Efficient Privacy Preserving Distributed K-Means for Non-IID Data

André Brandão[1], Ricardo Mendes[2][0000−0003−2077−7223], and João P. Vilela[1][0000−0001−5805−1351]

[1] CRACS/INESCTEC, CISUC and Dep. of Computer Science, Faculty of Sciences, University of Porto, Portugal
[2] CISUC, Dep. of Informatics Engineering, University of Coimbra, Portugal

Abstract. Privacy is becoming a crucial requirement in many machine learning systems. In this paper we introduce an efficient and secure distributed K-Means algorithm, that is robust to non-IID data. The base idea of our proposal consists in each client computing the K-Means algorithm locally, with a variable number of clusters. The server will use the resultant centroids to apply the K-Means algorithm again, discovering the global centroids. To maintain the client's privacy, homomorphic encryption and secure aggregation is used in the process of learning the global centroids. This algorithm is efficient and reduces transmission costs, since only the local centroids are used to find the global centroids. In our experimental evaluation, we demonstrate that our strategy achieves a similar performance to the centralized version even in cases where the data follows an extreme non-IID form.

Keywords: Privacy · Distributed Clustering · Federated Learning · Homomorphic Encryption · Secure Aggregation

1 Introduction

Ubiquitous devices allow for ever-growing data collection. This data is useful in machine learning to optimize services and to extract information about the population [17]. For example, sensor data from mobile phones can be used to infer transportation modes [8] or to accurately estimate traffic congestion [23].

One of the techniques to extract information is clustering, where algorithms partition objects into groups in order to find hidden structures in the data [27]. Clustering algorithms belong to the unsupervised learning class, *i.e.*, they can learn from unlabeled data. Labeling datasets is both costly and time consuming [31], therefore clustering plays a crucial role in the machine learning paradigm.

In order to increase the amount and diversity of the data, entities can jointly apply learning algorithms to the combined data. This is also used in the mobile scenario, where each user shares his collected data. Traditionally, in this context, learning is performed in a central trusted server, which receives the data from all entities. However, this approach requires data owners to trust the server

with their data, thus posing a privacy risk [25]. In order to overcome this issue, distributed privacy preserving mechanisms have been proposed.

Distributed privacy preserving mechanisms can be evaluated in three axis: privacy guarantees, efficiency and robustness to non-IID data. In the context of mobile/crowd-sourcing scenarios, robustness to non-IID data is particularly important, as the clients are the individuals collecting and storing the data. In turn, this data might belong to a single or to a small subset of clusters that may strongly vary between the different clients (non-IID case). This situation can reveal to the clustering server which cluster(s) the client's data belongs to, thus posing a risk to individuals' privacy. For example, in [11] information related to the users' app permission choices were clustered to create privacy profiles, where each user belonged to a single profile. Therefore, in this case, a server would know the privacy preferences of each user.

Existing distributed privacy preserving clustering approaches fall short at either privacy, efficiency and/or robustness to non-IID data. In this paper, we propose a strategy to apply distributed K-Means that, unlike previous work, is efficient, mutually private and robust to non-IID data. To reduce the data that is shared with the server and for robustness against non-IID data, clients compute the K-means locally, with a variable number of clusters, and only the centroids are sent to the server. To preserve privacy, the centroids are encrypted homomorphically, which still allows the server to compute the distance from the local centroids to the global centroids, over encrypted data. The distances are then sent to the clients who, after decryption, assign each local centroid to a global centroid. To update the global centroids in the server, secure aggregation is used, thus keeping the data private. Results show that the proposed strategy achieves a similar performance to the centralized K-means even for non-IID data.

The rest of this paper is organized as follows. Section 2 presents the related work. Section 3 details our proposal and presents the evaluation results, while section 4 concludes this work.

2 Related work

In this section we revise previous work in clustering and privacy preserving distributed clustering. We identify the advantages and disadvantages of current strategies and compare them according to their efficiency, privacy and robustness to non-IID data.

