}_i$.

Building a ROOM

Naive approach: Ignore database sparsity.

1. Server extends database with dummy elements to span entire key domain:

$$\mathbf{x} = \big(\perp, \ldots, \perp, \underset{\substack{\uparrow \\ \text{index } k_1}}{v_1}, \perp, \ldots, \perp, \underset{\substack{\uparrow \\ \text{index } k_2}}{v_2}, \ldots\big)$$

2. Server encrypts **x** element-wise and sends it to client:

$$\tilde{\mathbf{x}} = (\mathsf{Enc}_K(x_1), \ldots, \mathsf{Enc}_K(x_d))$$

3. For each query q_i, Client selects $\tilde{x}_{q_i}$ and the parties perform an MPC with inputs $\tilde{x}_{q_i}, K, \bar{v}_i$. The MPC
   a) Decrypts $x_{q_i} = \mathsf{Dec}_K(\tilde{x_{q_i}})$,
   b) Secret-shares x_{q_i} if $x_{q_i} \neq \perp$, otherwise $\bar{v}_i$.

Communication linear in the key domain!

Building a ROOM

Naive approach: Ignore database sparsity.

1. Server extends database with dummy elements to span entire key domain:

$$\mathbf{x} = (\perp, \ldots, \perp, \underset{\substack{\uparrow \\ \text{index } k_1}}{v_1}, \perp, \ldots, \perp, \underset{\substack{\uparrow \\ \text{index } k_2}}{v_2}, \ldots)$$

2. Server encrypts $\mathbf{x}$ element-wise and sends it to client:

$$\tilde{\mathbf{x}} = (\mathrm{Enc}_K(x_1), \ldots, \mathrm{Enc}_K(x_d))$$

3. For each query q_i, Client selects $\tilde{x}_{q_i}$ and the parties perform an MPC with inputs $\tilde{x}_{q_i}$, K, $\bar{v}_i$. The MPC
   a) Decrypts $x_{q_i} = \mathrm{Dec}_K(\tilde{x_{q_i}})$,
   b) Secret-shares x_{q_i} if $x_{q_i} \neq \perp$, otherwise $\bar{v}_i$.

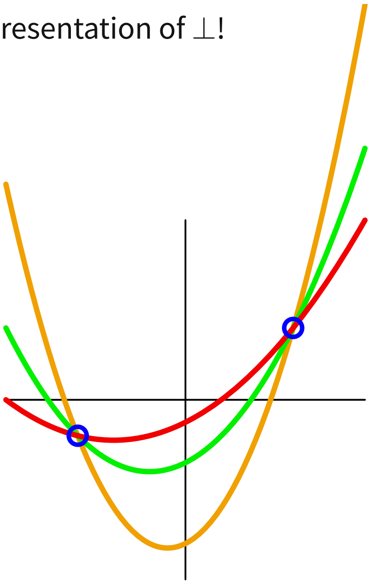
Communication linear in the key domain! ☹

Building a ROOM (2)

Idea: We don't need to have an explicit representation of $\perp$!

Building a ROOM (2)

Idea: We don't need to have an explicit representation of $\perp$!



Building a ROOM (2)

Idea: We don't need to have an explicit representation of $\perp$!

1. Server pads and encrypts each value in the database:

$$\tilde{\mathbf{v}} = \left(\mathrm{Enc}_K\left(v_1 || 0^s\right), \ldots, \mathrm{Enc}_K\left(v_l || 0^s\right)\right)$$

Building a ROOM (2)

Idea: We don't need to have an explicit representation of $\perp$!

1. Server pads and encrypts each value in the database:

$$\tilde{\mathbf{v}} = \left(\mathrm{Enc}_K\left(v_1 || 0^s\right), \ldots, \mathrm{Enc}_K\left(v_l || 0^s\right)\right)$$

2. Server interpolates and sends a polynomial P s.t. for all $i \in [l]$

$$P(k_i) = \tilde{v}_i.$$

Building a ROOM (2)

Idea: We don't need to have an explicit representation of $\perp$!

