SPDM801 – Master's thesis in Computer Science

Calculating pairwise euclidean distance matrix for horizontally partitioned data in federated learning environment

Md Shihab Ullah
mdull20@student.sdu.dk

Supervisor:
Richard Röttger

Co-supervisor:
Tobias Frisch

Department of Mathematics and Computer Science
Faculty of Science, University of Southern Denmark, Campus Odense

Contents

1 Abstract

Federated Learning has become popular as a privacy-preserving machine learning solution that encapsulates data but shares model parameters accordingly. Many machine learning algorithms have to perform pairwise euclidean distance calculations between all the data points among horizontally partitioned datasets. However, to ensure data privacy, most of the existing solutions embed encryption or cryptographic techniques so that the data points are shared and computed securely. In this paper, two approaches have been proposed to create two Euclidean distance matrices (FEDM and PEDM) from the pairwise euclidean distances between horizontally-partitioned data for ensuring "privacy by design". For creating FEDM, we have introduced the usage of artificial spike points. For creating PEDM, we perform regression between local euclidean distances of data points and FEDM, respectively. We have found that there can be significantly low loss of accuracy and highly positive similar distance matrix *w.r.t* actual euclidean distance matrix can be constructed using those approaches. As a result, it has enormous potential to change not only how different machine learning algorithm which requires euclidean distance calculation will be executed internally but also eliminate the need for separate integration of encryption techniques for ensuring privacy which can be fruitful in reduction of overall communication and computation cost in the federated learning environment. We have explained the underlying workflows of the method and construction of the matrix and clarified their performance and experimental outcome. We have also shown that PEDM is reliable and demonstrate a low loss of accuracy to be considered as aggregated euclidean distance matrix for many euclidean distance-related machine learning applications.

Keywords: horizontal federated learning, euclidean distance, federated distance calculation, privacy preserving clustering, pearson correlation coefficient, spearman correlation coefficient, horizontally-partitioned data, predict euclidean distance, aggregated euclidean distance matrix, spike points, regression, principal component analysis

Dansk resumé

Fødereret læring er blevet en populær machinelearning løsning, der bevarer privatlivets fred, da den indkapsler data, men stadig deler modelparametrene i overensstemmelse hermed. Mange machinelearning algoritmer skal udføre parvise euklidiske afstandsberegninger mellem alle datapunkter blandt vandret opdelte datasæt. For at sikre databeskyttelse indlejrer de fleste af de eksisterende løsninger imidlertid kryptering eller kryptografiske teknikker, så datapunkterne deles og beregnes sikkert. I denne afhandling er der blevet foreslået to tilgange til at skabe to euklidiske afstandsmatricer (FEDM og PEDM) ud fra de parvise euklidiske afstande mellem horisontalt opdelte data for at sikre "privacy by design". For at skabe FEDM introduceres brugen af kunstige spidspunkter. For at skabe PEDM udføres regression mellem henholdsvis lokale euklidiske afstande af datapunkter og FEDM. Det er opdaget, at der kan være et betydeligt lavt tab af nøjagtighed, og meget positiv lignende afstandsmatrix, uden at den faktiske euklidiske afstandsmatrix kan konstrueres ved hjælp af disse tilgange. Som et resultat heraf har det et enormt potentiale til ikke kun at ændre, hvordan forskellige machinelearning algoritmer, som kræver euklidisk afstandsberegning, vil blive udført internt, men det vil også kunne eliminere behovet for separat integration af krypteringsteknikker for at sikre privatlivets fred, hvilket kan være frugtbart i reduktion af de samlede kommunikations og beregningsomkostninger i det fødererede læringsmiljø. Heri er forklaret de underliggende arbejdsgange for metoden og konstruktionen af matrixen samt præciseret deres præstation og eksperimentelle resultat. Det er også vist, at PEDM er pålideligt og viser et lavt tab af nøjagtighed for at blive betragtet som aggregeret euklidsk afstandsmatrix for mange euklidiske afstandsrelaterede machinelearning applikationer.

2 Introduction

In today's world, the application of machine learning techniques and algorithms has proven to add realistic values and solutions in all spectrum of fields and aspects of life. If broadly classified, unsupervised machine learning algorithms are one of the major categories where the training data consists of input without any corresponding target values or labelling.

In traditional machine learning, data from different sites need to be sent and aggregated in a centralised location for training models which is highly vulnerable to data breaches and leakage in many ways. Furthermore, it may break many data privacy policies and related laws from many countries like GDPR (European Union), LGPD (Brazil), PDPA (Thailand), PDPB (India), PDPL (China), Digital Charter Implementation Act (Canada) etc. [1], which can be a major challenge for useful machine learning implementation.

Due to above mentioned reasons, privacy-preserving machine learning solutions like federated machine learning can be used where the machine learning algorithms are trained by using multiple local datasets to create a shared global model without exchanging training dataset in a central location. Consequently, allows personal data to remain in local sites, reducing the possibility of personal data breaches and also helping to provide personalized solutions by preserving data privacy.

One of the commonly used distance metrics in many supervised or unsupervised machine learning algorithms is known as Euclidean distance. In mathematics, the Euclidean distance between two points in Euclidean space is the length of a line segment between the two points. It can be calculated from the Cartesian coordinates of the points using the Pythagorean theorem, therefore occasionally being called the Pythagorean distance [2]. There are a variety of applications for Euclidean distance between real-valued vectors, such as nearest-neighbor search, clustering, numerical linear algebra etc.

However, the data points are exposed when calculating the euclidean distance of two points belonging in different dataset which makes them losing privacy and being highly vulnerable to data leakage. In order to impose data privacy, there are many privacy-preserving techniques have been introduced. For example, some of the encryption technique such as Secure Multi-Party Computation, Homomorphic encryption, Arithmetic Secret Sharing along with perturbation methods like Differential Privacy etc to basically provide security or privacy of the inputs i.e. dataset using encryption or perturbation techniques. Differential Privacy technique can be look as a trade-off

system between data accuracy and privacy where a system relying on this technique must sacrifice privacy for higher accuracy and vice-versa [3]. It tends to work well for large datasets where the tradeoff of adding noise to ensure data privacy against accuracy can be sustainable. Making an encryption scheme fully homomorphic is conceptually straightforward but securing it has been the hard part. Homomorphic Encryption comes with severe limitation of speed which cannot be still avoided. It can be setup with weaker encryption that might result into data leakage on attack [4].

Secure Multi-Party Computation (SMPC) introduces additional architectural components and complex computational process as well as it demands that all participants must be online at the same time which may be difficult in many environments. Considering these cases, Secure Multi-Party Computation can be expensive and less feasible for higher number of participants having datasets [5].

Nevertheless, all these above mentioned privacy preserving techniques creates communication, computational and network overhead while losing considerable amount of accuracy. Hence, two approaches to construct euclidean distance matrix between dataset has been proposed namely Federated Euclidean Distance Matrix (FEDM) and Predicted Euclidean Distance Matrix (PEDM) which ensure complete privacy of the data points among horizontally-partitioned dataset without sacrificing a high amount accuracy or embedding any separate data privacy mechanism. Privacy is ensured by design during the construction of the distance matrix while in most of cases the loss of accuracy of pairwise euclidean distances between data points are minimum.

3 Scientific background

3.1 Euclidean Distance

In general, for points given by Cartesian coordinates in **n**-dimensional space, the euclidean distance is:

$$\mathrm{d}(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

The Euclidean distance is the prototypical example of the distance in a metric space, and obeys all the defining properties of a metric space

It is ***symmetric***, meaning that for all points p and q, $d(p,q) = d(q,p) d(p,q) = d(q,p)$ i.e. no matter which point is consider as starting point or destination, the euclidean distance of two points will always be same.

It is ***positive***, which means the distance between two points cannot be negative or less than zero.

It obeys the ***triangle inequality***: for every three points p, q, and r, $d(p,q) + d(q,r) \geq d(p,r)$. Intuitively, distance between p and r via q cannot be any shorter than distance between p and r [6, 2].

3.2 Correlation Coefficients

In statistical terminology, correlation is a way of measuring association relationship between two continuous variables i.e. how the two variables can vary within a certain range regardless of the dataset distribution. Even though its quite loosely used in oral communication, it generally refers to any form of relevancy, connection or linkage between two variables how closely they co-vary. Correlation coefficient is a scalar quantity that takes a value in the range 1 to +1 to measure the correlation between two variables, which represents the strength of the presumed linear association between the variables [7, 8].

Depending on the magnitude and sign of the coefficient value, one can interpret how much two variables are directly or inversely related. The direction of the relationship is indicated by the sign of the coefficient; a + sign indicates a positive relationship and a – sign indicates a negative relationship. For example, if the value is 0.75 it means both variables have a strong direct or positive linear relationship while -0.75 will mean a strong inverse or negative relationship respectively [7].

Following table shows a rule of thumb to interpret correlation coefficient values:

Table 1: Rule of Thumb for Interpreting the Size of a Correlation Coefficient

Correlation Coefficient Value	**Interpretation**
(0.90 to 1.00) or (0.90 to 1.00)	Very high positive or negative correlation
(0.70 to 0.90) or (0.70 to 0.90)	High positive or negative correlation
(0.50 to 0.70) or (0.50 to 0.70)	Moderate positive or negative correlation
(0.30 to 0.50) or (0.30 to 0.50)	Low positive or negative correlation
(0.00 to 0.30) or (0.00 to 0.30)	Negligible correlation

3.2.1 Types of Correlation Coefficients

There are two main types of correlation coefficients: parametric and non-parametric. Parametric correlation can be measured by Pearson's product moment correlation coefficient and non-parametric one can be measured by Spearman's rank correlation coefficient respectively. Depending on the type of distribution of the variables, each type can project different value [7].

There are other types such as Covariance, Kendall and polychoric or tetrachoric correlation coefficients that can be relate to either Pearson or Spearman. They are often used when multiple variables are being considered [9].

3.2.2 Pearson's product moment correlation coefficient

Pearson's product moment correlation coefficient (or Pearson correlation coefficient, for short) is denoted as (ϱ) for whole population and as (r) for a sample. Firmness of linear association between two variables can be quantified by it. In essence, it aims to draw a best-fitted line through the data of two variables while stipulating how far away all these data points are to this line of best fit (i.e., how well the data points fit this new model/line of best fit). The formula for calculating the sample Pearson's correlation coefficient to get the correlation between variables x and y is given by:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where x_i and y_i are the pair of values of x and y for the ith individual.

It is used when both variables being studied are consider to be normally distributed. This coefficient is affected by extreme values, which may exaggerate or dampen the

strength of relationship, and is therefore inappropriate when either or both variables are not normally distributed [7, 10].

As we measured euclidean distance matrices denoted as FEDM and PEDM which are both pairwise distance matrix i.e similar type of values as they are both measuring the distances between the same data points, we tried to use Pearson's correlation coefficient to find linear similarities between them.

3.2.3 Spearman's rank correlation coefficient

Related to the Pearson correlation coefficient, the Spearman's correlation coefficient (ρ) is also known as the rank-based Pearson correlation coefficient that measures the relationship between two variables where the variables are not normal-distributed and have a non-linear relationship.

Its application is not only restricted to continuous data, but can also be used in analyses of ordinal attributes. It is suitable when one or both variables are skewed or ordinal and is robust when extreme values are present for variables i.e outlier(s). For a correlation between variables x and y, the formula for calculating the sample Spearman's correlation coefficient is given by:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where, d = the pairwise distances of the ranks of the independent variables x_i and y_i and n = the number of samples.

3.3 Federated Learning

Federated learning (also known as collaborative learning) is an approach used in machine learning where an ML model gets decentralized training in multiple device without having access to data of each device [11]. Standard machine learning model requires centralizing training data into a common server which has some down sides such as no privacy of the data and capacity of the client device etc. Federated learning stands in contrast to traditional centralized machine learning techniques where all the local dataset are uploaded to one server, as well as to more classical decentralized approaches which often assume that local data samples are identically distributed. The core idea is to enforce decentralized learning where the data are never send to the

server. After the initial deployment of pre-trained model, each client device independently train its own model using its own data which can reduce the communication overhead during the training process. To maintain the goodness of decentralized data and its knowledge, each client shares its model weights and parameter to the server which the server uses to retrain its existing global model and resend them to the clients after training. Depending on the policy, the server schedules the model update.

The idea of federated learning was originally introduced by Google in 2016[12, 13, 14]. Google's key objective was to train models in individual device and combined them meaningfully without losing privacy. Recently many improvisation has been proposed which overcomes various statistical challenges [15, 16] and enforcing more safety [17, 18] in federated learning environment. There are also experimental efforts to bring out personalized federated or collaborative learning [19, 15]. The above study give focus to on-device federated learning and how to optimize communication cost, improve device data security while building the model along with tackling unbalanced distribution of different types of data so that model can perform better in general. In addition, data is partitioned by user Ids or device Ids, therefore, horizontally in the data space [20].

Federated learning enables multiple participants to use and benefit from a common, robust machine learning model without sharing data, thus addressing censorious cases like data privacy and safety, data accessibility level and heterogeneous data etc. Its applications are exponentially emerging and spreading over many industries including bioinformatics, defense, telecom infrastructure, Internet of Things(IoT), IT Enterprise and Healthcare [11].