2.1 K-Means

K-Means [12], is one of the most known and widely used clustering algorithms. The goal of K-Means is to assign each observation to a single cluster minimizing the within-cluster Euclidean distance:

$$C_1^* \dots C_k^* = \underset{C_1 \dots C_k}{\operatorname{argmin}} \sum_{k \in K} \sum_{x_i \in C_k} ||x_i - c_k||^2 \tag{1}$$

where K is the set of cluster IDs, C_k is a set of points representing a cluster and c_k is the centroid value, i.e., the mean point for cluster k. The C_i^* are the resulting clusters.

The algorithm works as follows: given a set of k initial centroids $c_1^{(1)}, c_2^{(1)}, \ldots, c_k^{(1)}$, the algorithm iteratively executes the following two steps:

Assignment: Assign each observation to the cluster with the nearest mean, with respect to the Euclidean distance.

Update: Compute the new centroids' values:

$$c_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{x_p \in C_i^{(t)}} x_p \tag{2}$$

The algorithm repeats these steps until the shift between the new centroids and the previous ones is lower than a specified threshold ϵ, i.e. $||c^{(t+1)} - c^{(t)}|| \leq \epsilon$.

2.2 Privacy preserving Distributed K-Means

McMahan *et al.* presented in 2017 an efficient strategy to train neural networks from decentralized data [14]. This algorithm, designated by Federated Averaging takes advantage of the local clients' computing power to apply the gradient descent algorithm to the clients' data in their own devices. At each iteration, every client will send the gradient to the server where it is averaged and distributed to every client.

Triebe and Rajagopal joined mini batch K-Means [19] and Federated Averaging to create the **federated K-Means algorithm** [24], where clients share the centroids' position and number of observations per cluster, instead of the gradient. The server applies a weighted average over the centroids' positions based on the cluster size and send back the results to the clients. The main problem with this approach in the context of non-IID data is that averaging points in opposite extremes results in final centroids in the center of the dataset, thus achieving poor performance. Another problem of this strategy is the centroids initialization method. Since we do not have access to the data, we have to randomly select k points from the input space $\mathcal{X}$. However, this method achieves poor performance and may lead to empty clusters [2].

A different approach to privacy preserving clustering is taken by methods that resort to **homomorphic encryption** [10, 26, 28, 30]. This encryption technique allows the clustering to be done over the encrypted data, thus preserving privacy. However, in order for the server to update the global centroids, the clients must send the sum and number of points in each cluster to the server, which can disclose which cluster(s) the clients' data belongs to. Additionally, due to the amount of data that is sent to the server, it allows clients to apply trilateration to find the global centroids [15], thus it is not mutually private. Because all data is encrypted and computations are made over the encrypted data, these approaches are computationally expensive [5].

Another common strategy for private clustering is based on **differential privacy (DP)** [6, 13, 18, 22]. DP consists on adding "statistical noise" that is

significant enough to protect client's privacy, but small enough to not affect the model performance. Although this strategy is efficient and robust to non-IID data, it lacks a systematic methodology, due to the challenge of defining the amount of noise, as it highly depends on the dataset [3]. Other problems of differential privacy include the inherent uncertainty in the answer and the fact that the guarantees of immunity to background knowledge attacks are overstated [3]. It has also been shown recently, that DP can increase existing biases and have substantial impacts on the accuracy [4].

An alternative private distributed clustering approach is to select a subset of **local points (representatives)** and apply clustering over them. Soliman *et al.* [21] proposed running the K-Means algorithm locally on each client and using HyperLogLog counters to share the centroids and the approximate number of observations per centroid in a decentralized fashion with the other clients. Then a weighted averaging over all the centroids is done to find the global centroids. Januzaj *et al.* [9] have a similar strategy, where local representatives are extracted on site and then shared with the server, who finally performs a global clustering over the representatives. Both strategies are efficient and robust to non-IID data, but lack privacy. In the first strategy, the clients will know to which cluster(s) the other clients' data belongs to. In the second strategy, partial data, *i.e.* the representatives, is sent to the server.

Finally, two under-looked problems arise in privacy preserving K-Means, when there is no access to the data, which is how to choose the ideal number of clusters and compute the initial centroids. Additionally, because there is no access to the dataset, the typical metrics to evaluate the model cannot be used. In this paper we present an approach that overcomes the identified issues. Specifically, we propose a K-means algorithm that is efficient, secure and robust to non-IID data. We additionally propose secure inertia, a secure method that allows the estimation of the ideal number of clusters, the creation of the initial centroids and the evaluation of the final model.

