Degree Project in Computer Science and Engineering

Second cycle, 30 credits

Federated hierarchical clustering for RNA sequencing data

ASTRID NILSSON

Federated hierarchical clustering for RNA sequencing data

ASTRID NILSSON

Degree Programme in Computer Science and Engineering
Date: March 25, 2025

Supervisors: Boris Muzellec, Pawel Herman
Examiner: Ivy Peng
School of Electrical Engineering and Computer Science
Host company: Owkin
Swedish title: Federerad hierarkisk klustring för RNA-sekvenseringsdata

Abstract

Genetic data, and in particular RNA sequencing data analysis is crucial in our understanding of cancer, helping us prevent, diagnose and treat it. However, it is usually not easily accessible; because of its high sensitivity with regards to patient confidentiality, medical centers cannot often share it with research institutions. As a consequence, researchers are not able to leverage the power of big data in their models.

Federated computations have been introduced to find a trade-off between privacy and model quality. Federated algorithms are executed over distributed datasets, without the medical centers ever sharing their raw data directly, by communicating locally aggregated quantities instead. While this is rather straightforward for gradient-based methods, it is more challenging to apply to many unsupervised algorithms, especially those involving pairwise distances between points located in different data centers. Yet, genetic data analysis pipelines – and notably the quality control or exploratory steps – rely on hierarchical agglomerative clustering, which crucially depends on such distances.

We develop two approaches for federated agglomerative clustering of genetic data, and investigate their privacy properties, performance and complexity. We find the best quality for single and average linkage when testing them on bulk RNA sequencing data.

The first approach closely follows the classic agglomerative procedure; it approximates local point groups as their centroid once they reach a critical size in order to share them. We empirically observe that the critical size can be as much as 10% of the total data size while maintaining a great performance.

The second approach is based on computing the distance matrix centrally using random projections, a technique that can be employed for many other algorithms. It experimentally leads to satisfactory results for a significant dimensionality reduction. Privacy-wise, we attempt several attack models against this approach, which do not succeed on genetic data, indicating that collusion is the biggest threat for this method.

Keywords

Federated analytics, Hierarchical clustering, Transcriptomics

Sammanfattning

Genetiska data, och i synnerhet analys av RNA-sekvenseringsdata, är avgörande för vår förståelse av cancer och hjälper oss att förebygga, diagnostisera och behandla sjukdomen. Dessa data är dock vanligtvis inte lättillgängliga; på grund av deras höga känslighet när det gäller patientsekretess kan vårdcentraler ofta inte dela dem med forskningsinstitutioner. Följden blir att forskarna inte kan utnyttja kraften i stordata i sina modeller.

Federerade beräkningar har introducerats för att hitta en avvägning mellan integritet och modellkvalitet. Federerade algoritmer körs över distribuerade dataset utan att vårdcentralerna någonsin delar med sig av sina rådata direkt, utan kommunicerar istället lokalt aggregerade mängder. Även om detta är ganska enkelt för gradientbaserade metoder är det mer utmanande att tillämpa på många oövervakade algoritmer, särskilt de som involverar parvisa avstånd mellan punkter som ligger i olika datacenter. Ändå förlitar sig pipelines för analys av genetiska data - och särskilt kvalitetskontrollen eller de utforskande stegen - på hierarkisk agglomerativ klustring, som i hög grad beror på sådana avstånd.

Vi utvecklar två metoder för federerad agglomerativ klustring av genetiska data och undersöker deras integritetsegenskaper, prestanda och komplexitet. Vi finner den bästa kvaliteten för enkel och genomsnittlig koppling när vi testar dem på bulk RNA-sekvenseringsdata.

Den första metoden följer nära den klassiska agglomerativa proceduren; den approximerar lokala punktgrupper som deras centroid när de når en kritisk storlek för att kunna dela dem. Vi observerar empiriskt att den kritiska storleken kan vara så mycket som 10% av den totala datastorleken med bibehållen hög prestanda.

Det andra tillvägagångssättet bygger på att avståndsmatrisen beräknas centralt med hjälp av slumpmässiga projektioner, en teknik som kan användas för många andra algoritmer. Den leder experimentellt till tillfredsställande resultat för en betydande dimensionalitetsreduktion. När det gäller integritetsskydd försöker vi oss på flera modeller för att attackera denna metod, men dessa lyckas inte med genetiska data, vilket tyder på att hemliga överenskommelser är det största hotet mot den metoden.

Nyckelord

Federerade analyser, Hierarkisk klustring, Transkriptomik

Résumé

L'analyse de données génétiques, et en particulier de données de séquençage d'ARN, est cruciale pour notre compréhension du cancer, nous aidant à le prévenir, le diagnostiquer et le traiter. Cependant, ces données ne sont pas toujours facilement accessibles ; étant hautement sensibles et confidentielles, les centres médicaux peuvent rarement les partager avec des instituts de recherche. Par conséquent, les chercheurs ne peuvent pas tirer parti du potentiel des données massives pour créer leurs modèles.

Les calculs fédérés ont été introduits afin de trouver un compromis entre confidentialité et qualité des modèles. Les algorithmes fédérés sont exécutés sur des bases de données distribuées, sans que les centres médicaux aient à partager leurs données brutes directement, ceux-ci communicant plutôt des quantités agrégées localement. Si cela est relativement simple pour les méthodes de gradient, c'est plus difficile à appliquer à bon nombre d'algorithmes non-supervisés, notamment ceux impliquant des distances entre des points localisés dans des centres de données distincts. Or, l'analyse de données génétiques – particulièrement les étapes exploratoires ou de contrôle de qualité – repose généralement sur le regroupement hiérarchique agglomératif, qui dépend crucialement de telles distances.

Nous développons deux approches pour le regroupement agglomératif fédéré des données génétiques, et examinons leur degré de confidentialité, performance et complexité. Nous trouvons la meilleure qualité de regroupement pour les liens simple et moyen lors de tests sur des données de séquençage en masse d'ARN.

La première approche suit étroitement la procédure agglomérative classique ; elle remplace des groupes de points par leur centroïde lorsqu'ils dépassent une taille critique pour les partager. Nous observons empiriquement que la performance est maintenue pour une taille critique de 10% de la taille totale des données.

La seconde approche est basée sur le calcul de la matrice de distance via des projections aléatoires, une technique utile pour bien d'autres algorithmes. Expérimentalement, les résultats sont satisfaisants pour une réduction de dimension significative. En terme de confidentialité, nous tentons plusieurs attaques contre cette stratégie, qui échouent sur les données génétiques, indiquant que la collusion reste la principale menace pour cette méthode.

Mots-clés

Analyse fédérée, Regroupement hiérarchique, Transcriptomique

Acknowledgments

First, I would like to thank Boris Muzellec for all his guidance and kindness throughout this thesis. I learnt a lot and had a very enjoyable experience during this work thanks to him.

I would also like to thank every member of the Fundamental Machine Learning team at Owkin for their warm welcome and their valuable advice in this thesis.

Finally, many thanks to Pawel Herman for all his feedback and suggestions, which really helped improve the quality of this thesis.

Paris, France, March 2025
Astrid Nilsson

Contents

List of Figures

List of Tables

List of acronyms and abbreviations

ARI	adjusted Rand index
CCC	cophenetic correlation coefficient
DEA	differential expression analysis
DP	differential privacy
FedAVG	federated averaging
FL	federated learning
FMI	Fowkles-Mallows index
HC	hierarchical clustering
ICA	independent component analysis
JL	Johnson-Lindenstrauss
LSH	locality-senstive hashing
MDS	multidimensional scaling
ML	machine learning
MMD	maximum mean discrepancy
MSE	mean square error
MST	minimum spanning tree
PCA	principal component analysis
PDF	probability density function

RKHS	reproducing kernel Hilbert space
RNA-Seq	ribonucleic acid sequencing
SGD	stochastic gradient descent
SMPC	secure multiparty computation
TCGA	the Cancer Genome Atlas
VST	variance-stabilizing transformation

List of Symbols Used

The following symbols will be later used within the body of the thesis.

$\bar{x}$	Average of the coordinates of x,	page 36
$\lVert \cdot \rVert$	The Euclidean or ℓ_2 norm,	page 17
$\overset{d}{=}$	equals in distribution,	page 21
$S(p, \gamma)$	p-stable distribution of scale γ,	page 21

Chapter 1

Introduction

Genetic data analysis bolsters our understanding of how the human body functions and has many applications in healthcare. In particular, it can help prevent [1], diagnose [2], and treat [3] serious diseases such as cancer.

Here we focus on one data modality called ribonucleic acid sequencing (RNA-Seq) [4], which represents the gene expression as inferred from the sequencing of the RNA (*i.e* the resulting code from the transcription of the DNA). A particularity of RNA-Seq data is its high dimensionality, with tens of thousands of genes usually involved in the analysis. One of the main uses of RNA-Seq data is for differential expression analysis (DEA) [5], which compares the transcriptomic expression genewise and/or samplewise. DEA can help understand which genes are correlated and thus potentially involved in the same bodily functions. It can also uncover genes that are under- or over-expressed for certain subtypes, which is to say groups of people that share the same characteristics, such as a certain disease profile. A big part of the DEA process is quality control. It is notably achieved by checking visually or quantitatively that the data behaves as expected, using clustering or principal component analysis (PCA) for example.

In that regard, hierarchical clustering (HC) [6] – a stratified grouping procedure on data points – is often a key part of the genetic data analysis pipeline [7, 8]. Contrary to other clustering algorithms, it shows relationships between points at different levels, from strict to broad associations. The classic approach for this algorithm is bottom-up [9]; all points begin as their own cluster, and at each step, the two closest clusters are merged. The notion of cluster distance is determined by a parameter called the linkage. For instance, with single linkage, the distance between two given clusters is defined as the minimum distance across all pairs of points with one in the first cluster and another in the second cluster. HC can be used for several purposes, including identifying outliers or making sure that expected clusters are indeed present in the hierarchy.

In the context of RNA-Seq data analysis, there are unmistakable privacy concerns with the sharing of people' genetic code [10] so as to execute algorithms such as HC over them. Indeed, genetic sequences contains sensitive information that could be used to identify patients or discriminate them when in the wrong hands. For instance, an insurance company with access to someone's genetic information could decide to increase their pricing if they are likely to develop a certain disease. Given that it is critical to protect people's genetic confidentiality, a lot of genetic data is stored privately by medical institutions and cannot be leveraged for machine learning (ML) algorithms, or with increased difficulty. One way to preserve confidentiality throughout computations is secure multiparty computation (SMPC) [11]. It usually relies on cryptography tools to perform secure operations on the data, which is often quite costly in practice. Another privacy paradigm, Differential privacy (DP) [12], is a theoretical framework to analyze the anonymity provided by a mechanism. It is usually used when adding random noise to the outputs, in cases where the averaged output should not be affected too much by the extra noise. However, these approximations are not adapted for genetic data analysis, which requires precise computations.

Recently, the paradigm of federated learning (FL) has been proposed [13, 14] in order to deploy training of ML models over a network of distributed clients that want their data to remain private. To that end, the raw data stays in the local clients at all steps of the process, and only model parameters or non-sensitive information is communicated in the network. The typical FL setting is that of [13], in which the goal is to train a neural network among different clients who communicate with a central server. At each epoch, every client executes a training step with the current model, then sends its local gradient to the server. The latter averages all gradients to update the model, which can now be sent back to the clients. Lately, the concept of federated analytics [15] has been explored, in order to extend the principle of FL to other data science algorithms which are not gradient-based. However, federated algorithms are still lacking in many statistical pipelines, whether in visualization, clustering or specific to a certain application domain. Analysis of genetic sequencing data usually relies on the latter type of pipelines, meaning that it is currently hard to deploy a federated set-up to exploit genetic data securely. In particular, HC is a staple of genetic data analysis but has no standard federated equivalent. Therefore, our work aims at studying this specific issue to enhance federated pipelines on medical data.

1.1 Research Question

Federated analytics is a relatively new domain, with existing efforts on federating methods for PCA [16] or k-means clustering [17] for example. Currently, there is no well-established federated algorithm for HC. Therefore, the goal of this project is to study a federated approach to HC for practical use by data scientists. We propose the following research question:

What is the effect of federating the HC of genetic data on performance and complexity, compared with the equivalent unfederated algorithms? Additionally, how do such federated approaches measure up in terms of data privacy?

1.2 Purpose

Our purpose is to provide a federated algorithm for a standard HC approach with pooled-equivalent or close performance for a high privacy level on the genetic sequencing data.

Thus, the contribution is two-fold. On the one hand, we propose unsupervised federated analytics tools; while we focus on HC in this work, we assume the approach could potentially help federate other unsupervised algorithms utilized by bioinformaticians.

On the other hand, the work can bolster federated analytics pipelines in real applications, such as DEA in genetics. Clustering is a staple in exploratory data analysis, which can help uncover patterns, errors and outliers in the data before developing a model. Thus, having a federated HC algorithm would be very helpful in many projects where the data access is federated only. Contrary to many clustering algorithms, it does not require to input the number of clusters beforehand, making it a compelling alternative to k-means in the federated toolbox.

The project also adresses concerns about data privacy in artificial intelligence algorithms. It raises ethical questions in the sense that the privacy level of the algorithm has to be extensively studied so as to prevent data leaks.

1.3 Research Methodology

We develop federated methods for HC with the specific application of RNA-Seq data in mind. This has several implications. First, our method should be robust for high-dimensional data. Then, privacy-wise, we assume that genewise information is non-sensitive, whereas samplewise information is strictly confidential and should not be shared with the server.

In this work, we propose methods for both genewise and samplewise clustering. We analyze them in terms of performance – if not pooled-equivalent, using HC metrics on real RNA-Seq datasets – complexity and communication cost. We also examine the privacy level, by observing what quantities are shared and what can be learnt from them. Thus, we evaluate algorithms both in terms of theoretical aspects and empirical properties on real gene expression data.

1.4 Delimitations

First, all federated experiments in this project are simulated. While this means we cannot judge the time complexity and communication price empirically, it signifies we can focus more on the algorithmic perspective, as well as the practical quality and privacy in our experiments. Undoubtedly, the end goal is to adapt our developed methods into a federated software so that it can be used in practice.

Then, note that we only study HC from the federated viewpoint, and not other parts of the genetic data analysis pipeline. In particular, we do not federate preprocessing methods for RNA-Seq data. In our experiments, we assume that they are available and thus that our distributed data can be preprocessed as pooled while remaining in the medical centers.

Finally, we try to rely as little as possible on privacy-enhancing paradigms other than the federated one. Mainly, we would like to avoid SMPC because of its high complexity cost, and DP as adding noise to our outputs is often incompatible with the precision required in bioinformatics.

1.5 Structure of the thesis

Chapter 2 expands on the notions of federated computations, HC, RNA-Seq and private algorithms. Chapter 3 introduces the methodology we use to solve our problem. In particular, in section 3.3, we present our algorithms for federated clustering and analyze them theoretically. Chapter 4 displays our empirical results on real and synthetic data. Finally, we discuss our findings in chapter 5 and explore limitations and possible future work in chapter 6.

Chapter 2

Background

This chapter first describes the principles of federated computations. Then, it provides information about HC algorithms. It also presents RNA sequencing and gene expression analysis. Additionally, it describes private distance computation techniques as well as related privacy-preserving work.

2.1 Federated computations

Most ML algorithms are trained in a centralized fashion: the server aggregates the data from different sources before training a model with it. This creates obvious privacy issues when the data is sensitive, as is often the case in defense or pharmaceutical – including genomics-related [10] – applications. This has led researchers to investigate a *federated* setting, where the data remains in the data centers but the model still benefits from all datasets.

2.1.1 Federated learning

FL, introduced in [13, 14], is a paradigm in which a model is trained by a server using local datasets it does not have access to. In the classical federated setting, at the start of each epoch, the model parameters are shared with all centers, who perform one or several rounds of local training. Next, they each send their gradients to the central server, which performs an average of the local gradients to update the weights. Several strategies have been developed to better this federated averaging (FedAVG) scheme to reduce the number of communication rounds and improve performance in heterogeneous datasets [18, 19].

FedAVG is especially effective given that most deep learning problems optimize a seperable loss, *i.e* a loss $\mathcal{L}_{tot}$ that is a function of the individual losses $\mathcal{L}$ on each data point: $\mathcal{L}_{tot}(\theta, x) = f((\mathcal{L}_{tot}(\theta, x_i))_{1 \leq i \leq n})$ where $x \in \mathbb{R}^{n \times d}$ is the training data, θ the trained parameters and $f : \mathbb{R}^n \to \mathbb{R}$ a function. This is the case of popular losses like the mean square error (MSE) loss or the cross-entropy loss for instance.

Two major distribution settings exist in FL [20]:

- cross-device FL, in which a large number of small devices such as mobile phones store the data, and training can be completely decentralized
- cross-silo FL, where a small number of entities with high computational power own larger datasets

A significant difference between the two is that devices are usually allowed to drop out of training at any step, for instance when they have low battery or no internet connection. On the other hand, in cross-silo FL, clients are assumed to be stateful: they are present at every round of communication.

The cross-silo setting is especially relevant in healthcare, where hospitals have sensitive medical data which they do not want to share. With FL, they are able to create a powerful model based on more than their own patients' data [21].

2.1.2 Federated analytics

The federated paradigm has mostly been applied to deep learning. However, a lot of ML pipelines include other critical steps such as normalization, clustering, visualization or statistical methods. This is notably the case in bioinformatics and RNA-Seq data analysis. Thus, the federated framework has been extended to *federated analytics* [15].

For instance, [22] have developed a federated version of a preprocessing protocol, the Yeo-Johnson transform. They have reduced the problem to a log-likelihood maximisation problem solved with an exponential search. PCA has also been studied in this context, with the different federated algorithms analysed and compared in [16].

In terms of clustering, k-means has received the most attention. The authors of [17] have proposed a solution based on mini-batch gradient descent. In [23], a local $k^{(z)}$-means is run for each client $C^{(z)}$ and the cluster centroids are then sent to the server. The latter then performs one round of Lloyd's algorithm to cluster these into k groups.

Another clustering approach is explored in [24], where they circumvent the difficulties of unsupervised federated analytics by reducing the problem to a traditional FL one. More specifically, they first train an autoencoder on the data with FL and then share the latent space representation with the server, which can then use any clustering algorithm.

To our knowledge, there is no purely federated approach for HC in the literature. All privacy-preserving methods seem to rely on other notions such as cryptography or differential privacy (more details in section 2.5).

2.1.3 Attacks on federated computations

In the general case and in the absence of other privacy-preserving tools, FL is not guaranteed to protect the privacy of clients. Even when the raw data is not directly shared with the server, this does not mean that an attacker cannot learn anything. The quantities that are shared to the server during computations – $e.g$, the gradients in FL – can potentially leak information about the local data. Let us present a few common attacks:

- the property inference attack [25] aims at deducing some properties of the training data, such as class proportions or metadata
- the membership inference attack [26] tries to find whether a particular sample was used or not during model training
- the reconstruction attack uses model parameters, confidence scores and/or gradients [27] in order to retrieve a training sample

These attacks have mostly been developed for machine learning models and thus apply mostly to FL and less to federated analytics. Nevertheless, they provide us with useful notions for analyzing the privacy capacity of any potential strategy for federated clustering.

When defending against potential attacks, it is necessary to consider the trustworthiness of the different participants. If some clients are malicious, they might work with the server so as to decrypt the data of honest clients, meaning that great precautions should be taken. Another possible model is the honest-but-curious one, in which participants obey the instructions, but use any information sent by the server so as to learn about other centers' data.

2.2 Hierarchical clustering

HC [6] is a procedure which groups similar objects together to create a stratified structure of a dataset. The resulting hierarchical tree is often presented as a *dendrogram* (see Figure 2.1). The dendrogram helps understand the relationships between objects at different scales: clusters at the bottom correspond to narrow categories, whereas clusters at the top correspond to more high-level associations.

There are two main families of algorithms for HC:

- *agglomerative* clustering [9], where each point starts out as its own cluster. The algorithm successively picks two clusters to merge until there is one cluster left containing all the items
- *divisive* clustering [28], where all points are initially in the same cluster. At each step, a cluster is split in two, until all clusters consist of one object only

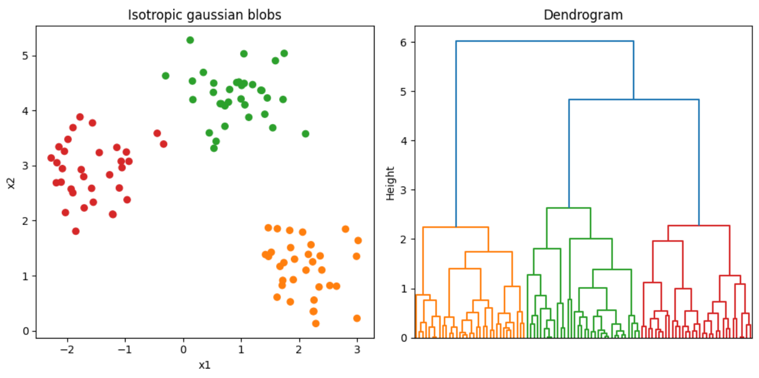

Figure 2.1: Dendrogram (right) generated by agglomerative clustering of isotropic Gaussian blobs (left) with complete linkage and Euclidean metric. Each inversed U-shaped link corresponds to a merge between two clusters, with the height indicating the distance between theses clusters.

One notable advantage of HC over other clustering algorithms is its lack of hyperparameters. In particular, compared to the k-means, there is no need to assign the number of clusters in advance: one can observe the produced hierarchy and then choose a cutoff to flatten the clustering. Besides, it is deterministic[1], where k-means requires an initializaton in order to be

[1] given strictly increasing cluster merge distances or a way to uniquely choose between two merges of equal cluster distance

reproducible.

Some common uses for HC include:

- identifying outliers in a dataset in order to filter them out
- qualitatively verify that data groups expected to be seperated according to true labels (such as one healthy, one with a disease) are effectively in distinct clusters
- serving as a baseline for other algorithms, such as classification or consensus clustering, usually by flattening the clustering first

Here we choose to focus on agglomerative clustering, as it is the most prevalent kind in RNA-Seq data analysis (more details available in 2.3).

2.2.1 Agglomerative clustering

Agglomerative clustering is the most prevalent kind of HC. It uses as input the pairwise distance matrix, which can be obtained with any metric such as the Euclidean metric (ℓ_2), city-block (ℓ_1) metric, or cosine dissimilarity[1] for instance. It requires a *linkage* criterion, which defines the distance between two data clusters, such that at each step of the algorithm, the two closest clusters are combined.

We adopt the SciPy [29] convention for the output of agglomerative clustering. The clustering of n points outputs a matrix $Z \in \mathbb{R}^{(n-1)\times 4}$ matrix, where each row represents a merging step, and stores in order the two indices of the merged clusters, the merge distance and the number of points in the new clusters.

2.2.1.1 Linkage methods and properties

The linkage method is a way to introduce a distance between clusters. Some common linkages are the single, complete and average linkage – respectively based on the minimum, maximum and average pairwise distance between points in the two clusters. A lot of linkages, including those, fall under the Lance-Williams algorithm family. This means cluster distances can be updated in constant time after each merge using the following recursive

[1]Note that cosine dissimilarity is actually a semi-metric: it does not fulfil the triangle inequality property. In this work, we use the terms distance and metric as an abuse of terminology and not in the true mathematical sense.

formula. More specifically, let A, B and C be three clusters such that A and B are selected to be merged. Then:

$$d(A \cup B, C) = \alpha_A d(A, C) + \alpha_B d(B, C) + \beta d(A, B) + \gamma |d(A, C) - d(B, C)|$$

where α_A, α_B, β and γ are parameters which may depend on the size of A, B and C. The definitions and update formulas for several usual linkages are presented in table 2.1. Note that centroid and Ward linkage are only defined for the Euclidean distance.

Method	$d(A, B)$	Lance-Williams update formula for $d(A \cup B, C)$
single	$\underset{a \in A, b \in B}{min} d(a, b)$	$min(d(A, C), d(B, C))$
complete	$\underset{a \in A, b \in B}{max} d(a, b)$	$max(d(A, C), d(B, C))$
average	$\frac{1}{\lvert A\rvert\lvert B\rvert} \sum_{a \in A, b \in B} d(a, b)$	$\frac{\lvert A\rvert d(A, C) + \lvert B\rvert d(B, C)}{\lvert A\rvert + \lvert B\rvert}$
centroid	$\lvert\lvert \vec{c_A} - \vec{c_B} \rvert\rvert$	$\sqrt{\frac{\lvert A\rvert d(A, C)^2 + \lvert B\rvert d(B, C)^2}{\lvert A\rvert + \lvert B\rvert} - \frac{\lvert A\rvert\lvert B\rvert d(A, B)^2}{(\lvert A\rvert + \lvert B\rvert)^2}}$
Ward	$\sqrt{\frac{2\lvert A\rvert\lvert B\rvert}{\lvert A\rvert + \lvert B\rvert}} \lvert\lvert \vec{c_A} - \vec{c_B} \rvert\rvert$	$\sqrt{\frac{(\lvert A\rvert + \lvert C\rvert) d(A, C)^2 + (\lvert B\rvert + \lvert C\rvert) d(B, C)^2 - \lvert C\rvert d(A, B)^2}{\lvert A\rvert + \lvert B\rvert + \lvert C\rvert}}$

Table 2.1: Cluster distance definition and update formula for several linkage methods

One desirable property of a linkage method is *reducibility*. It is verified when: for all clusters A, B, C such that A and B are nearest neighbours, $d(A \cup B, C) \geq min\,(d(A, C), d(B, C))$. Notice that this is true for single, complete, average and Ward linkage, but not for centroid linkage. A non-reducible linkage often leads to inversions in the dendrogram representation of the clustering, which can make it hard to interpret. An example of this can be seen on figure 2.2, as the centroid of points A and B is closer to C than are A or B.