3.3.1 Privacy in Federated Learning

Despite its apparent promise, Federated Learning suffers from several weaknesses. Although model parameters are shared, however, by modifying the global model parameters, an attacker with privileged access can easily obtain the training data and compromise the privacy of individuals [21]. As exchanging parameters of training models can lead to sensitive information being revealed, there are data privacy mechanisms available to tackle data privacy concerns in federated learning environment.

Differential privacy is a way for publicly sharing statistical analysis or information while protecting individual data which is achieved by injecting small amount of random noise into statistical computation to obscure the effect of each useful statistical signal to come through so that if an adversary is interacting with the system to learn about a data record, it would not learn much about the individual giving at least scope

for plausible deniability. This technique prevents leaking information about any particular data record everytime any related outcome of the model suddenly becomes likely or unlikely whether that particular data record is added or removed for the dataset. For example: Lets say we want to query if Alice have diabetes or not from a ML model. If Alice's record is present in the training data than assume we get the probability of 0.85 while without that we get the probability of 0.58. Thus, model reveals significant level of information when Alice's record is present as training data. If the ML model was differentially private it will apply certain threshold (ε) such that if any one query to know whether Alice has diabetes or not it will make similar prediction whether her record is included or excluded. The privacy budget i.e the threshold (ε) controls how much learning variance will considered. If the threshold is high, it means it will accept high variance resulting the fact that the model is more accurate and learning else if the threshold is low, it will mean that the model has high privacy but it will barely learn anything i.e. will have less accuracy.

Consequently, one cannot derive any individual data whether the data is included or excluded from a differentially private model or algorithm . In other words,there is assurance that anything the model might output on a database containing some individual's information is almost as likely to have come from a database with or without that individual's information. Most notably, this guarantee holds formally for any individual and any dataset. Differential Privacy has usefulness in many fields other than machine learning such as game theory and economic mechanism design, statistical estimation, and streaming etc. It is worth remarking that the technique usually works better on larger databases because as the number of records in a database grows, the effect of any single records on a given aggregate statistical output declines [22].

Homomorphic Encryption differs from typical encryption methods in that it allows computation to be performed directly on encrypted data without requiring access to a secret key. The result of such a computation remains in encrypted form, and can at a later point be revealed by the owner of the secret key. The security of the most practical homomorphic encryption schemes is based on the Ring-Learning With Errors (RLWE) problem, which is a hard mathematical problem related to high-dimensional lattices. Namely, the security assumption of these encryption schemes states that if the scheme can be broken efficiently, then the RLWE problem can be solved efficiently [23].

For example, predictive analytics in health care can be hard to apply via a third party service provider due to medical data privacy concerns, but if the predictive analytics service provider can operate on encrypted data instead, these privacy concerns are diminished. Moreover, even if the service provider's system is compromised, the data

would remain secure [24].

In case of **Secure Multi-Party Computation (SMC)**, a given number of participants or parties, p_1, p_2, ..., p_n, each have private data d_1, d_2, ..., d_n calculate the value of a public function on the data each of them will provide: $F(d_1, d_2, ..., d_n)$ without sharing or revealing their own data to other parties [25]. One of the common example to illustrate Secure Multi-Party Computation is the millionaire's problem i.e. a bunch of millionaire wants to know who's the richest without revealing their net worth. SMC allows the parties to share their respective details of wealth by masking them as a function. SMC has come a long way since its origin in 80s and now have been used in different industrial way such as sharing cryptographic keys between parties across the cloud or calculating winning bid and bidder for the auction without revealing bids and bidder respectively.

SMC security models tends to provide security in a well-defined communication architecture to guarantee complete zero knowledge meaning that each party only about its own input and the system output. However, this desired property usually bring complexity in computation protocol and become tough to achieve completely. In specific cases, partial knowledge exposure may be acceptable if security guarantees are provided. Furthermore, it is possible to build a model comprised with SMC having lower security requirements for exchange for efficiency [26, 20].

Over the past three decades, many different techniques have been developed for constructing MPC protocols with different properties, and for different settings such as Shamir Secret Sharing, Honest-Majority MPC with Secret Sharing, Threshold Cryptography etc [27]. It is less complex and computationally expensive than Homomorphic Encryption

Generally, the two important requirements on any secure computation protocol are *privacy* and *correctness*. The privacy requirement demands that nothing should be learned or exposed except the desired output information. The correctness requirement states that the adversary must not be able to cause the result of the computation to deviate from the function that the parties had set out to compute ensuring each parties receive the correct output [27].

Other security properties which are considered are as follows [27]:

1. All concerned parties should be able to give input independently without having any noise or influence

2. The adversary should not be able to disrupt the computation or temper with it

by carrying out a "denial of service" attack.

3. Parties should be treated fairly i.e. all parties should receive same outputs.

3.3.2 Categories

Federated learning can be classified into horizontal federated learning, vertical federated learning, and federated transfer learning based on how data is partitioned distributed among various parties in the feature and sample space along with the distribution characteristics of the data. Following figure projects various federated learning frameworks for only two-party scenario:

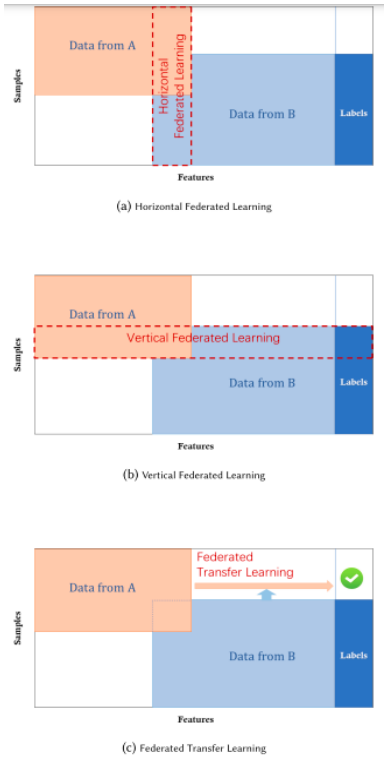

Figure 1: Categorization of federated learning [20]

In **Horizontal Federated Learning**, datasets among the different participants or entities have identical sets of features which is used to independently train a common model situated in centralized server without sharing the data. Consequently, it is also referred as sample-based or homogeneous federated learning.

For example, there are number of hospital which want to have a centralized model that can make common diagnosis but has restrictions or privacy concern for sharing same types of data with each other. As a result, the hospitals collaboratively learn a shared prediction model while keeping all training data decentralized. In 2017, Google proposed a horizontal federated-learning solution for Android phone model updates

where every mobile devices are trained locally and independently while sharing only subsets of updates of parameters to the centralized model server [20]. We have considered horizontal federated learning architecture for creating and calculating FEDM and PEDM.

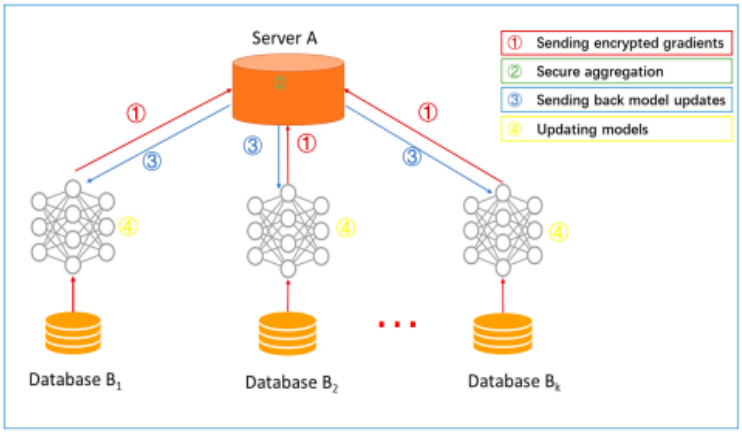

Figure 2: System architecture for horizontal federated learning [20]

In case of **Vertical Federated Learning** or feature-based federated learning, model(s) are trained on the dataset among different entities have dissimilar sets of feature i.e the data between the participants does not have same features or attributes of their records. Data of the same user or ID in different participants are securely aligned between the databases by technique called Private Set Interaction [28] which is a secure multi-party computation that allows parties to compute and only expose the intersection of sets (i.e. the ID of each data records) without revealing party's individual sets. Private Set Union can further increase privacy without even revealing the intersected values [29]. Vertical federated learning is the process of aggregating different features of same sample and performing model training by preserving privacy to collaboratively build a learning model with data from all parties. There are two major methods of model training: SplitNN, FedBCD where both requires engaging parties to share intermediate results in order to update model parameters using back propagation keeping their data private to each other [30].

For example, a privacy-preserving prediction model for product recommendation considers dataset of two different entities (bank and e-commerce platform) in same geographical location where both of their sets are likely to contain most of the residents of that location. However, each entities holds different information for the same user/ records. The dataset of the bank may contain the user's revenue, expenditure behavior and credit rating whereas the e-commerce retains the user's browsing and purchasing history [20].

Unlike horizontal and vertical federated learning, **Federated transfer learning (FTL)** is a showcase of performing transfer learning between parties when their datasets vary both in feature and sample space and small amount of feature or sample space is overlapped in between parties that is used to learn a local model and transfer it accordingly. Transfer learning allows the model to be trained and tested even the distribution, features and samples of different parties can diverge. It plans to create robust model for the target domain while leveraging knowledge from the other (source) domains. A typical architecture of federated transfer learning is shown in Figure 1(c).

Considering two parties A and B, where there is only a small overlap in feature space and sample space between A and B, a model learned on B is transferred to A by leveraging small overlapping data and features. We recall that horizontal federated learning is used when there is large overlap in the feature space between datasets and vertical federated learning is used when there is large overlap in user/sample space between datasets. In contrast to them, FTL is used when there is small overlap in both feature space and sample space (shown by dotted box in Figure 1(c)). FTL ingests a model trained on source domain samples and feature space.

Subsequently, FTL orients the model for reuse in target space such that model is used for non-overlapping samples leveraging the knowledge acquired from source domain non-overlapping features. Thus FTL covers the region in right upper corner of the Figure 1(c) by transferring knowledge from non-overlapping features from source domain to the new samples in the target domain. The ability to use the transferred knowledge on non-overlapping data in A makes FTL different from vertical transfer learning [31].

For example: Lets consider two institutions where one is China-based banking institution while the other is an e-commerce company located in the United States of America. Owing to geographical variance and legislature restrictions, the user groups of the two institutions have small overlapping sample space. On the other hand, owing to the difference in business model and nature of customer, minuscule portion of the feature space from both parties overlaps.As a result of the federated model, transfer-learning techniques can provide solutions for the entire sample space and feature space. Specifically, a federated transfer model learns a common representation used to portray feature spaces of different parties through a limited number of common samples and then used to predict for samples with one-sided features [20].

4 Related Work

Various privacy-preserving techniques or dataset transformations have been proposed in either federated or non-federated learning environment to mostly preserve the privacy of data points. Nevertheless, all of them ultimately proposed to use the actual or noised data points for calculating aggregated euclidean distance. To the best of my knowledge, the approaches offered in this paper for calculating aggregated euclidean distance matrix in federated setting are one of the first considering "privacy by design" without using or sending datasets of multiple data owners directly or indirectly.

4.1 Literature Review

Pandya in [32] considered adding random noises to dataset while minimizing information loss and create perturbed dataset based on orthogonal transformation where the participant generates an orthogonal matrix (a matrix with independent Gaussian random entries that uses QR, SVD, spectral or polar decomposition etc) and multiply with respective original dataset points. For each participants, the pairwise euclidean derived from the perturbed dataset are same as that of original dataset.

However, the study does not propose any solution for constructing aggregated euclidean distance matrix for combined dataset for multiple participant in federated learning settings so that pairwise distance of point AD1 and AD2 from dataset D1 and D2 can be calculated. Furthermore, sending pairwise euclidean distance matrix of perturbed dataset for individual participants to the central coordinator does not have substantial benefit privacy-wise rather than extra computational overhead in getting aggregated euclidean distance between AD1 and AD2.

Mohassel et al. in section 4.2 of [33], for having better computational efficiency, contemplated secured computation of squared euclidean distance (which is not a metric as it does not satisfy triangle inequality) instead of the square root of the sum of the squares of the differences between two points in each dimension because it does not affect the output of the algorithms using euclidean distance. Authors wanted to compute the secure Euclidean squared distance by which both parties obtain the arithmetic shared value of the output assuming both parties have arithmetic secret shared value of two points have same dimension. Among the three terms, two of them can be computed locally while the other one can be calculated by secured multiplication and an optimized solution is proposed to compute the mixed term in the amortized setting.

Furthermore, it can be inspected that after certain number of iteration the calculation is executed and direct approach are suggested to create aggregated distance matrix for clustering evaluation. Consequently, the above-mentioned method of sharing euclidean distance primarily relies on cryptographic methods which can introduce additional communication and computational overhead respectively.

In 2004, Ravikumar et al. [34] mentioned about using a secure intersection protocol [35] for secure computation of string distance matrices as euclidean distance between two feature vectors can also be expressed as a scalar product of two vectors and can be used to obtain the distance securely via computing the dot product of each and every data pair respectively. In contrast, many of the secure dot product algorithms are either unreliable or poorly suited for processing large amounts of data.

For example, the communication cost increases rapidly as the number of data items at each party increases. Furthermore, parallelizing the dot product calculations could negatively impact security. The stochastic estimates are used to compute the dot products in [34] which produces inaccurate results when the data size is not large enough (i.e., < 1000). Many medical datasets can have low number samples for example GSE84426 and GSE84433 contains 76 and 357 samples only.

