A Critical Review On Make Some ROOM for the Zeros: Data Sparsity in Secure Distributed Machine Learning

Shubhika GARG

Abstract— **The research paper proposes sparse data structures and secure computation protocols for efficiently handling data sparsity in various tasks, including linear algebra operations and secure analysis of non-parametric and parametric models. The protocols, utilizing a Read-Only Oblivious Map primitive (ROOM) and secret sharing, demonstrate significant efficiency improvements in experiments across applications. The system follows a modular architecture inspired by scientific computing and machine learning frameworks, resembling the Sparse BLAS standard.**

I. INTRODUCTION

Machine Learning techniques, driven by advances in statistical modeling, mathematics, computer science, software engineering, and hardware design, rely on standardized interfaces like BLAS (Basic Linear Algebra Subprograms). BLAS specifications provide fundamental routines for linear algebra, ensuring portability across scientific computing frameworks. Several libraries implement BLAS, offering users the flexibility to develop applications without being tied to specific implementations. These libraries, along with architecture-optimized variants, serve as the backend for popular machine learning frameworks like TensorFlow and PyTorch.

The paper presents a framework for privacy-preserving machine learning, offering confidential alternatives for basic linear algebra operations. The framework allows multiple parties to compute on shared inputs without revealing private data, ensuring privacy compliance. Hospitals jointly analyzing patient data and government agencies discovering discrepancies in databases are the two example scenarios that have these characteristics.

The framework's foundational building blocks are cryptographic two-party computation protocols, ensuring formal privacy guarantees. The research optimizes for scenarios where the sparsity level is not sensitive, such as in datasets with publicly known sparsity metrics. Similar to Sparse BLAS, exploiting sparsity allows substantial speedups in our secure protocols, enhancing the efficiency of higher-level applications.

Sparse linear algebra on clear data typically uses specific data structures like coordinate-wise or Compressed Sparse Row (CSR) representations for sparse matrices. In the context of Multi-Party Computation (MPC), the paper introduces a similar abstract representation called Read-Only Oblivious Map (ROOM) that allows for various back-end implementations of ROOM functionality, leading to different trade-offs and improvements in communication and computation. MPC experts can create secure computation instances for ROOM, seamlessly utilized by non-experts for higher-level tools, similar to data scientists using statistical modeling in machine learning frameworks.

The paper introduces a modular secure computation framework for sparse data, which they utilize to develop three secure two-party applications. The proposed framework, resembling scientific computing architectures, defines basic functionality, implements secure instantiations in MPC, builds linear algebra primitives, and uses them to create higher-level machine learning applications. The contributions include a Read-Only Oblivious Map (ROOM) for secure manipulation of sparse datasets, three ROOM protocol instantiations with varied trade-offs that include Basic-ROOM that prioritizes higher communication with minimal secure computation, Circuit-ROOM, which balances reduced communication with additional secure computation through sort-merge networks, and PolyROOM, leveraging fast polynomial interpolation and evaluation to reduce secure computation costs while maintaining low communication and implementations for sparse matrix-vector multiplication, gather, and scatter operations. The framework's performance is evaluated on k-nearest neighbors, stochastic gradient descent for logistic regression, and naive Bayes classification, showcasing significant improvements over existing protocols in terms of runtime and communication efficiency.

II. PROPOSED SETUP

The paper focuses on exploiting sparsity in secure distributed machine learning (ML). Sparsity, prevalent in real-world data, reduces storage overhead and forms the basis for efficient algorithms. Various data structures like Compressed Sparse Row (CSR) [1] are used for sparse data representation. The research aims to extend the benefits of sparsity to the secure distributed ML setting. The main challenge is concealing the locations of non-zero values in the data while revealing the sparsity metric. The privacy requirements involve revealing the sparsity metric while hiding the specific sparsity locations.

A. *Privacy Requirements and Threat Model*

The threat model assumes a two-party computation setting with semi-honest parties [2], adhering to the simulation-based paradigm for secure computation. The computations involve matrices and vectors in a finite domain. The sparsity metric of the input set is considered public, varying between the total number of zeros or zero rows/columns. The sparsity metric is disclosed for the batch when applied to dataset rows (batches).

B. *Multi-Party Computation (MPC) Preprocessing Model*

The preprocessing model involves a division into online and offline stages. The offline stage occurs before input availability, while the online stage depends on concrete input values. The research does not prioritize optimization for the online-offline setting but emphasizes minimizing the overall protocol cost.

III. IMPLEMENTATION

The method outlined in the paper centres around making sparse matrix-vector multiplication more efficient. The central insight is that when dealing with matrices containing numerous zero elements, calculations are unnecessary, especially when one of the factors is zero. The paper emphasizes the need to keep the locations of these zero elements private, particularly in situations where privacy is a significant concern.