3 Efficient Privacy Preserving Distributed K-Means for Non-IID Data

This section details our proposed approach towards an efficient and secure K-Means algorithm that is robust to non-IID data. In section 3.1 we describe our proposed approach and in section 3.2 we present a detailed evaluation of the privacy, efficiency and robustness to non-IID data of our strategy.

3.1 Description

We consider a setup where a set of clients, each possessing its own dataset of N points with two features, aims to cluster their data. In this scenario, we propose a mechanism that allows the clients to cluster their data, alongside the other clients without the server knowing the clients' points. The operation of our proposed mechanism is illustrated via an example in figure 1. In the first stage, each client

performs Diffie-Hellman Key Exchange so as to agree on a seed with the server. Then, they will compute the sum and number of points in their local dataset. These two statistics will be masked and securely sent to the server using secure aggregation ("Send Masked Statistics" step on figure 1). **Secure aggregation** presented by Bonawitz *et al.* [1] allows the secure sum of vectors using Diffie-Hellman Key Exchange, where the resulting secret will be used as a seed to generate random vectors. These random vectors will be the same for each pair of clients: one of the clients adds the random vector and the other subtracts the vector to their contributions and both send the result to the server. To the server each contribution will be indistinguishable from a random vector. But when the server adds all the contributions, the random vectors cancel out, retaining only the real sum of every client's contribution. Using this strategy the server is able to compute the center point of all clients' datasets. In the example from figure 1, $\frac{(28+56,50+58)}{100+200} = (0.28, 0.36)$. To improve the k-Means performance we choose k random points close to the center point as initial centroids. Since the server cannot choose k points from the data (known only to the clients) and choosing random points from the input space would result in a poor performance [2].

While the server is initializing the centroids, the clients will apply the K-Means to their local dataset to generate local clusters. Each client uses the silhouette score to estimate the best number of local clusters as follows. Given the following metrics:

$$a(i) = \frac{\sum_{j \in C_i, i \neq j} d(i,j)}{|C_i| - 1} \qquad b(i) = \min_{k \neq i} \frac{\sum_{j \in C_k} d(i,j)}{|C_k|} \qquad s(i) = \frac{a(i) - b(i)}{max\{a(i), b(i)\}}$$

where C_i is the set of points in cluster i and $d(\cdot)$ is the distance metric. We can interpret $a(i)$ has how well is point i assigned to its cluster and $b(i)$ as the smallest mean dissimilarity of any cluster except C_i. The silhouette score is defined as the mean $s(i)$, over all observations of the local dataset. We compute the silhouette score for $2 \leq k \leq K$, where K is the number of global clusters, defined by the server. Then, the number of clusters for the model with the maximum silhouette score is chosen as the optimal local number of clusters.

The resulting centroids for the local clusters will be sent to the server in a secure way, using homomorphic encryption. **Homomorphic encryption** allows one to perform calculations over encrypted data, in such a way that when the result is decrypted it yields the same result as if the operations were made over unencrypted data. The centroids will be encrypted and sent to the server, that will compute the distances between the encrypted local centroids and the unencrypted global centroids. To calculate the distances between the pairs of points, we will need a schema that is able to subtract scalars from encrypted numbers and multiply an encrypted number by itself. The CKKS schema [20] offers us that possibility. By only providing approximate results, it is more efficient, but still precise enough for machine learning models [20], as we also assess. Even with the encrypted centroids, the server would still be able to known how many clusters each client has. In order to hide this informantion, each client will always send information about k clusters, by adding random centroids whenever needed. In the example from figure 1, the server defined $k = 2$ as the number of

clusters and Client B found a single centroid, namely $(0.28, 0.29)$. So, the client is going to send $[c_1 = (0.28, 0.29), c_2 = (0.56, 0.89)]$, where the second centroid is randomly created to deceive the server.