Recently, Stausholm et al in [36] shows that, based on the Sparser Johnson-Lindenstrauss constructions by Kane and Nelson, their technique offers better privacy, efficiency, and accuracy when estimating the euclidean distance between two vectors when the construction is combined with either the Laplace or the Gaussian mechanism depending on the privacy parameters ε and δ.

Lee et al in [37] suggested method to calculate euclidean distance for vertically-partitioned data i.e where two same points can belong to different dataset where each dataset do not contains all their feature space. Authors stated that combining Euclidean distances from two different spaces does not equal to concatenating these two spaces and then computing the Euclidean distances in this space, i.e.

$$\sqrt{\sum_{i=1}^{k}(q_i - p_i)^2} + \sqrt{\sum_{i=k+1}^{n}(q_i - p_i)^2} \neq \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{1}$$

It is showed that Euclidean distances that come from two different spaces can be combined by first calculating the Euclidean distances in the concatenated space, then cross-referencing the distances from the feature spaces, i.e,

$$\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2 + \sqrt{\sum_{i=1}^{k}(q_i - p_i)^2 + \sum_{i=k+1}^{n}(q_i - p_i)^2}} \tag{2}$$

However, this might not be applicable in vertical federated learning without imposing encryption technique because one must know the pairwise euclidean distances of all the feature space in order to assimilate the overall distance. Moreover, concerns of privacy of data points were out of the span of research.

Using a Fourier transform, a technique is proposed by Mukherjee et al [38] to guarantee that, before and after the transformation, the distance between two points remains the same. This preserves data privacy while keeping its statistical properties intact. Authors strongly agree with one of their experimental outcomes i.e., for real-life scenarios, it is sufficient to take the first few high-energy coefficients from the Fourier transform of the entire dataset so that a few high-energy coefficients are also considered for every data point.

Permutations or rearrangement of these coefficients may further increase privacy. Nevertheless, it remains difficult to set a distance difference threshold for non-uniform or unknown dataset distributions. Although the paper evaluates the approach using k-means clustering, it does not guarantee any utility.

To be mentioned, there are many proposed approach in [39, 40, 41, 42] to calculate Euclidean Distance Matrix (EDM) between a single dataset D1 has n rows and m columns whether its square *(n = m)* or rectangular *(n != m)* considering parallel calculation in GPU or CPU by dividing data into chunks or submatrices respectively. However, these algorithms are not applicable for multiple dataset while data privacy concerns were not naturally considered as one of their research scope. Furthermore, all the proposed computational techniques were either distributed or parallelized. Overall, the principle focus of most of these studies was to increase either computational or memory efficiency to do large EDM calculation [43, 44].

5 Methods and Datasets

Unlike in many approaches mentioned in the related work section, no cryptographic technique such as Secure Multi-Party Computation, Differential Privacy or secret sharing techniques like Arithmetic secret sharing was used to find pairwise euclidean distance between the dataset. As a result, the pairwise euclidean distances of the dataset of N number of participant was considered for getting two types of aggregated distances in the federated setting, known as the Federated Euclidean Distance Matrix (FEDM) and Predicted Euclidean Distance Matrix (PEDM) respectively. Both FEDM and PEDM are (N x N) matrices where N is the total number of samples in all the participant.

Federated Euclidean Distance Matrix (FEDM) is calculated by adding random points in each participants dataset (referred in this paper as spike points) then creating pairwise distance matrices referred as Local Spike Distance Matrices(LSDMs) between the spike points and data points in each participant's end which are shared to the coordinator after concatenating LSDMs and returning the pairwise distances of between LDSMs. For each participants, LSDM is a (S x N) matrix where S is the total number of spike points and N is the total number of data samples available that participant.

In order to have some correction and increase the correlation of FEDM w.r.t aggregated or true distance matrix, the calculation of Predicted Euclidean Distance Matrix (PEDM) is proposed. One of the regressors such as Linear, Huber, Theil Sen are used to perform regression between the true pairwise distance matrix and the LSDM of each participant's dataset. The coefficients and the intercept derived from the regression performed in each participant's end are shared to the coordinator's end to construct PEDM by multiplying respective coefficients with each value FEDM and adding the intercept with it.

In overall, both FEDM and PEDM are constructed in coordinator's end which avoids both computational and network communication overhead for the participant(s).

The following subsections contains detailed explanation of how the matrices and the spike points generated. Also the evaluation process of FEDM and PEDM w,r.t the aggregated distance matrix and its reasoning are discussed for convenience.

5.1 Assumption

Some of the assumption we considered while designing and developing our respective methods are as follows:

1. Both participant(s) and coordinator are essentially computational resource(s)
2. There can be N number of participants and only one coordinator. The coordinator itself can act or function as a participant
3. Each participant will share information of only one dataset for each operation of finding distance matrices
4. Data points of all the participant's dataset are vectors and will have the same number of dimension
5. The number of spike points will not be greater than that of dimension

5.2 Spike Points Generation Methods

Participant can explicitly mention the number of spike points to be generated. If not, the proposed methods will generate the points based on its given parameter of reducing or increasing the number of spike points.

5.2.1 Overview

Spike points are artificial non-existing data points which are basically centroids or uniformly distributed random sample or combination of both. They are considered as the common points for all participant's dataset.

Lets assume $S_1...S_N$ are spike in points where $A_1...A_N \rightarrow \mathbb{R}$ are data points for participant P_1 and $B_1...B_N \rightarrow \mathbb{R}$ are data points for participant P_2 respectively.

Federated Euclidean Distance Matrix (FEDM) is basically a meta distance which can be denoted as:

euclidean_dist(euclidean_dist($A_1...A_N$,$S_1...S_N$),euclidean_dist($B_1...B_N$,$S_1...S_N$))

Distance between the distances of spike points with data points of two different datasets can provide approximation of true euclidean distance. Privacy of the data points are

preserved because the total number of spike points will always be less than the dimension of the data points.

In case of PEDM, the derived coefficient and intercept are used with each values of FEDM(which is itself a distance matrix).

Following subsections discuss three methods proposed for calculating and generating spike points

5.2.2 Centroid-based spike points

In this approach, we consider the given centroid(s) of each participant's dataset as spike points. The participant performs K-mean or any other clustering algorithms like Affinity Propagation over their respective dataset which may or may not require the number of cluster (k). Each participant shares their centroid(s) to the coordinator which then broadcast all of the centroids derived from all the participant as the final array of spike points to initial calculation of FEDM and PEDM respectively.

Algorithm 1: Generate spike points based on centroid(s) of each participant's dataset

Data: number_of_participants, number_of_cluster(optional)
Result: spike_point (Array)

```
initialization of spike_point [];
for each number_of_participants do
    perform K-means with k=number_of_cluster in participant's end
    return the cluster centers to coordinator
    append the cluster centers to spike_point (Array)
return spike_point (Array)
```

5.2.3 Uniformly distributed random spike points

Each participants will perform PCA on their dataset locally and based on the dataset of reduced dimension, it get uniformly distributed random spike points over the half-open interval where the lowest range include the lowest value of all the features of all samples and the highest range includes that of all the samples as its local spike points. After that, the spike points are inversely transformed back to its original magnitude of feature space and sent to the coordinator respectively. Upon receiving spike points generated by all participant in the system, the coordinator will append and then broadcast them to all the participant for creating FEDM

Algorithm 2: Generate spike points from each participants using PCA covering desired variance

Data: variance, dataset, no_of_spikes, reduce= 1, induce= 1
Result: generated_spikes

```
initialization of generated_spikes [];
dataset_pca = perform_PCA(variance, dataset)
generated_spikes = get_uniform_samples(low=min(dataset_pca),
  high=max(dataset_pca), size=(no_of_spikes, dimension_of_dataset_pca))
inverse_transform(generated_spikes)
return generated_spikes []
```

Algorithm 3: Generate spike points based on uniform random samples using each participant's dataset

Data: number_of_participants, generated_spikes_of_each_participants
Result: generated_spikes

```
initialization of generated_spikes [];
for each number_of_participants do
    perform Algorithm 2
    get the uniformly distributed random samples to coordinator
    append the random samples to generated_spikes []
return generated_spikes []
```

5.2.4 Random samples spike points with centroids

Each participant will create uniform random spike points based on Algorithm 2 along with centroids based on Algorithm 1. After that, the participant will append all of random samples and centroids as spike points for sharing it to the coordinator.

After receiving them from all the participants, the coordinator can either return a specific amount of the points or all the spike points respectively.

Algorithm 4: Generate spike points from each participants using PCA covering desired variance along with its centroid(s)

Data: variance, dataset, centroids, no_of_spikes, reduce= 1, induce= 1
Result: generated_spikes_with_centroids

```
initialization of generated_spikes_with_centroids [];
dataset_pca = perform_PCA(variance, dataset)
generated_spikes = get_uniform_samples(low=min(dataset_pca),
  high=max(dataset_pca), size=(no_of_spikes, dimension_of_dataset_pca))
inverse_transform(generated_spikes)
save the random samples in generated_spikes_with_centroids []
concatenate the centroids with generated_spikes_with_centroids []
return generated_spikes_with_centroids []
```

Algorithm 5: Generate spike points based on uniform random samples using each participant's dataset and its centroid(s)

Data: number_of_participants, generated_spikes_of_each_participants
Result: generated_spikes_for_all_participants

```
initialization of generated_spikes [];
for each number_of_participants do
    perform Algorithm 4
    get the uniformly distributed random samples to coordinator
    append the random samples to generated_spikes []
return generated_spikes_for_all_participants []
```

5.3 Federated Euclidean Distance Matrix

One of the aggregated distance matrix we can obtain from euclidean distance provided by different participant(s) is Federated Euclidean Distance Matrix (FEDM). It can be consider as the concatenated or combined euclidean distances of the pairwise distances between all the data points and the spike points of all the participant(s).

The steps involved in construction of FEDM are mentioned in order as follows:

1. For each participant(s), we generate spike points of their dataset(s) based on any methods. Different method can be used by each participant.
2. Share the generated spike-in points for all the participant(s) to the coordinator along with the size of their respective dataset(s)
3. The coordinator concatenate all the generated spike-in points received from each participant and broadcast the points to all participant(s)
4. Each participant(s) construct a pairwise distance matrix between their data points and the spike points which can be denoted as Local Spike Distance Matrices(LSDMs)
5. The coordinator concatenates all the LSDM(s) from each participant(s)
6. The coordinator construct FEDM by getting pairwise euclidean distance between each points of the concatenated LSDM(s) shared by all participant(s)

5.4 Predicted Euclidean Distance Matrix

We propose another euclidean distance matrix calculation known as Predicted Euclidean Distance Matrix (PEDM) from the euclidean distance provided by different participant(s) which minimize error in value of the pairwise euclidean distances between points in FEDM using regression model

The steps involved in construction of PEDM are mentioned in order as follows:

1. For each participant(s), calculate the pairwise euclidean distances i.e Local Distance Matrix (LDM) between its data points
2. Each participant(s) construct a pairwise distance matrix between their data points and the spike points, denoted as Local Spike Distance Matrices(LSDMs)

3. Participant compute the pairwise distance between each value of it's LSDM can be denoted as LSDM'

4. From the participant's end, perform Linear or Huber regression where LSDM' is the independent training data (x) and LDM is the dependent target value (y) respectively

5. Each participant shares the derived coefficient (M) and intercept (C) to the coordinator after performing regression locally

6. Construct PEDM using all the coefficients (M_n) and intercept (C_n) of N number of participant(s) for the corresponding value in FEDM

7. For calculating pairwise predicted distance $D1_AD2_B$ where A and B are two data points of different participant's dataset D1 and D2, the coordinator uses the mean($\overline{M}$) of all coefficients and intercept as to the respective value in FEDM

In Figure 3, generic systematic flow for constructing FEDM and PEDM can be seen in details.

