Privacy-Preserving Linear Regression

Ariadna Fernandez Sarah Athar

Abstract

Linear Regression is a Machine Learning (ML) algorithm that is widely used in practice to generate predictive models that are essential in almost every industry. Training such models requires massive amounts of data in order to achieve accurate results. However, extensive data collection that often combines data from different sources creates data privacy concerns.
Through our research, we implement the online phase of an existing privacy-preserving linear regression protocol [5]. Our implementation consolidates the work into a single C++ program, highlighting the previously developed techniques for secure arithmetic operations, all while being able to train using the MNIST database [1].

1 Introduction

Machine Learning techniques, like linear regression, are used to produce predictive models which rely on large amounts of sometimes sensitive data. This often hinders companies and organizations from contributing their data for model training, since they do not want to share sensitive information with other parties. Privacy-preserving machine learning provides a solution to this security issue, by enabling companies to perform the same algorithm without knowing the underlying content of other parties' data. In this work, we implement the online phase of the secure linear regression protocol detailed in previous work [5]. This protocol follows the two-server model, in which parties (clients) distribute their secret-shared data among two non-colluding servers who train the model on the client's joint data by means of multiparty computation (MPC). Multiparty computation allows different parties to jointly train a model on their combined data inputs, while keeping their respective data private or hidden from other parties and revealing only the resulting predictive model.

Summary of Contributions.

- We implement the Online Phase of a Secure Machine Learning Protocol to Compute a Two-Party Linear Regression Model using
- We consolidated the work into a single C++ program, to emphasize the previously developed techniques for secure arithmetic operations.
- We trained a secure linear regression model using the MNIST database.

1.1 Timeline

During the first week, we read literature on two possible research topics:

1. Improving the performance of private set union (PSU) using new techniques from private set intersection (PSI).
2. Supporting Data Deletion in Privacy-Preserving Machine Learning.

We initially decided to continue on the path of data deletion in machine learning, which we found to be really interesting and of high relevance due to increasing concerns for users' data privacy rights. In the second week, we started to familiarize ourselves with cryptography primitives, such as Oblivious Transfer, Garbled Circuits and Secret Sharing. We also got acquainted with Multi-Party Computation, which allows parties to jointly compute a function over their combined data inputs, while keeping their respective data private or hidden from other parties. During the third week, we dove deeper into finding potential implementations of Privacy-Preserving Data Deletion, however after giving some thought to the project we decided to shift gears in order to achieve a more practical approach. Hence, we started our fourth week acquainting ourselves with Privacy-Preserving Linear Regression and how it can be implemented. After consulting with Dr. Peihan Miao, we started coding the Online Phase of the Protocol in which we utilized the Eigen Library [2] for its matrix operations, and worked on it consistently for two whole weeks. During the last two weeks, we spent a valuable amount of time, polishing and testing our protocol with our own tests, as well as training on the MNIST Database.

1.2 Related Work

Keith Bonawitz et. al. [6] outline an approach to advancing privacy preserving machine learning by leveraging secure multiparty computation (MPC) to compute sums of model parameter updates from individual users' devices in a secure manner. They presented a practical protocol for securely aggregating data while ensuring that clients' inputs are only learned by the server in aggregate.
Our implementation of the secure linear regression machine learning model follows the protocol detailed in the work *Secure ML: A System for Scalable Privacy-Preserving Machine Learning* by Payman Mohassel and Yupeng Zhang [5]. Through the two-server model, clients first distribute their data inputs among two non-colluding servers via secret-sharing.
Secret-sharing is a cryptographic primitive in which a piece of data, or 'secret', is kept private by having distinct owners hold a share of the secret. Not one party can guess or generate the original secret, which can only be reconstructed by combining the shares together. A value x that has been secret shared by two parties holding x_0 and x_1 is denoted as $\langle x \rangle$.
After data distribution happens in the setup phase, clients outsource the computation to the two servers and are not involved in any further computation. The protocol consists of an offline and online phase. The former is data-independent and consists of generating shared multiplication triplets, which are shared matrices consisting of random numbers in order to mask the data during computation. However, we focus on implementing the online phase that trains the linear regression model using arithmetic secret sharing and multiplication triplets techniques.