1. Server pads and encrypts each value in the database:

$$\tilde{\mathbf{v}} = \left(\mathrm{Enc}_K\left(v_1||0^s\right), \ldots, \mathrm{Enc}_K\left(v_l||0^s\right)\right)$$

2. Server interpolates and sends a polynomial P s.t. for all $i \in [l]$

$$P(k_i) = \tilde{v}_i.$$

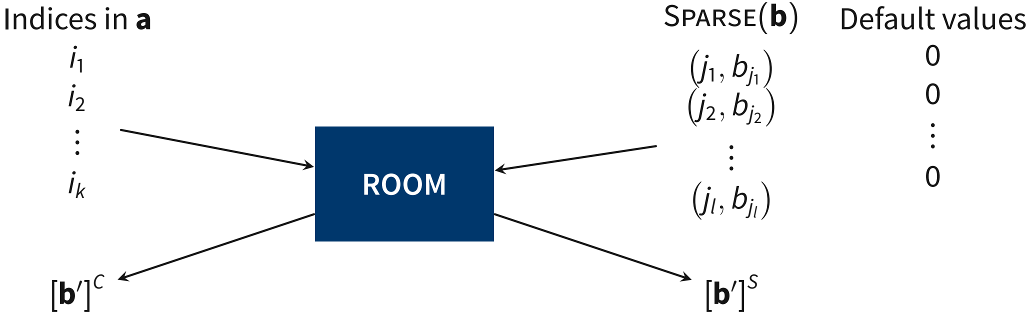
3. For each query key q_i, perform an MPC with inputs $\tilde{x}_{q_i} = P(q_i), K, \bar{v}_i$, that
   a) Decrypts $x_{q_i} = \mathrm{Dec}_K(\tilde{x_{q_i}})$,
   b) Secret-shares v if $x_{q_i} = (v||0^s)$, otherwise $\bar{v}_i$.

Sparse Inner Product from ROOM

Let $\textsc{Sparse}(\mathbf{a}) := \big((i, a_i)\big)_{a_i \neq 0}$, $\textsc{Sparse}(\mathbf{b}) := \big((j, b_j)\big)_{b_j \neq 0}$.

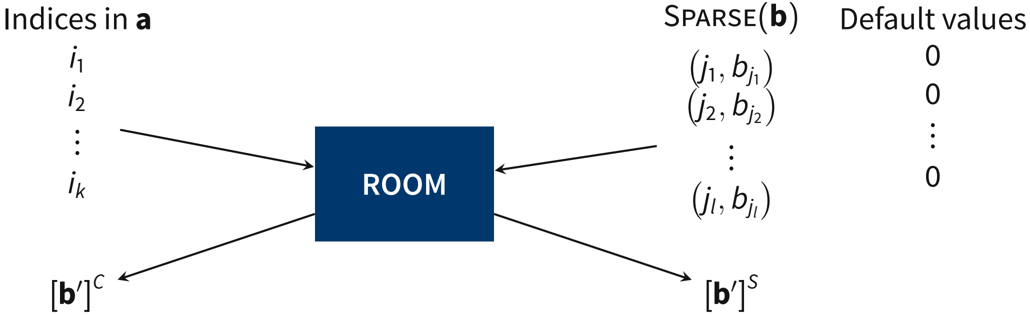
Sparse Inner Product from ROOM

Let $\textsc{Sparse}(\mathbf{a}) := \big((i, a_i)\big)_{a_i \neq 0}$, $\textsc{Sparse}(\mathbf{b}) := \big((j, b_j)\big)_{b_j \neq 0}$.