2.2.1.2 Time complexity

For a dataset with n points and d features, the initial distance matrix computation has a time complexity of $\mathcal{O}(n^2 d)$. The merging phase can always be implemented in $\mathcal{O}(n^3)$ for all linkages with distance update in constant time. This is achieved by computing the nearest cluster neighbours at each step and

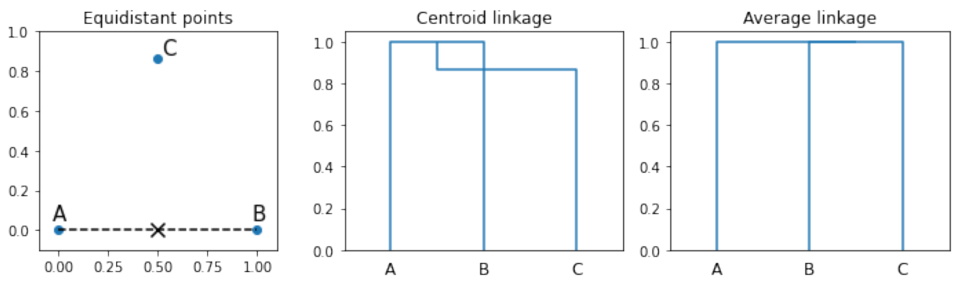

Figure 2.2: Dendrogram inversion caused by non-reducibility of centroid linkage. The centroid of points A and B is indicated by the cross.

merging them (see pseudo-code in 1). Using a heap to store pairwise distances between clusters, the complexity becomes $\mathcal{O}(n^2 \log n)$.

For some linkages – including single, complete, average and Ward – implementations in $\mathcal{O}(n^2)$ exist. Typically, one possible way is by using the nearest-neighbour chain algorithm [30, 31]. It applies to reducible linkages that are also insensitive to merge order, and works by successively merging clusters who are each other's nearest neighbour (but not necessarily the global pair of nearest neighbours).

Algorithm 1 Naive agglomerative clustering

```
Require: pairwise distance matrix $D_X$ between samples of $X$, metric $\mu$,
    linkage $L$
  for $k = 1, \dots, n$ do                    ▷ initialize clusters and distances
      $C_k \leftarrow \{k\}$
      for $l = 1, \dots, k-1$ do
          $d(C_k, C_l) \leftarrow D_X[k, l]$
      end for
  end for
  for $i = 1, \dots, n-1$ do                  ▷ merge the two closest clusters
      $k, l \leftarrow \arg\min_{k,l} d(C_k, C_l)$
      $C_k \leftarrow C_k \cup C_l$
      Inactivate $C_l$
      for $j = 1, \dots, n$ do
          if $C_j$ active then
              $d(C_k, C_j) = \textsc{UpdateDistance}(C_k, C_l, C_j, \mu, L)$
          end if
      end for
  end for
```

For single-linkage clustering, the classic implementation is based on the minimum spanning tree (MST) problem. Given a connected undirected graph of vertices, the MST is the subset of the edges that connects all points together without any cycle and minimizing the sum of edge weights in the subset.

Note that both problems are closely related [32]. Indeed, consider the subset E of edges that form the MST of a dataset and sort it by increasing weight. For simplicity, let us assume weights are strictly increasing. Then: the i-th edge of the MST connects points a and b with weight w *i.i.f* the i-th merge during single-linkage clustering is between the clusters that respectively contain a and b, with a cluster distance of $d(a, b) = w$. Thus, finding the MST of a dataset gives the merge order and distances for its single-linkage clustering.

2.2.2 Approximate agglomerative clustering

Because of the $\mathcal{O}(n^2)$ or $\mathcal{O}(n^3)$ time complexity, some approximate algorithms have been developed for practical clustering of large datasets. They are pertinent in the context of federated analytics, as they could potentially use techniques that are easier to federate than the standard algorithms. The lower complexity could also translate into smaller messages exchanged between the clients and the server, leading to lower communication times. This is critical as network communication is often a bottleneck in distributed computations.

[33, 34] propose faster algorithms for several linkages, relying on approximate nearest neighbours queries. Their scheme is based on locality-senstive hashing (LSH) [35], which consists in building hash functions such that closer objects have a bigger probability of ending up in the same hash bucket.

Another relevant approach for our work is [36], which computes only a subsample of the pairwise distances, and defines the cluster distance based only on the known object distances. More precisely, they sample random pivot objects and compute distances from the dataset to the pivots. Then, they calculate actual distances for all pairs whose distance to all pivots is small, and compute additional distances between randomly picked pairs. This approach can help in the cross-silo setting, where pairwise distances in each silo are locally available but distances cross-silo are hard to compute.

2.3 RNA-Seq data analysis

2.3.1 RNA sequencing

RNA-Seq [4] is a standard method to analyze gene expression in samples. RNA results from the transcription of the DNA in the cell, so its examination provides insights about genetic expression. Typically, *transcriptomics* – the study of RNA transcripts – lets us know which genes are more active in a given cell, helping us understand its role in the body for example. Given that RNA is then translated by the cell into proteins, RNA-Seq data can also serve as a proxy for protein levels, in order to further study phenotypes [37].

The popularity of RNA-Seq has grown over the years as sequencing technologies have become cheaper and more precise. There are three main kinds of procedures [38, 39]:

- *bulk* RNA-Seq analyzes the expression of a large number of cells at once, with cell membranes being destroyed before the sequencing. It is useful to study general patterns and it produces big amounts of data at a low cost, but does not give information about individual cells

- *single-cell* RNA-Seq can evaluate the expression of one cell at a time, which leads to a more powerful analysis and makes it easier to study gene expression in heterogeneous settings

- *spatial transcriptomics* provide localization information for single-cell expression data. In bulk and single-cell sequencing, the spatial context is lost. This new paradigm thus allows for even more precise investigation of gene expression

The methods above are presented in increasing order of cost and complexity. Even if spatial transcriptomics technically produce the most information, bulk RNA-Seq is still widely used as it is largely sufficient for certain purposes, having led to the discovery of biomarkers for cancer for instance.

Bulk RNA-Seq data is often modeled with Poisson or negative binomial law. Indeed these are discrete laws and sequencing data corresponds to the number of counts per gene, which are integers. Negative binomial is often preferred to Poisson as the latter has equal variance and mean. In practice, it is observed that RNA counts are overdispersed, *i.e* the variance grows faster than the mean. It is worth mentioning that RNA-Seq data is very high-dimensional, containing the expression of tens of thousands of genes. For reference, the human genome consists of about 100 000 genes.

2.3.2 Differential expression analysis

One of the most notable applications of bulk RNA-Seq is DEA [5]. This consists in finding genes that have different levels of expressions in varying conditions. For instance, some genes can be differentially expressed between cancerous and healthy cells, and thus become potential targets for cancer treatment. The DEA pipeline has several preprocessing steps; after normalization, data exploratory analysis is performed for quality control.

Clustering, especially agglomerative, is an essential part of the DEA framework and more generally RNA-Seq analysis [7, 8]. Note that agglomerative clustering can be used to cluster both genes and samples, which have different purposes. Clustering of genes helps visualize which genes have similar expression profiles, which could for instance mean they are involved in the same biological function. Clustering of samples is useful for quality control. For instance, it can be used to make sure the data is not clustered according to non-biological factors – such as who performed the sequencing of each sample. In the federated clustering setting, genewise clustering corresponds to a *vertical* partition of the data and samplewise clustering to a *horizontal* partition. For genewise clustering, the cosine distance and average linkage are often used together [40]. For sample clustering, the Euclidean, cosine and Pearson correlation are common metrics, using either average, complete or Ward linkage.

The best choice of metric and linkage is of course dataset-dependent. For instance, single linkage performs well for well-seperated clusters but has a chaining effect otherwise, while complete linkage produces compact clusters. Average and Ward linkages usually offer a balance between separability and compactness.

The dendrogram representation of HC is often combined with a heatmap (see figure 2.3) in order to visualize the differential gene expression across samples and genes.

2.4 Private distance computation

Agglomerative clustering only requires the pairwise distance matrix as input and not the coordinates. Therefore, instead of developing a federated clustering algorithm, another valid approach would be for the server to first privately compute pairwise distances, and then apply a standard clustering algorithm on the distance matrix. In the field of private distance computation, multiplicative perturbations have become popular for the ℓ_2 norm. Indeed,

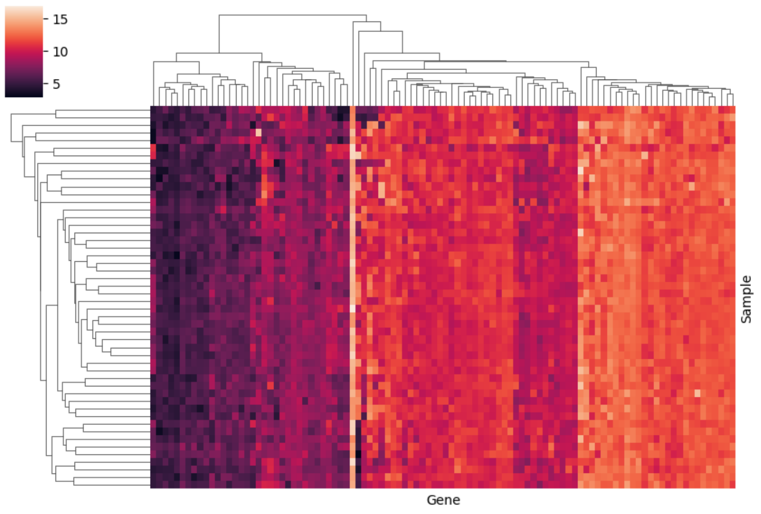

Figure 2.3: Heatmap of the gene expression of 50 samples across 100 genes from a breast cancer dataset (TCGA-BRCA [41]). Raw counts are preprocessed with the variance stabilizing transformation [42].

they allow the server to retrieve a distorted version of the dataset, but whose distance matrix is approximately preserved. They are also rather inexpensive compared to cryptography-based approaches and tend to preserve geometric properties better than additive noise for instance.

This can somewhat easily be adopted in the cross-silo federated framework: the centers just have to agree on a seed to generate the multiplication matrix and then send their transformed data to the server 2.

2.4.1 Orthogonal perturbations

One possible class of multiplicative matrix is rotations, or more generally orthogonal matrices. Recall that a matrix $R \in \mathbb{R}^{d\times d}$ is orthogonal if and only if $R^T R = RR^T = I_d$. Indeed, they are isometries for the ℓ_2 norm: if $X \in \mathbb{R}^{1\times d}$ then $||XR|| = ||X||$. By preserving pairwise distances, they have been used for many purposes such as clustering [43], visualization, classification [44] or kernel methods [45]. The authors of [45] even suggest to use a matrix $R \in \mathbb{R}^{d\times k}$ with $k > d$ such that $RR^T = I_d$ (but of course $R^T R \neq I_k$.) Then, the Gram matrix is still preserved but the number of features is not shared with the server.

Without prior knowledge, it is impossible for an attacker to infer the original data from the transformed dataset. However, this method is highly sensitive to collusion: if a center communicates the transformation to the

Algorithm 2 Private distance computation scheme

```
Require: $M$ centers, with dataset $X_{(k)} \in \mathbb{R}^{n_{(k)} \times d}$ in center $C_{(k)}$, final
    dimension $p$
    Centers $C_{(1)}, C_{(M)}$ agree on a shared seed $s$
    for center $k = 1, \dots, M$ do
        $Q = \textsc{RandomPerturbation}(d, p, s)$                ▷ $Q \in \mathbb{R}^{d \times p}$
        $Y_{(k)} \leftarrow X_{(k)} Q$
        Send $Y_{(k)}$ to the server
    end for
    for $k = 1, \dots, M$ do
        Server computes $D(Y_{(k)}, Y_{(k)})$          ▷ equal to $D(X_{(k)}, X_{(k)})$
        for $l = k + 1, \dots, M$ do
            Server computes $D(Y_{(k)}, Y_{(l)})$      ▷ equal to $D(X_{(k)}, X_{(l)})$
        end for
    end for
```

server, it can access the whole decrypted dataset. Additionally, several attacks have been developed for the case the attacker has additional knowledge.

The authors of [44] propose an attack based on independent component analysis (ICA) [46] in the case the attacker knows the distribution of the features. As ICA reconstruction does not protect the feature order, the distribution is used to match the reconstructed and original probability density functions (PDFs). This might be hard in practice in a high-dimensional space or if features have similar PDFs.

In [47], two attacks are developed. The first one assumes some inputs X and their projection Y are known. For each unknown point x of projection y, they derive the probability of reconstructing the input with a relative error ϵ. This probability depends on the distance from y to the vector subspace generated by Y. In the extreme case where y belongs to the generated subspace, the probability is 1 for all ϵ, *i.e* the reconstruction is exact.

The second attack assumes some samples from the same probability distribution as the inputs are known, and uses PCA to reconstruct the inputs. It has limited use in high dimension (as is the case with RNA-Seq data) as one of the steps has a complexity that includes a factor of $\mathcal{O}(2^d)$, where d is the dimension of the feature space.

2.4.2 Random projections

2.4.2.1 Definition

Random projections can alleviate some of the privacy issues of orthogonal projections, while approximately maintaining pairwise distances. The Johnson-Lindenstrauss (JL) lemma [48] states the following. Let $0 < \epsilon < 1, X \in \mathbb{R}^{n \times d}$ and $k > 8\ln(n)/\epsilon^2$. There exists a linear map $f : \mathbb{R}^d \to \mathbb{R}^k$ such that for all $u, v \in X$:

$$(1-\epsilon)||u-v||^2 \leq ||f(u)-f(v)||^2 \leq (1+\epsilon)||u-v||^2$$

It is notably verified with projection matrices $Q \in \mathbb{R}^{d \times k}$ with i.i.d Gaussian entries sampled from $N(0, \frac{1}{k})$; for all $u, v \in X$:

$$(1-\epsilon)||u-v||^2 \leq ||uQ-vQ||^2 \leq (1+\epsilon)||u-v||^2$$

Note that the sampling bound $k = 8\ln(n)/\epsilon^2$ depends on the number of samples n but not on the number of features d. This is very convenient for us: RNA-Seq data is usually comprised of tens of thousands of genes, while the number of samples in a given study is usually around a few hundreds – maybe thousands – at most. Depending on the complexity of the downstream task on the server, the dimensionality reduction can also be useful to curtail costs.

2.4.2.2 Bounds for specific problems

In the case of specific distance-based applications, some better sampling bounds have been found. In particular, [49] propose a lower bound for the minimum spanning tree (MST) problem. The authors give a sampling bound of $\mathcal{O}(\frac{1}{\epsilon^2}(d_X \log(\frac{1}{\epsilon}) + \log\log n))$, where $d_X \leq \log X$ is the *doubling dimension* of dataset X, representing its intrinsic dimensionality[1]. As seen in section 2.2.1.2, finding the MST of a dataset is equivalent to clustering it with single linkage. Therefore, this also gives us a sampling bound for single linkage when using the cost of the associated MST as a metric.

For k-means and Euclidean k-medians clustering – which respectively minimize the squared and classic Euclidean distance between points and their cluster centroids – the state of the art bound has been derived by [50]. They show that the cost of any clustering, including the optimal solution, is

[1]More precisely, let λ_X be the minimum $\lambda > 0$ such that for all $x \in X$ and $r > 0$, there exists $S \subseteq X$, $|S| \leq \lambda$ such that $B(x,r) \cap X \subseteq \bigcup_{s \in S} B(s, \frac{r}{2})$. The doubling dimension of X is defined as $d_X = \log \lambda_X$.

preserved with $1 + \epsilon$ error with a sampling dimension of $\mathcal{O}(\frac{1}{\epsilon^2} \log(\frac{k}{\epsilon}))$. This bound depends on k which is usually much smaller than n.

2.4.2.3 Privacy-preserving properties

Random projections have mostly been used for dimensionality reduction, by projecting data to a lower dimension $k < d$. Nevertheless, they can also be relevant for privately computing distances as they are more difficult to attack than orthogonal projections.

First, they produce approximations of the pairwise distances, and the sample size can be adjusted to get a trade-off between utility and privacy. Then, they are not as vulnerable to the attacks on orthogonal projections. Typically, the ICA attack as in [44] fails as the number of observed dimensions k is smaller than the number of sources d. Similarly, the PCA attack of [47] works because the covariance matrix of the projected dataset has the same eigenvalues as the covariance of the original dataset. However, the covariance matrix of a dataset projected with a Gaussian-distributed matrix converges to αI_d for some $\alpha \in \mathbb{R}$.

The authors of [51] have proposed several attacks based on prior knowledge of the data and on whether the random projection matrix is known or not. Their method used underdetermined ICA assuming the original data is sparse and mutually independent, so that it can be modelled with the Laplace distribution. They use maximum a posteriori estimation when the data is non-sparse and correlated, in which case the data is assumed to follow a multivariate Gaussian distribution.

As explained in section 2.3.1, RNA-Seq counts are discrete and skewed, so they are usually modelled with a Poisson or negative binomial distribution. Besides, they are neither sparse nor Gaussian distributed, so the aforementioned attacks are usually not well-suited for bulk RNA-Seq. On the contrary, single-cell RNA-Seq data is very sparse, so potentially exposed to the first attack from [51].

While most preprocessing methods in bulk RNA-Seq data preserve the skewness, feature Gaussianization – *e.g* with the Box-Cox or Yeo-Johnson transformation – is sometimes used, notably for survival analysis [52], even in the federated setting [22]. Such a procedure could make the random projection of RNA data vulnerable to the second attack presented in [51].

2.4.3 Multidimensional scaling

It is worth mentioning that possessing the pairwise Euclidean distance matrix $D = d(X, X)$ of a dataset X is perfectly equivalent to having a version of X up to affine isometry, *i.e* up to orthogonal projection and translation. Indeed, given a valid Euclidean distance matrix D, obtained from a dataset X, one can compute a dataset X' such that $d(X', X') = D$ with multidimensional scaling (MDS) [53, 54]. As X and X' have the exact same Euclidean distance matrix, they are equal up to affine isometry.

In the federated setting, this means that even assuming we can privately compute the distance matrix on the server – with cryptography techniques for instance – there could still be a privacy leakage through attacks based on distance-preserving transformations.

In the general case where the the norm is not l_2, MDS can be expressed as an optimization problem, but there is no trivial solution as in the classical Euclidean case.

2.4.4 Stable distributions

2.4.4.1 Definition

The random projection trick for ℓ_2 norm approximation is based on the centered normal distribution, which is a special case of a *stable* distribution. A distribution $\mathcal{D}$ over $\mathbb{R}$ is said to be stable if for all $X_1, X_2 \sim \mathcal{D}$ independent and $a, b > 0$, there exists $X \sim \mathcal{D}$ and $c > 0, d \in \mathbb{R}$ such that $aX_1 + bX_2$ and $cX + d$ have the same distribution.

In particular, let $p > 0$. $\mathcal{D}$ is *p-stable* if any realization $X \sim \mathcal{D}$ has a characteristic function $\mathbb{E}(e^{itX}) = e^{-\gamma|t|^p}$ where $\gamma > 0$ is the *scale* parameter. We note $\mathcal{D} \overset{d}{=} S(p, \gamma)$.

There are some famous analytical cases:

- the Gaussian distribution is 2-stable
- the Cauchy distribution is 1-stable
- the Levy distribution is $\frac{1}{2}$-stable

In the general case, p-stable distributions for $0 < p \leq 2$ are not analytical but they can still be simulated easily[55].

2.4.4.2 ℓ_p distance estimation

Let $\mathcal{D}$ be a p-stable distribution of scale 1. Then: for all $k \in \mathbb{N}$, for all $X, X_1, \ldots, X_k \sim \mathcal{D}$ i.i.d and for all $a_1, \ldots, a_k \in \mathbb{R}$:

$$\sum_{i=1}^{k} a_i X_i \overset{d}{=} \left(\sum_{i=1}^{k} |a_i|^p \right)^{\frac{1}{p}} X \tag{2.1}$$

The *scale* of the distribution above is then $\sum_{i=1}^{k} |a_i|^p$. This gives us the general scheme for a privacy-preserving approximation of the ℓ_p distance by estimating its scale. Precisely, let $u, v \in \mathbb{R}^d$, $X \in \mathbb{R}^{k \times d}$ with i.i.d entries of law $S(p, 1)$, then $y = Xu - Xv$ is a random vector with i.i.d entries of law $S(p, ||u - v||_p^p)$.

One possible estimator is the geometric mean, analyzed in [56, 57]:

$$\hat{\gamma} = \frac{\prod_{i=1}^{k} |y_i|^{\frac{p}{k}}}{\left[\frac{2}{\pi}\Gamma(\frac{p}{k})\Gamma(1 - \frac{1}{k}) \sin(\frac{\pi p}{2k})\right]^k} \tag{2.2}$$

It is unbiased and similarly to the JL lemma, it can approximate pairwise ℓ_p distances between n points up to a $1 \pm \epsilon$ factor for a sample size $k = \mathcal{O}\left(\frac{\log n}{\epsilon^2}\right)$. We are especially interested in the ℓ_1 distance, which is still regularly used in practice and is notably more robust to outliers than the ℓ_2 norm.

2.5 Related work

2.5.1 Secure multiparty computation

SMPC [11, 58] is a way for parties to compute a function over their private data without revealing it to the others. It is based on cryptography protocols, and can be used in place of or in addition to federated learning.

SMPC protects participants from external attackers, but also from fellow participants. It introduces two relevant adversary models to judge how clients may try to gather private data [59]:

- the *semi-honest* or honest-but-curious model, in which parties respect the protocol but use all the knowledge at their disposal – such as intermediate and final results – to infer information about the original data

- the *malicious* model, in which one or more participants actively disrupt the protocol, by colluding or feeding wrong inputs to the server for example

In the case of a central server performing computations on medical institutions' data, the semi-honest model is usually adapted, as in [60]. Indeed, hospitals want to protect their patients' data for moral, legal, and reputational reasons. They also do not trust outside entities with their data, and especially not the central server.

In the field of FL, a useful SMPC tool is secure aggregation, which lets users privately compute the sum over their inputs. This can notably be used by the server to average local model updates from clients [61]. Cryptography schemes can also be used for federated analytics, as in [62] which securely compute the cosine similarity matrix. Their technique relies on a random projection scheme followed by a secure Hamming distance protocol. Similarly to the random projection scheme for the ℓ_p norm, this could be applied to our problem as the cosine distance is often used for agglomerative clustering of RNA-Seq data.

We note that SMPC has been proposed for single-linkage clustering in the case of two communicating parties [63], with a scalable scheme for a large number of samples in a low-dimensional feature space (experiments are done for under 20-dimensional datasets). In our cross-silo setup for RNA-Seq data, we have more parties, connected to a central server, and the feature space is massive (tens of thousands of genes). Thus, in practice, the use of encryption techniques could have a prohibitive cost compared to other privacy-enhancing strategies.

2.5.2 Differential privacy

DP [12, 64] is another framework that aims to protect individual information in a group computation. The idea is usually to add noise to individual outputs so that their true value is masked, but that the aggregated statistic remains about the same. Typically, for FL, each client adds noise to their gradients before sharing them with the server.

Formally, a mechanism M is ϵ-differentially private if for any two datasets D_1 and D_2 that differ on one sample, and $S \subseteq \operatorname{im} M$:

$$\mathbb{P}(M(D_1) \in S) \leq e^{\epsilon}\, \mathbb{P}(M(D_2) \in S)$$

The intuition is that applying M to a dataset should give about the same result whether a single data point is present or not in the dataset. Choosing ϵ offers a utility-privacy trade-off: a smaller ϵ brings more privacy, but the added noise can affect the performance.

In the model above, each client is assumed to have access to the raw data of individuals. In case individuals do not trust the data curator, they can add noise to their data before sharing it. This is the setting of *local* differential privacy [65]; an algorithm M respects ϵ-local differential privacy if for all user data points x_1 and x_2, and all $S \subseteq \operatorname{im} M$:

$$\mathbb{P}(M(x_1) \in S) \leq e^{\epsilon}\, \mathbb{P}(M(x_2) \in S)$$

DP has many applications. It has been explored in the federated setting, for instance in federated learning [66], PCA [67] or k-means [68]. In [69], a locally DP algorithm for HC is developed, but is decentralized and not provably in polynomial time. The authors of [70] do propose a polynomial-time approximation in the global DP setting. Nevertheless, both solutions optimize Dasgupta's objective for HC [71], while we are more interested in the classic agglomerative algorithm with linkage functions, which is the standard in genetic analysis. Furthermore, we would ideally prefer an exact or close to exact solution for clustering of genetic data, whereas DP requires to add random noise to shared quantities.

Chapter 3

Methods

In this chapter, we provide an overview of the research process, the genetic expression dataset used in our experiments, as well as our experimental setup. We then present our original federated HC algorithms.

We first develop a straightforward method for genewise clustering based on the low privacy constraints on gene information.

For samplewise clustering, we decide to investigate two different tracks:

- one specifically focused on the standard agglomerative clustering algorithm, which introduces privacy while trying to be as close to the original procedure as possible
- one that can more generally apply to distance-based algorithms, by privately computing the distance matrix of the dataset

For all tracks, we focus on providing privacy under the federated framework, and do not want to largely base our privacy mechanisms on DP or SMPC. Indeed, usually, they respectively induce a loss of performance under noise addition in the first case, and high computation costs in the second. We do not forbid ourselves from resorting to them if appropriate, but we consider that they should not be at the heart of our algorithms as we want to give emphasis to real-world practical usage, which is not compatible with large additive noise or complex encryption schemes.