2 Privacy Preserving Linear Regression

Linear regression. It is a statistical approach for producing a linear function that models the relationship between independent and dependent variables and it is used to predict values based on some input. The model is trained on data value pairs consisting of a vector of d features $\boldsymbol{x_i}$,the independent variables, and an output label $\boldsymbol{y}$, the dependent variable. Training results in a coefficient vector $\boldsymbol{w}$ used in the model linear function g such that $g(\boldsymbol{x_i}) = \sum_{j=1}^{d} x_{ij}\, w_j = \boldsymbol{x_i} \cdot \boldsymbol{w}$. The goal of linear regression is to find the best coefficient vector $\boldsymbol{w}$ so that $g(\boldsymbol{x_i})$ is approximately equal to the label $\boldsymbol{y}$. This is done by optimizing a commonly used *cost function* that calculates the mean difference between the predicted value $\boldsymbol{y_i}^* = \boldsymbol{x_i} \cdot \boldsymbol{w}$ and the actual value $\boldsymbol{y_i}$:
$\mathrm{C}(\boldsymbol{w}) = \frac{1}{n}\sum \mathrm{C}_\mathrm{i}(\boldsymbol{w})$, where $\mathrm{C}_\mathrm{i}(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{x_i} \cdot \boldsymbol{w} - \boldsymbol{y_i})^2$.
An effective approximation method called Stochastic Gradient Descent (SGD) is used optimize the cost function by minimizing the difference between $\boldsymbol{y_i}^*$ and $\boldsymbol{y}$. In this method, $\boldsymbol{w}$ is initialized as a vector of random values, and is consecutively updated by the formula $w_j := w_j - \alpha(x_i \cdot w - y_i)x_{ij}$.

For more efficient computation, training data samples are stored in matrix $\mathbf{X}$ of size N x d storing the features and matrix $\mathbf{Y}$ of size N x 1 holding the corresponding output label. In each of the t iterations of the protocol, |B| number of data samples $(\boldsymbol{x_i}, \boldsymbol{y_i})$ are randomly selected and $\boldsymbol{w}$ is updated by this function now in vectorized form $\boldsymbol{w} := \boldsymbol{w} - (\frac{1}{|B|})\ \alpha\ \boldsymbol{X}^{\mathrm{T}}_{\mathrm{B}} \times(\mathbf{X_B} \times \boldsymbol{w} - \mathbf{Y}_{\mathrm{B}})$, where α is the learning rate defining the magnitude to move toward the minimum in each iteration, $\mathbf{X_B}$ and $\mathbf{Y}_{\mathrm{B}}$ are B x d and B x 1 submatrices of $\mathbf{X}$ and $\mathbf{Y}$ selected using indices in B, representing |B| samples of data and labels in an iteration.

Truncation. One of the major bottlenecks in privacy-preserving ML algorithms come from inefficient computation on shared decimal numbers, especially in multiplication operations since the range of the result grows exponentially to the number of multiplications and becomes almost impossible to support in practice. To resolve this issue we follow a simple solution detailed in [5], that supports decimal arithmetics in an integer field of l bits. When multiplying two decimal numbers x and y with l_D bits assigned to its fractional part. The decimal numbers are first scaled by a factor 2^{lD}, becoming $x' = 2^{lD} x$ and $y' = 2^{lD} y$. Multiplying then yields $z = x'y'$ with at most $2l_\mathrm{D}$ bits representing the fractional part.Then resulting product z is then truncated and the last l_D bits are removed so that only l_D represents the fractional part of z, without any significant change in accuracy results. A truncated value z is denoted by $\lfloor z \rfloor$.

2.1 Online Phase

This phase of the protocol assumes that the clients have already secret-shared their data to the two non-concluding servers and that multiplication triplets vital for computation have formerly been generated in the offline phase. However, our implementation concentrates on the online phase, and we achieve these two setup steps by simpler non cryptographic techniques.
Once each server holds their shares of the training data $\langle\mathbf{X}\rangle_\mathrm{i}$ and $\langle\mathbf{Y}\rangle_\mathrm{i}$, and random shared multiplication triplets $\langle\mathbf{U}\rangle_\mathrm{i}$, $\langle\mathbf{V}\rangle_\mathrm{i}$, $\langle\mathbf{Z}\rangle_\mathrm{i}$, $\langle\mathbf{V'}\rangle_\mathrm{i}$, $\langle\mathbf{Z'}\rangle_\mathrm{i}$ have been generated for $i \in \{0, 1\}$.