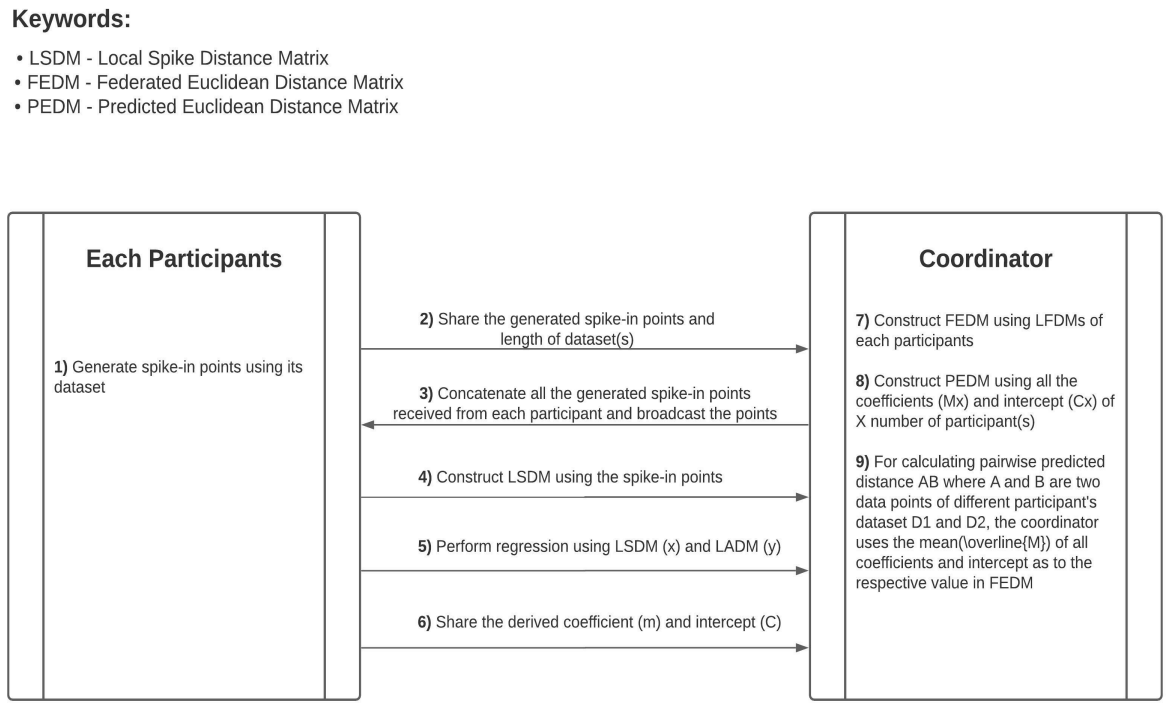

Figure 3: Visualizing Federated and Predicted Euclidean Distance Matrix calculation *w.r.t* the participant(s) and the coordinator

5.5 Datasets

For experimental and evaluation purpose, two types of datasets are used. They are artificial blobs of data points and gene expression dataset respectively.

5.5.1 Artificial datasets

For experimentation and analysis, we used two types of independent artificial datasets:

- Datasets which were isotropic Gaussian blobs which are clustered and the data points are non-uniformly distributed random samples respectively [45].
- Datasets where the data points are distributed uniformly at random [46].

Table 2: Types of the artificial datasets used along with their respective size of samples and dimensions

No# of Samples	Dimension	Type
100	1200	non-uniform
5000	10000	non-uniform
5000	50000	non-uniform
1200	1000	uniform
5000	100	uniform
5000	10000	uniform
5000	20000	uniform

5.5.2 Gene Expression Datasets

High-dimensional gene expression datasets available in the public domain usually requires efficient prediction and identification methods as it is challenging to convey accurate gene identification. Euclidean distance function is heavily utilized by many methods and used on such types of datasets [47]. For fair comparison, we use two labelled datasets where both of them were used for clustering evaluation in [48]. Both gene expression profiles have been considered to calculate both federated and predicted euclidean distance matrices for the respective dataset. It is possible to explain a large portion of the overall variation just by two principal components giving us a

simple overview of the data. In this particular case, we perform PCA upto first two principle components in order to transform the dataset, reduce it into two and three dimensional data while capturing as much as variance as possible for visualization purpose.

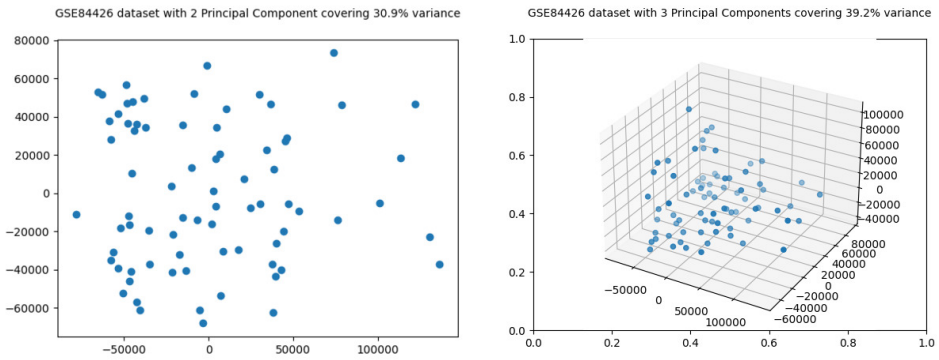

Figure 4: Visualization of first two principle components (PC) of GSE84426 dataset. The first two PC covers 30.9% and 39.2% variance of the overall dataset

Firstly, the gene expression profile of gastrectomy samples from 76 gastric cancer patients by HumanHT-12 v3.0 Expression BeadChip array (Illumina) known as GSE84426. Total RNA was extracted from the fresh-frozen gastrectomy specimens at the Yonsei University Severance Hospital (South Korea) between 2000 and 2010 [Figure 4] [49].

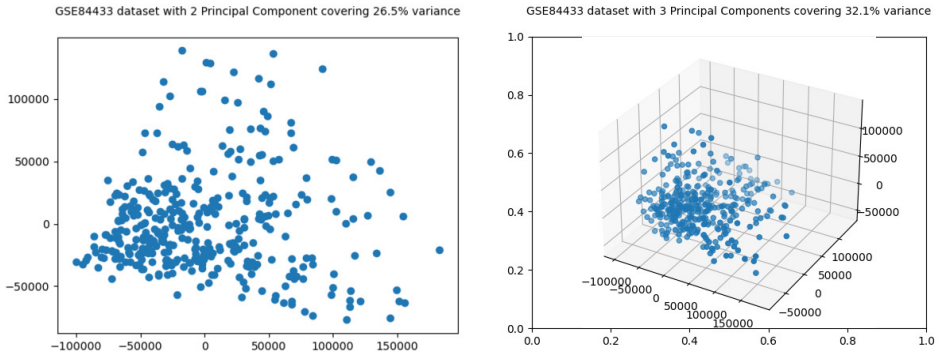

Figure 5: Visualization of first two principle components (PC) of GSE84433 dataset. The first two PC covers 26.5% and 32.1% variance of the overall dataset

Secondly, in Figure 5 we can see the gene expression profile of gastrectomy samples from 357 gastric cancer patients by HumanHT-12 v3.0 Expression BeadChip array (Illumina) known as GSE84433. Total RNA was extracted from the fresh-frozen gastrectomy specimens at the Yonsei University Severance Hospital (South Korea) between 2000 and 2010 [50].

The molecular subtypes and conserved modules in gastric cancer are present in each dataset was identified by unsupervised clustering algorithm.

6 Implementation

In order to simulate a centralized federated environment, every dataset during experimentation were randomly shuffled and then split into two non-overlapping sub-sets where the number of samples for each subset are unevenly distributed. Each sub-set was considered as a set of unique data records of two different participant sharing same feature spaces where the participant shares the intermediate results i.e Local Spike Distance Matrices (LSDMs) for coordinator-based computation. No network dependency and communication overhead was considered while simulating the federated learning environment and performing FEDM and PEDM calculation.

6.1 Computational Role

Considering the general concept stated in [51], there are two roles which are present in the federated learning environment for FEDM and PEDM calculation: **Participant(s)** and **Coordinator**. These roles were defined mostly based on what computation undergoes by them.

Participant(s) create local spike points based on chosen spike point generation method and broadcast them to the coordinator. Then they receive the updated global spike points coordinates from the coordinator and calculate Local Spike Distance Matrices (LSDMs) between its own data points and the spike points for supporting FEDM calculation. Furthermore, each participant performs regression considering all the values of its LSDM as independent variable (x) and that of its own data points as dependent variable (y). The slope and the coefficients are shared only as intermediate results to the coordinator for the calculation of PEDM.

It is assumed to have only one **Coordinator** based on FeatureCloud implementation [51] where the coordinator can also act as a participant. Like participants, there are also a number of operations performed by the coordinator:

- Concatenate all the spike points received from each participant and then broadcast them to all the participant(s).
- Calculate the FEDM and then the PEDM respectively.

6.2 System Configuration

Following system configuration were used to perform construction and evaluation of FEDM, PEDM w.r.t ADM. They were used for overall testing too.

System 01 was used in the development and initial testing purposes with smaller artificially created datasets along with GSE84426 dataset. **System 02** was used mostly for vigorous experimentation with mostly large dataset.

Table 3: System Configuration used for experimental purposes

System 01		**System 02**	
Processor	Intel(R) Core(TM) i5-4300U CPU @ 1.90GHz	**Processor**	Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz
Type	x64-based	**Type**	x64-based
RAM	8.00 GB (7.69 GB usable)	**RAM**	512 GB
HDD	250 GB (approx)	**HDD**	3.35 TB
Operating System	Windows 10 Home 20H2	**Operating System**	Ubuntu 20.04.3 LTS

6.3 Software Packages

Python has been used as the programming language for coding the implementation where Python version 3.9.7 was used in **System 01** and Python version 3.8.10 in **System 02** respectively.

As **System 02** was basically an Ubuntu Linux server, VIM - Vi IMproved 8.1 [52] was used as the code editor whenever it was necessary. Since **System 01** was primarily used for coding and initial prototype implementation, Visual Studio Code 1.61.2 was used as the code editor and the IDE (Integrated Development Environment) [53]. GitHub was used as the code hosting platform for version control of the codes and sharing purposes [54]. In order to ease the implementation of the federated settings along with the construction of spike points, FEDM and PEDM for experimental purposes, various software packages were used for code re-usability avoiding boilerplate and redundant code. They are mentioned in Table 4 respectively.

Table 4: Information about the python packages used in implementation

Package Name	**Version**	**Summary**
numpy	1.19.5	NumPy is the fundamental package for array computing with Python. [55]
pandas	1.2.4	Powerful data structures for data analysis, time series, and statistics. [56]
scikit-learn	1.0	A set of python modules for machine learning and data mining. [57]
scipy	1.6.3	Fundamental algorithms for scientific computing in Python. [58]
matplotlib	3.4.2	A comprehensive library for creating static, animated, and interactive visualizations in Python. [59]
seaborn	0.11.1	Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. [60]

7 Results

7.1 Artificial datasets

7.1.1 Non-uniformly distributed

In Figure 6, we visualize the pearson correlation between ADM and FEDM for three different datasets with varying amounts of samples and dimensions for a number of spike points with increasing order. In Figure 7 we visualize the same but between ADM and PEDM.

From Figure 6, it can be seen that the ADM can be almost 100% correlated based on Pearson coefficient by FEDM when it is constructed using centroids of different participant as spike points regardless of the amount of spike point used. On the other hand, the figure shows that for the first and the second dataset, FEDM created by just random spike points have a marginally better correlation than that created with both random points and centroids. However, for the third dataset which has a very high number of features, we see the performance of FEDM created by random points is unstable and often creates a lower positive correlation.

When the FEDM is created from random spike points or spike points generated via both random points and centroids, its pearson correlation *w.r.t* ADM increases as the number of spike point increases.

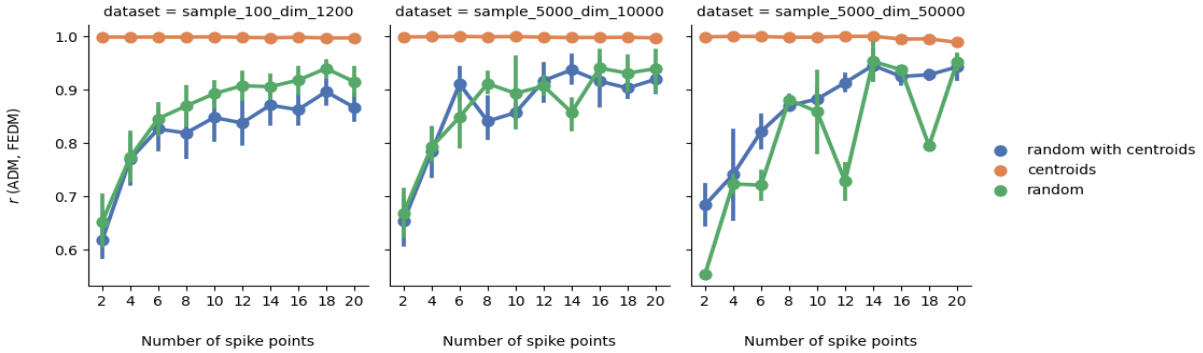

Figure 6: Visualizing Pearson coefficient of Federated Euclidean Distance Matrix (FEDM) *w.r.t* ADM for non-uniformly distributed datasets

In Figure 7, it can be observed that the correlation of PEDM *w.r.t* ADM for all the datasets are similar with that of FEDM in Figure 6. Noticeably, PEDM created from FEDM generated by spike points can be identical *w.r.t* ADM as the Pearson correlation coefficient is very close to 1.0

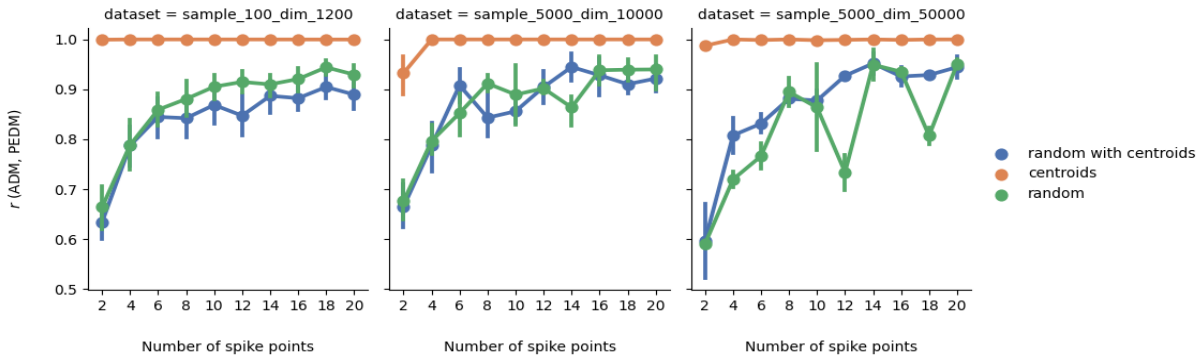

Figure 7: Visualizing Pearson coefficient of Predicted Euclidean Distance Matrix (PEDM) *w.r.t* ADM for non-uniformly distributed datasets

In Figure 8, we visualize the spearman correlation between ADM and FEDM for three different datasets with varying amount of samples and dimensions for number of spike points with increasing order. In Figure 9 we visualize the same but between ADM and PEDM.

