Federated Learning using PHE - Training on Shared Data

Vibhav Agarwal and Aayush Grover

CS 821: Privacy-Preserving Machine Learning

May 17, 2020

1 Introduction

Data is the backbone of the entire field of Artificial Intelligence and Machine Learning. It has been proved that the models trained on more data will at least perform as good as the models with lesser data, if not better. All kinds of domains have benefited from these ML models, but the medical domain is still struggling to truly benefit from these models. A major reason for this is the inability to share sensitive medical data.

In this project, we aim to show a secure way of sharing medical data from 'n' number of hospitals to a company X, who helps these hospitals to use machine learning techniques without having to ever disclose their actual data to the company. For our project, we solve the problem of classifying a breast cancer into benign or malignant cancer. We use the scikit-learn dataset: load-breast-cancer for our use case. Since the task is a binary classification, we build a Logistic Regression model and incorporate Homomorphic Encryption for shared training and encryption.

2 Problem Statement

The 'n' hospitals own sensitive medical information about their patients. This data is used to train a model that can predict if the patient has a benign or malignant cancer. The data cannot be shared between any two hospitals (not even in the encrypted format), whereas for a model to predict better, it needs to be trained on the complete dataset (i.e. combination of data from all the hospitals). We must also ensure that no hospital is able to identify the source of the data. In this project, we use Partially Homomorphic Encryption (PHE) scheme to solve the above problem.

There have been previous works where a secure protocol for training on shared data was used. Liu et al [1] uses PHE for distributed learning where each party sends the encrypted weights to the server, where homomorphic additions are done, and the total sum of weights is computed. It is then sent to the leader party and the total weights are then sent to all parties. Giacomelli et al [2] uses Linear Homomorphic Encryption (LWE) to obtain a common dataset from different parties. The Linear Regression model is then trained on this dataset. This allows exchange of data in the encrypted format which might not always be possible given many hospitals' privacy policies. On the other hand, in Aono et al [3], the data is sent to the server in the encrypted format, where data is processed using approximations and the model weights are then sent to the respective data owners. In this scenario, again, the encrypted data is being shared between parties and the approximation of cost function reduces the accuracy of the Logistic Regression model.

3 Types of Homomorphic Encryption

- Partial homomorphic encryption (PHE) schemes allow one selected mathematical operation **unlimited number of times** on the encrypted values.
- Somewhat homomorphic encryption (SHE) schemes, on the other hand, allow a limited set of operations to take place (more than one) but **only a limited number of times**.

- Fully homomorphic encryption (FHE) schemes allow any efficiently computable function (not just addition or multiplication) unlimited number of times. Although, it is the most general form of HE, it loses its value because of the latency. **The speed of an FHE is impractically slow.**

4 Our Approach

Given the task of classifying breast cancer into benign or malignant is binary, we train a Logistic Regression Classification model on the complete data of all the ‘n’ hospitals without accessing their databases or getting to know the actual data points. The hospitals will have different patient records but with the same feature set. In our problem statement, we split our data into ‘test’ and ‘train’ set. This train set is further split into n shares for our ‘n’ number of hospitals. This represents the idea of horizontal split of the dataset, i.e. different patients but same feature set. The ‘test’ data is common for all parties and for all experiments.

Our approach is to not share data in its true or encrypted format but instead share the derived information i.e. gradients (used for gradient descent) in the encrypted format. The company X creates the public-private key pair which then shares the public key to the hospitals for the gradient encryption and keeps the private key with itself. We use Paillier scheme (PHE) for our key generation because the only homomorphic computation that we need is addition (Section 3).

The idea is that each hospital computes its own gradient, encrypts it using the public key and then passes it to another hospital which aggregates this to its own computed encrypted gradient and so on. The last hospital passes the total aggregated encrypted gradient to the company X which then decrypts it and sends back the total gradients to all the ‘n’ hospitals. The hospitals then update their models using this aggregated gradient computed over the complete data.

Algorithm 1 Pseudocode

```
for hospital $h_i$ in n do
    $g_i \leftarrow$ compute gradient for $h_i$ on data $X_i$
    $ag_i \leftarrow g_i + ag_{i-1}$ where aggregated gradient $ag_{i-1}$ from $h_{i-1}$
end for
$d_{ag} \leftarrow$ Company decrypts $ag$
for hospital $h_i$ in n do
    $h_i$ performs gradient descent using $d_{ag}$
end for
$\therefore$ Model is trained on the whole data
```

This allows all the hospitals to have their models trained on the complete sensitive data without actually sharing any information about the patients’ personal data. No data leaves the hospital and yet, the model is trained over the entire dataset.