Sparse Inner Product from ROOM

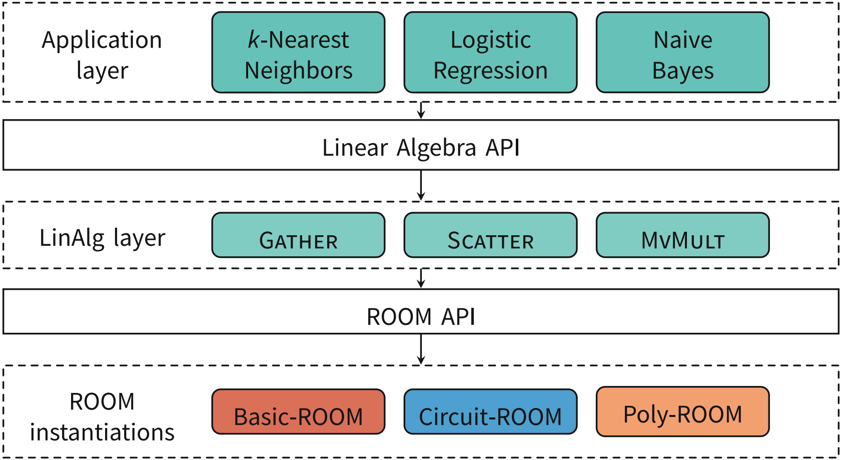
Let $\textsc{Sparse}(\mathbf{a}) := \left((i, a_i)\right)_{a_i \neq 0}$, $\textsc{Sparse}(\mathbf{b}) := \left((j, b_j)\right)_{b_j \neq 0}$.



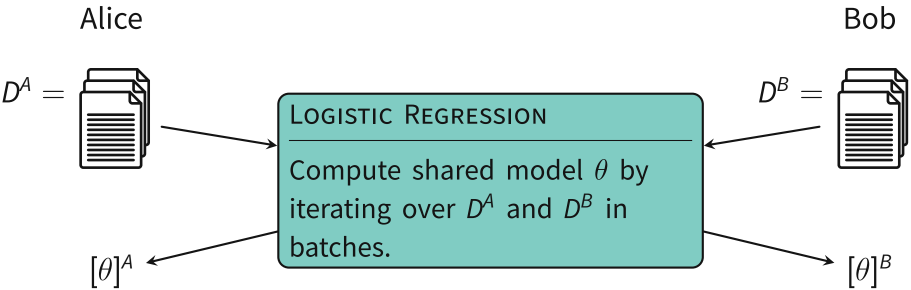
Now, $\mathbf{b}' = (b_i)_{a_i \neq 0}$. Let $\mathbf{a}' = (a_i)_{a_i \neq 0}$. Then $\mathbf{ab} = \mathbf{a}'\mathbf{b}'$.

Applications

ROOM Framework



Logistic Regression

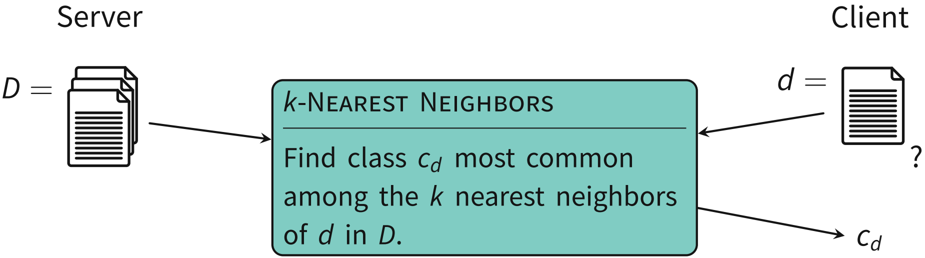


P. Mohassel and Y. Zhang. ‘SecureML: A System for Scalable Privacy-Preserving Machine Learning’. In: *IEEE Symposium on Security and Privacy*. 2017, pp. 19–38