In Figure 8, for the first two datasets and an increasing number of spike points, it can be seen that the spearman correlation of FEDM created by centroids as spike points are higher in average than that created by either just random points or both random points and centroids. However, when the sample size is 5000 and the number of dimensions are as high as 50000 the trend of correlation is unstable and random even with the increase of number of spike points used to create FEDM *w.r.t* ADM.

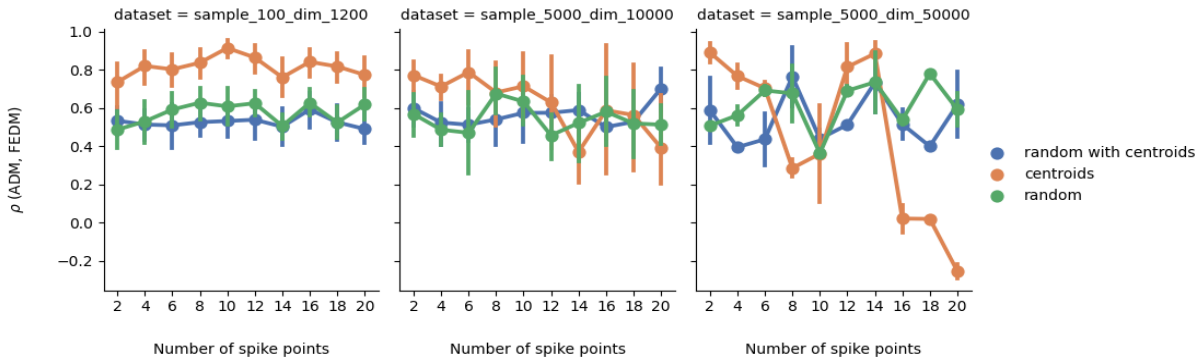

Figure 8: Visualizing Spearman coefficient of Federated Euclidean Distance Matrix (FEDM) *w.r.t* ADM for non-uniformly distributed datasets

In Figure 9, for all the datasets, it can be seen that the PEDMs can re-construct reasonably good distance matrices having very positive spearman's rank correlation coefficient values denoting that regression helps to minimize the drastic error of values for each pairwise distance value in FEDM. As usual, PEDM which are created indirectly

from FEDM generated by the centroids portrays the highest correlation while PEDM created indirectly with other spike point generation method struggles to be reasonably correlated *w.r.t* ADM and does not provide an increment in coefficient values with the increasing number of spike points.

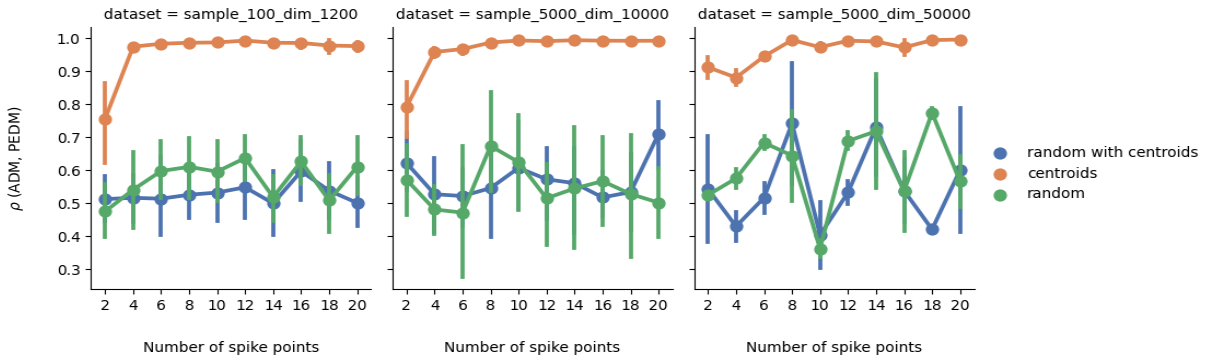

Figure 9: Visualizing Spearman coefficient of Predicted Euclidean Distance Matrix (PEDM) *w.r.t* ADM for non-uniformly distributed datasets

In Table 5, the average of Pearson and Spearman correlation coefficient values of both PEDM and FEDM *w.r.t* ADM regardless of the spike generation method. Except for the dataset having 5000 samples and 50000 dimensions, We can see that for all the other datasets, 6 - 14 spike points can generally create FEDM and PEDM possessing very high positive pearson correlation *w.r.t* ADM. However, the value of spearman correlation coefficient is lower than that of pearson correlation coefficient regardless of the number of spike points, spike point generation methods along with the size of sample and dimension of the datasets.

Table 5: Average Pearson and Spearman correlation coefficient of FEDM and PEDM *w.r.t.* ADM for non-uniformly distributed random datasets using all the spike point generation methods

Samples	**Dimension**	**Spike points**	**Pearson Correlation Coefficient**		**Spearman Correlation Coefficient**	
			ADM w.r.t PEDM	**ADM w.r.t FEDM**	**ADM w.r.t PEDM**	**ADM w.r.t FEDM**
100	1200	2	0.77	0.76	0.58	0.59
100	1200	6	0.90	0.89	0.70	0.63
100	1200	14	0.93	0.92	0.64	0.57
100	1200	20	0.94	0.92	0.68	0.62
5000	10000	2	0.76	0.77	0.66	0.64
5000	10000	6	0.92	0.92	0.65	0.59
5000	10000	14	0.94	0.93	0.70	0.49
5000	10000	20	0.95	0.95	0.73	0.54
5000	50000	2	0.75	0.78	0.68	0.69
5000	50000	6	0.86	0.84	0.71	0.61
5000	50000	12	0.89	0.88	0.74	0.67
5000	50000	20	0.97	0.96	0.72	0.32

7.1.2 Uniformly distributed

In Figure 10, we visualize the pearson correlation between ADM and FEDM for four different datasets with varying amounts of samples and dimensions for number of spike points with increasing order. In Figure 11, we visualize the same but between ADM and PEDM.

Due to the uniform distribution of random data points, in Figure 10, we can see that random spike points perform better in constructing a more correlated distance matrix *w.r.t* to ADM. When the dimension of the data points is lower, FEDM has lesser error in constructing matrix close to ADM and the pearson correlation gradually increases with the number of spike points. The FEDM created via other spike points generation methods have significantly lower pearson correlation coefficient values. As the data points are sparsely distributed and there is no dense region, it does not help to have centroids as spike points in order to gain more information of the data points respectively.

However, when we consider some random points along with the centroids as spike points the correlation increase with the dataset having lower dimension. FEDM calculation is more error-pruned as the dimension increases in a uniformly distributed dataset of random samples.

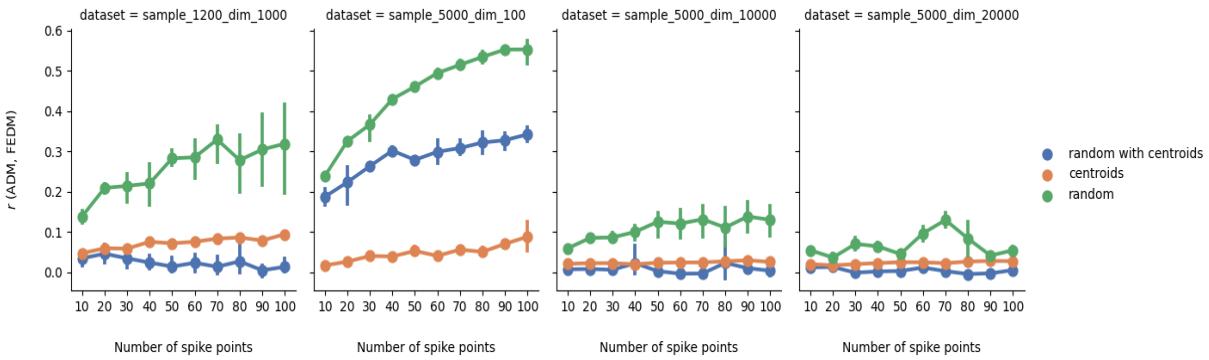

Figure 10: Visualizing Pearson coefficient of Federated Euclidean Distance Matrix (FEDM) *w.r.t* ADM for uniformly distributed random datasets

In Figure 11, we can see the potential of PEDM and regression. After performing regression, it gains almost 85% to 90% correlation *w.r.t* ADM making it useful for operational use-cases related to Euclidean Distance Matrix (EDM) more implementable and usable in the federated learning environment. PEDM generated from FEDM have similar pearson correlation coefficient values despite the selection of spike point generation method. It is noticeable that both FEDM and PEDM created based on dataset with lower dimensions have almost similar correlation *w.r.t* ADM.

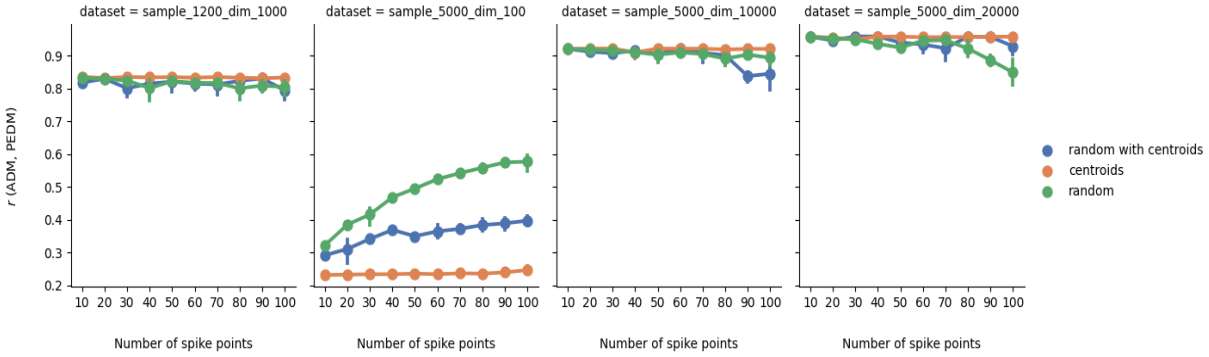

Figure 11: Visualizing Pearson coefficient of Predicted Euclidean Distance Matrix (PEDM) *w.r.t* ADM for uniformly distributed random datasets

In Figure 12, we visualize the spearman correlation between ADM and FEDM for four different datasets with varying amounts of samples and dimensions for an increasing number of spike points. In Figure 13, we visualize the same but between ADM and PEDM.

However, both Figure 12 and Figure 13 suggest that spearman correlation tends to remains approximately the same for FEDM and PEDM *w.r.t* regardless of spike point generation method. Except for the dataset having samples with lower dimensions i.e. 100, all the other data samples have very low spearman correlation coefficient (0.0 to 0.3) because of the difference of value along with the rank for each dimension of the data points is very high and the difference furthermore tends to get higher with the increase in number of dimensions respectively.