Methods are analyzed according to several criteria:

- their empirical performance on real RNA-Seq data, and at times on toy datasets to exhibit certain behaviours of the developed algorithms
- their theoretical complexity, including the communication cost between clients and the server

- their degree of privacy, by studying what is shared and what can potentially be recovered

3.1 The Cancer Genome Atlas

For our experiments on real RNA-Seq data, we use 8 datasets from the Cancer Genome Atlas (TCGA) program [41], each corresponding to a different cancer type. Under this program, a massive amount of genomic-related data has been published to help researchers improve cancer diagnosis, treatment and prevention. The cancer types in the atlas have been selected according to several reasons, including health impact, quality and quantity of the collected data, and adequate patient consent (more information about ethical concerns in section 5.2.1).

In our case, each data sample consists of two parts: the transcriptomic data and the metadata. The transcriptomic data corresponds to raw gene counts, which means the data is already aligned with a reference genome, so that we obtain a table with the number of counts for each gene. Each dataset is composed of hundreds of samples and around 50000 genes (see table 3.1).

Cancer type	Acronym	Number of samples	Number of genes
breast invasive carcinoma	BRCA	1188	53633
colon adenocarcinoma	COAD	506	53325
lung adenocarcinoma	LUAD	532	53439
lung squamous cell carcinoma	LUSC	517	53523
pancreatic adenocarcinoma	PAAD	179	51642
prostate adenocarcinoma	PRAD	452	53281
rectum adenocarcinoma	READ	170	50777
skin cutaneous melanoma	SKCM	427	53119

Table 3.1: The different transcriptomics datasets from the Cancer Genome Atlas

The metadata contains relevant information about the patient. It notably includes:

- potentially sensitive personal information such as gender, ethnicity or age

- the tissue source site, which is the medical center that collected the sample

- medical data related to the tumour, such as the stage of the cancer or a prior diagnosis

3.1.1 Federated simulations with TCGA

It is important to note that the tissue source site or even more broadly the geographic region offers a natural way for us to split samples into centers to simulate federated analytics. This is crucial because different clinical centers have different instruments and procedures, which can translate into noticeable heterogeneity between the datasets of these centers. By following a split by actual source site, or by grouping source sites together into artificially bigger centers, we have a more realistic setup than if we used a random split, or had a homogeneous dataset.

In this work, we decide to regroup clinical centers by region in order to mimic the federated cross-silo setting, and obtain a few centers with decently-sized datasets. More specifically, we follow the same grouping as in [72], in order to obtain seven centers: four from the United States – Northeast, South, West, Midwest – Canada, Europe and a client containing the rest (which includes clinical centers from different Asian countries, Brazil, Nigeria and Australia). This allows a split between artificial centers of heterogeneous sizes, with:

- the smallest center containing as low as 0.59-7.9% of the data
- the biggest center containing about 22-58% of the total data

3.1.2 RNA-Seq data preprocessing

Preprocessing of bulk RNA-Seq data is necessary in order to have comparable samples and/or genes. Several factors need to be accounted for[1], including:

- gene length: given two genes with similar levels of expression, the profiling method causes the gene with the longer sequence to have more counts, proportionally to that length
- sequencing depth: if two samples have the same level of expression per gene but the first is sequenced with a bigger depth, the read counts are proportionally bigger

[1]A more detailed description of the sequencing discrepancies and normalization methods can be found at https://hbctraining.github.io/DGE_workshop_salmon_online/lessons/02_DGE_count_normalization.html

One common normalization method for samplewise gene comparison is the median of ratios method [73] from the DESeq2 [74] package in R, which addresses the sequencing depth issue. We consider a dataset $X \in \mathbb{N}^{n \times d}$ consisting of n samples and d genes. The method assumes most genes are not differentially expressed, and creates a pseudo-reference sample corresponding to the geometric mean of all samples for each gene: $\mu_j = \left(\prod_{i=1}^{n} X_{i,j} \right)^{\frac{1}{n}}$. For a given sample, its *size factor* is defined as the median of the ratio of the sample's counts over the reference counts: $s_i = median \left(\frac{X_{i,j}}{\mu_j} \right)$. The normalized counts X_{norm} are then defined by $X_{norm,i,j} = \frac{X_{i,j}}{s_i}$. Note that genes with at least one sample with zero counts are filtered out for the size factor estimation, as the geometric mean ends up being 0.

As mentioned previously, the size factors method does not correct for gene effects. In contrast, the variance-stabilizing transformation (VST) [42] is a normalization method which uses the size factors but also gets rid of the variance's dependence on the mean. Indeed, RNA counts are over-dispersed: the count variance increases faster than the mean. It is usually useful to remove this dependence for visualization or clustering purposes. Specifically, VST takes as input the counts normalized with the median of ratios above, X_{norm}. It then fits a dispersion-mean relation, for example via a Gamma-distributed generalized linear model $f_{a,b}$ parametrized by $a, b \in \mathbb{R}$, so that the variance of $f_{a,b}(X_{norm})$ for each gene does not depend on the mean gene count. The counts normalized by VST are thus the $f_{a,b}(X_{norm})$. Note that VST returns data on the $\log_2$ scale, which is the scale used for differential gene expression analysis.

To show the impact of the preprocessing method empirically, we plot the PCA and explained variance per number of components for our cancer genome dataset, preprocessed with the median of ratios method (figure 3.1), and with VST (figure 3.2). We notably see that the dataset is captured by much less components for the former rather than the latter preprocessing method.

Concretely for our purposes, these normalization methods are available in PyDESeq2 [75], which is a Python package for DEA that implements the R package DESeq2 [74].

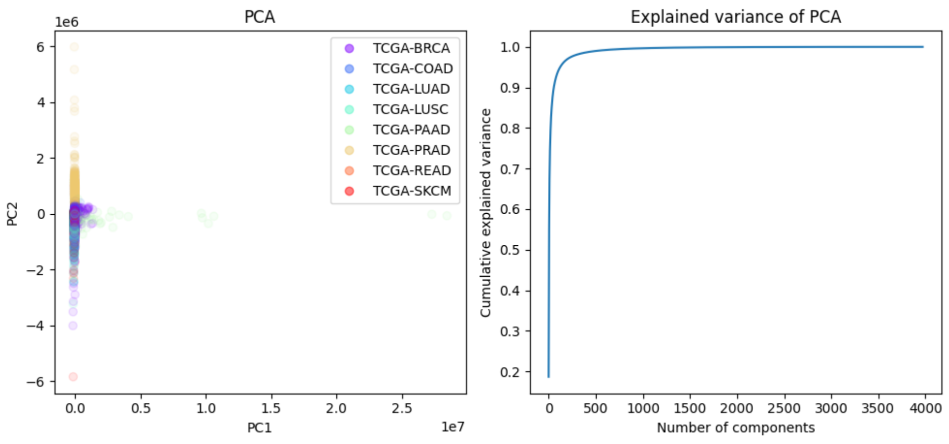

Figure 3.1: PCA of the TCGA dataset normalized with the median of ratios method.

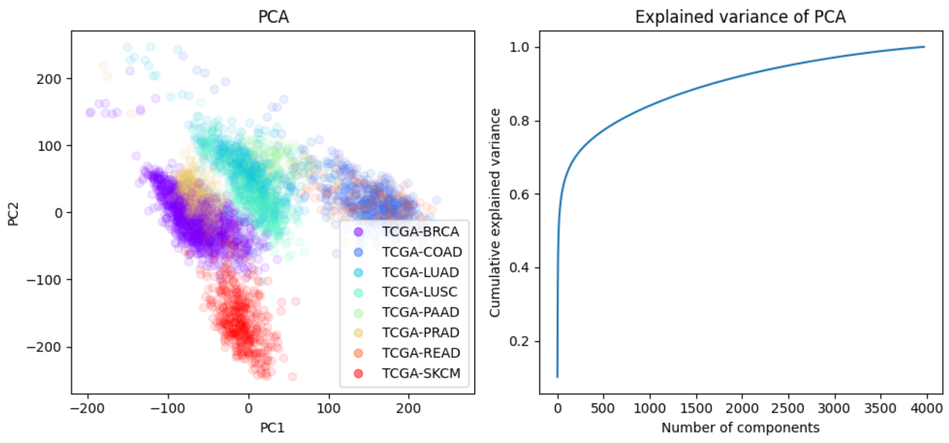

Figure 3.2: PCA of the TCGA dataset normalized with VST.

3.2 Experimental design

3.2.1 Simulation of the federated environment

In this work, all experiments are simulated and not done on a real distributed network. The advantage of this approach is that we can easily deploy our algorithmic models and assess their performance on datasets. With a federated learning software, there are more code design constraints and experiments take longer to execute.

Thus, we focus on developing a solid simulated algorithm, that can then be translated into any federated software for real-world applications. The obvious downside of doing simulations only is that we do not measure the empirical time complexity of our algorithms. This means that we should keep an eye on the theoretical communication complexity, as this would presumably be the biggest bottleneck in a real federated set-up.

We implement our simulated environment in Python. For this, we create two classes, FedCenter and FedServer. The FedCenter class describes a local center containing private data. It implements methods for data preprocessing and for local computations involving its data only. Then, the FedServer class contains a dictionary whose values are FedCenter instances and correspond to each data center.

In our code, the FedServer never accesses or modifies FedCenter attributes directly. It can only call centers' public methods, to get the result of local calculations which are designed not to share confidential information. Note that in reality, there is no notion of private attribute in Python, so the FedServer could technically access the private data from the FedCenter dictionary; this is why this is merely a simulation and not an actual federated implementation.

3.2.2 Complexity and communication cost

We determine the time complexity of each algorithm, theoretically, by using the big $\mathcal{O}$ formalism. This is useful to know whether it can be used in a reasonable length of time on a real dataset given its dimensions.

We do not compute the global space complexity on the server and data centers, assuming all of them have a large memory. We are however interested in the communication cost of our methods. Given that we simulate the federated network, we cannot empirically measure the duration of communications between the server and the centers. Instead, we compare our algorithms in terms of the total data size exchanged. More precisely, we examine the number of communication rounds involved in each method and the size of the message sent at each round.

3.2.3 Performance metrics

We use two complementary kinds of metrics to compare a federated clustering to its pooled equivalent: metrics based on dendrogram comparisons and metrics that study the cluster assignments when cutting the hierarchical tree at a certain level. The first kind takes into account the distances between clusters, while the second considers only if the groupings consist of the same data points.

3.2.3.1 Cophenetic distance measures

The first family uses the *cophenetic distance*, which is the pairwise distance induced by the dendrogram: the cophenetic distance between two points is the distance at which their respective clusters are merged during the clustering. For instance, in the figure 3.3, the cophenetic distance between point 3 and any of points 0, 4, or 5 is the same and is indicated by the dashed line.

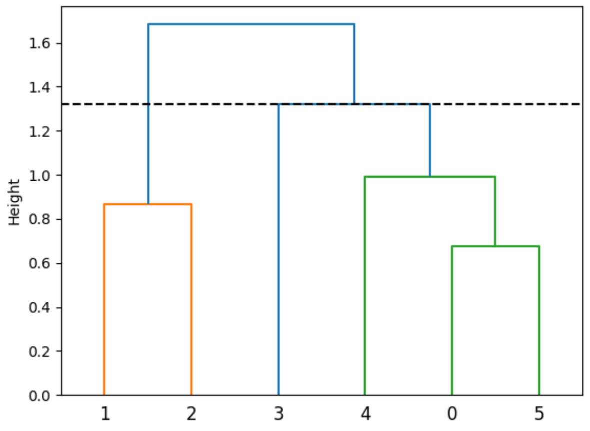

Figure 3.3: Dendrogram generated from the single linkage clustering of six points – the dashed line indicates the cophenetic distance between point 3 and points 0, 4 and 5.

The standard HC metric that uses the cophenetic distance is the cophenetic correlation coefficient (CCC). It is defined as the Pearson correlation coefficient c between the cophenetic distance d_{HC} and a reference distance d on the dataset X:

$$c = \frac{\sum_{1\leq i<j\leq n} \left(d_{HC}(X_i, X_j) - \bar{d}_{HC}\right)\left(d(X_i, X_j) - \bar{d}\right)}{\sqrt{\sum_{1\leq i<j\leq n} \left(d_{HC}(X_i, X_j) - \bar{d}_{HC}\right)^2 \sum_{1\leq i<j\leq n} \left(d(X_i, X_j) - \bar{d}\right)^2}}$$

It is typically used to compare the cophenetic distance to the actual pairwise distance with the metric of choice, but we can utilize it to compare the federated and pooled cophenetic distances. Its value is between -1 and 1; it is 1 when the distances are perfectly correlated, -1 when they are perfectly anticorrelated and 0 when there is no correlation.

As it is a correlation coefficient, all pairwise distances do not hold the same weight. Having a CCC close to 1 means that the biggest trends between pooled and federated clusterings are the same, but does not say anything about the concordance between small distances. More details about this behavior with a specific example are available in appendix A.

In our case, we are mostly interested in reproducing the bigger trends with federated clustering, especially as the last merges – which correspond to bigger distances for ultrametric linkages – are more informative than the first few. Therefore, the CCC is still relevant, more so in the case there are no strong outliers. If we want an index that assigns the same weight to all pairwise cophenetic distances, we can find a complementary measure in the mean relative error averaged over all federated and pooled cophenetic distances.

3.2.3.2 Flat clustering measures

The other kind of measure does not consider the merging distances, but only the cluster assignments at each step. Recall that the agglomerative clustering of n points has $n-1$ steps, and the k-th merge step, $1 \leq k \leq n-1$, produces $n-k$ different clusters.

Let π_1 and π_2 respectively be the flat clusterings obtained from the federated and pooled hierarchical trees cut to get k clusters. We can define the true/false positive/negative quantities as follows:

- TP the number of pairs of objects clustered together in π_1 and π_2
- FP the number of pairs of objects clustered together in π_1 but not π_2
- FN the number of pairs of objects clusterd together in π_2 but not π_1
- TN the number of pairs of objects that are in different clusters in π_1 and π_2

There are several measures we can use based on these quantities. The first one is the Fowkles-Mallows index (FMI), developed specifically for comparing two HC trees [76]. It has a value of

$$FMI = \frac{TP}{\sqrt{(TP+FP)(TP+FN)}}$$

corresponding to the geometric mean of precision and recall. It ranges from 0 to 1, and takes the value 1 when the clusterings are exactly the same.

Another possible measure is the adjusted Rand index (ARI), which is the Rand index

$$RI = \frac{TP+TN}{TP+FP+FN+TN}$$

corrected for chance. While the Rand index is between 0 and 1, this adjustment gives a value of 0 to a random assignment. It makes the index take values between -0.5 and 1, with the value 1 reached when the labelings are identical.

Note that to analyze the performance of federated clustering, we do not usually compute these indices for all possible number of clusters. Indeed, clustering mistakes during the first few merges are not as important as late mistakes, as the hierarchical tree is often cut to retrieve a small number of clusters. Therefore, we are more interested in getting indices close to 1 for the last few merges. Note that in the special case the dataset has a known number of clusters k, the index is most relevant for that given number of clusters.

3.2.4 Privacy analysis

We do not have a specific metric for privacy (as would be the case with DP, which provides a mathematical framework of privacy [77]). Instead, we perform a case-by-case analysis of whether each algorithm leaks local quantities with the server or other participants. If it does, we assess if the shared quantities reveal sensitive information about the data. Notably, we use different empirical attacks on RNA-Seq data, inspired by previously propsosed attacks on FL [78, 25] and data projections [47]. We can then see what a malicious actor could learn depending on the obtained information.

3.2.5 Hardware/Software to be used

We use a MacBook Pro with a 6-Core Intel Core i7 and 16GB memory for most purposes. For any machine learning training, we use a 8 CPU machine (n1-standard-8) with a NVIDIA Tesla T4 GPU and 25GB of RAM.

All code is written in Python[1] using VS Code. We rely on the following Python libraries:

- NumPy [79], SciPy [29] and Scikit-Learn [80] for most scientific code
- PyDESeq2 [75] for processing bulk RNA-Seq data
- PyTorch [81] for machine learning attacks
- pandas [82] for dataframes manipulations
- Matplotlib [83] and Seaborn [84] for visualization

[1]Python Software Foundation, https://www.python.org/

3.3 Federated hierarchical clustering approaches

Here we present the methods we have developed for federated HC in a cross-silo context: a central server communicates with a few data centers in order to execute some computation over the data, and send it back to each center.

As for the classic agglomerative HC procedure, our federated algorithms are parametrized by a *linkage* method (defined in section 2.2.1.1, which explicitates the most common linkages) and a *metric*. Both serve the same purpose in our federated approaches as in the usual unfederated algorithm, which we recall here. First, in terms of the linkage, single linkage identifies clusters that are spatially stretched out, for instance a set of parallel lines, each forming a cluster on which points are densely distributed. However, in case of compact clusters that might be close together – imagine for example two almost-colliding spheres with some noise at the intersection – single linkage might not recognize them as separate clusters, and chain them together, known as the chaining effect. In such a case, complete linkage is more adapted. Finally, linkages such as average, centroid and Ward have intermediary properties, providing an equilibrium between the separability of single-linkage clusters, and the compactness of complete-linkage clusters.

Second, for the metric, one can use ℓ_p distances, in which case the higher p is, the more the distance between two points will be based on the the dimension that differentiates them the most. Cosine and correlation metrics are also often used for HC; a big difference with ℓ_p metrics is that they are scale-invariant, with the correlation dissimilarity also being translation-invariant. Additionally, we note that average, Ward and centroid linkages are only defined for the ℓ_2 metric.

3.3.1 Federated genewise clustering

Genewise clustering is useful to identify correlated genes, as they may be involved in the same biological pathways and participate in the same functions in the body. Recall that federating the procedure of genewise clustering requires us to develop an algorithm in the case of vertically-partitioned data. Indeed, here the samples act as the features, distributed among the clients, while the genes are the clustered objects.

For genetic data, we assume that individual patient data is very sensitive, but that gene information is not. In particular, pairwise distances between genes with the usual metrics is not considered as confidential information.

Given that we investigate the cross-silo setting, we expect that there are several clients who each hold a decently-sized dataset – with an order of magnitude of at least 10-100. For each client, sharing the gene distances from their dataset does not break any privacy constraint.

Let us consider M centers $C^{(1)}$, ..., $C^{(M)}$, each of them containing a dataset $X^{(k)} \in \mathbb{R}^{n^{(k)} \times d}$. The equivalent pooled dataset is noted $X \in \mathbb{R}^{n \times d}$, with $n = \sum_{k=1}^{M} n^{(k)}$. Let us take a metric that is *separable* over its inputs, meaning that that the associated distance d between genes g_i and g_j is of the form

$$d(g_i, g_j) = \sum_{k=1}^{M} d^{(k)}(g_i, g_j)$$

where $d^{(k)}$ is a distance computed in center $C^{(k)}$. Per the assumption above, $d^{(k)}(g_i, g_j)$ are not sensitive and therefore we can privately compute the global distance in a federated way, by summing the contribution of each center. Let us now assess the separability of a few common metrics.

3.3.1.1 Federated gene pairwise distance

We will show that the ℓ_p, cosine and correlation distances are all separable over the data centers.

3.3.1.1.1 Minkowski distance For all genes g_i, g_j indexed by $1 \leq i, j \leq d$:

$$\begin{aligned} ||g_i - g_j||_p^p &= \sum_{l=1}^{n} |X_{l,i} - X_{l,j}|^p \\ &= \sum_{k=1}^{M} \underbrace{\sum_{l=1}^{n^{(k)}} |X_{l,i}^{(k)} - X_{l,j}^{(k)}|^p}_{\text{depends only on center } k} \end{aligned}$$

Thus, the ℓ_p distance can be computed as the sum of each center's contribution.

3.3.1.1.2 Cosine distance We can compute the cosine distance d_C using its relationship to the ℓ_2 norm. Let $x, y \in \mathbb{R}^d$ and let $\theta(\cdot, \cdot)$ indicate the angle between two points. Let us recall that:

$$\begin{aligned} d_C(x,y) &= 1 - cos(\theta(x,y)) \\ &= 1 - \frac{\sum\limits_{k=1}^{d} x_i \, y_i}{||x|| \, ||y||} \end{aligned}$$

It is easy to show that:

$$d_C(x,y) = \frac{1}{2} \left|\left| \frac{x}{||x||} - \frac{y}{||y||} \right|\right|^2$$

Therefore, the cosine distance can also be written as the sum of each center's input; specifically, for all genes g_i, g_j with $1 \leq i,j \leq d$:

$$d_C(g_i, g_j) = \frac{1}{2} \sum_{k=1}^{M} \underbrace{\sum_{l=1}^{n^{(k)}} \left(\frac{X_{l,i}^{(k)}}{||X_{l,i}^{(k)}||} - \frac{X_{l,j}^{(k)}}{||X_{l,j}^{(k)}||} \right)^2}_{\text{computed by center } k}$$

3.3.1.1.3 Correlation distance The correlation distance simply corresponds to the centered cosine distance. Once again, as gene information is not considered sensitive, we allow the server to compute each gene's sum of counts. We also assume that the server knows the total number of samples n, so that for gene g_i, it can compute $\bar{g}_i = \frac{1}{n} \sum\limits_{k=1}^{M} X_{l,i}^{(k)}$ and send it back to every center. The correlation distance for genes g_i, g_j with $1 \leq i,j \leq d$ is:

$$\begin{aligned} d_{corr}(g_i, g_j) &= d_C(g_i - \bar{g}_i, g_j - \bar{g}_j) \\ &= \frac{1}{2} \sum_{k=1}^{M} \underbrace{\sum_{l=1}^{n^{(k)}} \left(\frac{X_{l,i}^{(k)} - \bar{g}_i}{||X_{l,i}^{(k)} - \bar{g}_i||} - \frac{X_{l,j}^{(k)} - \bar{g}_j}{||X_{l,j}^{(k)} - \bar{g}_j||} \right)^2}_{\text{calculated by center } k} \end{aligned}$$

Again, the correlation distance can be expressed as the sum of centers' computations.

3.3.1.2 Algorithm description

We can now present a straightforward gene clustering algorithm valid for all linkage methods and for the metrics above (algorithm 3). Notice that the distance computation is exact, meaning that performance-wise, the algorithm is pooled-equivalent. Also, the algorithm has no more parameters than its unfederated counterpart, *i.e* a linkage method and a metric.

Algorithm 3 Federated genewise agglomerative clustering

```
Require: $M$ centers with dataset $X^{(k)} \in \mathbb{R}^{n^{(k)} \times d}$ in center $C^{(k)}$, metric $\mu$,
  linkage $L$
  $\bar{g} \leftarrow 0$                              ▷ average count per gene, $\bar{g} \in \mathbb{R}^d$
  if $\mu$ is correlation then
      $n \leftarrow 0$
      for center $k = 1, \ldots, M$ do
          On center k:
              $\bar{g}^{(k)} \leftarrow$ SumGeneCounts($C^{(k)}$)
              Send $\bar{g}^{(k)}, n^{(k)}$ to the server
          On the server:
              $\bar{g} \leftarrow \bar{g} + \bar{g}^{(k)}$
              $n \leftarrow n + n^{(k)}$
      end for
      $\bar{g} \leftarrow \bar{g}/n$
  end if
  $D \leftarrow 0$                                    ▷ global distance matrix
  for center $k = 1, \ldots, M$ do
      On center k:
          $D^{(k)} \leftarrow$ Distance($C^{(k)}, \bar{g}, \mu$)
          Send $D^{(k)}$ to the server
      On the server:
          $D \leftarrow D + D^{(k)}$
  end for
  On the server:
      AgglomerativeClustering($D, L$)
```

3.3.2 A centroid-based approach for federated samplewise clustering

Federating samplewise clustering is a much more complex issue than genewise clustering. With a horizontal partition of the data into federated centers, it is trivial to compare distances between points inside a center. However, we would also like to be able to compare samples from different centers, which is hard to achieve without leaking these samples.

We make the following assumption: sharing a single sample with the server and other centers is problematic, but communicating the coordinates of a centroid aggregated from enough samples does not leak sensitive information about those points. Given this, we can create an algorithm based around sharing the centroids of local clusters once they reach a critical size n_{min}, so that other centers can approximate points in these clusters as their average coordinates. n_{min} is the only new parameter of this federated approach compared to the classic algorithm, and lets the user select the right privacy/performance trade-off. Indeed, the smaller n_{min}, the closer the clustering is to the unfederated algorithm's output, but the more revealing the centroid potentially is, as it averages less points.

3.3.2.1 A first scheme with simultaneous sharing of local clusterings

A first simple idea is to compute the local clustering in each center, communicate the centroid of every local cluster to the server, and complete the clustering from these centroids. To be precise, we can simply build the hierarchical tree in each local center and cut it at the point for which all clusters have met the threshold size n_{min}. As is shown on figure 3.4 (right), when the cut is done at the height for which this criterion is met, some clusters can be pointlessly big. It is better to recursively look in each branch of a merge for the smallest allowed clusters (right).

Then, all centers share the centroids of their locally obtained clusters to the server. The partial dendrograms are also shared with the server, in order to reconstitute the merges in order of merge distance. Now, the agglomerative clustering can resume using the centroids, weighted by their number of points. In a way, the server sees each centroid c aggregated from n_C points as if it were n_C points with the same coordinates c.

Note that Ward and centroid linkages are based on the distances between the clusters' centroids (cf. table 2.1), so approximating clusters as their centroids gives exact cluster distance computations in the global phase.