From the security viewpoint, we consider all parties to be “honest but curious”. Even by seeing the true value of total gradient, no hospital will be able to point out where patients’ data originated. This is true if this protocol is run by at least 3 hospitals, which prevents the reconstruction of each others’ gradients by simply computing the difference. The server ‘X’ can also not learn anything about the underlying data from the total aggregated gradient, thereby preserving patients’ privacy.

5 Results

We test all the hospitals on a common test data. None of the data points in the test data have been seen by any of the hospitals before. We compare the performance of the models that are trained on their individual database against our approach where the model is trained on the complete data as seen in Section 4.

We run the experiments with different number of hospitals. We notice that as the number of hospitals increase, the improvement in average accuracy across hospitals also increase (Table 1). We also compare the results with scikit-learn’s inbuilt Logistic Regression on the complete plaintext training data (Table 1). for all the experiments, we used a key length of 1024 bits.

Number of Hospitals (n)	Avg Acc (LL) (in %)	Train Time (LL) (in s)	Avg Acc (FL) (in %)	Train Time (FL) (in s)	Avg Acc (sklearn) (in %)	Train Time (sklearn) (in s)
3	**96.27**	0.003	95.80	2.821	95.80	0.123
7	**96.00**	0.006	95.80	6.315	95.80	0.028
10	95.38	0.008	**95.80**	9.987	**95.80**	0.033
20	94.93	0.161	**95.80**	18.343	**95.80**	0.036
27	94.17	0.209	**95.80**	24.681	**95.80**	0.038

Table 1: Comparison of our model's performance on breast cancer data with the one without combining data and also, against scikit-learn's Logistic Regression. The best performances are indicated in bold. LL-Local Learning ; FL-Federated Learning

We can also see that the boost in performance comes at the cost of training time but the focus here remains at getting a secure model that can achieve higher accuracy. The sklearn Logistic Regression model acts as good as our model but it is faster as it operates on plaintext data. The difference in accuracies for lower 'n' values could be because of the particular dataset. Hence, we also compare the performances of our model against the local or independent learning model on the grad-admissions dataset (Table 2).

Number of Data Owners (n)	Avg Acc (LL) (in %)	Train Time (LL) (in s)	Avg Acc (FL) (in %)	Train Time (FL) (in s)	Avg Acc (sklearn) (in %)	Train Time (sklearn) (in s)
3	70.40	0.003	**86.40**	0.777	**86.40**	0.024
7	80.46	0.006	**86.40**	1.709	**86.40**	0.029
10	79.28	0.014	**86.40**	2.594	**86.40**	0.027
20	75.24	0.017	**86.40**	5.143	**86.40**	0.030
27	77.16	0.020	**86.40**	6.802	**86.40**	0.024

Table 2: Comparison of our model's performance on grad-admission data with the one without combining data and also, against scikit-learn's Logistic Regression.

Hence, we can conclude that our model will outperform the local learning model and will give as good an accuracy as the scikit-learn model.

6 Conclusion

Therefore, our model can predict outputs as good as the scikit-learn model on plaintext data while preserving the privacy of all the data sources. Although, the time taken to train is more than independent learning because of the PHE scheme and more data points, by around a factor of 1000 for the breast cancer and 300 for the grad admission dataset. This depends on the size of the dataset and the number of parties. The protocol proposed by us also ensures that no single party (including the server) can infer anything about the data or its source, thereby preserving the patients' privacy.

References

[1] Changchang Liu, Supriyo Chakraborty, and Dinesh Verma. Secure model fusion for distributed learning using partial homomorphic encryption. In *Policy-Based Autonomic Data Governance*, pages 154–179. Springer, 2019.

[2] Irene Giacomelli, Somesh Jha, Marc Joye, C David Page, and Kyonghwan Yoon. Privacy-preserving ridge regression with only linearly-homomorphic encryption. In *International Conference on Applied Cryptography and Network Security*, pages 243–261. Springer, 2018.

[3] Yoshinori Aono, Takuya Hayashi, Le Trieu Phong, and Lihua Wang. Scalable and secure logistic regression via homomorphic encryption. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pages 142–144, 2016.