The server computes the squared distance between the encrypted local centroids and every global centroid. The encrypted distances are sent back to the users, who decrypt them and assign each local centroid to a global cluster. Now each client sends to the server the sum and the number of points for each cluster – this information is needed for the server to compute the new centroids (equation (2)). In the example from figure 1, the resulting statistics are $[((0.28, 0.29), 1), ((0, 0), 0)]$, for client B, where $(0.28, 0.29)$ represents the sum of points in this cluster and 1 the number of points in the cluster. The distances for the random centroid are ignored and both the sum and the number of points are set to zero. Since only the local centroids are sent to the server, this will difficult the trilateration attack, since in order to infer the global centroids from the distances, the client needs to have more points than the number of features [15]. To update the global centroids in the server without revealing the clients' individual statistics, secure aggregation is employed again to send the masked statistics. With these statistics, the server is able to compute the new global centroids' values by dividing the total sum by the total number, just like in equation (2). These steps will be repeated until the new centroids are equal to the previous ones or a maximum number of iterations is performed.

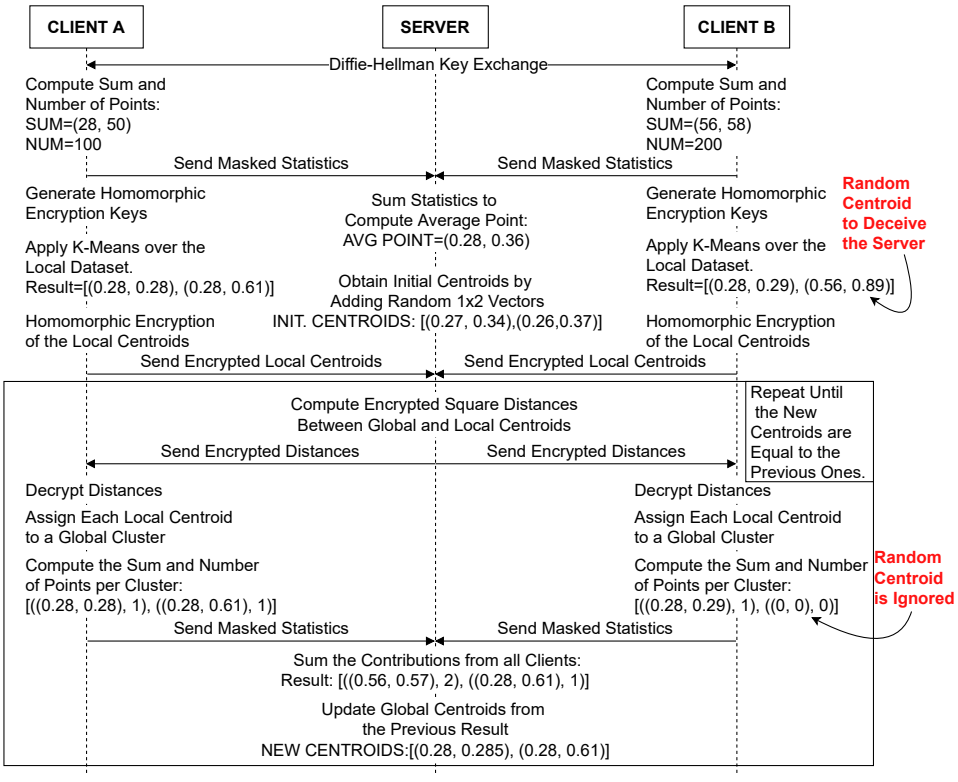


Fig. 1: Sequence diagram (with example) of the proposed algorithm.

3.2 Evaluation

To evaluate this strategy we ran the algorithm through 114 artificial benchmark datasets[3]. To simulate an environment of non-IID data, we created 20 clients, where each has a random number of clusters from 1 to k (depending on the dataset). Every client has a random number of 70 to 90 observations from one cluster and a random number of 1 to 30 observations for each of the remaining clusters, except in the case of only one cluster, with 100 observations from the same cluster. All random numbers are generated from a uniform distribution.

