uTile PET: a Privacy Preserving Solution for Collaborative Data-Driven Projects

Daniel Hurtado Ramírez, Luis Porras Díaz, Álvaro Calzado Pérez, Borja Irigoyen Peña, Alexander Benítez Buenache, Juan Miguel Auñón García, Ana María García Sánchez and Pablo González Fuente

April 18, 2021

UTILE PET: A PRIVACY PRESERVING SOLUTION FOR COLLABORATIVE DATA-DRIVEN PROJECTS

D. Hurtado, L. Porras, Á. Calzado, B. Irigoyen, A. Benítez, J. M. Auñón, A. M. García, P. González

GMV
Department of Artificial Intelligence and Big Data
Isaac Newton, 11, Tres Cantos, Madrid, Spain

ABSTRACT

There are an increasing number of data-driven space projects. In these projects, the solution is as important as the quality of the data. Moreover, the more data available, the better the performance, so it is normal for different entities to collaborate on a common solution. However, this can be a problem in terms of privacy and it may not always be possible to share data between the different parties. Therefore, we present uTile PET, a solution for the collaborative development of Artificial Intelligence algorithms without the need to compromise the privacy of each of the parties. In addition, *SHK-means*, a clustering algorithm that works with distributed data and maintains privacy at all times, is presented as an example.

***Index Terms*—** Privacy Preserving, Multi-party computation, Federated Learning, K-means clustering

1. INTRODUCTION

Artificial Intelligence is becoming more and more relevant in space domain projects. The current potential of Machine Learning (ML) solutions allows their use to complement (or even replace in some cases) classical techniques for solving tasks such as signal processing [16] or anomaly detection [12], to cite a couple of examples. However, these techniques require the collection and use of historical data to train the algorithms. This is undoubtedly one of the most problematic aspects due to the required privacy with projects data. In some cases it can be solved by means of complex confidentiality agreements. Nevertheless, this may limit or even preclude the use of the data. In addition, in order to obtain optimum performance, it is necessary to have as much data as possible. Thus, it is very common to create consortiums between entities, companies and/or universities to jointly address a project. But again, problems arise when it comes to sharing data due to privacy issues. In view of the above, new approaches have emerged with the aim of solving privacy problems during the training of ML models, among which Secure Multi-party Computation (SMPC) and Federated Learning (FL) stand out.

SPMC, and in particular additive secret sharing, allows a secret data to be segmented into parts in such a way that none of the participating parties has the ability to reconstruct the original secret data, but all of them benefit from the result of the sharing. SMPC has evolved a lot from its beginnings in the 80s, starting from theoretical research (see Yao and Shamir seminar papers [13, 17]), to real applications that allow the use of this technology to end users [2]. This evolution has been possible thanks to the effort of different research groups, which have tried to develop and optimize the protocols responsible of the computation (see [4]).

On the other hand, FL is an algorithmic solution that allows the training of ML models by sending copies of a model (commonly, trainable model parameters, such as the weights or gradients during training) and performing training to the location where the data resides, eliminating the need to share data on a central server. Sometimes even sharing this information can be sensitive in terms of privacy, as the original data can be reconstructed from such model information. In these cases, SMPC can be used to share this information. Although FL solutions are relatively recent, it is a line of research with great interest from the scientific community due to the great potential of its use. For the interested reader, the article [7] is recommended reading.

Both approaches allow different entities to collaborate in obtaining a joint solution to a common problem without the need to share their data, thus complying with established privacy protocols. This is the backdrop for the presentation of uTile PET (Privacy-Enhancing Technologies), the GMV's solution for the development of collaborative data-driven projects while preserving privacy between different parties.

The paper is organized as follows: Section 2 briefly introduces uTile solution. Section 3 focuses on one of these solutions, namely K-means, describing the solution and showing its advantages by means of a synthetic example. Some conclusions and the description of future lines of research close the paper in Section 4.

Thanks to Spain's Ministerio de Economía y Empresa (TSI-100906-2019-2) and GMV for funding, and to the OpenMined community, whose efforts have provided a solid base on which to experiment and iterate.

2. UTILE PET

uTile PET is a GMV-developed solution for harnessing confidential and private data to improve ML algorithms and analytical models, complying at all times with organizational requirements, guaranteeing data privacy as well as current regulations. There is no need to choose between data privacy and usability, as it leverages advanced cryptographic methods that keep data encrypted while all the necessary computations are performed. In this way, uTile PET enables the possibility of organizations' sensitive data will never be exposed or transferred across departments, organizations or different countries.