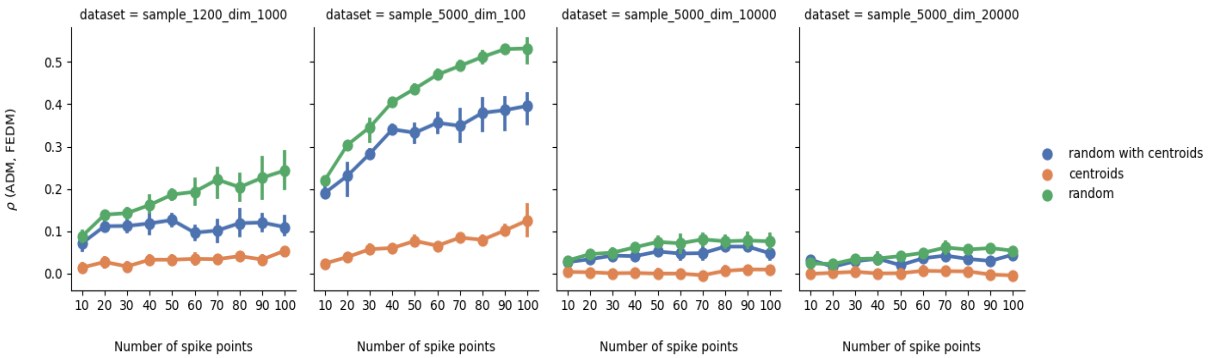

Figure 12: Visualizing Spearman coefficient of Federated Euclidean Distance Matrix (FEDM) *w.r.t* ADM for uniformly distributed random datasets

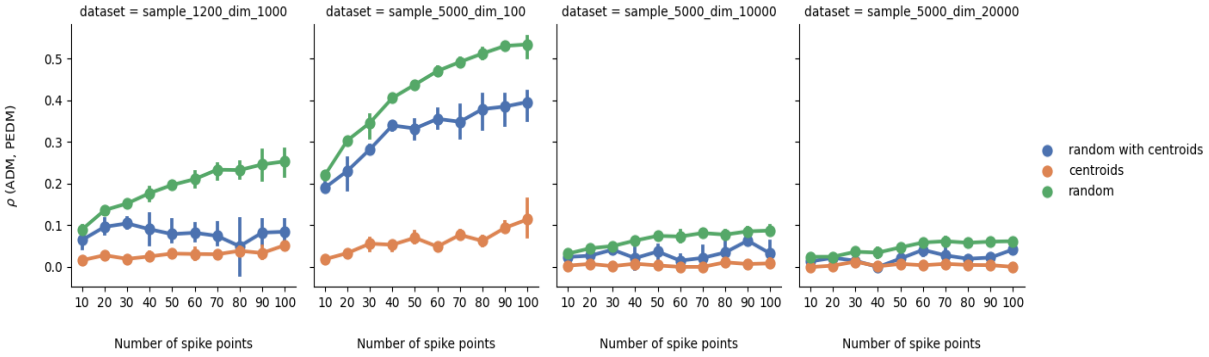

Figure 13: Visualizing Spearman coefficient of Predicted Euclidean Distance Matrix (PEDM) *w.r.t* ADM for uniformly distributed random datasets

Table 6: Average Pearson and Spearman correlation coefficient of FEDM and PEDM *w.r.t.* ADM for uniformly random dataset using all the spike point generation methods

Samples	Dimension	Spike points	Pearson Correlation Coefficient		Spearman Correlation Coefficient	
			ADM w.r.t PEDM	ADM w.r.t FEDM	ADM w.r.t PEDM	ADM w.r.t FEDM
1200	1000	10	0.83	0.07	0.05	0.05
1200	1000	40	0.82	0.10	0.07	0.08
1200	1000	70	0.82	0.13	0.10	0.10
1200	1000	100	0.81	0.13	0.11	0.11
5000	100	10	0.28	0.15	0.14	0.14
5000	100	40	0.36	0.25	0.27	0.27
5000	100	70	0.38	0.30	0.30	0.31
5000	100	100	0.41	0.33	0.35	0.35
5000	10000	10	0.92	0.03	0.02	0.02
5000	10000	40	0.91	0.05	0.03	0.03
5000	10000	70	0.91	0.05	0.03	0.04
5000	10000	100	0.89	0.05	0.04	0.15
5000	20000	10	0.96	0.03	0.01	0.01
5000	20000	40	0.95	0.03	0.01	0.01
5000	20000	70	0.95	0.04	0.02	0.03
5000	20000	100	0.93	0.03	0.02	0.02

From Table 6, we can get an overall picture of how the number of spike points regardless the method used to generate them has an effect in constructing more correlated euclidean distance matrix similar to ADM. When more spike points have been used, the pearson and spearman correlation tends to increases upto a certain degree of value. From the above calculation, we can inspect and further assume that, 40 - 70 spike points can produce PEDM with very high linear correlation *w.r.t* ADM as it was produced by perform regression on FEDM and ADM respectively.

7.2 Gene Expression Datasets

For each method of spike point generation, an even number of spike points have been considered. For each N number spike points, the average value was considered after calculating the coefficient values for 10 to 15 times.

7.2.1 GSE84426

In Figure 14, we visualize the pearson correlation between ADM and FEDM and between ADM and PEDM for GSE84426 dataset with increasing number of spike points. In Figure 15, we visualize the spearman correlation instead.

From Figure 14, we can see that the spike points generated by centroid-based and random method lead to higher Pearson correlation between ADM and FEDM gradually with the increasing number of spike points. Whereas the spike points consisting of both random points and centroids generates FEDM with lower Pearson correlation for every given number of spike points because many of the random points happen to emerge very close to centroids which results in less information gain by LSDM(s) thus creating less accurate FEDM respectively.

However, when we have PEDM which is a corrected version of FEDM where FEDM is corrected via the slope and intercept obtained from regression performed by ADM and FEDM, we can observe for every given number of spike points, PEDM has approximately similar pearson correlation regardless of spike point generation method used by the FEDM. When the FEDM was created with centroid based spike points, then PEDM has a better pearson correlation than that by random points or centroids if the number of spike points is equal or greater than 28 which can happen as randomly generated may reduce information gain due to the randomness of placement.

In Figure 15, we observe that for every number of spike points, the FEDM created by both random points and centroids along with the resulted corresponding PEDM having significantly lower spearman correlation *w.r.t* to ADM.

It is visible that even though FEDM and PEDM resulted using random spike points and that using centroids have a similar correlation for different number of spike points, the centroid-based method produces slightly better correlation as the spike points in this case consider average distances between the data points whereas random spike point generation method creates multiple spike points in close proximity which may reduce information gain due to the randomness of placement.

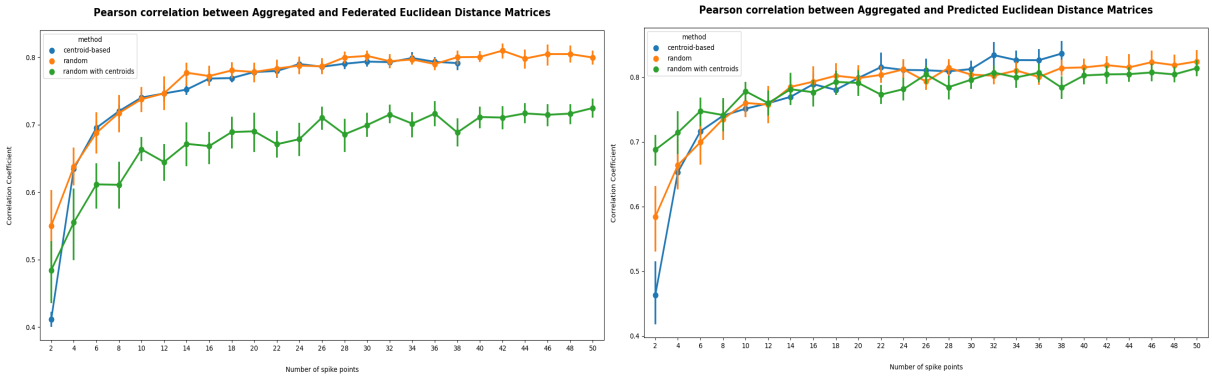

Figure 14: Pearson correlation coefficient of Federated Euclidean Distance Matrix (FEDM) and Predicted Euclidean Distance Matrix (PEDM) *w.r.t* Aggregated Distance Matrix (ADM) for GSE84426 dataset

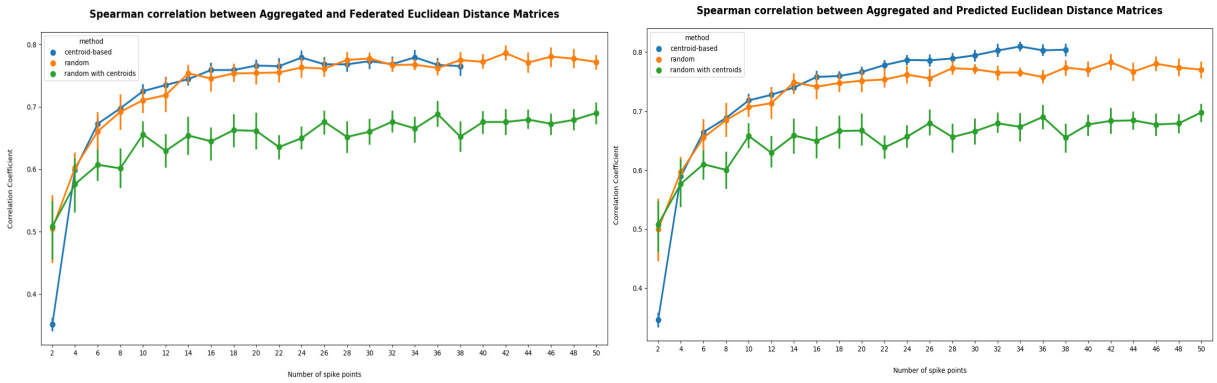

Figure 15: Spearman correlation coefficient of Federated Euclidean Distance Matrix (FEDM) and Predicted Euclidean Distance Matrix (PEDM) *w.r.t* Aggregated Distance Matrix (ADM) for GSE84426 dataset

Figure 16 projects that the relation of FEDM and PEDM with ADM is more linear as the correlation of both matrices decreases monotonically *w.r.t* ADM. Furthermore, PEDM has higher linear correlation than FEDM while FEDM can obtain similar pearson or spearman relation *w.r.t* ADM. It suggests that there is a notable difference between the rank of value of dimensions for considerable amount of the data points which is resulting in higher spearman correlation coefficient meaning that some data points have very high value for their dimensional space.

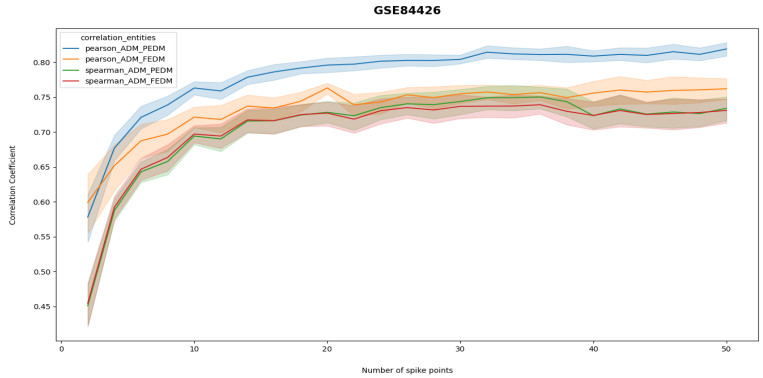

Figure 16: Pearson and Spearman correlation coefficient of Federated Euclidean Distance Matrix (FEDM) and Predicted Euclidean Distance Matrix (PEDM) *w.r.t* Aggregated Distance Matrix (ADM) upto 50 spike points for GSE84426 dataset

7.2.2 GSE84433

In Figure 17, we visualize the pearson correlation between ADM and FEDM and between ADM and PEDM for the GSE84433 dataset with an increasing number of spike points. In Figure 18, we visualize the spearman correlation instead.

From Figure 17, it can be perceived that for every number of spike points, FEDM created by all the spike generation methods have progressively increasing the value of Pearson correlation coefficient. However, the FEDM created by uniformly distributed random spike points have higher linear correlation with ADM for every number of spike points because a high number of data points are uniformly scattered [Figure 5] along with the fact that randomness of spike point captured more correlation of data points for FEDM which is the distance of the distance between spike points and the data points of different participant(s). When the spike points are also distributed similarly the gap of information gain via LSDM(s) decreases.

However, for each and every number of spike points, PEDM regressed or corrected from FEDM which are created from the spike points generated via random points along with random points with centroid have better pearson correlation with ADM than that regressed from FEDM which are created from the spike points generated from centroids only. Although we observe that PEDMs which have the highest pearson correlation are produced by FEDM resulted from spike points that are both centroids and random points. As a considerable number of data points are available in the range of value of -10000 and 0 [Figure 5], its difference of the coefficient values with PEDM originated from FEDM created by only random points is negligible.

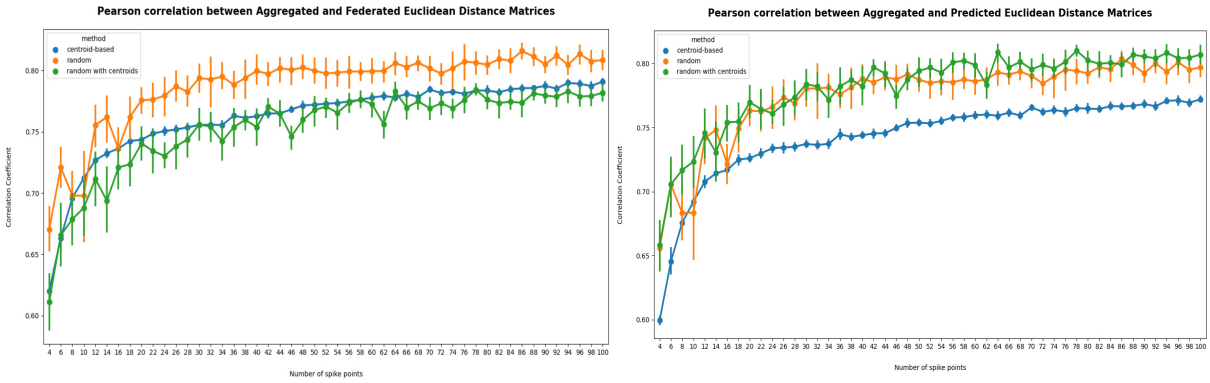

Figure 17: Pearson correlation coefficient of Federated Euclidean Distance Matrix (FEDM) and Predicted Euclidean Distance Matrix (PEDM) *w.r.t* Aggregated Distance Matrix (ADM) for GSE84433 dataset

In Figure 18, we can note that FEDM created with the pairwise distance between random spike points and data points have higher and better monotonic correlation *w.r.t* ADM while FEDM created by the other two spike generation methods have lesser correlation because when spike points are only randomly created and uniformly distributed it decreases the difference of the rank of data points as the value of data points have high variances and vice versa for the spike points i.e which are mostly centroids or partially used beside uniform random spike points. Although, centroid-based FEDM have slight better correlation with the ADMs, the FEDMs created by spike points that are either centroids or random points with centroid have very close respective correlation coefficient values against ADM. We can also observe the increase in spearman correlation regardless of the method used for spike point generation as the number of spike point increases.