Logistic Regression: Time and Communication

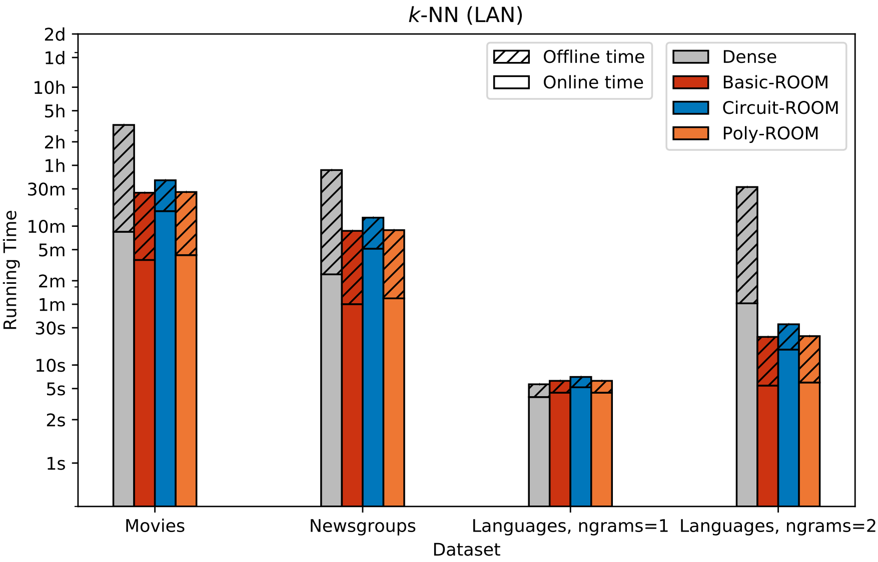
Dataset	*Total Time*		*Communication*	
	SecureML	*Ours*	*SecureML*	*Ours*
Movies	6h29m28.37s	**2h43m46.09s**	4.8 TiB	**187.42 GiB**
Newsgroups	1h42m38.14s	**42m37.68s**	1.26 TiB	**47.63 GiB**
Languages, ngrams=1	**5.9s**	29.89s	790.9 MiB	**500.61 MiB**
Languages, ngrams=2	1h3m7.12s	**6m17.51s**	797.85 GiB	**3.69 GiB**

k-Nearest Neighbors



P. Schoppmann, A. Gascón and B. Balle. *Private Nearest Neighbors Classification in Federated Databases*. Cryptology ePrint Archive, Report 2018/289. `https://eprint.iacr.org/2018/289`. 2018

k-Nearest Neighbors: Time



Conclusion

- To scale secure machine learning, we have to exploit characteristics in the *setting* and the *data*.

- We show that for *data sparsity*, using a dedicated data structure helps speed up multiple applications.

Conclusion

- To scale secure machine learning, we have to exploit characteristics in the *setting* and the *data*.

- We show that for *data sparsity*, using a dedicated data structure helps speed up multiple applications.

- Future directions:
  - Improve access times: LowMC, Cuckoo Hashing.
  - Adapt other primitives, e.g. Labeled PSI, flavors of PIR.

References I

P. Mohassel and Y. Zhang. 'SecureML: A System for Scalable Privacy-Preserving Machine Learning'. In: *IEEE Symposium on Security and Privacy*. 2017, pp. 19–38.

B. Chor, O. Goldreich, E. Kushilevitz and M. Sudan. 'Private information retrieval'. In: *Proceedings of IEEE 36th Annual Foundations of Computer Science*. IEEE. 1995, pp. 41–50.

S. Angel, H. Chen, K. Laine and S. T. V. Setty. 'PIR with Compressed Queries and Amortized Query Processing'. In: *IEEE Symposium on Security and Privacy*. IEEE, 2018, pp. 962–979.

B. Chor, N. Gilboa and M. Naor. 'Private Information Retrieval by Keywords'. In: *IACR Cryptology ePrint Archive* 1998 (1998), p. 3.

Y. Gertner, Y. Ishai, E. Kushilevitz and T. Malkin. 'Protecting Data Privacy in Private Information Retrieval Schemes'. In: *J. Comput. Syst. Sci.* 60.3 (2000), pp. 592–629.

References II

H. Chen, Z. Huang, K. Laine and P. Rindal. 'Labeled PSI from Fully Homomorphic Encryption with Malicious Security'. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2018, pp. 1223–1237.

P. Schoppmann, A. Gascón and B. Balle. *Private Nearest Neighbors Classification in Federated Databases*. Cryptology ePrint Archive, Report 2018/289. https://eprint.iacr.org/2018/289. 2018.

P. Schoppmann, A. Gascón, M. Raykova and B. Pinkas. *Make Some ROOM for the Zeros: Data Sparsity in Secure Distributed Machine Learning*. Cryptology ePrint Archive, Report 2019/281. https://eprint.iacr.org/2019/281. 2019.

Source Code: https://github.com/schoppmp/room-framework.