Then the online phase of the linear regression protocol is as follows:
1: Each server computes $\langle \mathbf{E} \rangle_i = \langle \mathbf{X} \rangle_i - \langle \mathbf{U} \rangle_i$ for $i \in \{0, 1\}$ to mask their data shares. Then parties exchange shares of $\langle \mathbf{E} \rangle_i$ and reconstruct, to recover **E**.
2: **for** $j = 1, \ldots, t$ each server does the following:
3: A set of random indices corresponding to data samples is created, and each server selects the mini-batch $\langle \mathbf{X}_{Bj} \rangle$, $\langle \mathbf{Y}_{Bj} \rangle$, and $\mathbf{E}_{Bj,}$.
4: Compute $\langle \mathbf{F}_j \rangle_i = \langle \mathbf{w} \rangle_i$ - $\mathbf{V}[j]$ using the *jth* column of matrix **V** to mask $\mathbf{w}_{i.}$ Then parties exchange shares of $\langle \mathbf{F} \rangle_i$ and reconstruct, to recover **F**.
5: Calculate the predicted output $\langle \mathbf{Y}^*_{Bj} \rangle_i = -i \cdot \mathbf{E}_{Bj} \cdot \mathbf{F}_i \cdot \mathbf{E}_{Bj} \cdot \langle \mathbf{w} \rangle_i + \langle \mathbf{Z}_j \rangle_i$ using the *jth* column of matrix **Z** for $i \in \{0, 1\}$.
6: Compute the difference between the predicted output and real output $\langle \mathbf{D}_{Bj} \rangle_i = \langle \mathbf{Y}^*_{Bj} \rangle_i - \langle \mathbf{Y}_{Bj} \rangle_i$ for $i \in \{0, 1\}$.
7: Compute $\langle \mathbf{F'} \rangle_i = \langle \mathbf{D}_{Bj} \rangle_i - \langle \mathbf{V'}_j \rangle_i$ for $i \in \{0, 1\}$ to mask the calculated difference. Then parties exchange shares of $\langle \mathbf{F'} \rangle_i$ and reconstruct, to recover **F**.
8: Compute $\langle \Delta \rangle_i = -i \cdot \mathbf{E}^T_{Bj} \times \mathbf{F}'_j + \langle \mathbf{X}^T_{Bj} \rangle_i \times \mathbf{F}'_j + \mathbf{E}^T_{Bj} \times \langle \mathbf{D}_{Bj} \rangle_i + \langle \mathbf{Z'}_j \rangle_i$ for $i \in \{0,1\}$ to calculate the change of $\mathbf{w}_i$.
9: Truncate $\langle \Delta \rangle_i$ to get $\lfloor \langle \Delta \rangle_i \rfloor$ for $i \in \{0,1\}$.
10: Compute $\langle \mathbf{w} \rangle_i := \langle \mathbf{w} \rangle_i - \frac{\alpha}{|B|} \lfloor \langle \Delta \rangle_i \rfloor$ for $i \in \{0,1\}$.
11: Then parties exchange shares of $\langle \mathbf{w} \rangle_i$ and reconstruct, to obtain **w**.

3 Experiments

The training of our implementation of the privacy-preserving linear regression system based on the proposed protocol [5] is detailed in this section. Deploying and testing this protocol to achieve optimal accuracy results is an important direction for future work.

Datasets. The MNIST database [1] (Modified National Institute of Standards and Technology database) is a large collection of handwritten digits (0 through 9) popular for training various image processing systems. It is a common practice to train an algorithm on the MNIST dataset to test a new protocol, hence, we trained our secure linear regression model using the database, to ensure its performance.
MNIST is split into two datasets: the training set containing 60,000 training images and its corresponding labels, and the test set containing 10,000 testing images and its corresponding labels. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset. Each of the training and testing samples have 784 features with a value ranging from 0 to 255 representing a pixel of the 28 x 28 image. All of its images are the same size, and within them, the digits are centered and size normalized.

Experimental Results. To test the accuracy of our implementation of the protocol we multiplied the resulting coefficient vector **w** times a testing data sample **X** and rounded the result to the nearest integer. We then compared this with the corresponding testing label and if equivalent it was added to a total count of correct predictions that was then divided by the number of total testing samples. Multiple training experiments on the MNIST database did not yield accepted accuracy results, suggesting the need for improvements in performance.

4 Discussion

At present, the main options for privacy preserving machine learning consist of linear regression models, logistic regression models, and neural network training. While our work here focuses solely on the implementation of the linear regression protocol, our future plans involve testing the model with the MNIST dataset, in order to achieve optimal accuracy. As of now our implementation yields unexpected results, which we have deduced to originate from a system compatibility issue, hence investigating this in detail is crucial for the practical use of our implementation.

5 Conclusion

In this work, we consider a solution to the problem of Data Privacy when training Machine Learning Models, with a focus of Privacy-Preserving Linear Regression. We implement the online phase of an existing secure linear regression protocol. We consolidated the work into a single C++ program, to emphasize the previously developed techniques for secure arithmetic operations. It is our hope that these insights will allow different parties to jointly train a model on their combined data inputs, while keeping their respective data private or hidden from other parties and revealing only the resulting predictive model.
We have just scratched the surface of implementing the Privacy-Preserving Linear Regression protocol. Implementing the offline phase of the protocol, consisting of multiplication triplet generation as well as understanding learning rate adjustment procedures, is an important direction for future work. Code for reproducing our protocol can be found at https://github.com/sarahathar/SecureML-Privacy-Preserving-Linear-Regression.

Acknowledgements

A huge thank you to our research advisor Dr. Peihan Miao for sharing her knowledge and expertise with us, graduate mentor Shweta Srinivasan for the helpful advice she provided us with and to Break Through Tech for organizing this inspiring learning experience.

References

[1] MNIST database. http://yann.lecun.com/exdb/mnist/

[2] Eigen library. http://eigen.tuxfamily.org/

[3] Source code for the paper.

[4] Shreya's Implementation on github. https://github.com/shreya-28/Secure-ML

[5] Payman Mohassel and Yupeng Zhang. Secure ML: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy*.

[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *2017 ACM SIGSAC Conference on Computer and Communications Security.*