We observe the same pattern and results for the PEDM calculated after performing regression by each participant and applying the respective coefficient and intercept for each value of FEDM. In case of PEDM, those which are originated from random and centroid-based FEDM have slightly better correlation with the ADMs even though PEDM generated from FEDM which are created by spike points that are either centroids or just random points have high similarities *w.r.t* correlation coefficient values against ADM.

It is noticeable from Figure 19 that the average Pearson correlation of FEDM and PEDM *w.r.t* ADM has an increasing order for each number of spike points. However, FEDM has slightly better correlation coefficient value. As good number of data points were very close to eachother, it gave slightly better FEDM compared to the PEDM which used the predicted intercept(s) and coefficient(s) after performing regression on LSDM(s)

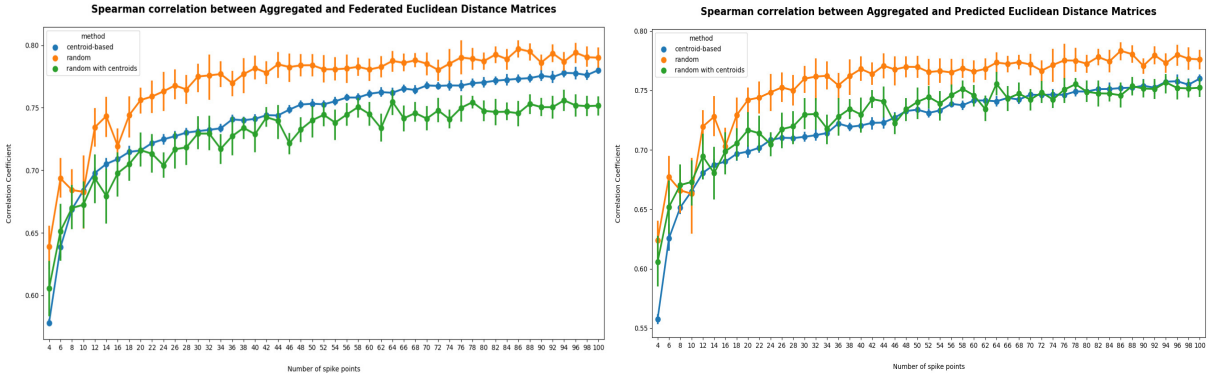

Figure 18: Spearman correlation coefficient of Federated Euclidean Distance Matrix (FEDM) and Predicted Euclidean Distance Matrix (PEDM) *w.r.t* Aggregated Distance Matrix (ADM) for GSE84433 dataset

It can also be perceived that the average Spearman correlation of FEDM and PEDM rise with the increment of spike points while the Spearman correlation between FEDM and ADM is notably higher than that between PEDM and ADM respectively. It may tell us that gap of values along with the difference between the dimension of the data points were lower in FEDM and a high number of spike points can generate distance matrices very close to ADM.

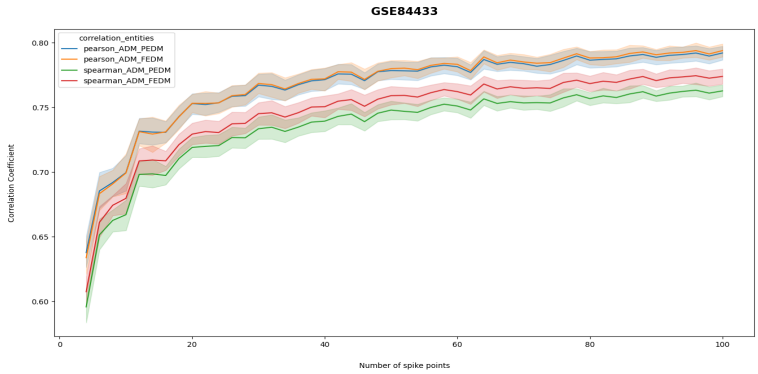

Figure 19: Pearson and Spearman correlation coefficient of Federated Euclidean Distance Matrix (FEDM) and Predicted Euclidean Distance Matrix (PEDM) *w.r.t* Aggregated Distance Matrix (ADM) upto 100 spike points for GSE84433 dataset

Even though the Euclidean distance becomes less effective with an increase in the dimensionality, it shows good performance as many of the datasets have similarities with non-central t-distribution, for which euclidean distance function can be much effective for higher dimensions (10,000 and more). Additionally, by increasing the number of samples, the Euclidean distance becomes more effective which assists in overcoming the curse of dimensionality. [61]

7.2.3 K-medoids clustering evaluation of GSE84426 dataset

In order to demonstrate one of the use-cases of FEDM and PEDM, K-medoids clustering has been performed on a gene-expression dataset, namely GSE84426. As the number of unique labels was available within the dataset, it was used as the number of cluster (k) to cluster the dataset respectively.

In Figure 4, we can see two and three-dimensional view of the dataset. Although, they are not covering a significant amount of variation, the distribution of the data point can more or less be visualized. We can observe that the dataset does not look as the best candidate to provide good clustering results as they are scattered both horizontally and vertically.

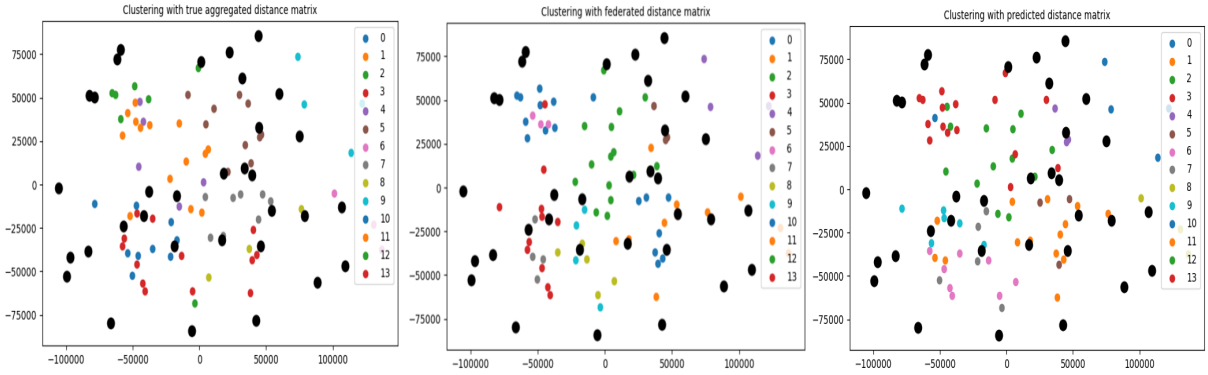

Figure 20: K-medoid clustering using ADM, FEDM and PEDM visualized in two-dimensional space

In Figure 20, the black points are the artificially generated spike points which were later used to form FEDM and PEDM. Important to be mentioned, the clustering was performed on the original raw dataset which is actually split between participant(s) to simulate federated settings even though the figure represents the clustering visualization in two-dimensional space.

Due to the distribution of the data points, we have considered K-medoids instead of K-means because the clustering is done considering the medoid which is more robust and less influenced by outliers (i.e. points far away from the other members of the cluster) than the arithmetic mean (centroid). K-means will select the "center" of the cluster, while k-medoid will select the "most centered" member of the cluster. In a cluster with outliers, k-means will place the center of the cluster towards the outliers, whereas k-medoid will select one of the more clustered members (the medoid) as the center which can yield in better clustering results even if there were some huge errors in the dataset.

Table 7: Evaluation of K-medoids Clustering for GSE84426 dataset

Distance Matrix Type	Adjusted Random Score	Adjusted Mutual Info Score	F1 Score
Aggregated Distance Matrix (ADM)	3.32	8.12	3.0
Federated Euclidean Distance Matrix (FEDM)	32.66	46.28	6.6
Predicted Euclidean Distance Matrix (PEDM)	35.77	52.21	1.3

Table 7 shows that Adjusted Random Score (ARI) and Adjusted Mutual Info Score (AMI) for ADM is very low as if the labels were given randomly. However, when the clustering is performed based on FEDM and PEDM and evaluated them *w.r.t* the labelling of ADM, we find around 30% of all the data points are similarly clustered and about 50% similarity between two labels of the same data based on AMI score respectively. The F1 score tends to zero which shows the lack of precision and robustness of the K-medoid clustering in this particular dataset.

8 Discussion

Overall, we have multiple observations from the derived pearson and spearman correlation coefficient value of FEDM and PEDM *w.r.t* ADM of different dataset. Suppose both the FEDM and PEDM belongs to data points which are random and uniformly distributed. In that case, it will tend to have a low positive spearman correlation due to the high difference of rank of each value of the dimension. However, PEDM will be very reasonable with a strong linear correlation to the ADM due to the regression performed locally on LSDMs.

Importantly, the experiment shows using randomly distributed spike points for constructing FEDM ultimately and overall returns good PEDM. For a dataset which portrays more visible clusters or have multiple high-density regions comprised of the value of the data points, centroids as the spike points will provide more strongly correlated FEDM and PEDM respectively. If the spike points are either uniformly distributed random points or uniformly distributed random points along with centroids, a dataset with lots of outliers can have very highly correlated FEDM and PEDM.

If the data consists of isotropic gaussian blobs where all data points belong to one of the cluster and all cluster have the same standard deviation, more information is gained from spike points which are centroid-based [62]. Hence, the FEDM and PEDM for this type of dataset could have very high positive correlation *w.r.t* ADM regardless of the number of spike points if and only if the spike points are generated from centroids. However, when the blobs have samples with higher dimensions, the value of spearman correlation coefficient is unstable even with the increase in number of the spike points used because the pariwise euclidean distances of LSDMs are less interpretable due to the curse of dimensionality along with insufficient coverage of monotonic relationship. Also, in this particular case, PEDM created indirectly with 8 - 10 spike points gives a high positive correlation *w.r.t* ADM in aspects of pearson and spearman correlation.

When FEDM is created from an array of spike points where some of them are centroids and others are random points, it tends to have either low or moderate positive correlation with ADM considering both pearson and spearman correlation. It is due to the fact when the spike points are generated, many of the random ones happen to emerge very close to centroids which results in little or no information gain of distance by LSDM(s), thus, creating less accurate FEDM respectively. PEDM can overcome the gap of error and produce a highly or very highly positive correlated matrix *w.r.t* ADM.

Spike points generated via centroids provided by participant(s) works well with dataset having multiple high or low dense clusters. For a dataset that has skewed distribution

and high number of outliers, spike points generated with centroids and random points together create better correlated FEDM than spike points which are just centroids. Nonetheless, despite the nature of the dataset, spike points generated randomly have a higher chance to contribute for the construction of a highly correlated distance matrix.

In general, we expected to have a gradual increase in either pearson and spearman correlation when the number of spike points used also increase. Apparently, it is not the case and in fact, after 6 to 24 spike points there are no consequential improvement of either pearson or spearman correlation. When the dataset distribution is unknown, participant(s) can optimally consider 20 spike points created via random method and expect to have highly but linearly correlated PEDM if not FEDM and use it for further processing or use-cases. It is much preferable that the pearson correlation coefficient value of either FEDM or PEDM is greater than 0.80, in order to be useful for less erroneous implementation of different use-cases of euclidean distance matrix in federated learning environment.

when a dataset contains uniformly distributed high-dimensional random samples, the complete shift of pearson correlation coefficient between PEDM and FEDM was surprising but insightful due to the showcase of how FEDM can be far away from ADM but performing regression can almost reconstruct a distance matrix (PEDM) having more than 80% similarity and did not expect there will be drastic variance between Pearson and Spearman correlation coefficient values. FEDM or PEDM for a dataset with uniformly distributed lower-dimensional random samples were not able to be strongly correlated with ADM.

8.1 Limitation

There are few architectural and privacy constraints that can limit the potentials and practicality of the proposed solution to approximately reconstruct the original euclidean distance matrix as FEDM or PEDM via spike points.

The proposed solution to calculate FEDM and PEDM is initially based on the concept that the datasets will be obtained from different sources but have similar structures. Thus, our approach can be applied only to horizontally partitioned data across participants, i.e., each participant has unique data points, but all the data points are vectors belonging to the same feature spaces having the same number and order of dimension respectively. It will not be possible to create FEDM and PEDM if there is a mismatch with the number of features or dimensions of the datasets in any of the participants. Furthermore, the order of the features for each dataset should be sequentially same.

Euclidean distance is calculated by taking the square root of the summation of differences between each dimension value of two data points. In order to ensure the security and privacy of the raw data, the number of spike points cannot be more than the number of dimensions due to the property of euclidean distance. Suppose any dishonest party can access all the spike points and LSDM of any participants where the number of spike points is more than that of dimension. In that case, it can calculate the value of the original data points of that respective participants. However, if the number of spike points is less than that of the dimension of data points of the participants, due to the nature of the euclidean distance formula, the attacker would not be able to re-construct the data points of any participants. Hence, making the construction of the euclidean distance matrices "private by design".