A notable aspect discussed is the recognition that, in many instances, there is a publicly known upper limit on the number of non-zero elements. This understanding of sparsity provides valuable information for optimization purposes.

The proposed approach involves encoding the sparse vector using a Read-Only Oblivious Map (ROOM) data structure. This strategic choice enhances privacy by allowing access to vector information without revealing specific details. Furthermore, the paper recommends an efficient implementation of matrix-vector multiplication achieved through batched oblivious map access. Combining this methodology with ROOMs underscores a concerted effort to preserve privacy while optimizing computations, particularly for sparse matrices.

A. *Read-Only Oblivious Maps (ROOM)*

The research paper introduces Read-Only Oblivious Maps (ROOMs), a 2-party functionality involving a server and a client. These ROOMs operate on fixed finite sets of keys (K) and values (V), where the server holds key-value pairs, and the client issues queries. The output of a ROOM is an array of results based on the queries, with values retrieved from the server's list or default values assigned by the server. The shared output is secret-shared between the parties, and a designated output variant allows for a single party to obtain the result.

B. *Building a ROOM*

The ROOM protocol is implemented in three instantiations: Basic-ROOM, Circuit-ROOM, and Poly-ROOM. Basic-ROOM, a naive approach, extends the key domain and encrypts all potential queries, with linear overhead in the domain. Circuit-ROOM employs secure computation using a sort-merge join technique, significantly reducing the comparison problem to one-to-one. Poly-ROOM introduces a polynomial construction that reduces the dependency on the domain, allowing for efficient evaluation and decryption, with runtime similar to Basic-ROOM but without the linear dependency on the domain during initialization. Each instantiation is analyzed in terms of security and computational cost, showcasing various trade-offs and highlighting their respective strengths.

i) Naive Approach

In the naive approach, database sparsity is ignored.

1) The server augments the database with dummy elements to cover the entire key domain:

$$x = (\perp, \ldots, \perp, v_1, \perp, \ldots, \perp, v_2, \ldots)$$

where $\perp$ represents dummy elements, and $v_1, v_2, \ldots$ are actual values associated with specific keys.

2) The server encrypts each element of x individually and transmits the resulting encrypted vector to the client:

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

3) For each query q_i, the client selects $\tilde{x}_{q_i}$, and the parties engage in a secure multi-party computation (MPC) with inputs $\tilde{x}_{q_i}$, K, and $\overline{v}_i$. The MPC entails:
   a) Decrypting $x_{q_i} = \text{Dec}_K(\tilde{x}_{q_i})$,
   b) Secret-sharing x_{q_i} if $x_{q_i} \neq \perp$, otherwise using $\overline{v}_i$.

This process involves linear communication in the key domain.

ii) CIRCUIT-ROOM Protocol

- **Inputs:**
  - Server: d, β
  - Client: q
- **Protocol Steps:**
  1) Compute vectors b and c.
     - Vector b stores selected values based on key matches in adjacent entries.
     - Vector c denotes whether the i-th pair involves a key from q.
       * $c_i > 0$, it corresponds to the index of that key in q otherwise $c_i = 0$.
       * If $c_i > 0$, return the corresponding answer from b.
       * If a match is found, the answer is the value from d; otherwise, it's the value from β.
  2) Obliviously shuffle vectors b and c to prevent information leakage.
  3) Output shares of entries in b corresponding to the client's keys, along with entries in c, are shared between parties.
  4) Parties can map their output shares back to the order of the inputs.

ii) Polynomial-Based ROOM Construction

No need for an explicit representation of $\perp$.

1) The server employs padding and encryption for each value in the database where 0^s denotes the zero-padding:

$$\widetilde{v} = (\text{Enc}_K(v_1 \| \| 0^s), \ldots, \text{Enc}_K(v_l \| \| 0^s))$$

2) The server conducts interpolation and transmits a polynomial P such that:
   for all $i \in [l]$, $P(k_i) = \widetilde{v}_i$.
3) For each query key q_i, engage in a secure multi-party computation (MPC) with inputs $\widetilde{x}_{q_i} = P(q_i)$, K, and $\overline{v}_i$, which involves:
   a) Decrypting $x_{q_i} = \text{Dec}_K(\widetilde{x}_{q_i})$,
   b) Secret-sharing v if $x_{q_i} = (v \| \| 0^s)$, otherwise using $\overline{v}_i$.

By adopting this polynomial-based construction, the need for explicit representation of $\perp$ is eliminated, and the protocol achieves efficient ROOM functionality without linear dependency on the key domain during initialization.