Since we have access to the cluster labels of the benchmark datasets, we used the Adjusted Rand Index (ARI) metric to compare our strategy with the centralized K-Means over the entire dataset [7]. Let K_t be the clustering ground truth and K_p the clustering done by the model. If a is the number of pairs of elements that are in the same set in K_t and K_p and b the number of pairs of elements that are in different sets in K_t and K_p, then

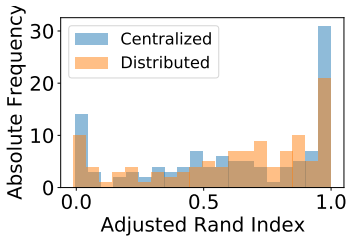
$$\mathrm{RI} = \frac{a+b}{C_2^{n_{\text{samples}}}} \qquad (3) \qquad\qquad \mathrm{ARI} = \frac{\mathrm{RI} - E[\mathrm{RI}]}{\max(\mathrm{RI}) - E[\mathrm{RI}]} \qquad (4)$$

Since the Random Index (RI) in (3) does not guarantee a value close to zero for random label assignments, we resort to the ARI in (4) that counters this effect by subtracting the expected RI. We run the strategy 30 times for each dataset and the one with lowest inertia is chosen. The inertia metric corresponds to the sum of the distances of all points within a cluster to the respective centroid [16]. Each client computes the local inertia and, using secure aggregation, the server obtains the total inertia, without knowing individual contributions. This metric can be used to compare models and allows the server to use the *elbow method* to estimate the best number of clusters in the dataset [29]. We chose the best inertia because our goal is to prove that our strategy can achieve a good performance with few repetitions. Additionally, this procedure replicates a real world scenario where one computes the inertia securely in order to choose the best model. We compared the results of our strategy to the centralized version in terms of efficiency and robustness to non-IID, since the centralized version achieves the best results in these two criteria.

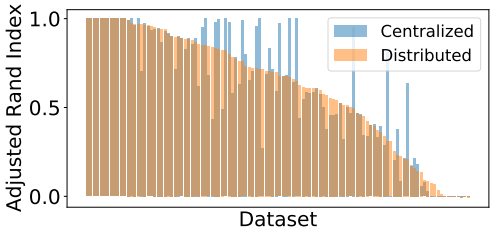
Robustness to non-IID Data Figure 2a shows the distribution of the ARI over the 114 datasets of the proposed strategy (orange) versus the centralized K-Means (blue). From this plot, we can see that our strategy and the centralized K-Means have a similar distribution, however our proposal achieves smaller values at the extremes. In fact, our strategy was not able to achieve ARI values above 0.9 in around 10 datasets. However, it also produced fewer ARI values closer to 0 and achieved more ARI values between 0.6 and 0.9. Overall, we can conclude that this strategy is consistent with the centralized version of K-Means.

Figure 2b presents the ARI score for every dataset. The number of datasets for which our approach performed better than the centralized model is 56, worse in 43 and had the same performance in 15 of the 114 datasets. As aforementioned,

[3] `https://github.com/deric/clustering-benchmark`



(a) Adjusted rand index distribution.



(b) Adjusted rand index by dataset.

Fig. 2: Adjusted rand index distribution and adjusted rand index by dataset.

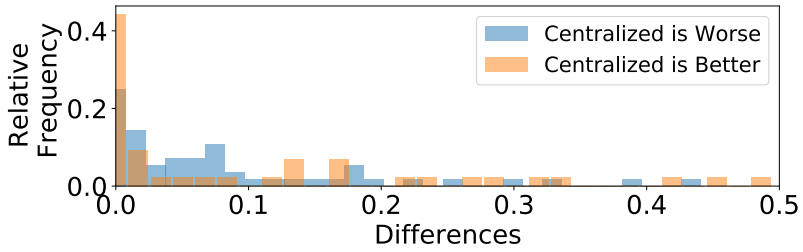


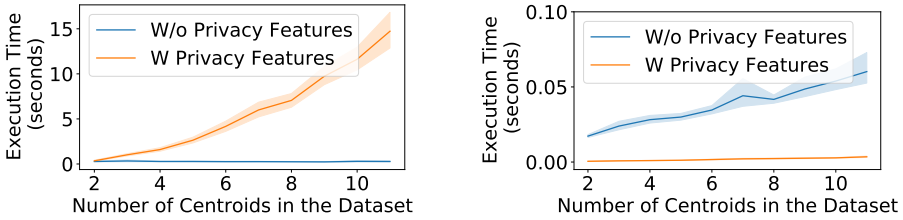
Fig. 3: Adjusted rand index differences.

when the centralized model is better, it is usually better by a larger difference than when the centralized model is worse.