3.3.2.2 Analysis

This simple method unfortunately has a big shortcoming. Let us consider the case where a small cluster – whether an outlier, or a number of points lesser than n_{min} – is far from other points in a center. During the local clustering phase, it will remain its own cluster until the last step, where it will be merged

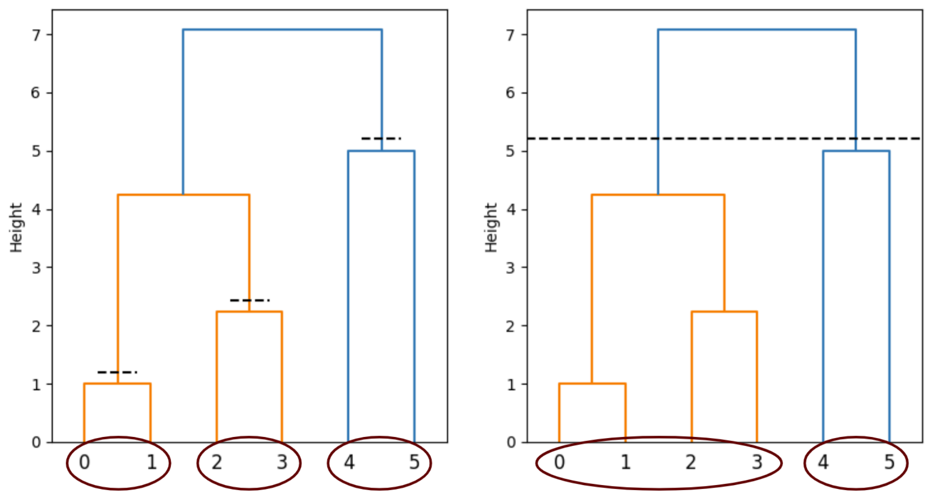

Figure 3.4: Local cut to get clusters of size at least 2, either recursive (left) or by height (right). Given the dashed cut, resulting clusters are circled in red

with other points. Therefore, the cut of the HC will return only one cluster with all points in it.

Ideally, we would like cluster centroids to be shared at the lowest feasible size, in order to maximize interaction with other centers and be as close as possible to the pooled clustering. This is what leads us to the next algorithm.

3.3.3 Gradual sharing of local clusters for the win

We thus consider a second approach that is still based on approximating clusters as their centroids, but that is closer to the original standard agglomerative clustering algorithm. Here the idea is to merge the two closest communicating clusters at each step. At first, clusters are only local and each center shares the distance between its two closest clusters in order to find the closest pair overall.

Then, once a cluster's size n_C surpasses the threshold size n_{min}, its centroid c is shared with other centers, who now see a cluster with n_C points with the coordinates of c (see figure 3.5). In the center of origin of the cluster, distances remain unchanged, but in other centers, the distance to the new cluster is computed according the linkage.

This cluster is now considered global, and can be fused with either other global clusters or local clusters. When a local cluster is integrated into a global cluster, its members join a waitlist of unshared samples attached to the center. That way, when the waitlist reaches the minimum size n_{min} in a center, its

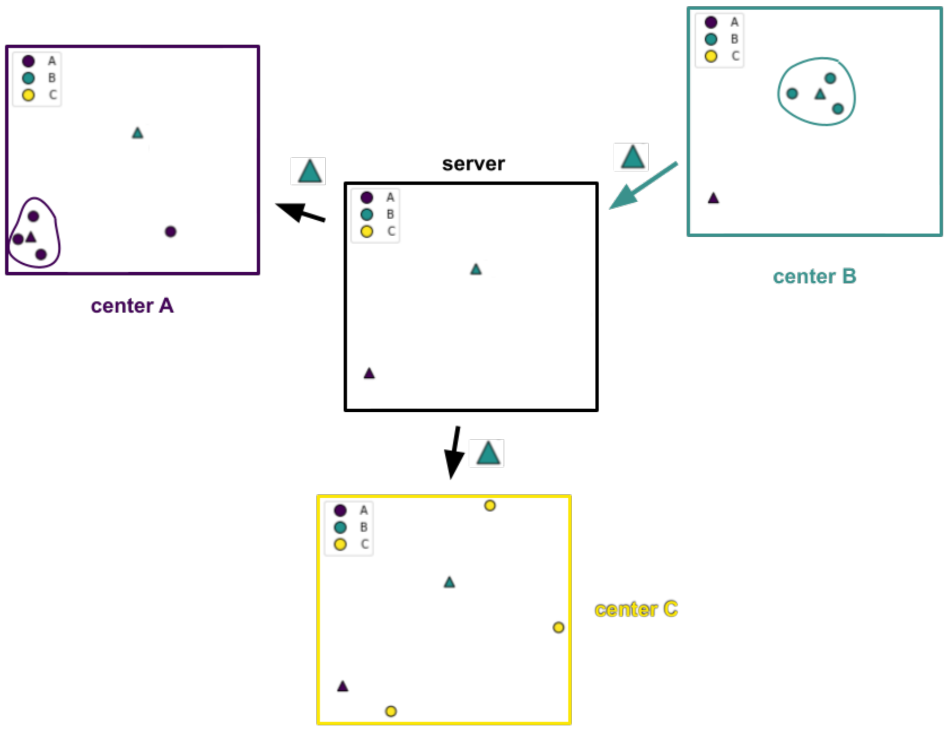

Figure 3.5: Scheme of the gradual sharing of centroids. Centroids are represented with a triangle. Here a centroid from center A has already been publicized to act as a cluster proxy, and a centroid from center B is currently being communicated to all participants.

objects are averaged and the centroid is published. Distances with clusters in other centers are then updated as if the samples had joined the cluster but with their centroid's coordinates.

3.3.3.1 Federated distance update

We now give more details about the distance update schemes. Recall that when two clusters A and B are combined, we usually have to compute the distance $d(A \cup B, C)$ to all other clusters C, using $d(A, C)$ and $d(B, C)$. This is straightforward for local/local or global/global merges. Indeed, when two local clusters from the same center are merged, only distances in that center and to global clusters are updated. The linkage rule is used for the update, using the already known distances. Similarly, when two global clusters are fused, the distance update is necessary for other global clusters, as well as distances to local clusters in every center. Again, this is done according to the linkage and with existing cluster distances.

However, let us examine a global/local merge between a global cluster C_G and a local cluster C_L. For clusters in the center of origin of C_L and for all other global clusters than C_G, there are no issues as all distances to C_L and C_G already exist. However, let us consider a cluster C from another center. We would also like to compute the distance from $C_G \cup C_L$ to local clusters in other centers, but only $d(C_G, C)$ is defined. We decide to perform the update as if $d(C_L, C) = d(C_G, C)$. The intuition is that this should be a reasonable approximation as C_L and C_G are the closest clusters at that merging step, so they should be similar enough compared to other clusters. With this approximation, there is no need to update the distance for single, complete and centroid linkages. An update is nevertheless required for average and Ward linkages as their formula is based on cluster sizes.

3.3.3.2 Validity for approximate centroid and Ward linkages

We note that with our algorithm, the distance update formula is always well-defined even with approximate distances. This is not trivial at first glance for centroid and Ward linkage, whose update formula involves the square root of a value created from the distance (see table 2.1). We show that the value is always positive and that this is due to the fact that we always fuse the closest cluster pair at each step. Let A and B be two clusters that are merged at the current step and C be another cluster. A and B are merged which means they are the two closest clusters: $(A, B) = \underset{A', B'}{argmin}\ d(A', B')$. In particular, $d(A, B) \leq d(A, C)$ and $d(A, B) \leq d(B, C)$ when $d(A, C), d(B, C)$ are defined. The only case when one is not defined – say $d(A, C)$ without loss of generality – is if A is local and B is global, in which case we previously defined $d(A, C) := d(B, C)$, which means the assumption is still true.

We now prove that this is sufficient for the square root to be defined. For centroid linkage:

$$\begin{aligned}\frac{|A|d(A,C)^2 + |B|d(B,C)^2}{|A| + |B|} &\geq \frac{|A|d(A,B)^2 + |B|d(A,B)^2}{|A| + |B|} = d(A,B)^2 \\ &\geq \frac{|A||B|d(A,B)^2}{(|A| + |B|)^2}\end{aligned}$$

as $|A|, |B| \geq 1$.

For Ward linkage:

$$(|A| + |C|)d(A,C)^2 + (|B| + |C|)d(B,C)^2 \geq |C|d(A,B)^2$$

seeing as $|A|, |B| \geq 1$.

Therefore, the quantities are always positive and the square root is defined.

3.3.3.3 Centroid sharing and distance correction

We now describe the process of sharing objects' centroids as a way to approximate clusters. When a new global cluster emerges C_G, we share its centroid c_G with every client. The centroid is notably used for initializing the distance to local clusters in other centers. Let us take another global cluster C'_G and let $l^{(k)}$, $1 \leq k \leq M$ denote its (potentially empty) local waitlists in each center. Before C_G became global, its distance to C'_G may have contained approximations corresponding to the local waitlists. Specifically, whenever a local cluster was merged into C'_G, it was approximated as the public centroid c'_G of C'_G at that time, as we had no better available information. Now that the centroid of C_G is shared, the distance from c_G to a waitlist $l^{(k)}$ in center (k) does not have to rely on centroid c'_G anymore, and we should get a better approximation of $d(C_G, l^{(k)})$ by computing it in center (k) as the distance from c_G to the cluster made of all points in $l^{(k)}$. Thus, we correct the distance estimation as follows.

Let us note $C'_G = G' \cup \bigcup_{1 \leq k \leq M} L^{(k)}$, where G' contains all the points published as part of a centroid, and $L^{(k)}, 1 \leq k \leq M$ contains the unshared points in center (k). The distance between C_G and C'_G can be updated in the following way. For single linkage,

$$d(C_G, C'_G) \leftarrow \min\left(d(C_G, C'_G), \min_{1 \leq k \leq M} d(C_G, L^{(k)})\right)$$

Similarly for complete linkage,

$$d(C_G, C'_G) \leftarrow \max\left(d(C_G, C'_G), \max_{1 \leq k \leq M} d(C_G, L^{(k)})\right)$$

For average linkage, before correction,

$$d(C_G, C'_G) = \frac{|G'|}{|C'_G|} d(C_G, G') + \frac{|C'_G| - |G'|}{|C'_G|} \delta$$

where δ is the term we would like to replace. δ is of the form $\underset{1\leq k\leq M}{\delta_k}$ where δ_k can be stored in center (k) and appropriately modified every time a local/local approximation is made. Therefore we do:

$$d(C_G, C'_G) \leftarrow d(C_G, C'_G) + \frac{|C'_G| - |G'|}{|C'_G|}\left(\sum_{1\leq k\leq M} \frac{|L^{(k)}|}{|C'_G| - |G'|} - \delta\right)$$

For centroid and Ward linkages, it is not possible to keep a separate track of the accumulated error given the distance formula. Therefore, we recompute $d(C_G, C'_G)$ from scratch.

A centroid can also become public as part of an already existing global cluster, and the distance correction works similarly as above. The main difference is with average linkage, for which the weighted average also contains a fixed element corresponding to the distance to the global cluster before integration of the centroid. The public centroid of the merged cluster is also updated as the weighted average of all shared centroids of the cluster.

3.3.3.4 Algorithm summary

We summarize our samplewise clustering algorithm in 4. In a nutshell, at each step, the closest communicating clusters, are merged: either two global clusters, one global and one local cluster, or two local clusters in the same center. Distances are updated only for clusters that are allowed to communicate. When a cluster has a local part bigger than the threshold size n_{min}, we share its centroid with everyone so as to initialize or improve cluster distance approximations.

3.3.4 Random projections for federated samplewise clustering

The previous approach is very much shaped for agglomerative clustering and hard to generalize to other federated analytics pipelines. We also investigate a different technique, based on computing an approximation of the pooled distance matrix through random projections. The random projection scheme for private computations has been extensively studied in the literature and described in section 2.4. It has been proposed for single-linkage clustering [49], but the experiments consider the cost of the associated MST, and are only valid for the Euclidean metric. Here, we propose a more thorough

Algorithm 4 Centroid-based federated clustering

```
Require: $M$ centers with $X_{(k)} \in \mathbb{R}^{n_{(k)} \times d}$ in center $(k)$, minimum cluster size $n_{min}$
Ensure: $\exists 1 \leq k \leq M$, such that $n_{min} \leq n_{(k)}$
  for center $k = 1, \ldots, M$ do
      Initialize local clusters and distances
  end for
  for step $s = 1, \ldots, n-1$ do
      $i, j \leftarrow \arg\min_{i,j} d(C_i, C_j)$          ▷ closest communicating active clusters
      $C_i \leftarrow C_i \cup C_j$
      for cluster $l$ do
          if $C_i$ and $C_l$ communicate then
              $d(C_i, C_l) = \textsc{UpdateDistance}(C_i, C_j, C_l)$
          end if
      end for
      for center $k = 1, \ldots, M$ do
          $C_i[k] \leftarrow C_i[k] \cup C_j[k]$          ▷ merge private sublists in the center
          if $|C_i[k]| \geq n_{min}$ then
              $\textsc{ShareCentroid}(C_i, C_i[k])$
              $C_i[k] \leftarrow \varnothing$
              for cluster $l$ not in center $k$ do
                  $d(C_i, C_l) = \textsc{CorrectDistance}(C_i, C_l)$
              end for
          end if
      end for
  end for
```

benchmark, analyzing more linkages and metrics, and centered specifically on HC measures (presented in 3.2.3).

The random projection method for the Euclidean distance is quite general, as many statistical pipelines can be computed using only the Euclidean distance matrix – or equivalently with MDS, the dataset up to orthogonal transformation. Some notable applications include kernel methods, visualization methods (t-SNE, UMAP), clustering (k-means, DBSCAN) and nearest neighbours. Using the random projection principle, we only add the projection dimension d' as additional parameter to federate the HC method. We see intuitively that a lower value of d' yields more noisy distances, meaning a higher degree privacy but a lower resemblance to the unfederated clustering result.

3.3.4.1 Algorithm description

We detail the algorithm for clustering with any linkage L and with the ℓ_p norm, $0 < p \leq 2$, based on sampling from the p-stable distribution with scale 1, noted $S(p, 1)$. Let us consider a dataset $X \in \mathbb{R}^{n \times d}$ split into M centers, with each center $C_{(k)}$ containing a dataset $X_{(k)} \in \mathbb{R}^{n_{(k)} \times d}$. Let $n = \sum_{i=1}^{M} n_{(k)}$. Let d' be the final dimension of the dataset.

First, all centers settle on a random seed s using SMPC. Though this requires the use of cryptography, we consider it to be reasonable for practical purposes. Now, all centers can sample dd' i.i.d variables from the stable law $S(p, 1)$ with the shared seed s, so as to form the same matrix $Q \in \mathbb{R}^{d \times d'}$. Each center sends its encrypted dataset $Y_{(k)} = X_{(k)}Q$ to the server.

The server concatenates all received sets to get $Y \in \mathbb{R}^{n \times d'}$. It then computes the pairwise differences between all objects in Y as a $n \times n \times d'$ matrix Δ. Now, let $1 \leq i, j \leq n$. $\Delta_{i,j}$ is a random vector of d' variables following the stable law $S(p, ||X_i - X_j||_p^p)$.

Therefore, we get an approximation of $||X_i - X_j||_p$ by estimating the scale of this stable distribution using $\Delta_{i,j}$. For the ℓ_2 norm, this simply corresponds to the standard deviation of the distribution, which can be estimated as

$$||Y_i - Y_j||_2 = \sqrt{\frac{1}{n}\sum_{k=1}^{d'}(Y_{i,k} - Y_{j,k})^2}$$

For the ℓ_1 norm, several estimators exist and are analyzed in [57]: the median, geometric mean and maximum likelihood estimators. The first two are asymptotically equivalent, with the second converging faster. The last estimator is more efficient but requires to solve a maximum likelihood equation. We opt for the geometric mean estimator (equation 2.2) for its rather low computational cost and adequate performance (80% of the maximum likelihood estimator according to the authors). While we only benchmark it for the ℓ_1 norm, it can be extended to the ℓ_p norm , $0 < p \leq 2$ (a detailed account of estimators can be found in [56]).

Finally, once the total distance matrix is estimated, it can be fed to a standard agglomerative clustering algorithm. The summary as pseudocode of the complete algorithm can be found in 5.

The algorithm is written for the ℓ_p norm, but we can also extend it to the cosine and correlation distances. This is done by reducing them to the Euclidean distance with the tricks explicited in sections 3.3.1.1.2 and 3.3.1.1.3.

Algorithm 5 Sampling stable distributions for HC

```
Require: $M$ centers, with dataset $X_{(k)} \in \mathbb{R}^{n_{(k)} \times d}$ in center $C_{(k)}$, final
    dimension $d'$, exponent $p$ for the $\ell_p$ norm, linkage $L$
    Centers $C_{(1)}, \dots, C_{(M)}$ agree on a shared seed $s$
    for center $k = 1, \dots, M$ do
        Sample $Q \in \mathbb{R}^{d \times d'}$ with i.i.d entries from $S(p, 1)$ with seed $s$
        $Y_{(k)} \leftarrow X_{(k)} Q$
        Send $Y_{(k)}$ to the server
    end for
    On the server:
        $\Delta \leftarrow$ PairwiseDifference$(Y)$      ▷ $\Delta \in \mathbb{R}^{n \times n \times d'}$ where $n = \sum_{i=1}^{M} n_{(k)}$
        $D \leftarrow 0$                                 ▷ distance matrix, $D \in \mathbb{R}^{n \times n}$
        for $i = 1, \dots, n-1$ do
            for $j = i+1, \dots, n$ do
                $D(i,j) \leftarrow$ EstimateStableScale$(\Delta_{i,j}, p)$
                $D(j,i) \leftarrow D(i,j)$
            end for
        end for
        AgglomerativeClustering$(D, L)$
```

3.3.4.2 Privacy analysis through attacks

While it is easy to understand what is shared with the server for the centroid-based approach, it is not clear what the server can recover from the random projection or even the orthogonal projection method, especially given prior knowledge on the data. This is why we attempt several attacks to see if the server or a malicious entity would be able to retrieve more information than expected from the projected data.

3.3.4.2.1 Known input-output attack on orthogonal projections We first test the known input-output attack from [47], which we will now describe. It requires no prior knowledge, and assumes that the attacker holds two pieces of information: the projected dataset on the server, as well as some of the original inputs.

More precisely, let $X \in \mathbb{R}^{n \times d}$ be the original pooled dataset, Y its orthogonal projection and k the number of known inputs. Let $X_k \in \mathbb{R}^{k \times d}$ be the objects known by the attacker and Y_k the associated output. For all $x \in \mathbb{R}^d$, let $d(x, X_k)$ refer to the distance from x to the vector space generated by the rows of X_k.

We also assume the attacker can identify Y_k, *i.e* they know the index matches between the partial inputs and the total output. If they do not, they can usually easily find it by matching the vectors norms and pairwise distances: they are the same between the inputs and the outputs as the projection is orthogonal. In most datasets, even with vectors normalized to 1, enough pairwise distances are different for there to be no ambiguity in the matching.

We now describe the attack: given all orthogonal matrices $Q \in \mathbb{R}^{d\times d}$ such that $X_kQ = Y_k$, the attacker can sample one uniformly one at random, denoted $\hat{Q}$. The authors of [47] then infer the probability of the reconstruction $\hat{x} = y\hat{Q}^{-1}$ being a good reconstruction of x when $y = xQ$. Specifically, the probability that $||\hat{x} - x|| \leq \epsilon||x||$ for an input $x \in X$ is:

$$\mathbb{P}(x, \epsilon) = \frac{2}{\pi} \arcsin\left(\frac{||x||\epsilon}{2d(x, X_k)}\right) \text{ if } ||x||\epsilon < 2d(x, X_k) \text{ else } 1 \tag{3.1}$$

In particular, for all $x \in X$, we can now compute the value of ϵ for which the probability of a breach is 1, which corresponds to the maximum relative error of the reconstruction when sampling a random solution valid on X_k:

$$\epsilon_x = \frac{2d(x, X_k)}{||x||} \tag{3.2}$$

Notice that when x belongs to the subspace generated by X_k, then $d(x, X_k) = 0$ and the reconstruction is exact.

3.3.4.2.2 Property inference attack with a classifier Gene expression data is obviously very sensitive, and encodes information such as sex or genetic diseases. The authors of [85] benchmarked different machine learning algorithms for the task of recovering patient information from their RNA-Seq data. They notably obtain good results when training a neural network for gender and cancer type, but not with age, race[1] and cancer stage.

Even if their RNA-Seq preprocessing setup is different than ours, this suggests that we could potentially learn sensitive information from gene expression. In our case, we have an additional difficulty in the fact that the gene expression is projected with a random matrix, whether orthogonal or following a stable distribution.

We consider a setting where an attacker has access to the projection $Y \in$

[1]Prediction of race also had a good accuracy, but the dataset was very imbalanced (80% Caucasian), making it somewhat inconclusive

$\mathbb{R}^{n \times k}$ of a dataset $X \in \mathbb{R}^{n \times d}$ ($k \leq d$), the preprocessing method, the selected genes as well as the projection method (here we only focus on Gaussian and orthogonal matrices). We assume the attacker also has access to a dataset $X' \in \mathbb{R}^{n' \times d}$ with similar distribution as X, and their labels $z \in A^{n'}$ where A is the label space. The labels can represent gender or cancer type for instance.

The attacker can now use (X', z) to train a classifier f that predicts the label from the genetic data. To account for the unknown projection, the classifier receives as input a batch from X' projected with a random matrix from the relevant matrix group. In other words, we try to see if it is possible to train a projection-invariant classifier to retrieve patient data, so that we can feed it Y to retrieve its labels.

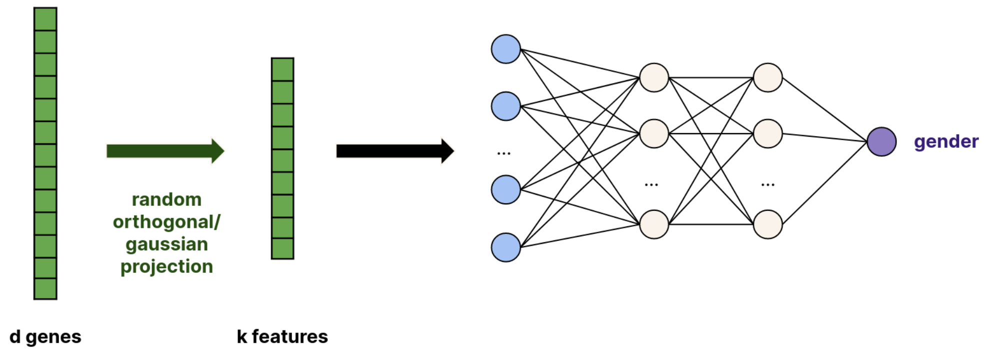

Figure 3.6: Outline of the projection-invariant classifier for predicting gender from RNA-Seq data.

In a second part, we also try a simpler attack that supposes stronger knowledge from the attacker, which is that they know the labels of objects transformed with the right projection. This corresponds for instance to the case some labels (and the indices of their matching samples) are leaked, or a center colludes with the server and shares the projection matrix. This is less realistic than the previous attack, but it can still be insightful to check whether it is effective. In that case, there is no need to sample a new projection at each training step, as labeled projected data is available.

3.3.4.2.3 Reconstruction attack RNA-Seq data follows a certain distribution, usually modeled as a negative binomial law. Even after preprocessing with the median of ratios or VST method, we can assume that gene expression data has a given distribution. We can thus try to use that prior information in order to inverse the projection of the data and reconstruct it in the original space. To achieve this, we should find the inverse transformation that best matches the distribution of the reconstruction to the prior distribution.

We express this as an optimization problem. We are looking for a distance between probability distributions, and it should be expressed as a differentiable objective so as to find the best inverse transformation through gradient descent. We choose the maximum mean discrepancy (MMD) for this purpose.

The MMD between distributions p and q is defined as the distance between their representations μ_p and μ_q in a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$:

$$\text{MMD}(p, q) = ||\mu_p - \mu_q||_{\mathcal{H}}$$

Let $X \in \mathbb{R}^{m\times d}$ be n vectors sampled from the distribution p and $Y \in \mathbb{R}^{n\times d}$ be m vectors with distribution q. Let $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be the kernel associated to $\mathcal{H}$. An unbiased estimator of the squared MMD is:

$$\widehat{\text{MMD}}^2(X, Y) = \frac{1}{m(m-1)}\sum_{i=1}^{m}\sum_{j\neq i}^{m} k(x_i, x_j) + \frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j\neq i}^{n} k(y_i, y_j) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(x_i, y_j)$$

We use the Gaussian kernel, defined as follows: for all $x, y \in \mathbb{R}^d$,

$$k(x, y) = \text{exp}\left(-\frac{1}{2\sigma^2}||x - y||^2\right)$$

where $\sigma \in \mathbb{R}^+$ is a parameter to tune.

We now go back to our attack. We assume that the attacker has access to the projected dataset $Y \in \mathbb{R}^{n\times k}$, originating from the dataset $X \in \mathbb{R}^{n\times d}$: $Y = XQ$. Now, the attacker also possesses some data $X' \in \mathbb{R}^{m\times d}$ with same distribution as X. They want to solve the following minimization problem:

$$\hat{f} = \text{arg}\min_{f\in\mathcal{F}} \text{MMD}\left(X', f(Y)\right)$$

where $\mathcal{F}$ is a certain parametric function space. We experiment with two function spaces:

- the deprojection space, in which any function is of the form $f : Y \in \mathbb{R}^k \to YR \in \mathbb{R}^d$ where $R \in \mathbb{R}^{k\times d}$. In the case of orthogonal projection, we want $R = Q^{-1}$

- a more complex decoder space, in which functions follow a typical decoder architecture, alternating between linear and activation layers

3.3.4.2.4 Attack models training and parameters We now detail the neural network architectures and training parameters for the classifier and reconstruction attacks.

First, for the classifier, we use four blocks made of one linear layer, one ReLU layer and one batch normalization [86] layer. We double the number of features at each linear step, starting with an output dimension of d_{NN} in the first block. We choose $d_{NN} = 2048$ except for the Gaussian projection in dimension 100, where we use $d_{NN} = 1024$. The last block is a linear layer followed by an activation layer – sigmoid for binary classification, else softmax. We use the Adam optimizer [87]. We tune the learning rate with a grid search.

