Experimental Secure Multi-Party Computation on Real Data using SPDZ

Abstract

Secure Multi-Party Computation (MPC) is an area of theoretical cryptography research that has the power to revolutionize computation between shared parties holding private data. However theoretical protocols are often not scalable and efficient given large quantities of real world data. This project investigates an MPC protocol called SPDZ secure against malicious adversaries in comparison to two other semi-honest MPC protocols Obliv-C and SecureML in machine learning and regression applications. We demonstrate its potential to be used in real world applications with experiments to show that it outperforms the other two methods while maintaining a more rigorous level of security.

1 Introduction

Regression analysis and other machine learning techniques aim to build a model that fits a set of predictors to the dependent variable. Models built from a large set of collected data can be used to predict future behavior. With the advent of big data, it is becoming increasingly common for entities to store vast amounts of user data, run regression analyses and build models of this data to gain insights. A desired use case could be, for example, the Center for Disease Control is interested in identifying disease outbreak and many individual hospitals have their own patient data which could benefit this study. The problem is that openly sharing this data for prediction or model-building purposes would be against modern day privacy laws as it would leak private individual data. This application of secure MPC is one that would be extremely valuable in today's world.

1.1 Multi-Party Computation

Secure MPC addresses this problem by providing a mechanism through which different parties supply their data for joint computation without revealing individual values from each database or even intermediate values. This need for MPC motivates our study to bridge theory and application through experiments on real-world data.

In general, MPC protocols predominantly come in two forms: security against semi-honest and malicious adversaries. In semi-honest security, a more naive model, it is assumed that the party will follow the protocol as specified but will merely gather information intermediate values. Typically semi-honest security is more efficient on computation as it has fewer requirements. In malicious security, dishonest parties may attempt to deviate from the specified protocol, and the protocol must be able to detect such cheating and abort the computation - thus any successful output is guaranteed to have privacy preserved. This notion can be rigorously proved.

For this study, we chose the SPDZ framework, which ensures against malicious security, and compare against two semi-honest methods Obliv-C and SecureML. SPDZ performs the checks necessary to detect dishonest parties through using random data in the hopes of making computation more efficient. Obliv-C is based on Yao's garbled circuits and introduces optimizations on inner products to speed up regression calculation. SecureML's focus is with developing MPC-friendly alternatives to non-linear activation functions which are often intensive to compute.

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

Since secure computation slower and could be limited in accuracy when compared to computation in plaintext, there are a number of modifications to plaintext algorithms that can be made to obtain a reasonable degree of accuracy in more efficient time.

2 ML Functionalities

Our goal was to conduct a thorough investigation of the SPDZ framework in the applications of regression and machine learning algorithms in a similar fashion to that done in [1,2], evaluating in terms of runtime and accuracy. The main algorithms that we considered were, LDLT Decomposition, Cholesky, Conjugate Gradient Descent (CGD), Stochastic Gradient Descent (SGD), Logistic Regression. The algorithms were implemented in the SPDZ framework as a proof-of-concept as well as in python as a plaintext verification of the accuracy of the secure version.

2.1 Direct vs. Iterative Decomposition

LDLT and Cholesky are both variants of direct decomposition methods which decompose a Hermitian, positive-definite matrix into a lower triangular matrix and its conjugate transpose. The algorithms are cubic in complexity with asymptotic run time of $O(d^3)$. This can become difficult to work with when looking at real world data that has large number of features and entries. The difference between LDLT and Cholesky is that Cholesky requires a square root. We looked at two different methods to approximate the square root function in SPDZ. The Babylonian method involves iteratively dividing and averaging an initial guess so on the next iteration can be represented as $x_{n+1} = \frac{1}{2}(x_n + \frac{S}{x_n})$. The Newton Method is similar but finds the square root using multiplications and not divisions: $\sqrt{S} = S \cdot (1/\sqrt{S})$.

In terms of an iterative approach to regression, [1] proposed a modification for the normalization in the iterative version of CGD that would be more stable with fixed-point numbers.

We applied the same algorithm to SPDZ to see if the same modification would work. The goal of comparing iterative versus a direct method would be to justify the potential accuracy trade off for a much smaller run time.