Hence, uTile PET is a set of solutions that, in addition to the aforementioned technologies (SMPC and FL), includes Private Set Intersection (PSI) [1]. PSI is a cryptographic technique that allows finding the intersection between several vertically distributed datasets (the data are distributed among the parties by columns) without having to expose the data, thus protecting data privacy.

In this way, uTile PET creates an environment on which to develop ML solutions while preserving data privacy. Among the algorithms already implemented are the following:

- K-means [5] clustering, a well known unsupervised method for finding clusters of points.
- Principal Componnent Analysis (PCA) [15], algorithm to reduce the data dimensionality.
- Neural network architectures.

These solutions have been developed on top of PySyft [11], a Python library for secure and private ML, where most of the Additive Secret Sharing and secureNN [14] routines are implemented. It is also in charge of managing the communication between the parties involved in the computation.

3. PRIVACY PRESERVING K-MEANS CLUSTERING

As an illustrative example, this section describes the implemented solution for privacy-preserving K-means when data is horizontally distributed, i.e., several parties P_j have different entries (rows), however all of them keep the same schema. For the reader interested in vertical partitioning (several parties have different features), we recommend reading [9], where both approaches are described in detail.

3.1. Problem definition

In Earth Observation (EO) problems it is common to use K-means [6, 8] to identify areas with their own characteristics from the received data (images or signals). Nevertheless, for demonstration purposes, we have generated a dummy dataset X consisting of 4 subsets of 100 samples, where each subset follow a 2-dimensional Gaussian distribution of means $\mu_0 = (5, 3)$, $\mu_1(5, -5)$, $\mu_2(-5, 5)$ and $\mu_3(-3, -5)$; and a standard deviation $\sigma_0 = \sigma_1 = \sigma_2 = \sigma_3 = 1$.

The usual K-means algorithm assumes that the we have full access to the data, leaving aside privacy concerns. However, in this case we will study its application in horizontal distributed data. Following the usual notation, the famous characters Alice (A) and Bob (B) [10] are used as parties for horizontal distribution as follows: $X_{400\times 2} = X^{(A)}_{200\times 2} \cup X^{(B)}_{200\times 2}$, being the intersection between $X^{(A)}$ and $X^{(B)}$ an empty set ($X^{(A)} \cap X^{(B)} = \emptyset$). Notice that the difference between both comes from the samples, the number of columns is kept. Figure 1(a) shows this data distribution.

3.2. Centralized K-means

First, the operation of the classic K-means is recalled (iterating until mean does not change):

1. Select k random centroids C, being k the number of desired clusters. In this example, we know the number of groups, so we set $k = 4$, but in a real problem, it would be a parameter to explore.
2. Calculate the distance between data points and centroids, assigning each data point to the closest centroid.
3. Re-calculate the clusters centroids performing the mean of the data belonging to such centroid.

Considering that both parties cannot share their data, the following flow is defined to obtain the results:

- **Training phase**. Alice wants to train the centralized *K-means* algorithm with her data, $X_{train} = X^{(A)}$, so she will train the algorithm. The output of this step is the centroids C.
- **Testing phase**. For the testing phase, Alice and Bob reach an agreement and Alice sends the model (C in this case) to Bob, then Bob just have to analyze the results with his own data (data never seen for the algorithm): $X_{test} = X^{(B)}$. Bob calculates the distance between C and X_{test}.

Figure 1(b) shows these steps in a single snapshot. The *training phase* shows that points have been assigned to 4 clusters. It can be seen how the model trained locally with one party's data may not generalize for the other party's data.

3.3. SPMC approach: Secure Horizontal K-means

Secure Horizontal K-means (*SHK-means*) deals with horizontally distributed data, translating the core steps from original K-means algorithm into the SMPC framework. It takes all the data as an additively shared matrix, which allows the data to be arbitrarily distributed between all parties. Its operation is described in Algorithm 1, however, some aspects of

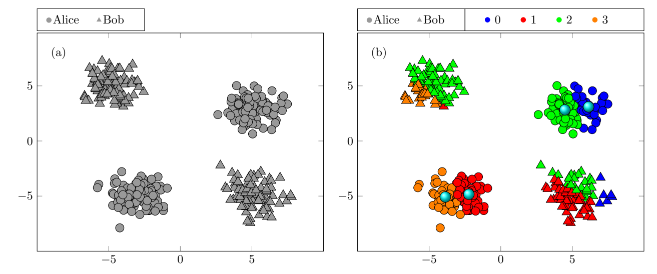

Fig. 1. (a) Dataset distribution between Alice and Bob. (b) Labeled data points after training the centralized K-means with data from Alice (○) and testing with data from Bob (△). Centroids C from training phase are marked in cyan.

notation and privacy are clarified below, in addition to a brief description of the protocols, which have been used as building blocks. For a more detailed description about protocols, it is advisable to read [9].