Figure 3 compares the differences between ARI scores when the centralized model is worse than our strategy (blue distribution) and when the centralized model is better than our strategy (orange distribution). We can observe, as previously discussed, that the two distributions are similar. More than 50% of the times, when the centralized model is better, the ARI differences are lower than 0.1 and the same happens when the centralized model is worse. However, we see a more uniform distribution in the interval between 0 and 0.1 when the centralized model is worse, as to when the centralized model is better, where approximately 45% of the times the difference is practically 0.

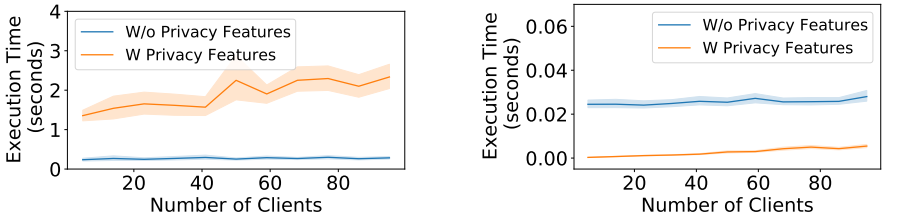
From figures 2 and 3 we conclude that the proposed strategy achieves a similar performance to the centralized K-Means. Specifically, it achieves a better ARI score in more datasets than the centralized version, but the majority of the differences are lower than 0.1. Therefore, our strategy, while decentralized, is robust against non-IID data.

Efficiency In order to understand the efficiency of our final strategy we compared it to the strategy without privacy features, *i.e.* without secure aggregation and homomorphic encryption. To measure the efficiency, we measured the execution time, without taking into account the network delay, in three dimensions: number of points, number of clusters and number of clients. For each dimension we measured the time spent by the clients and by the server. To facilitate the efficiency evaluation, we assume the server to know how many clusters there are in the dataset. Since the steps executed by the clients are done in parallel, the



(a) Clients' execution time. (b) Server's execution time.

Fig. 4: Execution time versus the number of centroids on the clients and server side (95% confidence intervals are shaded). The case with privacy features corresponds to our proposed method.



(a) Clients' execution time. (b) Server's execution time.

Fig. 5: Execution time versus the number of clients on the clients and server side (95% confidence intervals are shaded). The case with privacy features corresponds to our proposed method.

reported clients' execution time corresponds to the time of the *slowest* client. We trained the model 30 times and in the following figures is presented the mean values and the 95% confidence interval.

In Fig. 4 we present the execution time according to the number of centroids in the dataset. We can observe that for the strategy with privacy features, most of the work is done by the clients (see Fig. 4a). This is expected, since the clients perform local clustering followed by the encryption of the centroids. The server execution time shown in Fig. 4b is much higher without privacy features than with privacy. This result is expected since without privacy features the server needs to apply the full clustering algorithm as opposed to only computing the distances and updating the global centroids. Nevertheless, for the server side, both strategies have a rather low server execution time – below 0.1 seconds. Thus, the predominant execution time is the clients' time.

In Fig. 5 we present the execution time according to the number of clients. The strategy with privacy takes more time to execute in the client side and slowly increases with the number of clients. Specifically, its execution times are 1 to 2 seconds higher than the strategy without privacy features. This latter strategy has a higher server execution time, but the difference is low, below 0.025, and thus the total execution time is mostly affected by the clients' execution time.

We additionally measured the execution time as a function of the number of points in the dataset. Our results indicate that on both strategies, the execution time is not affected by the number of points in the dataset (between 100 and

50000). On the clients' execution time, the strategy with privacy features takes around 1 second more than the strategy without privacy features. On the server side, this latter strategy takes around 0.025 seconds more than the strategy with privacy features. We omit these plots due to the lack of space.

Overall, we consider the algorithm to be efficient. While it takes more time than the strategy without privacy features it is only significant when the datasets have many centroids. In this case, our strategy will take longer to execute.