For the reconstruction attack, the architecture of the complex decoder is composed of three blocks of a linear layer and a ReLU activation layer, and one final linear layer. Again, we double the hidden dimension at each step, starting with an output dimension of $d_{NN} = 4096$ in the very first linear layer. We use Adam for this advanced decoder, while we try both Adam and stochastic gradient descent (SGD) for the simple linear decoder. We tune the learning rate and the standard deviation σ of the Gaussian kernel with a grid search. A good value for σ is usually the median of pairwise distances in the dataset, so we try for values of σ a few orders of magnitude below and above that constant.

Note that all presented experiments for these attacks are run with 5 different seeds, and performance metrics are averaged across runs unless specified otherwise.

Chapter 4

Results and Analysis

In this chapter, we present the theoretical analysis of our algorithms as well as the results to our experiments and discuss them. Note that all plots showing the evolution of a measure with quantiles exhibit the mean value as well as the 10-90% and 25-75% quantiles.

We recall that our clustering performance metrics (CCC and FMI) compare all federated clusterings with their classic unfederated equivalents, and not to any clustering based on data labels.

4.1 Federated genewise clustering

Our proposed approach has pooled-equivalent clustering results, so we only analyze its complexity and privacy.

4.1.1 Complexity analysis

We now analyze the complexity of the algorithm. In each center k:

- computing the sum of counts per gene is done in $\mathcal{O}(n^{(k)}d)$
- computing the gene pairwise distance takes $\mathcal{O}(n^{(k)}d^2)$

Note that even if the centers work in parallel, we are in the cross-silo setup, so we assume that at least several centers have datasets of length in the same order of magnitude as the total size. More precisely, let $n_{max} = \max\limits_{1\leq k\leq M} n^{(k)}$. Then we assume $n_{max} = \Theta(n)$. Thus, all centers' computations are done in $\mathcal{O}(nd^2)$.

Now, we suppose that we have a star network: data centers communicate only with the server and not with each other. With this structure, every center communicates with the server in a sequence – whether to send the distance matrix or gene count, or to receive the average gene count. In total, the communication cost is therefore $\mathcal{O}(Md^2)$.

Finally, on the server, the time complexity of the agglomerative clustering is that of the classic algorithm: $\mathcal{O}(n^2)$ for optimized procedures (single, complete, average and Ward linkages) and $\mathcal{O}(n^2 log\, n)$ in the general case.

4.1.2 Privacy analysis

To summarize, the centers share the following information with the server:

- the average counts per gene for the correlation metric
- the pairwise gene distance per center for all metrics

This information is considered non-sensitive in our case, hence the privacy of the algorithm.

Note that we can extend this scheme to a more general case where the different summands are considered sensitive, but the global aggregation is not. For genetic data, this could be the case in a cross-device setting with the correlation metric: indeed, computing the average gene count when every client holds one data point means sharing said data point with the server.

In that case, secure aggregation [61] should be used in order to obtain the global summation (either for the gene average or the distance matrix) without accessing any of the individual summands.

4.2 Centroid-based clustering approach

4.2.1 Complexity and privacy analysis

4.2.1.1 Complexity of the algorithm with simultaneous sharing of local centroids

This first centroid-based method allows clusters from different centers to interact, and has a low communication cost. Let us consider a dataset $X \in \mathbb{R}^{n \times d}$ partitioned into M centers, such that center (k) has $X^{(k)} \in \mathbb{R}^{n_{(k)} \times d}$. Given the threshold size n_{min}, center (k) communicates:

- its centroids, which represents at most a $\lfloor \frac{n^{(k)}}{n_{min}} \rfloor \times d$ matrix

- its merge information, which is at most a $(n^{(k)} - 1) \times 4$ matrix (storing the cluster indices, merge distance and new number of points for each merge)

Therefore, the communication cost is in $\mathcal{O}(nd/n_{min} + n)$.

Complexity-wise, the clustering in the centers and the server can be done with any optimized procedure. Cutting the tree, computing the centroids and building the final dendrogram amounts to $\mathcal{O}(nd)$, which is smaller than the cost of the clustering. Therefore, the total complexity is that of the pooled clustering.

4.2.1.2 Complexity of the algorithm with gradual sharing of local centroids

We now compute the complexity and communication costs of our second centroid-based approach. Let us consider M centers, each having a dataset $X_{(k)} \in \mathbb{R}^{n_{(k)} \times d}$. Given the threshold size n_{min}, we define $n_G = \lfloor \frac{n}{n_{min}} \rfloor$, which corresponds to the maximum number of active global clusters at any step of the algorithm.

Local cluster distances are first initialized in $\mathcal{O}(n_{(k)}^2 d)$ in every center k. We detail below the complexity of a single merge operation. Keep in mind that there are $n - 1$ merges in total and that at most n_G centroids are shared.

Each center k finds the min distance between a local and a local or global cluster in $\mathcal{O}(n_{(k)}(n_{(k)} + n_G))$. The center communicates this distance with the server. With a star network, the communication cost is $\mathcal{O}(M)$. The server also computes the min distance between global clusters in $\mathcal{O}(n_G^2)$.

The server now communicates to centers whether they are the one which contains the overall min distance, and thus have to perform the merge. If the merge is between global clusters, the server can send the indices of the two clusters. In any case, the cluster distance update is performed in $\mathcal{O}(n)$.

We now describe the cost of sharing a centroid globally. When a subcluster reaches the critical size n_{min} in center k, its centroid is computed in $\mathcal{O}(n_{(k)}d)$ and sent in $\mathcal{O}(d)$ to the server. The center must also send the distances between this local group and the global clusters in $\mathcal{O}(n_G)$. For centroid and Ward linkages, these distances can be recomputed on the server in $\mathcal{O}(n_G d)$ by using the received local centroid.

Then, the server distributes the centroid coordinates with a communication cost of $\mathcal{O}(Md)$, and each center k computes or makes a correction on distances in $\mathcal{O}(n_{(k)}d)$.

In total, the algorithm thus has a $\mathcal{O}(n^2(d+n))$ complexity cost, as does the naive pooled version when including its distance computation cost. The total communication cost is $\mathcal{O}(nM + \frac{n}{n_{min}}(Md + \frac{n}{n_{min}}))$.

4.2.1.3 Privacy analysis

Both algorithms share the following significant information with the server and the centers:

1. merge information: which clusters are merged and at which distance
2. centroids of groups of at least n_{min} points (divulged only to the server in the first centroid-based method)

Centers also disclose the smallest distance involving one of their clusters, but it is hardly useful, especially if the cluster index is not shared.

We analyze the sharing of merge information first. We assume that agglomerative clustering is performed in a setting where having the pooled dendrogram is not sensitive. Otherwise, there is no point in clustering private data. This is a fair assumption given that at worse, the dendrogram reveals a few pairwise distance between points – note that the distance between two clusters with one point only is exactly the distance between the two points for all common linkages.

Thus, in our federated setting, having access to pairwise distances between clusters does not reveal anything when the position of the clusters is completely unknown. Now, recall that with our federated approach, we share centroids coordinates. By extension, in the gradual-centroid-sharing method, we sometimes share the distance from a centroid to a local cluster: this happens when a cluster C_G that has just become global is merged with a local cluster C_L. In the extreme case where C_L contains only one point, we can now place said point in a ball of radius $d(C_G, C_L)$.

One way to alleviate this issue is to impose a threshold d_{min} on the dendrogram distances, such that points can only be placed in a ball of radius d_{min}. Let (k) be a center with minimum cluster distance $d_{(k)}$ at a given step; then (k) sends max$(d_{min}, d_{(k)})$ instead of $d_{(k)}$ to the server.

We now discuss our second privacy point: the shared centroids. Intuitively, the bigger n_{min} is, the bigger the privacy level. For $n_{min} = 1$, the algorithm even corresponds to the classic pooled version. While n_{min} is not a standard privacy parameter – typically, it is not part of a privacy framework such as DP – it is still a flexible parameter that can be chosen according to the privacy needs and the data distribution. Genetic data is in dimension close to 50 000 and with a distribution that is not excessively concentrated. Therefore, it seems difficult to infer the coordinates of RNA-Seq data points from their average, even with a low number of points.

4.2.2 Performance on a toy example: the bimodal dataset

We first use a very simple toy dataset with two well-seperated clusters in order to outline the behavior of our centroid-based algorithm. This lets us illustrate the relationship between performance (*i.e* similarity to the pooled classic clustering, which very easily identifies the two clusters) and the minimum cluster size n_{min}. In particular, it gives us an idea of the maximum value n_{min} can take in order to still reproduce the important clustering trends in the data.

We use a 2D bimodal dataset of 100 points where the two cluster centers have a distance of 1 and points follow a Gaussian distribution of deviation 0.1 around the centers (see figure 4.1). Each cluster has 50 points. Points are randomly assigned to 3 federated centers so that the centers have around the same number of samples (33/33/34 split), equitably distributed between the two modes. We compute the CCC for different values of n_{min} for the Euclidean metric and the single, complete and average linkages (cf. figure 4.2).

We observe a step-like evolution of the metric: while the CCC is very close to 1 at first, it suddenly drops at $n_{min} = 18$ to stagnate around 0.4-0.5. Given the strong seperation between the two clusters, the CCC is almost 1 when the algorithm effectively forms the two clusters at the penultimate merge. Federated centers have a size of 33-34, hence 16-17 points in each mode. When $n_{min} \geq 18$, clusters are forced to include points from the two modes long before the second to last merge, leading to a poor HC.

While this is a simple example, it illustrates an upper bound on n_{min} to obtain a reasonable performance in the case of clearly separated clusters. Ideally, for each important cluster distributed among at least two centers, n_{min} should be smaller than the maximum number of points in a center that belong to said cluster.

4.2.3 Performance on TCGA

We now study the performance of our centroid-based approach on real RNA-Seq datasets, as the goal of our work is to contribute to federated HC methods specifically in the context of genetic data analysis. We use datasets from

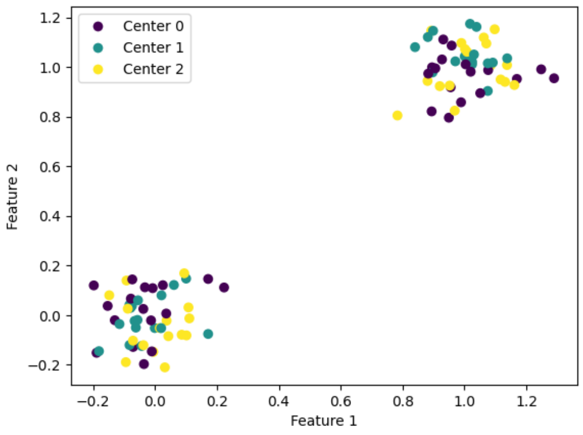

Figure 4.1: 2D bimodal dataset divided in 3 federated centers.

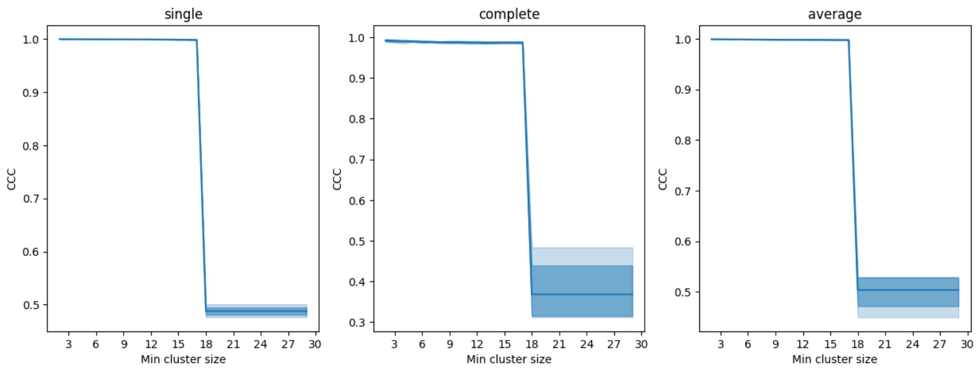

Figure 4.2: CCC evolution with the threshold size for the bimodal dataset for the single (left), complete (middle) and average (right) linkages with the gradual sharing algorithm.

TCGA (presented in 3.1). Each dataset is divided artificially between different federated data centers by using an intuitive geographic split, explained in section 3.1.1.

4.2.3.1 Performance of the algorithm with simultaneous sharing of local centroids

We first examine the performance of the naive centroid-based approach, when the local clustering is executed in each center until all clusters pass the critical size, and then their centroids are sent to the server to continue the clustering. For the FMI, we take the average over the last 10 significant merges (*i.e* we omit the last one, which is 1 as there is one cluster containing all points for both pooled and federated algorithms). We compute the CCC and this average FMI for the LUSC dataset preprocessed with the median of ratios method (figure 4.3), for $n_{min} \geq 2$. We use the Euclidean metric.

We observe that the CCC and FMI range from mediocre to decent depending on the linkage when $n_{min} = 2$. They then quickly drop to low values before stagnating for complete, average and Ward linkages as n_{min} increases. In the case of single linkage, the metrics are constant from $n_{min} = 2$, meaning that the local clustering already has to merge all points together in order to naturally force all objects into a cluster of size more than one. This is the problem explained in section 3.3.2.2: in practice, the rule rarely allows clusters from different centers to fuse unless they contain all the points in their center. Here, even for a dataset of 517 points, the clustering quality stagnates at very poor values for $n_{min} = 15$ at most for all linkages.

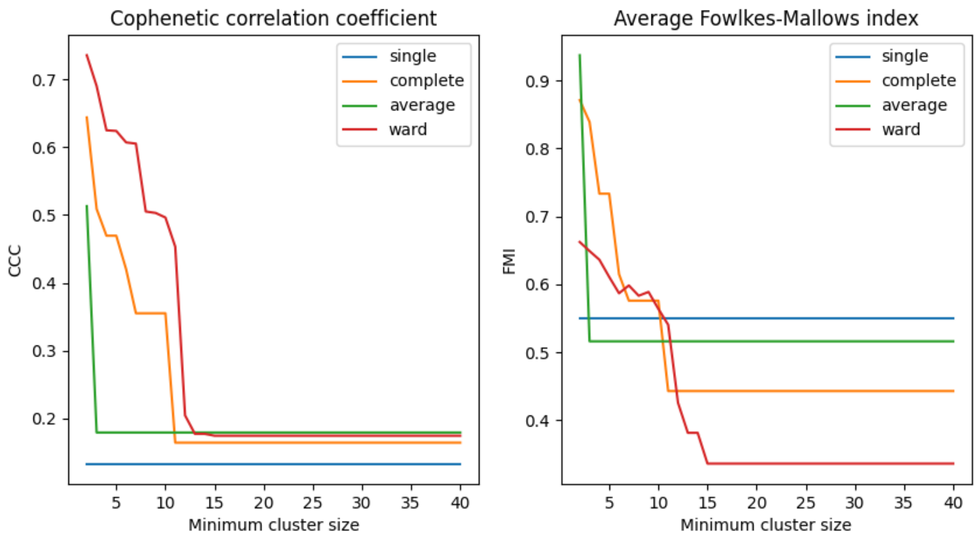

Figure 4.3: Evolution of two clustering metrics (CCC and FMI) with the minimum cluster size for the simultaneous sharing of local clusterings scheme. Here we use the LUSC dataset preprocessed with size factors. Results are shown for the single, complete, average and Ward linkages.

4.2.3.2 Performance of the algorithm with gradual sharing of local centroids

We now check the performance of the improved centroid-based method on the different TCGA datasets. Again, we compute the CCC as well as the FMI averaged over the last 10 relevant merging steps. For each dataset, we consider values from $n_{min} = 2$ to $n_{min} = n_f$ where n_f is 10% of the size of the equivalent pooled dataset. Here we show the performance quantiles for the TCGA datasets of similar size around 500 – COAD, LUAD, LUSC, PRAD, SKCM – until $n_f = 42$ (10% of SKCM's size, the smallest of these datasets). We plot the performance in three cases: Euclidean metric for both preprocessing methods, and cosine metric for VST (figure 4.4). For this last case – Euclidean and VST – we also plot the measures at 10% for all datasets in figure .

The federated clustering for single and average linkages on the large RNA-Seq datasets is the closest to the original unfederated agglomerative grouping. Notably, for all preprocessings and metrics, and for all tested values of n_{min}, the FMI averaged over the 10 last merges remains above 0.95. This means that the algorithm makes few errors in forming the major clusters and then merging them. The CCC also stays quite close to 1, standing above 0.9 for single linkage and 0.8 for average linkage. The performance for the smaller datasets is about as good for READ, but quite lower for PAAD. It would be worth investigating more the relationship between dataset size/distribution and performance.

For complete and Ward linkage, the quality is much more variable between the 5 datasets, and less stable between consecutive values of the minimum cluster size, no matter the preprocessing (and metric for complete linkage). This is somewhat surprising for Ward linkage, as its formula for cluster distances is the distance between the clusters' centroids, multiplied by a factor depending on their sizes. With our centroid-based approach, approximating a cluster as its centroid does not change its distance to other clusters. It seems that the local merges at the start influence the structure of the clustering too much too early, and so the initial bad choices cannot be corrected when the centroids are shared.

4.2.3.3 Inversion issues

One major drawback of our approach is that it causes inversions to appear in the dendrogram, even for ultrametric linkages like single, complete, average and Ward linkages. In other words, the pooled dendrogram only has increasing merging heights whereas the federated dendrogram is not guaranteed to. In particular, let us consider the case where a global cluster is created and its centroid is very close to a point clustered on its own in another center. Then, there is a good chance the merge will have a distance smaller than the merge distance of the global cluster, and an inversion will appear.

When significant inversions occur in the dendrogram, the latter is much

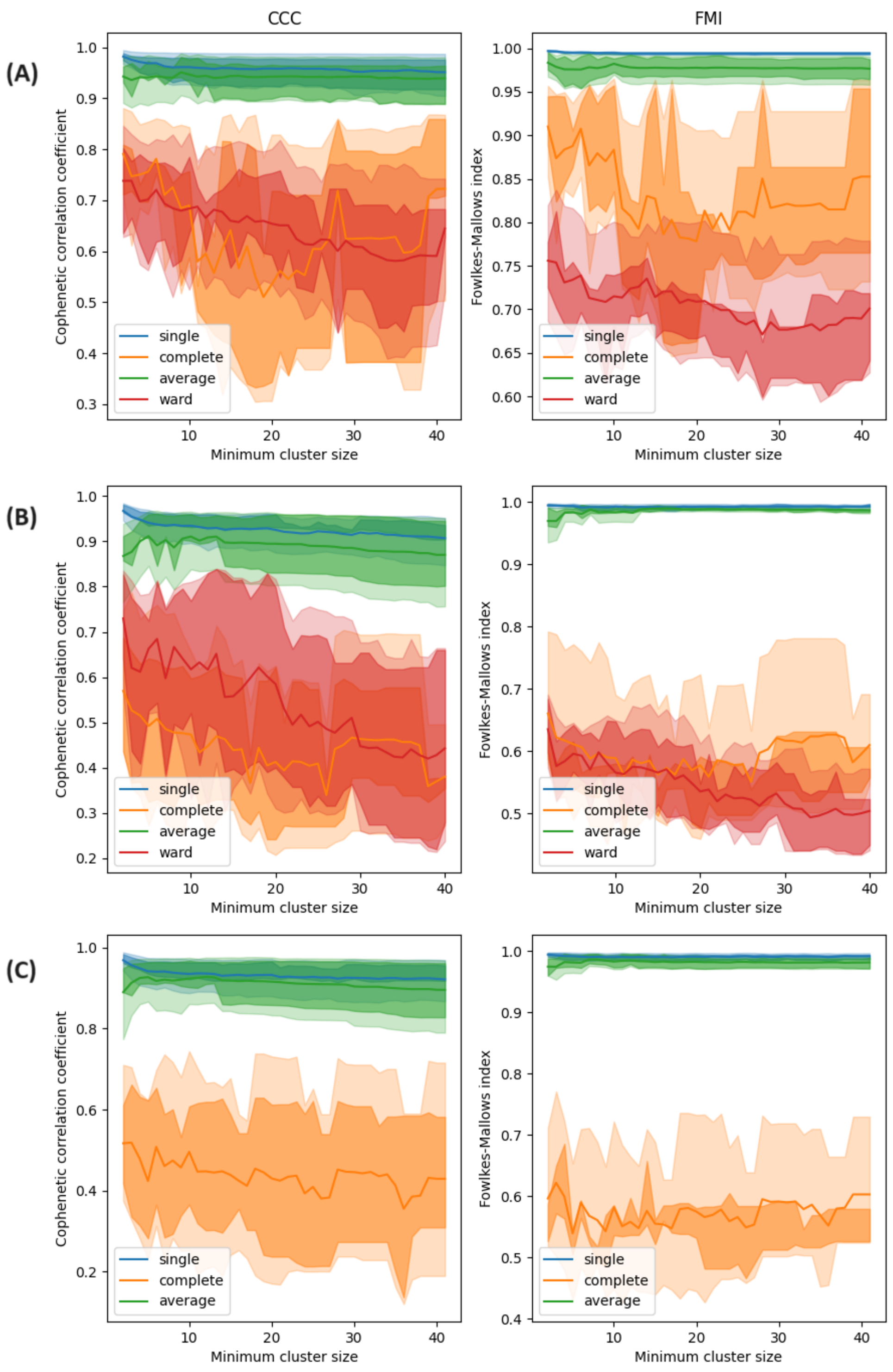

Figure 4.4: Clustering performance with gradual sharing for similarly-sized datasets for single, complete, average and Ward linkages. Different preprocessings and metrics are studied: (A) size factors, Euclidean (B) VST, Euclidean (C) VST, cosine.

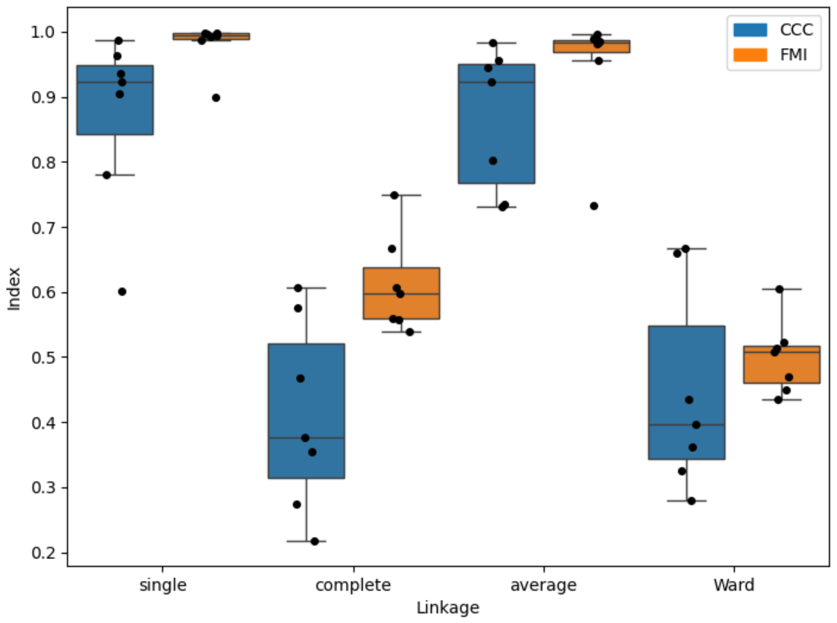

Figure 4.5: Clustering performance at $n_{min} = 10\%$ with gradual sharing for all VST datasets for Euclidean metrics, using the single, complete, average and Ward linkages.

harder to interpret, as can be seen on figure 4.6. Inversions make it difficult to visualize the order of merges and to compare merging distances. One solution to this issue is to enforce the monotonicity of merge distances for ultrametric linkages, as we know that the associated pooled dendrogram has monotonic merge distances. This can be achieved as a post-process after executing the federated clustering algorithm. The question is now how this affects the performance of the federated algorithm.

In order to determine this, we compute the difference in CCC between the algorithm with imposed monotonicty and without. We plot the average over all TCGA datasets for $2 \leq n_{min} \leq 50$ (figure 4.7); each dataset is only included in the average until n_{min} reaches 10% of the size of the equivalent pooled dataset. We observe that there is no major discrepancy in the CCC, with a maximum absolute difference of 0.016. For complete linkage, enforcing a monotonic behaviour even improves the mean CCC with regards to the pooled clustering; for the other linkages, depending on the value of n_{min}, one or the other is better.

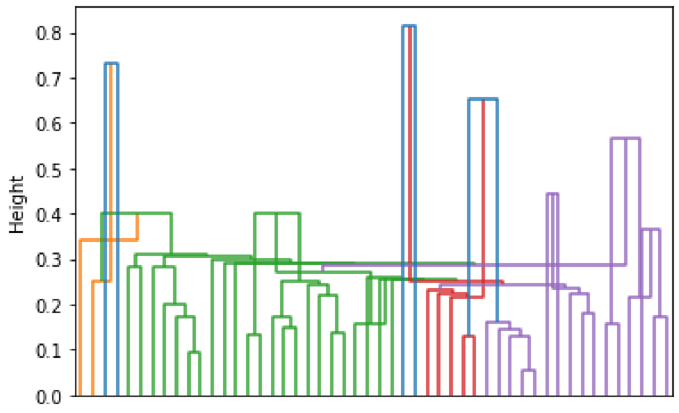

Figure 4.6: A dendrogram with inversions: pairs of clusters merge to form a cluster with lower height than the individual clusters composing it.

4.3 Random projection clustering

4.3.1 Complexity and communication cost analysis

First, all centers project their data in $\mathcal{O}(n_{(k)}dd')$. For the cosine and correlation distances, an added cost of $\mathcal{O}(n_{(k)}d)$ is necessary to normalize the data.

For a cross-silo network in which centers communicate only with the server, the communication cost is that of all centers sending their transformed dataset to the server: $\mathcal{O}(nd')$. Now, the server computes the approximated pairwise distances in $\mathcal{O}(n^2d')$ and the HC with the complexity of the classic pooled approach: either $\mathcal{O}(n^2)$ or $\mathcal{O}(n^2 \log n)$ depending on the implementation.