C. *ROOM For Secure Sparse Linear Algebra*

The ScatterInit protocol leverages ROOM (Read-Only Oblivious Maps) and OT (Oblivious Transfer) Extension to achieve secure and efficient computations involving vector shares. The goal of the ScatterInit protocol is to securely compute and obtain a modified vector share $[\![v']\!]$ from the original vector v while preserving privacy and integrity.

1) For each index $i \in [n]$, party P_2 generates a random value s_i.
2) P_2 acts as the sender, and P_1, the receiver, engages in a $(n-l)$-out-of-n OT protocol. P_1 obtains:

$$u = \{(i, s_i) \mid i \notin \{i_1, \ldots, i_l\}\}$$

3) The parties run ROOM where P_2 becomes the server, and P_1 is the client with inputs $q = (i_1, \ldots, i_l)$, $d = \{(i, s_i)\}_{i \in n}$, and $\beta = \perp^n$. The parties obtain shares of the vector:
   $\bar{u} = (s_i \mid i \in \{i_1, \ldots, i_l\})$.
4) The parties engage in a two-party computation with inputs $[\![v]\!]_{P_1}$, $[\![v]\!]_{P_2}$, $[\![\bar{u}]\!]_{P_1}$, and $[\![\bar{u}]\!]_{P_2}$. The vector v and the masked values $\bar{u}$ are reconstructed, revealing the difference $\bar{s} = v - \bar{u}$ to P_1.
5) P_2 sets $[\![v']\!]_{P_2} = (s_i)_{i \in [n]}$, and P_1 sets $[\![v']\!]_{P_1} = s$ with values s_i determined as follows:
   a) For $i = i_j \in \{i_1, \ldots, i_l\}$, $s_i = \bar{s}_j$.
   b) For other indices, $s_i = -s_i$, where $(i, s_i) \in u$.

IV. APPLICATIONS

The paper highlights its framework applications, focusing on naive Bayes, k-nearest neighbours (k-

NN), and two-party logistic regression with stochastic gradient descent (SGD).

A. *k-Nearest Neighbours (k-NN)*

There's a database D with labelled documents distributed among servers. Each document $d \in D$ is represented using TF-IDF (Term Frequency-Inverse Document Frequency) encoding. A client wants to classify a document d against D. The k-NN algorithm [3] computes similarities between d and each document in D, then assigns a class to d based on a majority vote among the k most similar documents.

The k-NN protocols Basic-ROOM and Poly-ROOM significantly outperform the baseline, with up to 82x total time improvement, 2–5x faster online time and a remarkable 40x speedup in reduction time.

B. *Logistic Regression*

The paper introduces the application of logistic regression training in a two-party setting, where parties P1 and P2 hold horizontally partitioned databases. Each party possesses a sparse dataset with features and binary target labels. The goal is to collaboratively build a shared model (θ) that accurately predicts target values for unlabeled records while keeping the local training datasets private. The logistic regression model is trained using minibatch stochastic gradient descent (SGD) [4], minimizing the empirical loss iteratively. The protocol involves secure two-party gradient descent on sparse distributed training data, utilizing secure matrix multiplication and an approximation of the logistic function. The security of the protocol is maintained by keeping the model parameters secret shared throughout the process. The paper provides details on the instantiation of the protocol and emphasizes the sparsity considerations in real-world datasets.

The logistic regression training evaluation compares a dense matrix multiplication protocol with the proposed sparse matrix multiplication protocols. While the dense approach achieves rapid online computation, it incurs significant offline computation and communication costs.

In contrast, the proposed solution significantly reduces total runtime and communication across diverse datasets, showing enhanced efficiency. The implementation also shows scalability, benefitting from increased batch sizes in synthetic dataset experiments. The sparse approach shows potential for additional optimizations without sacrificing accuracy.

V. CONCLUSIONS

The research highlights the essential role of exploiting characteristics in secure machine learning, particularly emphasizing the effectiveness of a dedicated data structure for handling data sparsity and accelerating various applications. To scale secure machine learning, we must exploit characteristics in the setting and data.

The paper also underlines the broad relevance of privacy-preserving techniques, showcasing significant advancements in efficiency and scalability. The modular framework performs better than existing techniques and lays the foundation for various machine learning algorithms, with the potential for ongoing improvements.

REFERENCES

[1] https://pnxguide.medium.com/compressed-sparse-row-motivation-and-explanation-cd92c71b7cfa

[2] https://medium.com/boltlabs/intro-to-secure-multi-party-computation-15875dc5a24a

[3] https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

[4] https://sweta-nit.medium.com/batch-mini-batch-and-stochastic-gradient-descent-e9bc4cacd461