When a participant has a dataset with lots of outliers and performs K-means or other clustering algorithms to create spike points based on centroids or random points and centroids, there is a chance that an outlier itself or very similar data points can be considered as centroid and returned as spike point for consideration. The coordinator has no way of determining it as an original datapoint and skips it for constructing LSDMs for all the participants meaning that there is a chance to reveal outlier data points [63]. From the experimentation, we can state that using random points as spike points does not decrease the similarities of FEDM and PEDM *w.r.t* ADM. It is advisable to use random spike generation methods for constructing LSDMs unless the overall dataset presented by participants are expected to be similar to isotropic Gaussian blobs where outliers do not have much influence on centroid calculations.

Last but not least, it is difficult to find either an optimum or a generic number of spike points for creating FEDM or PEDM as a participant(s) can have different numbers and distribution of samples. As shown in the experimental outcome, for some datasets, even 4 spike points are enough, and for some, we needed 40-70 spike points to get a reasonable PEDM which can be considered for its use-case by the system and its users. More research and experimentation are required to come up with general suggestions.

8.2 Future Work

Further attempts could prove quite beneficial, such as investigating how to calculate other similarity measures like Cosine or Jaccard similarity along with overall generalization of other Minkowski distance functions, Canberra distance functions and their association between them might prove beneficial and effective as a utility for implementing different algorithms in the federated learning environment.

Regardless, future research should continue to explore how to increase the pearson and spearman correlation coefficient for a more skewed and uniformly distributed dataset and even better if very few spike points are used in the construction of FEDM. It will also be crucial to explore the correctness and performance of FEDM and PEDM in the production environment of federated settings.

More research and in-depth analysis should be conducted to see how FEDM and PEDM can be utilized for implementing different clustering algorithms under either horizontal or vertical federated settings. The possibility of breaching privacy, especially when some [64] or all the datapoints of a participant are known accidentally, warrants further investigation [65, 66].

9 Conclusion

In summary, we wanted to approximate and calculate the aggregated euclidean distance matrix between the datapoints from one or multiple datasets in a federated learning system without sharing the value of datapoints when the data samples are horizontally partitioned and distributed across different participant(s) in the system. Consequently, we have proposed two types of euclidean distance matrices originated from spike points and regression named Federated Euclidean Distance Matrix (FEDM) and Predicted Euclidean Distance Matrix (PEDM), which uses the pairwise distance between the data points and spike points locally for each participant. We have been able to construct distance matrices which have around 80% to 99% similarities *w.r.t* the aggregated distance matrix (distance matrix containing pairwise distance of all the datapoints of all participants) based on the application of regression and help of spike points created with various methodologies. We believe that, rather than using encryption or similar cryptographic technique along with any other existing high computational methods, our suggested approach for the construction of the FEDM and PEDM will make federated learning more practical and usable while reducing the prevailing bottleneck of high communication overhead in many existing solutions along with preserving data privacy and security.

10 Acknowledgement

My supervisor, Associate Professor Richard Röttger, deserves the most profound gratitude for suggesting this exciting, trendy topic for my dissertation and for all the insightful discussions he held with me as I developed the ideas presented in this paper. My genuine appreciation for co-supervisor Tobias Frisch for detail elaboration of problem. His helpful feedback on my intuitions and directions in organizing my task objectives encouraged me to reach my thesis goals by pre-reviewing and overseeing my deliverables and progress constantly as per necessity. The academic and professional guidance my study advisor Rune Wulff Christensen bestowed me to complete my degree program even before starting at Syddansk Universitet is something I will never forget. I would like to specially thank Aman Kushwaha for his advices and discussion with me regarding my solutions. Steffan Ravn Edwardsen of the IT department at SDU was also of great assistance to me to set up a remote access account to perform thesis-related experiments.

Personally, my wholehearted gratitude and sincere appreciation goes out to my uncle, Mr. Aziz Ahmad. He has continuously supported my education and shared his wisdom, opinion, and thoughts about becoming a better person on many levels while still doing it selflessly with genuine care and affection. My earnest regards and respect to Thomas Muus Thorkilsen for giving me all the flexibility, motivation and encouragement I needed to balance my study along with my job while always having faith and belief on me and lifting up my spirit. Last but not least, an extra-ordinarily special thanks goes to the most important person of my life, my mother, Syeda Shahida Akhter, who continuously inspires and supports me throughout my life in all of my endeavors.

References

[1] https://insights.comforte.com/13-countries-with-gdpr-like-data-privacy-laws.

[2] Wikipedia contributors, "Euclidean distance — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Euclidean_distance, 2021.

[3] J. Liu and X. Meng, "Survey on privacy-preserving machine learning," *Journal of Computer Research and Development*, vol. 57, no. 2, p. 346, 2020.

[4] https://www.pnas.org/content/112/28/8515.

[5] K. B. Frikken, "Secure multiparty computation," in *Algorithms and theory of computation handbook: special topics and techniques*, pp. 14–14, 2010.

[6] L. (https://math.stackexchange.com/users/18111/laciel), "Euclidean distance proof." Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/75207 (version: 2011-10-23).

[7] M. M. Mukaka, "Statistics corner: A guide to appropriate use of correlation coefficient in medical research," *Malawi Medical Journal*, vol. 24, no. 3, pp. 69–71, 2012.

[8] Ethan, "Pearson vs spearman vs kendall." https://datascience.stackexchange.com/questions/64260/pearson-vs-spearman-vs-kendall/64261, 2019.

[9] Pınar Ersoy, "Types of correlation coefficients- different kinds of correlation coefficients in a deeper look." https://towardsdatascience.com/types-of-correlation-coefficients-db5aa9ea8fd2, 2021.

[10] Laerd Statistics, "Pearson product-moment correlation." https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php, 2018.

[11] Wikipedia contributors, "Federated learning — Wikipedia, the free encyclopedia." https://en.wikipedia.org/wiki/Federated_learning, 2021.

[12] J. Konečnỳ, H. B. McMahan, D. Ramage, and P. Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[13] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[14] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[15] V. Smith, C.-K. Chiang, M. Sanjabi, and A. Talwalkar, "Federated multi-task learning," *arXiv preprint arXiv:1705.10467*, 2017.

[16] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[17] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "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.

[18] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," *arXiv preprint arXiv:1712.07557*, 2017.

[19] F. Chen, M. Luo, Z. Dong, Z. Li, and X. He, "Federated meta-learning with fast convergence and efficient communication," *arXiv preprint arXiv:1802.07876*, 2018.

[20] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[21] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pp. 308–318, 2016.

[22] H. U. P. T. Project, "Differential privacy." `https://privacytools.seas.harvard.edu/differential-privacy`, 2021.

[23] `https://homomorphicencryption.org/introduction/`.

[24] Wikipedia contributors, "Homomorphic encryption — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/w/index.php?title=Homomorphic_encryption`, 2021.

[25] Wikipedia contributors, "Secure multi-party computation — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/w/index.php?title=Secure_multi-party_computation`, 2021.

[26] W. Du, Y. S. Han, and S. Chen, "Privacy-preserving multivariate statistical analysis: Linear regression and classification," in *Proceedings of the 2004 SIAM international conference on data mining*, pp. 222–233, SIAM, 2004.

[27] Y. Lindell, "Secure multiparty computation," *Communications of the ACM*, vol. 64, no. 1, pp. 86–96, 2020.

[28] D. Dachman-Soled, T. Malkin, M. Raykova, and M. Yung, "Efficient robust private set intersection," in *International Conference on Applied Cryptography and Network Security*, pp. 125–142, Springer, 2009.

[29] K. Frikken, "Privacy-preserving set union," in *International Conference on Applied Cryptography and Network Security*, pp. 237–252, Springer, 2007.

[30] R. M. Vector Institute, "Ryan marten - vertical federated learning." `https://www.youtube.com/watch?v=JSCBRvysMFU`, 2021.

[31] S. Saha and T. Ahmad, "Federated transfer learning: concept and applications," *Intelligenza Artificiale*, vol. 15, no. 1, pp. 35–44, 2021.

[32] B. Pandya, U. Singh, and K. Dixit, "An analysis of euclidean distance preserving perturbation for privacy preserving data mining," *Int. J. Res. Appl. Sci. Eng. Technol*, vol. 2, pp. 33–35, 2014.

[33] P. Mohassel, M. Rosulek, and N. Trieu, "Practical privacy-preserving k-means clustering.," *Proc. Priv. Enhancing Technol.*, vol. 2020, no. 4, pp. 414–433, 2020.

[34] P. Ravikumar, W. W. Cohen, and S. E. Fienberg, "A secure protocol for computing string distance metrics," *PSDM held at ICDM*, pp. 40–46, 2004.

[35] J. Vaidya and C. Clifton, "Secure set intersection cardinality with application to association rule mining," *Journal of Computer Security*, vol. 13, no. 4, pp. 593–622, 2005.

[36] N. M. Stausholm, "Improved differentially private euclidean distance approximation," in *Proceedings of the 40th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pp. 42–56, 2021.

[37] W.-J. Lee, R. P. Duin, A. Ibba, and M. Loog, "An experimental study on combining euclidean distances," in *2010 2nd International Workshop on Cognitive Information Processing*, pp. 304–309, IEEE, 2010.

[38] S. Mukherjee, Z. Chen, and A. Gangopadhyay, "A privacy-preserving technique for euclidean distance-based mining algorithms using fourier-related transforms," *The VLDB journal*, vol. 15, no. 4, pp. 293–315, 2006.

[39] Q. Li, V. Kecman, and R. Salman, "A chunking method for euclidean distance matrix calculation on large dataset using multi-gpu," in *2010 Ninth International Conference on Machine Learning and Applications*, pp. 208–213, Ieee, 2010.

[40] D. Chang, N. A. Jones, D. Li, M. Ouyang, and R. K. Ragade, "Compute pairwise euclidean distances of data points with gpus," in *Proceedings of the iASTED international Symposium on Computational Biology and Bioinformatics*, pp. 278–283, 2008.

[41] T. Rechkalov and M. Zymbler, "A study of euclidean distance matrix computation on intel many-core processors," in *International Conference on Parallel Computational Technologies*, pp. 200–215, Springer, 2018.

[42] S. Kim and M. Ouyang, "Compute distance matrices with gpu," 2012.

[43] A. S. Arefin, C. Riveros, R. Berretta, and P. Moscato, "Computing large-scale distance matrices on gpu," in *2012 7th International Conference on Computer Science & Education (ICCSE)*, pp. 576–580, IEEE, 2012.

[44] M. Angeletti, J.-M. Bonny, and J. Koko, "Parallel euclidean distance matrix computation on big datasets," 2019.

[45] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html`.

[46] `https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.random.random.html`.

[47] A. Ghosh and S. Barman, "Application of euclidean distance measurement and principal component analysis for gene identification," *Gene*, vol. 583, no. 2, pp. 112–120, 2016.

[48] J. Cao, J. Gong, X. Li, Z. Hu, Y. Xu, H. Shi, D. Li, G. Liu, Y. Jie, B. Hu, *et al.*, "Unsupervised hierarchical clustering identifies immune gene subtypes in gastric cancer," *Frontiers in Pharmacology*, vol. 12, 2021.

[49] `https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE84426`.

[50] `ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE84433`.

[51] J. Matschinske, J. Späth, R. Nasirigerdeh, R. Torkzadehmahani, A. Hartebrodt, B. Orbán, S. Fejér, O. Zolotareva, M. Bakhtiari, B. Bihari, *et al.*, "The featurecloud ai store for federated learning in biomedicine and beyond," *arXiv preprint arXiv:2105.05734*, 2021.

[52] `https://www.vim.org/download.php`.

[53] `https://code.visualstudio.com/`.

[54] `https://github.com/`.

[55] https://numpy.org/doc/stable/reference/.

[56] https://pandas.pydata.org/docs/reference/index.html.

[57] https://scikit-learn.org/stable/.

[58] https://scipy.github.io/devdocs/index.html.

[59] https://matplotlib.org/stable/contents.html.

[60] https://seaborn.pydata.org/api.html.

[61] S. Xia, Z. Xiong, Y. Luo, G. Zhang, *et al.*, "Effectiveness of the euclidean distance in high dimensional spaces," *Optik*, vol. 126, no. 24, pp. 5614–5619, 2015.

[62] G. K. C. (https://stats.stackexchange.com/users/217591/girish-kumar chandora), "What is the meaning of isotropic gaussian blobs , which are generated by sklearn.datasets.makeblobs?." Cross Validated. URL:https://stats.stackexchange.com/q/534543 (version: 2021-07-14).

[63] P. K. Alisa Chang, "Practical differentially private clustering," 2021.

[64] C. R. Giannella, K. Liu, and H. Kargupta, "Breaching euclidean distance-preserving data perturbation using few known inputs," *Data & Knowledge Engineering*, vol. 83, pp. 93–110, 2013.

[65] S. Yu, J. Zheng, J. Chen, Q. Xuan, and Q. Zhang, "Unsupervised euclidean distance attack on network embedding," in *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*, pp. 71–77, IEEE, 2020.

[66] S. Yu, J. Zheng, Y. Wang, J. Chen, Q. Xuan, and Q. Zhang, "Network embedding attack: An euclidean distance based method," in *MDATA: A New Knowledge Representation Model: Theory, Methods and Applications*, pp. 131–151, Springer, 2021.

A Source Code, Figures and Results

The source code of implementation is available in this public repository hosted in Github.

Supplementary figures are available in .png format in this google drive link.

All the experimental results are available in .csv format in this google drive link.