4.3.2 Performance on toy datasets

To understand the performance of the random projection scheme for HC, we apply it to two very different types of datasets: one that is very unstructured and one that admits a well-defined clustering. For the first type, we take 100 uniformly sampled points in $[0, 1]^{500}$. For the second, we sample 100 points from 5 isotropic Gaussian blobs of standard deviation $\sigma = 0.1$, whose centers are sampled uniformly in $[0, 1]^{500}$. We evaluate the single, complete, average and Ward linkages for the Euclidean metric.

For each evaluated sketch size, we do 100 runs, sampling a different dataset and projection matrix each time. We observe that the clustering performance is great for the easy clustering problem (A) while it is very poor for the uniform

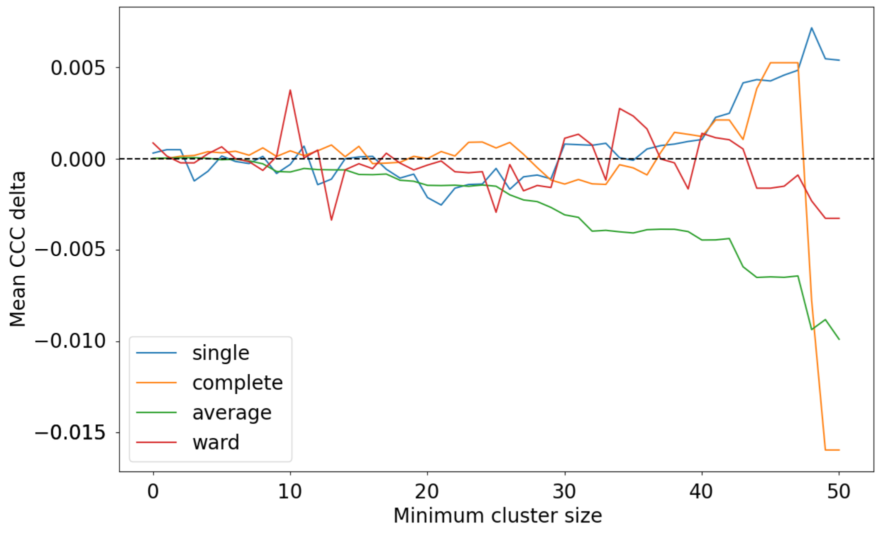

Figure 4.7: Mean CCC difference between enforced monotonocity and original federated algorithm for several linkages (single, complete, average, Ward), averaged over all TCGA datasets.

data (B) (cf. figure 4.8).

With the Gaussian blobs, the CCC reaches 0.9 for a sketch size below 25 for single and average linkage, and around dimension 50-60 for complete and Ward linkage, corresponding respectively to around a 90% and 80% dimension reduction. Recall that the bound on the sampling size for the convergence of pairwise distances depends on the number of samples (100 here) and not the original dimension (500 here). Thus, the scheme is all the more effective for high-dimensional data.

On the other hand, for the uniform dataset, the CCC barely even reaches 0.8 for single linkage when the sketch size equals the original dimension, and is under 0.3 for complete and Ward linkage. This highlights the unstability of agglomerative clustering, as the Euclidean random projection scheme more or less returns a perturbed version of the dataset, which then affects the clustering. The authors of [88] study the stability of HC, and find that single linkage is the most robust, while complete and average linkages are unstable. This is concordant with our experiments, in which single linkage obtains the best performance.

We can also look at the convergence of the ARI for the clustering of the 5 isotropic Gaussian groups when cutting the hierarchical tree at 5 clusters (figure 4.9). The index reaches a value of 1 for all linkages for a sketch size between 5 and 8 (dimension reduction of 98-99%): this corresponds to a

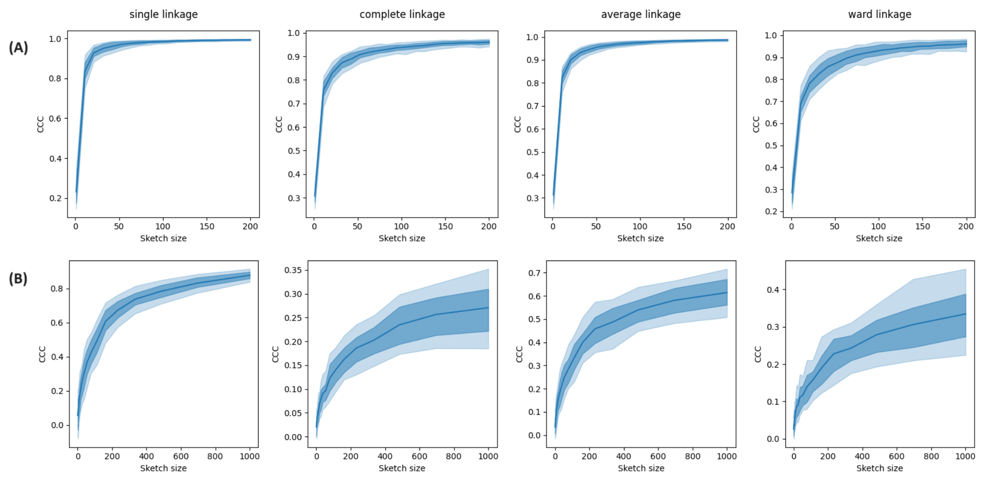

Figure 4.8: Cophenetic correlation coefficient evolution for different linkage functions using two toy datasets: (A) 5 isotropic Gaussian clusters, up to a sketch size of 200 (B) uniform data, up to a sketch size of 1000.

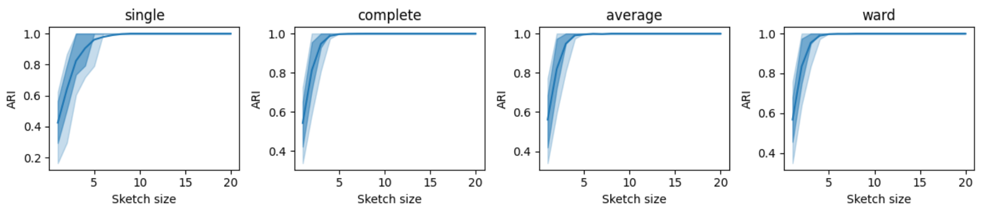

Figure 4.9: Convergence of Adjusted Rand index between pooled and federated clusterings, with a dataset made of 5 isotropic Gaussian clusters for the single, complete, average and Ward linkages.

perfect agreement between the clusters of the pooled and federated instances.

As this is an easy clustering problem (centers are chosen in $[0, 1]^{500}$ and $\sigma = 0.1$), we would also like to analyze the quality as the clustering becomes harder. For this, we compute the threshold size for which the average CCC over 20 runs surpasses 0.95 as σ increases (figure 4.10). Interestingly, while average and single linkage are better at first, we see that as the deviation increases, average and Ward linkages have a slow growth, contrary to single and complete linkages. This is for the specific problem of a few Gaussian clusters, so it would be appropriate to check other instance problems to see if that behaviour remains.

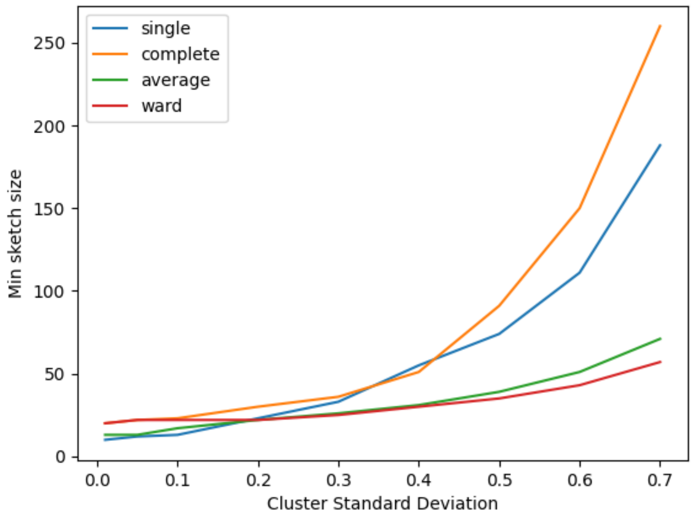

Figure 4.10: Adjusted Rand index convergence with a dataset made of 500 points sampled from 5 isotropic Gaussian clusters. We plot the min sketch size required for the CCC to exceed 95% depending on the standard deviation of the gaussian distributions. Four linkages are represented: single, complete, average and Ward.

4.3.3 Performance on TCGA

We now apply our random projection scheme to real RNA-Seq data from TCGA (described in 3.1), with the same federated split as before.

4.3.3.1 Clustering performance for the Euclidean metric

We now analyze the performance of the random projection algorithm on all TCGA datasets, with a sketch size from 10 to up to 1000. We examine the CCC, which we can compare to the pairwise distance correlation between the pooled dataset and its projection. As with the centroid-based algorithm, we also compute the FMI averaged over the last 10 significant cluster merges. For this experiment, BRCA is excluded[1]. Data is preprocessed with VST and experiments are run with 100 different seeds.

We evaluate the performance for the Euclidean metric, and show the plots for the best and worst performing datasets overall (figure 4.11).

We get great results for single and average linkages. For all datasets, the FMI surpasses 0.9 for a sketch size of 40 in the worst case for average linkage,

[1]Probably due to the size of the dataset, there was a system crash when using BRCA which we did not have time to investigate.

and is always above 0.94 for single linkage for all sketch sizes starting at 10. The CCC approaches 1 with a similar convergence to the pairwise distance correlation for single linkage. It reaches 0.9 for a sketch size of at worst 1000 for all datasets except SKCM, where it is valued at 0.76.

In the case of complete and Ward linkage, the performance varies a lot across datasets, with the CCC and FMI respectively converging towards values as low as around 0.4 and 0.6. As with the toy examples in section 4.3.2, it seems that for a non-trivial clustering problem, the lack of stability of some linkages really impacts the clustering performance of our algorithm. As a result, the random projection method does not appear viable for complete and Ward linkages.

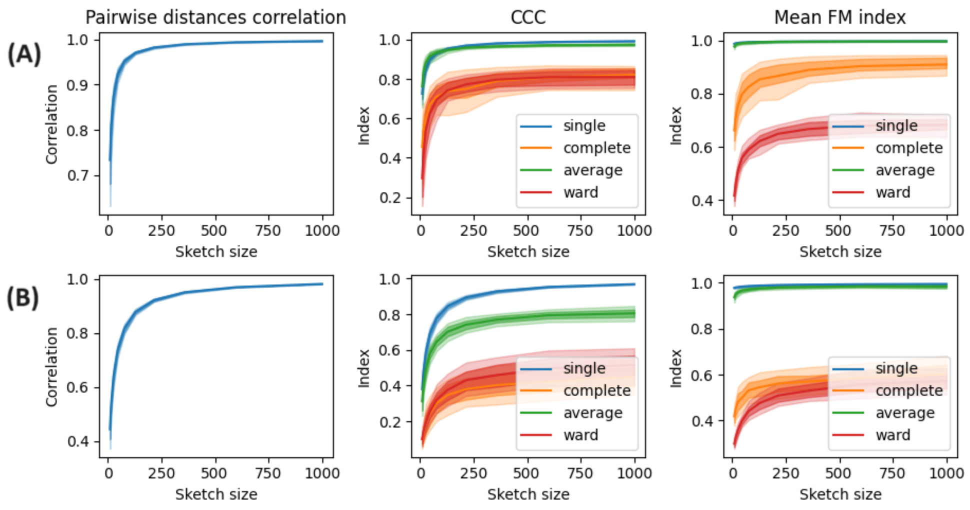

Figure 4.11: Performance of random Euclidean projection scheme on VST counts using the single, complete, average and Ward linkages for two RNA-Seq datasets: (A) COAD (B) SKCM.

We can also look at the convergence of the mean relative error of the cophenetic distance with regards to the sketch size. As opposed to the CCC, this gives the same weight to all cophenetic distances. We plot the relative error over 100 runs for BRCA counts preprocessed with the size factors method (figure 4.12).

We observe that the error for single linkage converges, passing below 5% around dimension 100 for all datasets. For complete and Ward, the error stays above 10% for all sketch sizes up to 1000; even for average linkage, the mean error across runs only drops below 10% for one dataset (SKCM). This empirically confirms the stability of single linkage. It also suggests that for other linkages, our algorithm recovers at best the main clustering trends, but not the low-level details.

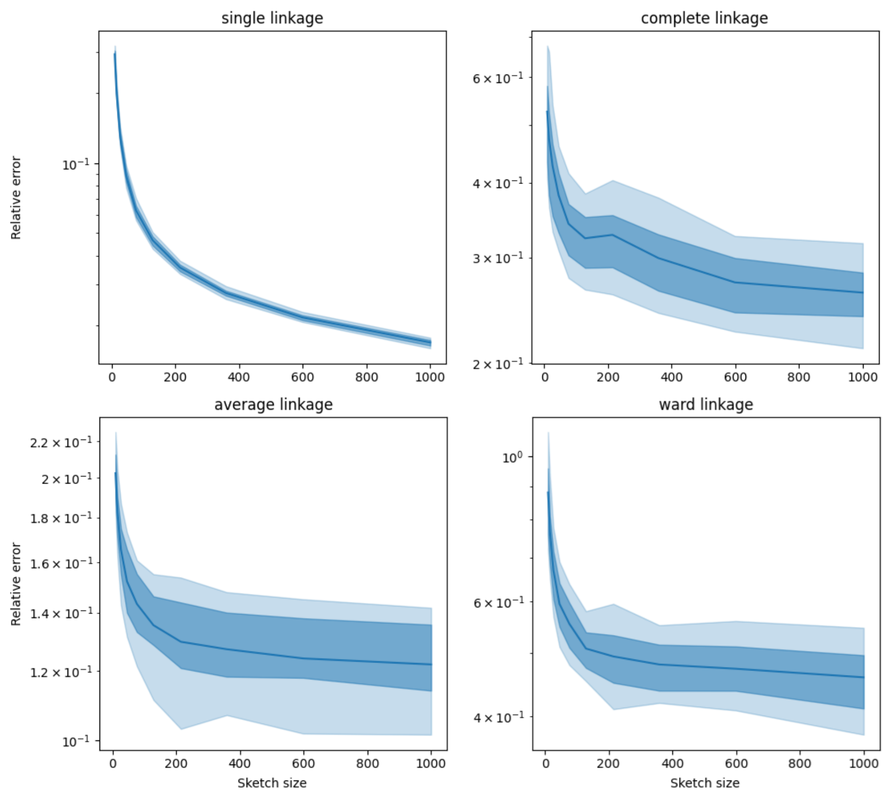

Figure 4.12: Relative cophenetic error of clustering with Euclidean random projection with different linkages for BRCA preprocessed with size factors.

4.3.3.2 Performance comparison across metrics

We extend our analysis to other useful distance metrics for RNA-Seq, and now compare our algorithm for the Euclidean, cityblock and cosine metric. We plot the performance over 100 seeds of the method on the PAAD dataset preprocessed with VST (figure 4.13).

As with the Euclidean metric, the algorithm performs best for single and average linkage. The convergence of distances – both cophenetic and usual – for the cosine metric is as quick as with the Euclidean metric, which we expect as they use the same type of random projection and estimation. Over 100 seeds, all CCC for the Euclidean and cosine metrics are above 0.95 for a projection dimension of 250.

On the other hand, the convergence of ℓ_1 distances is slightly slower, with 25% of runs giving a CCC below 95% at a sketch size of 500. This is surely due to the distance estimator being less efficient than for ℓ_2.

However, the FMI shows quick convergence for all metrics, staying above 0.95 for sketch sizes above 20, corresponding to a dimension reduction of 99.96%.

As for complete linkage, we observe the same somewhat unreliable behaviour as with the Euclidean metric, with the average CCC stagnating around 0.6 for instance for PAAD clustered with the cityblock metric.

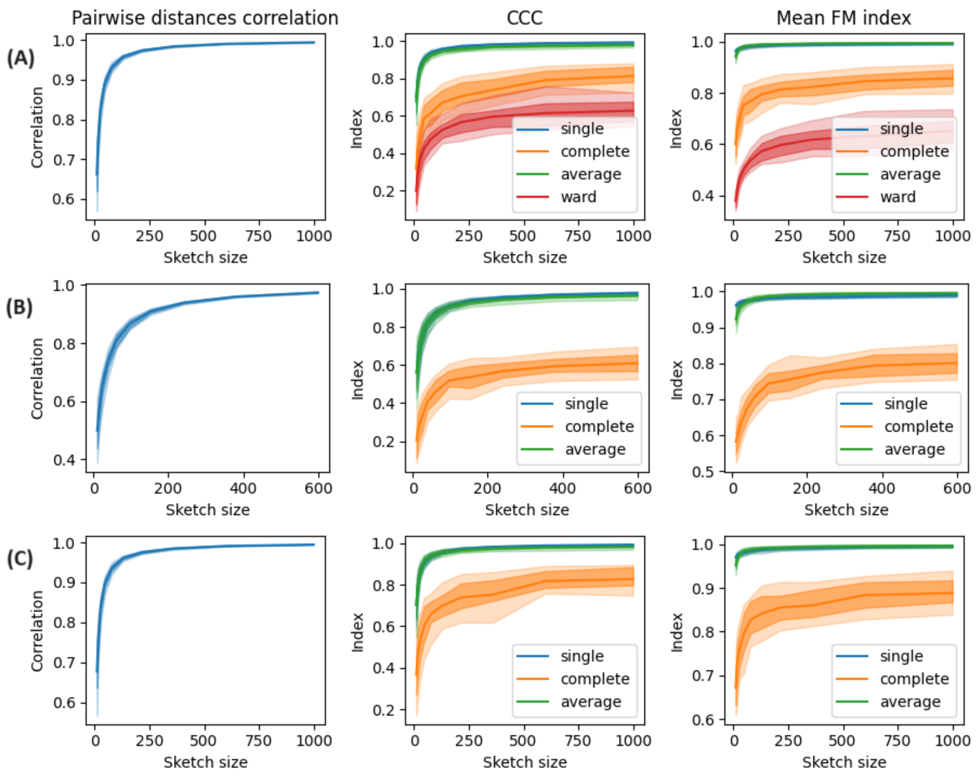

Figure 4.13: Evaluation of random projection scheme with the sketch size on PAAD counts preprocessed with VST for different metrics: (A) Euclidean (B) cityblock (C) cosine. Results are presented for the single, complete, average, and Ward linkages.

4.4 Attacks on random projections

4.4.1 Known input-output attack

We first perform the known input-output attack on orthogonal projections from [47] (described in 3.3.4.2.1). For this attack, the malicious party knows all projected data, as well as some of the input data and their matching projection.

Here we compute the maximum relative error that is guaranteed when the attacker knows a certain percentage of the dataset (equation 3.1). We do this separately for each TCGA dataset and preprocessing method. For each percentage that we consider, we randomly sample the known objects. Each experiment is repeated 10 times with a different seed. We show the results for the two datasets for which we get the worst and best results for the size factors (figure 4.14) and VST (figure 4.15) methods.

We find that the privacy breach is higher with the VST preprocessing than the size factors method for a small known proportion, while the opposite is true for a high proportion. For the size factors method, the relative error does get under 10% for 50% of the points when half the data is known. However, one might wonder how realistic it is for an attacker to acquire such a large portion of the original dataset. Indeed, if we imagine that most of the data is split more or less evenly between silos, they would have to obtain data from several centers. One case where this attack would actually be effective is in a federated setting with private centers where public data is also used, and the public data represents a majority of the global dataset.

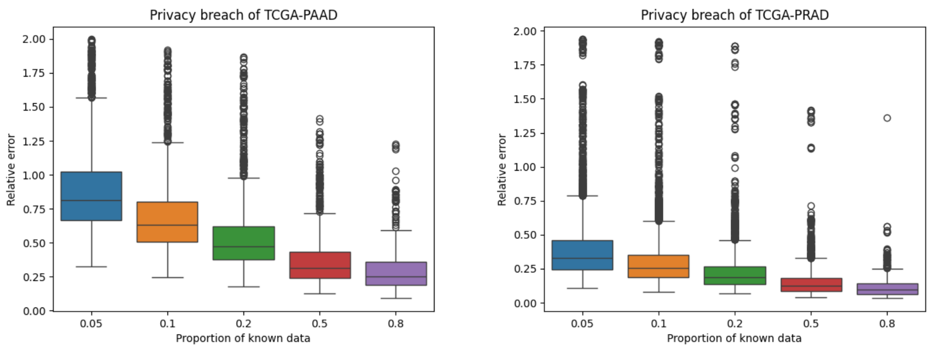

Figure 4.14: Relative error of the known input-output attack with the size factors normalization for the PAAD (left) and PRAD (right) datasets.

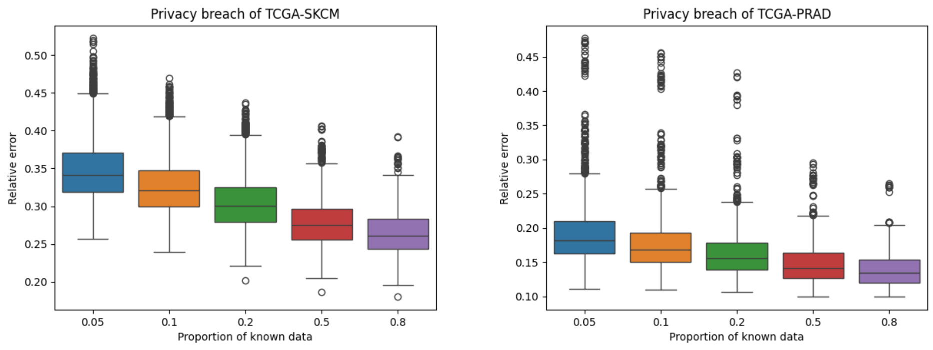

Figure 4.15: Relative error of the known input-output attack with the VST normalization for the SKCM (left) and PRAD (right) datasets.

4.4.2 Property inference attack

We execute the property inference attack explained in 3.3.4.2.2, in which an attacker with access to the projected data uses it to derive personal information on patients, such as gender or age. We perform experiments for the task of gender prediction from the TCGA data (BRCA, COAD, LUAD LUSC, PAAD, READ, SKCM)[1]. Here we use the normalization with size factors.

We take the decision of reducing the number of genes, originally around 50 000, by taking the genes filtered by the preprocessing. These are the genes which have no zero count in any sample. While this is not a realistic filtering step, it should allow us to keep the most important genes and helps reduce the size of the feature space, letting us do more experiments (with computations being obviously faster, especially for the orthogonal projection).

Preprocessing is done separately for the training/validation sets and the test set, to imitate the fact that the attacker and the federated centers do their computations separately. For the centers, we do the preprocessing as if their data was pooled, which supposes the existence of federated RNA normalization methods (which are not the focus of this work).

We then take the intersection of training and testing genes when there is no dimensionality reduction, *i.e* for orthogonal or no projection. We then have around 10 000 - 15 000 genes after filtering (the exact number depends on the random split between training and testing). We do a 60/20/20 split between training/validation/test sets.

[1]We do not include PRAD *i.e* prostate cancer patients, for which gender information is missing – presumably because all patients were cisgender male. We do include BRCA (breast cancer) as there are still a few male patients and not just female patients.

We compute the accuracy for four kinds of projection – orthogonal, and Gaussian for the projected dimensions 100, 1000 and 10000. We compare this to the accuracy of the standard classifier when there is no projection.

In the case that the attacker only has data in the original space, we observe that the model does not manage to learn projection invariance (figure 4.16). While the mean accuracy without projection is 76.5%, the accuracy for all projections mostly stays in the 40-60% range, which is rather poor given that the balance between labels is 60/40. The difference in testing accuracy between the classifier without projection and with any of the projections is statistically significant with a paired t-test (for all conditions, $p \leq 0.003$[1]). Note that even without projection, the accuracy is not excellent, which is probably due to the filtering-out of a majority of genes.

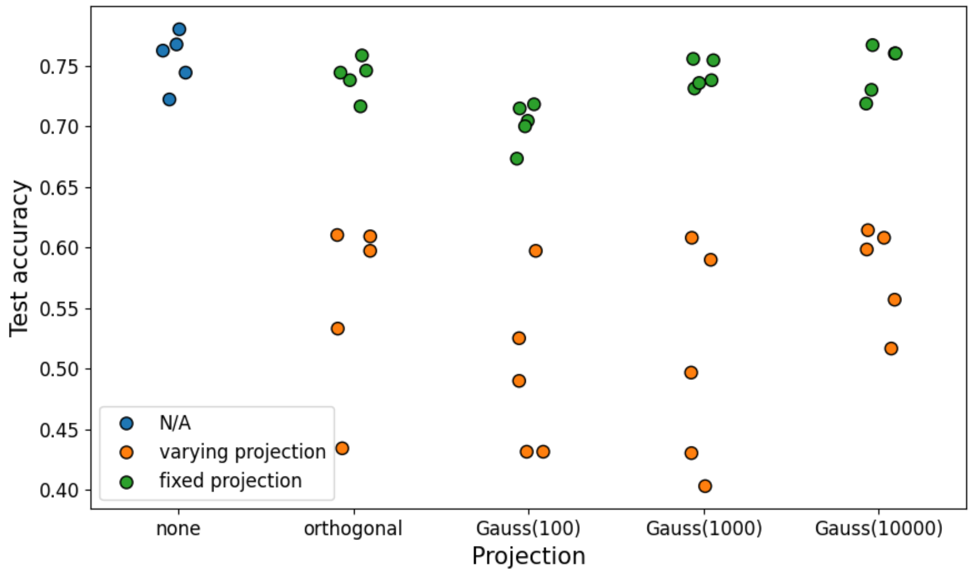

Figure 4.16: Test accuracy of gender prediction over 5 runs for the projection-invariant classifier (orange) VS the fixed-projection classifier (green), compared with the classifier in the absence of projection (blue). Dimensions 100, 1000 and 10000 are tested for the Gaussian projection.