To conclude, our proposal is robust to non-IID data while preserving the privacy of individual contributions, as the server only accesses the encrypted local centroids. Additionally, it is efficient at the expense of a slightly higher execution time, yet within reasonable bounds when compared to the centralized version. In particular, letting n_{pts}, n_{col}, n_{cent} and n_{cli} respectively represent the number of points, coordinates per point, centroids and clients, our method achieves a complexity of $O(n_{cent}^2 \times (n_{pts} \times n_{col} + n_{col}^2 + 1) + n_{pts} \times n_{col})$ for each client and $O(n_{cli} \times n_{cent}^2 + n_{cli} \times n_{cent} \times n_{col})$[4] for the server. When compared to the strategy without privacy features, with a complexity of $O(n_{pts} \times n_{cent} \times n_{col})$ for each client and $O(n_{cli} \times n_{cent}^2 \times n_{col})$ for the server, this confirms that the cost of our scheme lies in the increase of the client execution time, with the server execution time becoming even lower than for the base strategy without privacy.

4 Conclusion

In this paper, we propose a privacy preserving clustering algorithm that is private, efficient and robust to non-IID. As far as we known, there is no method that can fully fulfill the three. Our strategy based on local representatives, homomorphic encryption and secure aggregation is capable of outperforming or matching the centralized version in more than half of the datasets in terms of Adjusted Random Index in a non-IID scenario. In terms of efficiency, the time complexity moves from the server to the clients, yet leading to an overall time complexity within reasonable bounds when compared to the centralized version. Moreover, given the lack of secure metrics to evaluate models in a distributed environment, we introduced *secure inertia*, a method to compute the inertia of the model without sharing individual contributions. Our experiments show that our strategy can effectively be made practical in real world settings.

Acknowledgements

The work presented in this paper was carried out in the scope of project COPMODE, that has received funding from the European Union's Horizon 2020 research and innovation programme under the NGI_TRUST grant agreement no 825618, and the project AIDA: Adaptive, Intelligent and Distributed Assurance Platform (`POCI-01-0247-FEDER-045907`), co-financed by the European Regional Development Fund through the COMPETE2020 program and by the

[4] We assume a constant time complexity for multiplication between the encrypted centroids and the plaintext global centroids, according to [20].

Portuguese Foundation for Science and Technology (FCT) under the CMU Portugal Program. Ricardo Mendes wishes to acknowledge the Portuguese funding institution FCT - Foundation for Science and Technology for supporting his research under the Ph.D. grant SFRH/BD/128599/2017.

References

1. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1175–1191 (2017)
2. Celebi, M.E., Kingravi, H.A., Vela, P.A.: A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert systems with applications **40**(1), 200–210 (2013)
3. Clifton, C., Tassa, T.: On syntactic anonymity and differential privacy. In: 2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW). pp. 88–93. IEEE (2013)
4. Farrand, T., Mireshghallah, F., Singh, S., Trask, A.: Neither Private Nor Fair: Impact of Data Imbalance on Utility and Fairness in Differential Privacy, p. 15–19. Association for Computing Machinery (2020)
5. Graepel, T., Lauter, K., Naehrig, M.: Ml confidential: Machine learning on encrypted data. In: Information Security and Cryptology – ICISC 2012. pp. 1–21. Springer Berlin Heidelberg (2013)
6. Hu, X., Lu, L., Zhao, D., Xiang, J., Liu, X., Zhou, H., Xiong, S., Tian, J.: Privacy-preserving k-means clustering upon negative databases. In: International Conference on Neural Information Processing. pp. 191–204. Springer (2018)
7. Hubert, L., Arabie, P.: Comparing partitions. Journal of classification **2**(1), 193–218 (1985)
8. Jahangiri, A., Rakha, H.A.: Applying machine learning techniques to transportation mode recognition using mobile phone sensor data. IEEE transactions on intelligent transportation systems **16**(5), 2406–2417 (2015)
9. Januzaj, E., Kriegel, H.P., Pfeifle, M.: Towards effective and efficient distributed clustering. In: Workshop on Clustering Large Data Sets (ICDM2003). vol. 60 (2003)
10. Jiang, Z.L., Guo, N., Jin, Y., Lv, J., Wu, Y., Liu, Z., Fang, J., Yiu, S.M., Wang, X.: Efficient two-party privacy-preserving collaborative k-means clustering protocol supporting both storage and computation outsourcing. Information Sciences **518**, 168–180 (2020)
11. Liu, B., Andersen, M.S., Schaub, F., Almuhimedi, H., Zhang, S.A., Sadeh, N., Agarwal, Y., Acquisti, A.: Follow my recommendations: A personalized privacy assistant for mobile app permissions. In: Twelfth Symposium on Usable Privacy and Security (SOUPS 2016). pp. 27–41 (2016)
12. Lloyd, S.: Least squares quantization in pcm. IEEE transactions on information theory **28**(2), 129–137 (1982)
13. Lu, Z., Shen, H.: A convergent differentially private k-means clustering algorithm. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 612–624. Springer (2019)
14. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Int. Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282. PMLR (20–22 Apr 2017)