2.2 Gradient Descent

Stochastic gradient descent is also an iterative method to minimize a differentiable cost function and is the mechanism behind most neural networks. In linear regression the cost function is derived from the mean squared error. The update consists of forward propagation to calculate the predicted output $y_i^* = x_i \cdot w$ and a backward propagation phase to adjust weights accordingly given differential error and learning rate $\alpha(y_i^* - y_i)x_{ij}$ for an update function of $w_j = w_j - \alpha(x_i \cdot w * -y_i)x_{ij}$.

We implemented mini-batch SGD so the weights would converge more quickly and in smoother fashion to the minimum. Since SPDZ had limited functionality for matrix manipulations and vectorization, we were not able to take advantage of the time speed-ups in this sense. The update function for the batched version is:

$$w = w - \frac{1}{|B|}\alpha X_B^T \times (X_B \times w - Y_B)$$

2.2.1 Non-Linear Activations

For logistic regression, the activation function is defined as the logistic function $f(u) = \frac{1}{1+e^{-}u}$. Using the cross entropy function, we derive a similar update function for mini-batched SGD as $w = w - \frac{1}{|B|}\alpha X_B^T \times f(X_B \times w - Y_B)$. To approximate the activation function, [2] proposed this following piecewise function:

$$f(u) = \begin{cases} 0 & \text{if } u < -0.5 \\ u + 0.5 & \text{if } -0.5 \leq u \leq 0.5 \\ 1 & \text{if } u > 0.5 \end{cases}$$

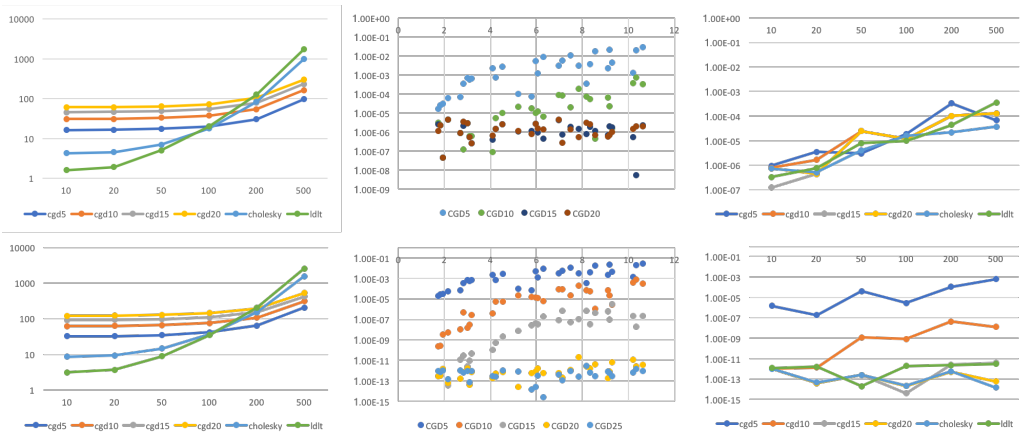

Figure 1: (Left) Run time comparison in seconds as a function of input dimension. (Middle) Condition number as a function of accuracy. (Right) Accuracy as a function of the input dimensionality d. (Top) Fixed-point numbers with the $b = 80$ bits, with 60 in the fractional part. (Bottom) $b = 41$, with 28 in the fractional part

We compare the results of this MPC-friendly piecewise function to a more standard approximation approach of taking the Taylor Series expansion to varying degrees.

3 Experiments

3.1 Experimental Setup

The main metrics of evaluations were the error in accuracy and the RMSE for prediction error. We varied the precision after the decimal point (32 and 64 bits) or less depending on the comparison.

To simulate the players, we needed to identify a source of data. We used both a combination of existing, well-known datasets (MNIST, Arcene, and 9 other UCI open-source datasets) as well as synthetically generated data. The advantage to using synthetic data is being able to control characteristics of the data like the dimensions (d = 10, 20, 50, 100, 200, 500), condition number (cd = [1,10]), and number of examples (n = 1000, 100000).