We now analyze the case where the projection is fixed, corresponding to the setting where the attacker obtains labeled data in the projection space (figure 4.16). As expected, for Gaussian projections, as the output dimension increases, the accuracy also improves, with statistical relevance between

[1]computed using $n = 5$ samples per condition, yielding degrees of freedom between 4.48 and 6.19

dimensions 100 and either 1000 or 10000 (paired t-test: $p \leq 0.00712$[1]), but not between 1000 and 10000 (paired t-test: $p = 0.711$[2]). The performance for the orthogonal projection is slightly lower than the performance without projection, but not with statistical significance ($p = 0.269$[3]).

Unsurprisingly, the attack for a fixed projection is more effective than the projection-invariant attack. We verify this by computing a paired t-test on the test accuracies, comparing them between a fixed and variable projection, which gives statistically significant results ($p_{orthogonal} = 0.00474$, $p_{gauss(100)} = 0.00196$, $p_{gauss(1000)} = 0.00420$, $p_{gauss(10000)} = 0.000192$[4]).

4.4.3 Reconstruction attack

We now try to reconstruct samples from their projection by matching the distribution of the reconstruction and other RNA-Seq data in the original space (more details in 3.3.4.2.3). It is critical for us to evaluate the performance of this attack. Indeed, if it succeeds in uncovering the original data, then the projection scheme is unviable as a federated approach, as it means that the raw dataset is essentially shared with the central server.

4.4.3.1 Median of ratios preprocessing tests

We perform the reconstruction attack on all TCGA datasets at once, assuming that they have an overall close distribution as RNA-Seq datasets, even if they correspond to different cancer types. We use a 50/50 random split between the projected data of the server and the data known by the attacker. We analyze three types of projections: orthogonal as well as Gaussian for a sketch size of 1000 and 10000. To evaluate the reconstruction attack, we compute the relative error between the reconstruction and the original sample.

Our first test is done on data preprocessed with the size factors method, and filtering the genes as for the classifier attack in 4.4.2, meaning we have around 10 000 - 15 000 genes. We compare the reconstruction error for the simple linear decoder and a more complex decoder with several layers (figure 4.17). We see that in all cases, the reconstruction is unsatisfactory: the median error is above 50% and the minimum error above 25% in all cases. We still observe that for all types of projections, the complex decoder performs better than the linear decoder.

[1] $n = 5$ samples, 6.78 and 7.73 degrees of freedom
[2] $n = 5$ samples, 6.06 degrees of freedom
[3] $n = 5$ samples, 7.07 degrees of freedom
[4] $n = 5$ samples, respective degrees of freedom 4.33; 4.51; 4.12 and 6.00

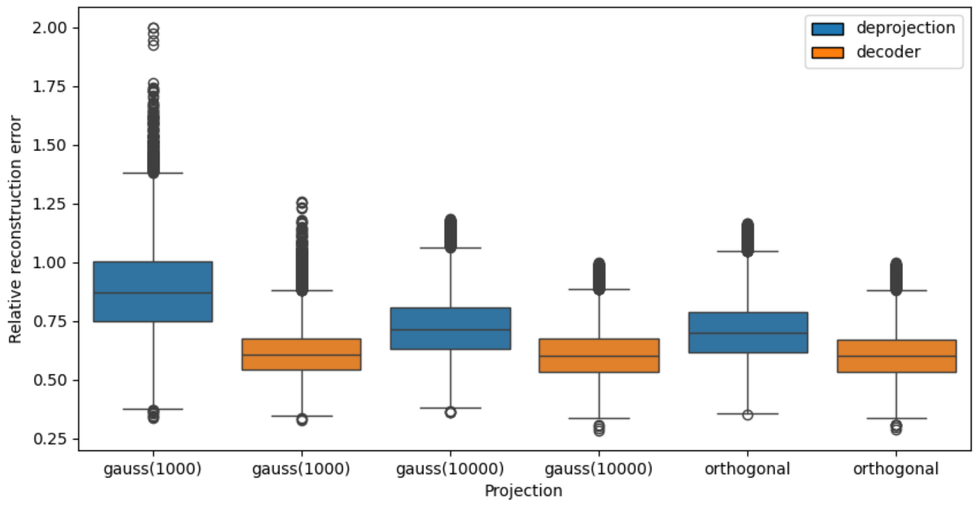

Figure 4.17: Comparison of the reconstruction error on TCGA when using a simple VS complex decoder. Samples are normalized with the median of ratios method. Dimensions 1000, and 10000 are tested for the Gaussian projection.

We notice that the reconstruction is bad even for the orthogonal projection, which theoretically admits an exact solution with error 0 for the linear decoder (which is the inverse of the orthogonal projection). For Gaussian projections, it is not obvious what the best possible error is for the linear decoder. We can take our dataset and compute the reconstruction error using the pseudo-inverse as a decoder (figure 4.18). We see that even with a Gaussian projection of dimension 10000, the relative error is around 40%. We also try initializing the weights in the optimization as the pseudo-inverse of the projection (figure 4.18). Interestingly, the pseudo-inverse serves as a good initialization for the gradient descent, and the relative error even decreases during training. In dimension 10000, half of the data is reconstructed with an error around 50% with this initialization.

4.4.3.2 VST preprocessing tests

We now compute the reconstruction error when the attack is performed on TCGA data normalized with VST. This time, we use all the genes (around 50 000). We also compare it to the error for the size factors preprocessing, selecting all the genes as well. The reconstruction is much better with VST (figure 4.19), as it is around 20% for most samples compared to around 75% for the previous preprocessing method. The discrepancy is statistically significant, as determined by a paired t-test ($p = 0.0000491$ for the mean

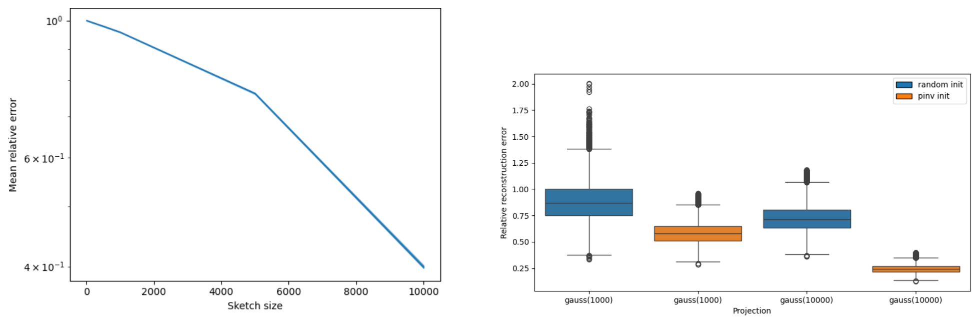

Figure 4.18: Initializing the decoder as the projection's pseudo-inverse improves reconstruction. Left: relative reconstruction error using the pseudo-inverse of the projection, averaged over 10 random Gaussian projections for each sketch size and TCGA dataset. Right: comparison of the reconstruction error on TCGA with a linear decoder when initializing the weights randomly or as the pseudo-inverse, for projection dimensions 1000 and 10000.

reconstruction error from a Gaussian projection of dimension 1000; computed from $n = 5$ sampled runs, giving 8 degrees of freedom). The reconstruction is not perfect, but low enough that it is unclear if the attacker could learn something from it.

We try to combine the reconstruction and property inference attack to see if the VST reconstruction is meaningful. We suppose that the attacker trains a classifier on their data to predict the gender attribute. They can then feed the reconstruction of the server data to the network to recover the label. We compare the testing accuracy between the original federated dataset and the reconstruction from its projection. The experiment is run for 5 seeds.

While the original dataset has a great accuracy, with a mean value of $0.90\,(\pm 0.06)$, the reconstructed dataset's average accuracy is valued at $0.57\,(\pm 0.09)$. Even though the original label imbalance is about a 60/40 split, the prediction on the reconstruction always favours one label: we find that the most predicted label is chosen from 87 to 99$\%$ of the time. Thus, we cannot conclude that the reconstruction is meaningful from this experiment.

4.5 Summary

4.5.1 Algorithms performance analysis

Our experiments in sections 4.2.3.2 and 4.3.3 suggest that both algorithms – based on centroids and random projections – perform quite well for single

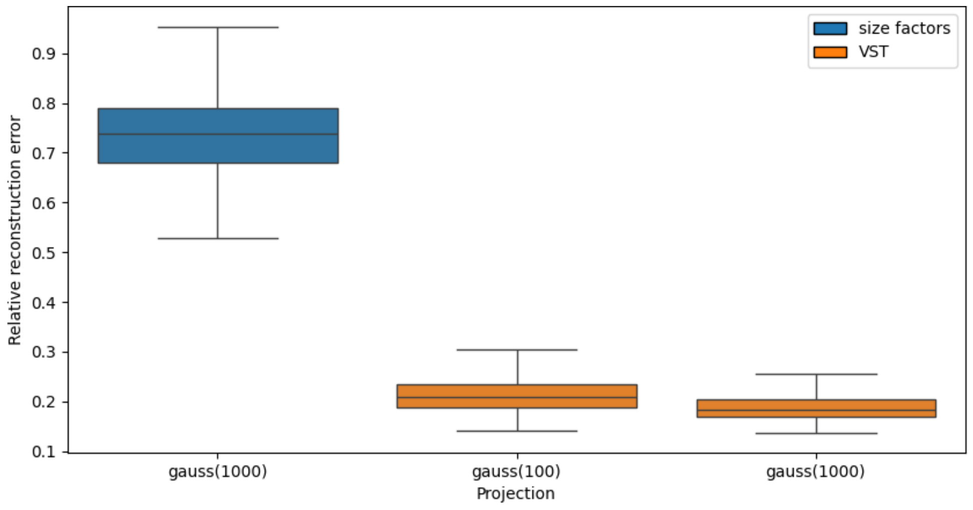

Figure 4.19: Comparison of the reconstruction error on TCGA between the size factors and VST methods, using all genes. Gaussian projections of dimension 100 and 1000 are studied here.

and average linkages on RNA-Seq data. This performance is maintained even for a high minimum public cluster size n_{min} in the first case (see figures 4.4, 4.5), or for large dimensionality reduction in the second case (cf. figures 4.11 and 4.13). Therefore, our algorithms seem to offer a reasonable privacy/performance trade-off for single and average linkages.

For complete and Ward linkages however, performance varies a lot across datasets, as shown in figures 4.4 and 4.3 for instance. While the clustering is great in some cases, its low performance in other cases makes it unpredictable for now. It could be worth investigating the difference in performance between datasets. The lack of stability of these linkages is probably a large factor in the low clustering quality of our algorithms. Indeed, these linkages lead to satisfactory results in easy clustering problems (*e.g* clear-cut Gaussian clusters as in figure 4.9), where perturbing the dataset rarely affects the clustering trends.

Thus, for difficult clustering problems, our methods seem to be adapted for single and average linkages only. We note that single linkage is rarely used in real instances, including gene expression analysis. On the other hand, average linkage is actually often employed in practice and works with all metrics; this makes our algorithms viable for real-case usage in terms of performance.

4.5.2 Attacks analysis

Our different attacks on random projections in section 4.4 are overall rather ineffective. We know a prior exists on genetic data and we examined cases with zero loss of information (orthogonal) or low loss (Gaussian projections with almost no dimension reduction). Therefore, we could expect to recover much more sensitive information than what we obtained. Genetic data is quite complex, and we suppose that its high dimension makes attacks much more difficult given the discrepancy between the number of samples and the number of genes in our attacks.

Specifically, the projection-invariant classifier (see figure 4.16) probably cannot be trained successfully with such a small dataset. For the reconstruction attack, it might be that the reconstruction follows the distribution of RNA-Seq data but remains far from its original match. This is something that could be investigated, as our preliminary analysis makes it unclear whether we can learn information from reconstructions that are still around 20% off (figure 4.19). We could not manage to accurately predict the gender of reconstructed samples, and most samples were classified together (cf. section 4.4.3.2), which does not indicate a meaningful reconstruction. One could also check if differentially expressed genes between the original samples and their reconstruction match to further test the significance of reconstructions.

4.6 Samplewise clustering methods comparison

As we have developed two different approaches for samplewise clustering, we ponder their differences to understand the benefits and drawbacks of each method.

To begin with, the result of the centroid method depends on the data distribution between the centers, while the output of the random projection method does not. In particular, the centroid-based method benefits from a heterogeneous partition of the data, in which every cluster is mostly contained in one center. On the other hand, there are more approximations when one cluster is split between a lot of centers. Contrary to this, the second method returns exactly the same results no matter the distribution, given an identical projection matrix.

Then, the centroid-based approach is quite focused on HC and can at best be generalized to federating other clustering algorithms. As explained before, the projection-based algorithm has many known applications other than clustering, such as visualization or kernel methods.

Another difference between the approaches is in the network communication. The random projection method requires only one round of communication (assuming a seed has been agreed upon previously) whereas the centroid approach has $\mathcal{O}(n)$ communication rounds. In practice, this can create a bottleneck in the computation, which should be investigated empirically with real federated software.

We can also consider variants of our centroid-based algorithm to reduce communication. Let us suppose that communication is the biggest time factor and local computation is negligible. Instead of having a communication round for each merging step, we could imagine ways to enforce less frequent checkpoints. For instance, instead of executing only the merge with minimum cluster distance, one could execute the top α merges, $1 \leq \alpha \leq n-1$, which would add parallelization and lower the number of communication steps. Another possibility would be for all big centers to merge clusters until one private group is of size n_{min} at least, then share all those groups, and repeat the process.

Privacy-wise, it seems that the first algorithm offers a reasonable privacy/performance trade-off, as n_{min} can be rather high without dropping in quality, so that centroids are averaged over many points. It is also quite straightforward to understand what it is shared to the server. On the other hand, for the projection algorithm, it is unclear what is leaked. As the server receives the complete transformed dataset, the data is highly vulnerable to collusion by a malicious center with the server. Other than that, our experimental analysis suggests that it is hard to attack transformed bulk RNA-Seq data using prior knowledge. Of course, just because our attack strategies have failed does not mean none would succeed. Therefore, it is difficult to be certain that the projection-based approach is truly secure for high-dimensional data.

To conclude this part, we summarize several aspects of both approaches in table 4.1.

Approach	Centroid-based	Projection-based
Privacy parameter	min size of shared centroid	projection dimension
Complexity	$\mathcal{O}(n^2(d+n))$	same as pooled
Communication rounds	$\mathcal{O}(n)$	$\mathcal{O}(1)$

Table 4.1: Algorithm comparison between the centroid and projection-based approaches

Chapter 5

Discussion

5.1 Summary of our findings and their relevance

5.1.1 Key findings

We have shown that genewise federated agglomerative clustering can be achieved without difficulty for all linkages and metrics and while preserving samples' privacy. Given that our approach yields pooled-equivalent performance and complexity, it should be preferred over other privacy-protecting methods. Indeed, SMPC guarantees equal performance but at a higher computational cost, while DP produces approximate results. Recall that our method is only valid in the case that feature information is non-sensitive, as is the case with genes; in other settings, SMPC-based techniques might be more adapted.

With regards to samplewise federated clustering, we have developed two distinct approaches: one based on approximating groups of points as their centroid on the server, and one based on random projections. Both have great performance for several common metrics when using single and average linkages.

For single linkage, a SMPC approach had been proposed [63], but it was tested on low-dimensional data and adapted to the case of two parties. Here, our methods can operate on high-dimensional data and in the cross-silo setting, *i.e* with more than two data centers. The authors of [32] did propose random projections for single linkage, but their experiments focused on the cost of the associated MST. Here, we confirm the empirical usefulness of this technique for clustering by using HC-specific metrics.

Though single linkage is somewhat uncommon in practice, this is not the case of average linkage, which is one of the most popular linkages, notably in RNA-Seq data analysis. This suggests that our algorithms can effectively be useful in practice as they will be in direct correspondence with centralized genetic analysis pipelines. We expect that this equivalence makes them rather likely to be adopted, compared to the previously mentioned DP approaches [69, 68] , which optimize the lesser known Dasguspta's objective.

Regrettably, our methods do not perform consistently well in our experiments for complete and Ward linkages, which are also widely used. This suggests that a SMPC-based approach might be required for these unstable linkages, so as to guarantee equivalent performance with pooled clustering.

Privacy-wise, our centroid approach offers a very reasonable performance/privacy trade-off. It has a satisfactory quality attained for as high as 10% of dataset sizes selected as the minimum number of points in a shared centroid. Meanwhile, the random projection method works well with high dimension reduction. Our empirical analysis suggests that it is difficult to attack in a semi-honest setting when dealing with bulk RNA-Seq data. However, in a malicious setting, it is obviously vulnerable to collusion between a center and the server.

To complement our work, in the literature, the privacy properties of random projections have also been investigated from a theoretical viewpoint, via the DP setting [89, 90]. They offer insights into what level of dimension reduction is required in order to preserve a certain degree of DP.

5.1.2 Impact of our work

5.1.2.1 Federated analytics

From the viewpoint of federated analytics, our contribution is two-fold. First, we have proposed a novel clustering approach by approximating groups of points as their centroids in external centers. This is one way to solve the impossibility of sample-to-sample comparison in different data centers which could serve for other algorithms.

To our knowledge, previous privacy-enhancing methods for HC relied on either on SMPC or DP. In the first case, the proposed method was for two parties only [63], which means an inability to face a case with several data centers, which is common in practice, in addition to heavy communication costs because of cryptography protocols. In the second case, the proposed techniques [69, 70] optimized Dasgupta's objective instead of the traditional agglomerative algorithms with linkages. Yet, the latter are the most prevalent in RNA-Seq analysis, meaning that bioinformaticians might not be able to interpret their results as easily. Thus, our centroid-based solution could be the most suited for multi-party agglomerative HC on genetic data, for scientists aiming to have an algorithm as close to what they usually use but with a bigger focus on privacy.

Then, we have studied the use of random projections for clustering. Random projections are a well-known technique, but they have seldom been applied to federated works. Yet, as they offer a way to compute pointwise distances on the server, they can potentially aid in several domains, such as clustering, visualization or kernel methods. However, their privacy guarantees are somewhat blurry, and they are especially vulnerable to collusion between malicious parties. Several attacks from the existing literature can be used against them [47, 25], and we have also proposed a new reconstruction attack, based on distribution matching in the input space. We have applied all these attacks to randomly-projected RNA-Seq data, so that our experimental assessment of the privacy/performance trade-off of random projections can help others understand whether or not to employ them in federated projects given their specific needs.

Federated clustering remains underdeveloped. While federated methods for k-means exist [17, 23], other clustering techniques such as density-based still do not have a standard federated equivalent, which is why our work fills a critical gap in the federated landscape.

5.1.2.2 Gene expression analysis

Gene expression analysis has had a crucial impact on the understanding of biological functions. It has helped us in the prevention, diagnosis and treatment of diseases such as cancer or cardiovascular technologies. Given the sensitivity of genetic data, we need to adapt our data analysis pipelines so as to protect the privacy of patients, and leverage currently unaccesible data pools.

Agglomerative clustering plays a critical role in these pipelines, as it benefits the exploratory and quality control steps. We propose algorithms for gene and samplewise clustering, allowing researchers to assess the relationships between genes and between patient data. Therefore, our federated algorithms can improve privacy-preserving procedures and help uncover new insights about diseases using previously unavailable data.

5.2 Ethical and sustainability considerations

5.2.1 Concerns about data privacy regarding the TCGA dataset

The TCGA gene expression and metadata we use is part of the open-access TCGA database. It only comprises data that does not cause any risk of patient re-identification. In our project, we do not utilize the controlled-access database, which contains more sensitive genetic information and is only available to qualified researchers who have agreed to certain contractual obligations.

Participants of the TCGA project underwent a comprehensive informed consent protocol, which includes an extensive discussion about the risks and benefits. Note that the ethical protocol used by TCGA is also stricter than the guidelines from the National Institutes of Health for Human Research Protections in the United States.

More information about TCGA's ethical policies can be found on the National Cancer Institute website[1].

5.2.2 Ethical questions

Analysis of genetic data comes with ethical concerns given its sensitive nature. Genetic expression contains information about a person's predispositions to diseases, and could thus be used to discriminate against that person. Given the hereditary nature of genetic data, when a patient's information is divulged, it is also potentially revealing about their relatives' information. Therefore, it is extremely important that any genetic data analysis protocol follow strict guidelines with clear privacy properties.

In this work, we have presented several algorithms with different privacy mechanisms. While not standard, the privacy parameter of our centroid-based method is intuitive enough. It can be tweaked so as to guarantee a high level of privacy in a context where the clustering results are deemed acceptable to share. It is also rather unaffected by malicious centers or a server who decide to collude.

On the other hand, random projections rely on the assumption that the transformation cannot be reversed by the server. This depends heavily on the data distribution in the general case, and it cannot be strictly proven that

[1]https://www.cancer.gov/ccg/research/genome-sequencing/tcga/history/ethics-policies

this procedure is safe. It is also quite vulnerable to collusion, which has serious consequences. In the orthogonal case, it leads to a complete decryption of the data. In the Gaussian case, our results suggest that the server could successfully train a classifier to predict information about patients. In this context, we are not convinced that hospitals would accept to use the random projection protocol in practice.

5.2.3 Sustainability issues

By contributing to privacy-enhancing algorithms for DEA, we aim at furthering research efforts on the treatment of diseases, in order to improve quality of life. For instance, early detection of pathologies can sometimes help curing them more quickly and at a lower cost. According to the World Health Organization, global cancer rates could face a 77% increase by 2050, highlighting the need for more effective and targeted medical care. Again, by allowing more data to be leveraged in computations by preserving their privacy, we should be able to move towards more efficient treatments.

5.3 Limitations

A limitation of our work is its specific application to genetic data. Indeed, vertically-partitioned clustering would be harder if feature-wise privacy was required. Moreover, our samplewise clustering algorithms apply to datasets $X \in \mathbb{R}^{n \times d}$ with typical orders of magnitude $n \sim 10 - 1000$ and $d \sim 10000 - 100000$. Recall that while the random projection algorithm has optimal complexity, the centroid-based algorithm requires a $\mathcal{O}(n^3)$ cost in terms of n, which can be prohibitive for large-scale applications. Note that this is often not the case, given that in the classical setting, the best-case scenario comes with a $\mathcal{O}(n^2)$ complexity that can already be quite restrictive.

It is worth mentioning that we focused specifically on the bulk RNA-Seq modality. While this sequencing method remains very popular, single-cell sequencing and transcriptomics are increasingly used for their better precision. Thus, it could be useful to have federated clustering algorithms adapted to these. A big difference would be in terms of privacy constraints, as single-cell data has stricter priors, and is notably very sparse [91]. Thus, our algorithms could potentially be easier to attack with such data.

Here we focused on approximate algorithms so as to get acceptable complexity for the cross-silo setting while maintaining a certain privacy threshold. The private alternative for exact clustering would be to use SMPC, which has higher communication costs but yields pooled-equivalent results. Even then, privacy concers remain: the quantities computed on the server have to be non-sensitive in the first place. In our case, computing the distance matrix on the server amounts to having the pooled dataset up to isometry, which can be attacked similarly as random projections. However, there is no collusion issue in that case.

Chapter 6

Conclusions and Future work

In this chapter, we summarize our findings and their limitations, and present possible avenues for future work.

6.1 Conclusions

In this work, we presented methods for cross-silo federated clustering both in the genewise and samplewise settings. We studied their complexity, privacy characteristics and empirical performance on high-dimensional genetic data. Our methods have proven reliable for RNA-Seq data for common parameters of the agglomerative clustering procedure, meaning that they can be integrated into federated data analysis pipelines for real-case projects.

In this thesis, we asked ourselves how federating the HC of genetic data affects performance and complexity, measured against equivalent unfederated algorithms. We also inquired to what extent these federated approaches respect data privacy.

First, we found that federated gene-wise HC can be achieved with pooled-equivalent results at a low complexity, while preserving sample-wise privacy. Therefore, there is no performance, complexity or privacy cost to federating the classic agglomerative clustering algorithm for gene-wise comparisons. Second, for sample-wise HC, our federated procedures had to be approximate versions of their classic counterparts, in order to retain low enough complexity and communication costs. Empirically, for genetic data, the clustering results of our approaches were consistently close to the unfederated agglomerative algorithms, for both single and average linkage, and this, at adequate levels of data privacy for the centroid-based method. For the random-projection approach, satisfactory performance was obtained for dimensions where reconstruction attacks failed, but where collusion between a data center and the server would have severe effects on privacy.

In brief, federating the HC of data samples without using SMPC methods has a negative effect on performance. It also involves a higher complexity, with back-and-forth communication between devices, but with a reasonable performance/privacy trade-off for our first algorithm. While tt requires acceptable communication costs for a low-dimensional dataset (either by nature or through Gaussian projection) for our second algorithm, this comes at a potential price in genetic data privacy.

Federated analytics certainly comprise different challenges to those of FL. In particular, the quantities to compute in such algorithms are sometimes more closely related to the original data, and thus harder to protect. We cannot simply average local gradient contributions in order to mask unique inputs as is possible with FL. In particular, the pairwise point relationships are often crucial in order to compute distance matrices for clustering or visualization techniques. This is a great difficulty when points are distributed across several clients.

We note that federated clustering is only useful as long as the distributed data can be properly preprocessed in a federated manner. RNA-Seq data is famously sensitive to batch effects – effects due to the different sequencing conditions in different experiments – which must be removed for analysis. Normalization methods such as VST are also significantly useful. Therefore, we could focus next on federating gene expression preprocessing methods in order to move towards a complete federated RNA-Seq analysis pipeline.

6.2 Future work

The obvious next thing to do is to implement our algorithms in a software for federated computations. Indeed, this would let us test it in practice, and especially verify if the compute time is reasonable. It would also mean that it would be available for researchers and thus could serve in federated projects that include a HC step.