From [14] we borrow the notation and the following protocols: *secure matrix multiplication, DReLU* and *division*. Note that *DReLU* stands for Derivative of *ReLU*; that is, *DReLU*$(x) = 0$ if $x < 0$ and *DReLU(x)* $= 1$ if $x > 0$

Altought the protocols in SecureNN are presented for the 2-party case, they can be easily extended to work with p parties. The protocols require the presence of an extra party that assists in the computations without providing data; we take P_0 as the assistant party, i.e., the responsible of generating triplets for Beaver multiplication [3], and $P_1, P_2, \ldots, P_p$ to be the parties holding the data.

We assume the aggregate of all parties' data consists of n samples of d-dimensional data, and this data is shared additively across parties $P_1, P_2, \ldots, P_p$. $\langle X \rangle_j$ refers to P_j's share of X, such that $X = \sum_{j=1}^{p} \langle X \rangle_j$ (over $\mathbb{Z}_L$, a finite field with size L). $X[i]$ refers to sample i, $X[i][j]$ refers to coordinate j of sample i $(1 \leq i \leq n, 1 \leq j \leq d)$.

The goal is to split the data into k clusters. These clusters are defined by their centroids C, where $C[j]$ refers to the centroid of cluster j $(1 \leq j \leq k)$. The centroids will also be shared across all parties, so $C = \sum_{j=1}^{p} \langle C \rangle_j$.

We wish to reveal as little data as possible; this includes the original samples, the resulting clusters, the label of any given sample, and distances between the samples and the cluster centroids. We consider acceptable revealing how many samples correspond to each cluster (but not which ones) since nothing useful can be extracted from this information while providing a significant speedup of the algorithm.

As one of the core operations for the secure K-means algorithm, algorithm *ElementWiseMatMul* addresses the elementwise secure matrix multiplication, i.e., $Z_{n \times d} = X_{n \times d} \odot Y_{n \times d}$ ($\odot$ symbol representing the elementwise multiplication), where parties P_j $(j \geq 1)$ hold shares of X, Y. This algorithm is the elementwise version of Π_{MatMul} (see Reference [14]), extended to p parties.

Algorithm 1: SHK-means

```
Input  : A shared matrix $X_{n \times d}$ of points, the public
         number $k$ of clusters, a public number $\epsilon > 0$
         for the stopping criterion
Output: A shared matrix $C_{k \times d}$ of cluster centroids
  1. Select random public integers $l_1, l_2, \ldots, l_k$,
     $1 \leq l_j \leq n$. Each party $p$ sets $C_p[j] = X_p[l_j]$,
     for all $j, 1 \leq j \leq k$
while *true* do
    2. Compute $H :=$ *LabelSamples*$(X, C)$ using the
       *LabelSamples* protocol
    3. Set $t_j := \sum_{i=1}^{n} H[i][j]$, the total number of
       samples that go to cluster $j$
    4. Set $T_{k \times d} :=$ *MatMul*$\left(H^T, X\right)$. Intuitively, row $j$
       of $T$ is the sum of samples belonging to cluster $j$
    5. Compute the new centroids as $\tilde{C}[j] := T[j]/t_j$.
       This division can either be done with a secure
       division protocol (completely secure, but slow)
       or by first revealing the values $t_j$ to both parties
    6. For each $j \in \{1, \ldots, k\}$, use the $MatDist$
       protocol to compute $D_j =$ *MatDist*$\left(C[j], \tilde{C}[j]\right)$
    7. Set $C := \tilde{C}$
    8. Compute the additively shared value
       $\Delta = \sum_{j=1}^{k} D_j$, the total movement of all centroids
    9. Compute $s :=$ *DReLU*$(\epsilon - \Delta)$ and reconstruct its
       value. If $s = 1$, stop and return $C$;
       if $s = 0$, return to step 2
end
```

On the other hand, algorithm *MatDist* computes the secure squared euclidean distance d^2 between two shared vectors $\mathbf{x}$, $\mathbf{y}$: $d^2(\mathbf{x}, \mathbf{y}) = (x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_d - y_d)^2 = \sum_{j=1}^{d} (x_j - y_j)^2$, namely between a data point $X[i]$ and a centroid $C[j]$.

Finally, algorithm *LabelSamples* addresses the "label" of a data point $X[i]$, meaning that a point will belong to a cluster if and only if the distance between this point and the $j^{th}-$cluster is closer than the rest.