15. Navidi, W., Murphy Jr, W.S., Hereman, W.: Statistical methods in surveying by trilateration. Computational statistics & data analysis **27**(2), 209–227 (1998)
16. Palacio-Niño, J., Berzal, F.: Evaluation metrics for unsupervised learning algorithms. CoRR **abs/1905.05667** (2019), http://arxiv.org/abs/1905.05667
17. Sarker, I.H., Hoque, M.M., Uddin, M.K., Alsanoosy, T.: Mobile data science and intelligent apps: Concepts, ai-based modeling and research directions. Mobile Networks and Applications pp. 1–19 (2020)
18. Schellekens, V., Chatalic, A., Houssiau, F., De Montjoye, Y.A., Jacques, L., Gribonval, R.: Differentially private compressive k-means. In: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 7933–7937. IEEE (2019)
19. Sculley, D.: Web-scale k-means clustering. In: Proceedings of the 19th International Conference on World Wide Web. p. 1177–1178. WWW '10, Association for Computing Machinery (2010)
20. Microsoft SEAL (release 3.5). https://github.com/Microsoft/SEAL (Apr 2020), Microsoft Research, Redmond, WA. Accessed: 2020-11-26.
21. Soliman, A., Girdzijauskas, S., Bouguelia, M.R., Pashami, S., Nowaczyk, S.: Decentralized and adaptive k-means clustering for non-iid data using hyperloglog counters. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 343–355. Springer (2020)
22. Su, D., Cao, J., Li, N., Bertino, E., Jin, H.: Differentially private k-means clustering. In: Proceedings of the sixth ACM conference on data and application security and privacy. pp. 26–37. ACM (2016)
23. Thiagarajan, A., Ravindranath, L., LaCurts, K., Madden, S., Balakrishnan, H., Toledo, S., Eriksson, J.: Vtrack: Accurate, energy-aware road traffic delay estimation using mobile phones. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems. p. 85–98. SenSys '09, Association for Computing Machinery (2009)
24. Triebe, O.J., Rajagopal, R.: Federated K-Means: clustering algorithm and proof of concept (2020), online preprint: https://github.com/ourownstory/federated_kmeans/blob/master/federated_kmeans_arxiv.pdf. Accessed: 2020-11-26.
25. Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over vertically partitioned data. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 206–215. KDD '03 (2003)
26. Xing, K., Hu, C., Yu, J., Cheng, X., Zhang, F.: Mutual privacy preserving k-means clustering in social participatory sensing. IEEE Transactions on Industrial Informatics **13**(4), 2066–2076 (2017)
27. Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE Transactions on neural networks **16**(3), 645–678 (2005)
28. Yin, H., Zhang, J., Xiong, Y., Huang, X., Deng, T.: PPK-means: Achieving privacy-preserving clustering over encrypted multi-dimensional cloud data. Electronics **7**(11), 310 (2018)
29. Yuan, C., Yang, H.: Research on k-value selection method of k-means clustering algorithm. J—Multidisciplinary Scientific Journal **2**(2), 226–235 (2019)
30. Yuan, J., Tian, Y.: Practical privacy-preserving mapreduce based k-means clustering over large-scale dataset. IEEE Transactions on Cloud Computing **7**(2), 568–579 (2019)
31. Zhang, W., Li, C., Peng, G., Chen, Y., Zhang, Z.: A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load. Mechanical Systems and Signal Processing **100**, 439–453 (2018)