Typically in a two player setting, both parties were run on one machine but we also performed tests where both parties were deployed on separate Amazon EC2 instances. It is important to note that the distribution of players across machine only affects runtime and not the accuracy of the results. We also looked at up to four parties as a form of validation of the SPDZ protocol.

3.2 Results

This section presents empirical results for real-world and synthetic data evaluated on the SPDZ protocol, comparing the five different algorithms in terms of accuracy and run time for various parameters.

For LDLT, Cholesky and various iterations of CGD, we evaluated on synthetically generated data of varying sizes and condition numbers. The direct decomposition methods grew exponentially in run time on the log scale as the matrix size increased, which means that it is not suitable for large size real data. Alternatively, the iterative CGD runtime increases at a much slower rate. In the middle column of Figure 2, we find that about 20 iterations is sufficient to reach maximum accuracy given the number of allocated bits.

Figure 3, which compares MNIST and Arcene results, demonstrates that the number of bits of precision needed to get good accuracy is highly dependent on the dataset. In terms of logistic regression, for SPDZ, we did not find that the new activation function was a better alternative to taking a Taylor Series approximation for the exponential function as shown in Table 1.

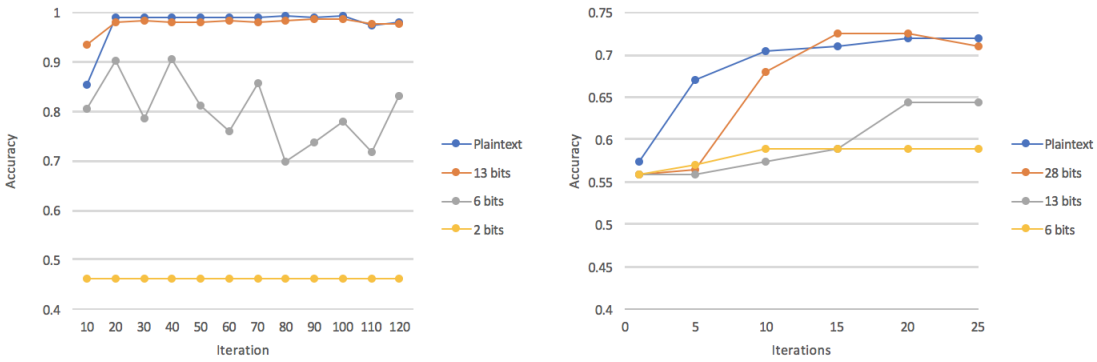

Figure 2: Varying number of bits and recreating results for MNIST (left) and Arcene (right) in [2].

Table 1: Comparing the validation accuracy for different activation functions for logistic regression for a fixed train/test dataset using 32 bits (for 1 epoch and learning rate of 0.01)

	Plaintext	New Activation function	Polynomial Approximation			
			degree 2	degree 5	degree 7	degree 10
MNIST	99.9%	95%	97%	85%	91%	99.5%
Arcene	72.0%	44.0%	44.0%	44.5%	65%	72%

3.3 Comparison

Through this work, we demonstrated that SPDZ outperforms other protocols Obliv-C and SecureML in terms of accuracy, as evidenced in Table FILL, efficiency, and security. In future work, we plan to expand the selected algorithms beyond linear and logistic regression to include neural networks and deep neural networks. This investigation into the applicability of secure MPC on real world data demonstrates its potential change the way data is used in a variety of applications in today's world.

References

[1] Gascón, A., Schoppmann, P., Balle, B., Raykova, M., Doerner, J., Zahur, S., & Evans, D. (2017). Privacy-preserving distributed linear regression on high-dimensional data. *Proceedings on Privacy Enhancing Technologies, 2017(4)*, pp. 345-364.

[2] Mohassel, P., & Zhang, Y. (2017). SecureML: A system for scalable privacy-preserving machine learning. *38th IEEE Symposium on Security and Privacy (SP)*, pp. 19-38. IEEE.

Table 2: Runtime results against

Datasets	Runtime (seconds)		
CGD (64 bit)	Ours	Ours (EC2)	Obliv-C
d=100	144	243	$>10^4$
d=500	527	2701	$>10^3$
SGD: MNIST	Ours	SecureML	
	10200	10330	