Bearing all the above in mind, *SHK-means* learns how data points are grouped while keeping privacy. Taking advantage of ML terminology, this phase is usually denoted as *training phase*. Figure 2(b) shows the resulting global model using *SHK-means*. In the *testing phase*, i.e., when centroids are already determined and new data points Y (additively

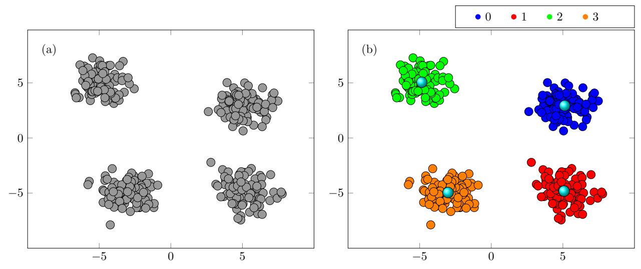

Fig. 2. (a) Global dataset distribution. (b) Labeled data points after training the global *SHK-means* with data from Alice and Bob. Centroids C are marked in cyan.

shared across parties) need to be labeled, it is enough to compute $H = \mathit{LabelSamples}(Y, C)$ and reconstruct it. Thanks to the fact that H is a one-hot encoded shared matrix, it could also be directly used as an input to some other secure protocol that builds on top of K-means, without losing any privacy.

4. CONCLUSIONS AND FURTHER RESEARCH

With this work, we have shown how uTile PET bridges the gap between the low-level SMPC protocols and the high-level work of the data scientist. We have shown a general approach for adapting traditional ML algorithms to a secure setting, exemplifying in detail the case of K-Means.

Besides the adaptation of other learning algorithms for privacy-preserving, we are analyzing the effects that such solutions can have on some singular problem families, such as imbalanced data.

REFERENCES

[1] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pages 86–97, New York, NY: ACM, 2003. doi: 10.1145/872757.872771.

[2] D. W Archer, D. Bogdanov, Y. Lindell, L. Kamm, K. Nielsen, J. I. Pagter, N. P Smart, and R. N Wright. From Keys to Databases-Real-World Applications of Secure Multi-Party Computation. *The Computer Journal*, 61 (12):1749–1771, 2018. doi: 10.1093/comjnl/bxy090.

[3] D. Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *Advances in Cryptology*, pages 420–432, Berlin, Heidelberg: Springer, 1992.

[4] R. Cramer, I. B. Damgård, and J. B. Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015. doi: 10.1017/CBO9781107337756.

[5] K. Fukunaga. Chapter 11 - clustering. In K. Fukunaga, editor, *Introduction to Statistical Pattern Recognition (Second Edition)*, pages 508 – 563. Boston, MA: Academic Press, 1990. doi: https://doi.org/10.1016/B978-0-08-047865-4.50017-X.

[6] F. Georgescu, C. Vaduva, D. Raducanu, and M. Datcu. Feature extraction for patch-based classification of multispectral earth observation images. *IEEE Geoscience and Remote Sensing Letters*, 13(6):865–869, 2016. doi: 10.1109/LGRS.2016.2551359.

[7] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3):50–60, 2020. doi: 10.1109/MSP.2020.2975749.

[8] R. Lucas et al. The earth observation data for habitat monitoring (EODHAM) system. *Intl. J. Applied Earth Observation and Geoinformation*, 37:17–28, 2015. doi: https://doi.org/10.1016/j.jag.2014.10.011.

[9] D. Hurtado-Ramírez, and J. M. Auñón. Privacy preserving k-means clustering: A secure multi-party computation approach, 2020.

[10] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.

[11] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, and J. Passerat-Palmbach. A generic framework for privacy preserving deep learning, 2018.

[12] M. Sakurada and T. Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. MLSDA'14, page 4–11, New York, NY: ACM, 2014. doi: 10.1145/2689746.2689747.

[13] A. Shamir. How to share a secret. *Commun. ACM*, 22 (11):612–613, 1979. doi: 10.1145/359168.359176.

[14] S. Wagh, D. Gupta, and N. Chandran. Securenn: Efficient and private neural network training. *IACR Cryptol. ePrint Arch.*, 2018:442, 2018.

[15] S. Wold, K. Esbensen, and P. Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[16] B. Yan and W. Qu. Aero-engine sensor fault diagnosis based on stacked denoising autoencoders. In *2016 35th Chinese Control Conf.*, pages 6542–6546, 2016. doi: 10.1109/ChiCC.2016.7554387.

[17] A. C. Yao. Protocols for secure computations. In *Proc. 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.