With regards to the centroid-based algorithm, a real implementation would also let us examine the empirical compute cost when communication between the server and the centers occurs. We could then decide whether we need to reduce the communication cost using variants of the algorithm as mentioned in section 4.6. It might be that less frequent checkpoints are possible at a small price in performance, given the observed stability of single and average linkages in our experiments.

We now focus on the random projection algorithm. We should look into detail for possible cryptography schemes that allow centers to choose a secret common seed. Cryptography was out of the scope of this work, but such a scheme would be required for a complete implementation of our federated algorithm. We would then need to analyze the complexity and communication cost of the selected mechanism.

As mentioned previously, the JL projection method has already been studied under the DP framework [89, 90]. This could be useful in order for us to have a theoretical way to examine the privacy of our random projection algorithm for the Euclidean distance. We could then try to extend it to the cityblock, cosine and correlation distances. In terms of practical experiments, we would have to test on real RNA-Seq data what DP constants are required to reasonably protect it. Indeed, if the level of noise deemed necessary for privacy purposes is too high, it means that our algorithm may not actually have a satisfactory privacy/performance balance.

In relation to this, there is also room for improvement in the attacks on random projections. First, we did not have time to try all attacks on all preprocessing methods and using the whole gene set. Besides, as mentioned previously, we did not exhaustively check whether a somewhat medium reconstruction error (less than 20%) is enough to leak sensitive information about the data. We could also focus more on the collusion problem for Gaussian or Cauchy projections. We obtained satisfactory clustering quality with high dimensionality reduction, given that the projection bound depends on the number of samples and not the dimension. Therefore, it would be interesting to assert what can actually be learnt when the projection matrix is divulged.

Finally, it would be useful to perform a more extensive performance benchmark of our algorithms. With regards to the centroid-based algorithm, we could analyze its performance depending on the splits between centers, both in terms of data heterogeneity and local dataset size. Additionally, not only could we use other gene expression datasets, but we should try to evaluate the biological relevance of the main clusters found. Indeed, we could check whether the clusters correspond to actual subgroups corresponding to precise medical information. This could potentially help reinforce our findings on the empirical quality of the clustering.

References

[1] O. Ginsburg, P. Ashton-Prolla, A. Cantor, D. Mariosa, and P. Brennan, "The role of genomics in global cancer prevention," *Nature Reviews Clinical Oncology*, vol. 18, no. 2, pp. 116–128, Feb 2021. doi: 10.1038/s41571-020-0428-5 [Page 1.]

[2] J. Khan *et al.*, "Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks," *Nature Medicine*, vol. 7, no. 6, pp. 673–679, Jun 2001. doi: 10.1038/89044 [Page 1.]

[3] J. E. Dancey, P. L. Bedard, N. Onetto, and T. J. Hudson, "The genetic basis for cancer treatment decisions," *Cell*, vol. 148, no. 3, pp. 409–420, Feb 2012. doi: 10.1016/j.cell.2012.01.014 [Page 1.]

[4] Z. Wang, M. Gerstein, and M. Snyder, "RNA-Seq: a revolutionary tool for transcriptomics," *Nat. Rev. Genet.*, vol. 10, no. 1, pp. 57–63, Jan. 2009. doi: 10.1038/nrg2484 [Pages 1 and 15.]

[5] C. Soneson and M. Delorenzi, "A comparison of methods for differential expression analysis of RNA-seq data," *BMC Bioinformatics*, vol. 14, no. 1, p. 91, Mar. 2013. doi: 10.1186/1471-2105-14-91 [Pages 1 and 16.]

[6] F. Murtagh and P. Contreras, "Algorithms for hierarchical clustering: an overview," *WIREs Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 86–97, 2012. doi: 10.1002/widm.53 [Pages 1 and 10.]

[7] C. M. Koch, S. F. Chiu, M. Akbarpour, A. Bharat, K. M. Ridge, E. T. Bartom, and D. R. Winter, "A Beginner's Guide to Analysis of RNA Sequencing Data," *American Journal of Respiratory Cell and Molecular Biology*, vol. 59, no. 2, pp. 145–157, Aug. 2018. doi: 10.1165/rcmb.2017-0430TR Publisher: American Thoracic Society - AJRCMB. [Pages 1 and 16.]

[8] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: a survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1370–1386, 2004. doi: 10.1109/TKDE.2004.68 [Pages 1 and 16.]

[9] D. Müllner, "Modern hierarchical, agglomerative clustering algorithms," Sep. 2011, arXiv:1109.2378 [cs, stat]. [Pages 1 and 10.]

[10] D. Greenbaum, A. Sboner, X. J. Mu, and M. Gerstein, "Genomics and privacy: implications of the new reality of closed data for the field," *PLoS Comput. Biol.*, vol. 7, no. 12, p. e1002278, Dec. 2011. [Pages 2 and 7.]

[11] A. C. Yao, "Protocols for secure computations," in *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, Nov. 1982. doi: 10.1109/SFCS.1982.38 pp. 160–164, iSSN: 0272-5428. [Pages 2 and 22.]

[12] C. Dwork, "Differential Privacy," in *Automata, Languages and Programming*, ser. Lecture Notes in Computer Science, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds. Berlin, Heidelberg: Springer, 2006. doi: 10.1007/11787006_1. ISBN 978-3-540-35908-1 pp. 1–12. [Pages 2 and 23.]

[13] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. PMLR, Apr. 2017, pp. 1273–1282, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a.html [Pages 2 and 7.]

[14] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. Association for Computing Machinery, 2015. doi: 10.1145/2810103.2813687. ISBN 9781450338325 p. 1310–1321. [Pages 2 and 7.]

[15] A. R. Elkordy, Y. H. Ezzeldin, S. Han, S. Sharma, C. He, S. Mehrotra, and S. Avestimehr, "Federated Analytics: A survey," Feb. 2023, arXiv:2302.01326 [cs]. [Pages 2 and 8.]

[16] A. Hartebrodt and R. Röttger, "Federated horizontally partitioned principal component analysis for biomedical applications," *Bioin-*

formatics Advances, vol. 2, no. 1, p. vbac026, 04 2022. doi: 10.1093/bioadv/vbac026 [Pages 3 and 8.]

[17] H. H. Kumar, K. V R, and M. K. Nair, "Federated k-means clustering: A novel edge ai based approach for privacy preservation," in *2020 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, 2020. doi: 10.1109/CCEM50674.2020.00021 pp. 52–56. [Pages 3, 8, and 79.]

[18] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, Jul. 2020, pp. 5132–5143. [Online]. Available: https://proceedings.mlr.press/v119/karimireddy20a.html [Page 7.]

[19] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020. [Page 7.]

[20] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2021. [Page 8.]

[21] N. Rieke *et al.*, "The future of digital health with federated learning," *npj Digital Medicine*, vol. 3, no. 1, p. 119, Sep. 2020. doi: 10.1038/s41746-020-00323-1 [Page 8.]

[22] T. Marchand, B. Muzellec, C. Béguier, J. Ogier du Terrail, and M. Andreux, "Securefedyj: a safe feature gaussianization protocol for federated learning," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 36 585–36 598. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/ed3c686f9cda57e56cc859402c775414-Paper-Conference.pdf [Pages 8 and 20.]

[23] D. K. Dennis, T. Li, and V. Smith, "Heterogeneity for the Win: One-Shot Federated Clustering," in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, Jul. 2021, pp. 2611–2620, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v139/dennis21a.html [Pages 8 and 79.]

[24] J. Zhou *et al.*, “Ppml-omics: A privacy-preserving federated machine learning method protects patients’ privacy in omic data,” *Science Advances*, vol. 10, no. 5, p. eadh8601, 2024. doi: 10.1126/sciadv.adh8601 [Page 9.]

[25] K. Ganju, Q. Wang, W. Yang, C. A. Gunter, and N. Borisov, “Property Inference Attacks on Fully Connected Neural Networks using Permutation Invariant Representations,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. Toronto Canada: ACM, Oct. 2018. doi: 10.1145/3243734.3243834. ISBN 978-1-4503-5693-0 pp. 619–633. [Pages 9, 33, and 79.]

[26] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, “Membership inference attacks against machine learning models,” in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017. doi: 10.1109/SP.2017.41 pp. 3–18. [Page 9.]

[27] B. Zhao, K. R. Mopuri, and H. Bilen, “idlg: Improved deep leakage from gradients,” 2020. doi: 10.48550/arXiv.2001.02610 [Page 9.]

[28] L. Kaufman and P. J. Rousseeuw, *”6. Divisive Analysis (Program DIANA)”*. Wiley, 2009. ISBN 978-0-470-31748-8 [Page 10.]

[29] P. Virtanen *et al.*, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020. doi: 10.1038/s41592-019-0686-2 [Pages 11 and 33.]

[30] B. Jean-Paul, *Construction d’une classification ascendante hiérarchique par la recherche en chaîne des voisins réciproques*, 1982, vol. 2. [Page 13.]

[31] J. Juan, “Programme de classification hiérarchique par l’algorithme de la recherche en chaîne des voisins réciproques,” vol. 2, pp. 219–225, 1982. [Page 13.]

[32] J. C. Gower and G. J. S. Ross, “Minimum spanning trees and single linkage cluster analysis,” *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 18, no. 1, pp. 54–64, 1969. [Online]. Available: http://www.jstor.org/stable/2346439 [Pages 14 and 77.]

[33] A. Abboud, V. Cohen-Addad, and H. Houdrouge, “Subquadratic High-Dimensional Hierarchical Clustering,” in *Advances in Neural*

Information Processing Systems, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://papers.nips.cc/paper_files/paper/2019/hash/d98c1545b7619bd99b817cb3169cdfde-Abstract.html [Page 14.]

[34] H. Koga, T. Ishibashi, and T. Watanabe, *Fast Hierarchical Clustering Algorithm Using Locality-Sensitive Hashing*, Oct. 2004, vol. 3245. ISBN 978-3-540-23357-2 Pages: 128. [Page 14.]

[35] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing - STOC '98*. Dallas, Texas, United States: ACM Press, 1998. doi: 10.1145/276698.276876. ISBN 978-0-89791-962-3 pp. 604–613. [Page 14.]

[36] M. Kull and J. Vilo, "Fast approximate hierarchical clustering using similarity heuristics," *BioData Mining*, vol. 1, no. 1, p. 9, Sep. 2008. doi: 10.1186/1756-0381-1-9 [Page 14.]

[37] F. Edfors *et al.*, "Gene-specific correlation of RNA and protein levels in human cells and tissues," *Mol. Syst. Biol.*, vol. 12, no. 10, Oct. 2016. doi: 10.15252/msb.20167144 [Page 15.]

[38] X. Yu, F. Abbas-Aghababazadeh, Y. A. Chen, and B. L. Fridley, "Statistical and bioinformatics analysis of data from bulk and single-cell RNA sequencing experiments," in *Methods in Molecular Biology*, ser. Methods in molecular biology (Clifton, N.J.). New York, NY: Springer US, 2021, pp. 143–175. [Page 15.]

[39] P. L. Ståhl *et al.*, "Visualization and analysis of gene expression in tissue sections by spatial transcriptomics," *Science*, vol. 353, no. 6294, pp. 78–82, Jul. 2016. doi: 10.1126/science.aaf2403 [Page 15.]

[40] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein, "Cluster analysis and display of genome-wide expression patterns," *Proceedings of the National Academy of Sciences*, vol. 95, no. 25, pp. 14 863–14 868, Dec. 1998. doi: 10.1073/pnas.95.25.14863 Publisher: Proceedings of the National Academy of Sciences. [Page 16.]

[41] Cancer Genome Atlas Research Network *et al.*, "The cancer genome atlas Pan-Cancer analysis project," *Nat. Genet.*, vol. 45, no. 10, pp. 1113–1120, Oct. 2013. doi: 10.1038/ng.2764 [Pages 17 and 26.]

[42] B. P. Durbin, J. S. Hardin, D. M. Hawkins, and D. M. Rocke, "A variance-stabilizing transformation for gene-expression microarray data," *Bioinformatics*, vol. 18, no. suppl_1, pp. S105–S110, Jul. 2002. doi: 10.1093/bioinformatics/18.suppl_1.s105 [Pages 17 and 28.]

[43] S. Oliveira and O. Zaïane, "Privacy Preserving Clustering By Data Transformation," *JIDM*, vol. 1, pp. 37–51, Feb. 2010. [Page 17.]

[44] K. Chen, G. Sun, and L. Liu, "Towards Attack-Resilient Geometric Data Perturbation," in *Proceedings of the 2007 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, Apr. 2007. doi: 10.1137/1.9781611972771.8. ISBN 978-0-89871-630-6 978-1-61197-277-1 pp. 78–89. [Pages 17, 18, and 20.]

[45] A. Hannemann, A. B. Ünal, A. Swaminathan, E. Buchmann, and M. Akgün, "A Privacy-Preserving Federated Learning Approach for Kernel methods," Jun. 2023, arXiv:2306.02677 [cs]. [Page 17.]

[46] P. Comon, "Independent component analysis, A new concept?" *Signal Processing*, vol. 36, no. 3, pp. 287–314, Apr. 1994. doi: 10.1016/0165-1684(94)90029-9 [Page 18.]

[47] K. Liu, C. Giannella, and H. Kargupta, "An Attacker's View of Distance Preserving Maps for Privacy Preserving Data Mining," in *Knowledge Discovery in Databases: PKDD 2006*, D. Hutchison *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, vol. 4213, pp. 297–308. ISBN 978-3-540-45374-1 978-3-540-46048-0 Series Title: Lecture Notes in Computer Science. [Pages 18, 20, 33, 46, 47, 68, and 79.]

[48] W. B. Johnson and J. Lindenstrauss, "Extensions of Lipschitz mappings into a Hilbert space," in *Contemporary Mathematics*, R. Beals, A. Beck, A. Bellow, and A. Hajian, Eds. Providence, Rhode Island: American Mathematical Society, 1984, vol. 26, pp. 189–206. ISBN 978-0-8218-5030-5 978-0-8218-7611-4 [Page 19.]

[49] S. Narayanan, S. Silwal, P. Indyk, and O. Zamir, "Randomized Dimensionality Reduction for Facility Location and Single-Linkage Clustering," in *Proceedings of the 38th International Conference on Machine Learning*. PMLR, Jul. 2021, pp. 7948–7957, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v139/narayanan21b.html [Pages 19 and 43.]

[50] K. Makarychev, Y. Makarychev, and I. Razenshteyn, “Performance of johnson-lindenstrauss transform for k-means and k-medians clustering,” in *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2019. Association for Computing Machinery, 2019. doi: 10.1145/3313276.3316350. ISBN 9781450367059 p. 1027–1038. [Page 19.]

[51] Y. Sang, H. Shen, and H. Tian, “Effective reconstruction of data perturbed by random projections,” *IEEE Transactions on Computers*, vol. 61, no. 1, pp. 101–117, 2012. doi: 10.1109/TC.2011.83 [Page 20.]

[52] I. Zwiener, B. Frisch, and H. Binder, “Transforming RNA-Seq Data to Improve the Performance of Prognostic Gene Signatures,” *PLOS ONE*, vol. 9, no. 1, p. e85150, Jan. 2014. doi: 10.1371/journal.pone.0085150 Publisher: Public Library of Science. [Page 20.]

[53] J. B. Kruskal, “Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis,” *Psychometrika*, vol. 29, no. 1, pp. 1–27, Mar. 1964. doi: 10.1007/BF02289565 [Page 21.]

[54] W. Torgerson, “Multidimensional Scaling: I. Theory and Method,” *Psychometrika*, vol. 17, pp. 401–419, Feb. 1952. doi: 10.1007/BF02288916 [Page 21.]

[55] J. M. Chambers, C. L. Mallows, and B. W. Stuck, “A Method for Simulating Stable Random Variables,” *Journal of the American Statistical Association*, vol. 71, no. 354, pp. 340–344, Jun. 1976. doi: 10.1080/01621459.1976.10480344 [Page 21.]

[56] P. Li, “Estimators and tail bounds for dimension reduction in l_α $(0 < \alpha \leq 2)$ using stable random projections,” in *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA ’08. Society for Industrial and Applied Mathematics, 2008. doi: 10.5555/1347082.1347084 p. 10–19. [Pages 22 and 45.]

[57] P. Li, T. J. Hastie, and K. W. Church, “Nonlinear Estimators and Tail Bounds for Dimension Reduction in l_1 Using Cauchy Random Projections,” in *Learning Theory*, N. H. Bshouty and C. Gentile, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. ISBN 978-3-540-72927-3 pp. 514–529. [Pages 22 and 45.]

[58] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, and Y.-a. Tan, “Secure Multi-Party Computation: Theory, practice and applications,” *Information Sciences*, vol. 476, pp. 357–372, Feb. 2019. doi: 10.1016/j.ins.2018.10.024 [Page 22.]

[59] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, 2004. [Page 22.]

[60] X. Dong, D. A. Randolph, C. Weng, A. N. Kho, J. M. Rogers, and X. Wang, “Developing high performance secure multi-party computation protocols in healthcare: A case study of patient risk stratification,” *AMIA Summits Transl. Sci. Proc.*, vol. 2021, pp. 200–209, May 2021. [Page 23.]

[61] K. Bonawitz *et al.*, “Practical secure aggregation for privacy-preserving machine learning,” in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’17. Association for Computing Machinery, 2017. doi: 10.1145/3133956.3133982. ISBN 9781450349468 p. 1175–1191. [Pages 23 and 52.]

[62] J. Scott, M. Yeo, and C. H. Lampert, “Cross-client label propagation for transductive and semi-supervised federated learning,” *Transactions on Machine Learning Research*, 2023. [Online]. Available: https://openreview.net/forum?id=gY04GX8R5k [Page 23.]

[63] X. Meng, D. Papadopoulos, A. Oprea, and N. Triandopoulos, “Private Hierarchical Clustering and Efficient Approximation,” in *Proceedings of the 2021 on Cloud Computing Security Workshop*. Virtual Event Republic of Korea: ACM, Nov. 2021. doi: 10.1145/3474123.3486760. ISBN 978-1-4503-8653-1 pp. 3–20. [Pages 23, 77, and 78.]

[64] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, “Our Data, Ourselves: Privacy Via Distributed Noise Generation,” in *Advances in Cryptology - EUROCRYPT 2006*, ser. Lecture Notes in Computer Science, S. Vaudenay, Ed. Berlin, Heidelberg: Springer, 2006. doi: 10.1007/11761679_29. ISBN 978-3-540-34547-3 pp. 486–503. [Page 23.]

[65] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, “What can we learn privately?” in *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, 2008. doi: 10.1109/FOCS.2008.27 pp. 531–540. [Page 24.]

[66] K. Wei *et al.*, “Federated learning with differential privacy: Algorithms and performance analysis,” *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3454–3469, 2020. doi: 10.1109/TIFS.2020.2988575 [Page 24.]

[67] A. Grammenos, R. Mendoza Smith, J. Crowcroft, and C. Mascolo, “Federated Principal Component Analysis,” in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 6453–6464. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/47a658229eb2368a99f1d032c8848542-Abstract.html [Page 24.]

[68] V. Cohen-Addad, A. Epasto, V. Mirrokni, S. Narayanan, and P. Zhong, “Near-Optimal Private and Scalable k-Clustering,” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 10 462–10 475. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/43f55776896a2e33239c2954519f605e-Paper-Conference.pdf [Pages 24 and 78.]

[69] A. Kolluri, T. Baluta, and P. Saxena, “Private Hierarchical Clustering in Federated Networks,” in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event Republic of Korea: ACM, Nov. 2021. doi: 10.1145/3460120.3484822. ISBN 978-1-4503-8454-4 pp. 2342–2360. [Pages 24 and 78.]

[70] J. Imola, A. Epasto, M. Mahdian, V. Cohen-Addad, and V. Mirrokni, “Differentially private hierarchical clustering with provable approximation guarantees,” in *Proceedings of the 40th International Conference on Machine Learning*, ser. ICML’23. JMLR.org, 2023. [Pages 24 and 78.]

[71] S. Dasgupta, “A cost function for similarity-based hierarchical clustering,” in *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC ’16. New York, NY, USA: Association for Computing Machinery, 2016. doi: 10.1145/2897518.2897527. ISBN 9781450341325 p. 118–127. [Page 24.]

[72] J. Ogier du Terrail *et al.*, “FLamby: Datasets and Benchmarks for Cross-Silo Federated Learning in Realistic Healthcare Settings,” in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 5315–5334. [Page 27.]

[73] S. Anders and W. Huber, "Differential expression analysis for sequence count data," *Genome Biology*, vol. 11, no. 10, p. R106, Oct 2010. doi: 10.1186/gb-2010-11-10-r106 [Page 28.]

[74] M. I. Love, W. Huber, and S. Anders, "Moderated estimation of fold change and dispersion for rna-seq data with deseq2," *Genome Biology*, vol. 15, no. 12, p. 550, Dec 2014. doi: 10.1186/s13059-014-0550-8 [Page 28.]

[75] B. Muzellec, M. Teleńczuk, V. Cabeli, and M. Andreux, "PyDESeq2: a python package for bulk RNA-seq differential expression analysis," *Bioinformatics*, vol. 39, no. 9, p. btad547, 09 2023. doi: 10.1093/bioinformatics/btad547 [Pages 28 and 33.]

[76] E. B. Fowlkes and C. L. Mallows, "A method for comparing two hierarchical clusterings," *Journal of the American Statistical Association*, vol. 78, no. 383, pp. 553–569, 1983. [Online]. Available: http://www.jstor.org/stable/2288117 [Page 32.]

[77] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3-4, pp. 211–407, 2013. doi: 10.1561/0400000042 [Page 33.]

[78] Y. Chen, Y. Gui, H. Lin, W. Gan, and Y. Wu, "Federated learning attacks and defenses: A survey," in *2022 IEEE International Conference on Big Data (Big Data)*, 2022. doi: 10.1109/BigData55660.2022.10020431 pp. 4256–4265. [Page 33.]

[79] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. doi: 10.1038/s41586-020-2649-2 [Page 33.]

[80] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Page 33.]

[81] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d. Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf [Page 33.]

[82] Wes McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference*, Stéfan van der Walt and Jarrod Millman, Eds., 2010. doi: 10.25080/Majora-92bf1922-00a pp. 56 – 61. [Page 33.]

[83] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. doi: 10.1109/MCSE.2007.55 [Page 33.]

[84] M. L. Waskom, “seaborn: statistical data visualization,” *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. doi: 10.21105/joss.03021 [Page 33.]

[85] S. Kweon, J. H. Lee, Y. Lee, and Y. R. Park, “Personal health information inference using machine learning on RNA expression data from patients with cancer: Algorithm validation study,” *J. Med. Internet Res.*, vol. 22, no. 8, p. e18387, Aug. 2020. doi: 10.2196/18387 [Page 47.]

[86] S. Ioffe and C. Szegedy, “Batch normalization: accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML’15. JMLR.org, 2015, p. 448–456. [Page 50.]

[87] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015. [Page 50.]

[88] G. Carlsson and F. Mémoli, “Characterization, stability and convergence of hierarchical clustering methods,” *Journal of Machine Learning Research*, vol. 11, no. 47, pp. 1425–1470, 2010. [Online]. Available: http://jmlr.org/papers/v11/carlsson10a.html [Page 62.]

[89] J. Blocki, A. Blum, A. Datta, and O. Sheffet, “The Johnson-Lindenstrauss Transform Itself Preserves Differential Privacy,” in *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, Oct. 2012. doi: 10.1109/FOCS.2012.67 pp. 410–419, iSSN: 0272-5428. [Pages 78 and 85.]

[90] K. Kenthapadi, A. Korolova, I. Mironov, and N. Mishra, “Privacy via the Johnson-Lindenstrauss Transform,” *Journal of Privacy and Confidentiality*, vol. 5, no. 1, Aug. 2013. doi: 10.29012/jpc.v5i1.625 ArXiv:1204.2606 [cs]. [Pages 78 and 85.]

[91] G. A. Bouland, A. Mahfouz, and M. J. T. Reinders, “Consequences and opportunities arising due to sparser single-cell rna-seq datasets,” *Genome Biology*, vol. 24, no. 1, p. 86, Apr 2023. doi: 10.1186/s13059-023-02933-w [Page 81.]

Appendix A

Limitations of the CCC with an example

While the CCC is popular for comparing two HC results, it is important to note that it is mostly meaningful in term of large trends. It is mostly the big distances that influence this measure.

We take a 2D toy dataset specifically designed to illustrate this behaviour. We look at the case where we have a few well-separated clusters and one stark outlier (figure A.1). The outlier is merged at the end, with a very high cluster distance.

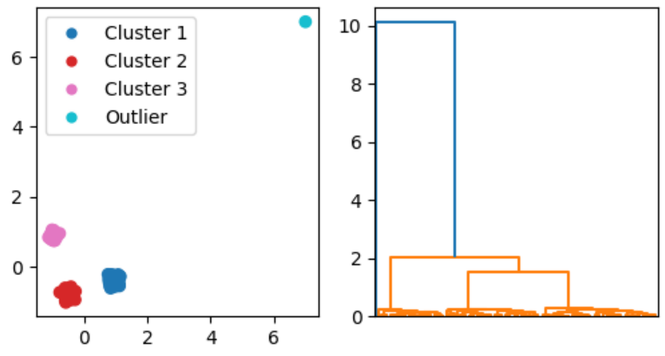

Figure A.1: Dataset containing 3 blobs and one strong outlier (left) and its clustering representation (right) - the outlier is merged with the other points at the very last step

Consider the case where all the federated merge heights are the same as pooled, but we assign them to a random pair of clusters to be merged at each step, except for the last merge which must be to the outlier. Without considering the outlier, we get an average CCC of 8.9e-5 over 100 shuffling seeds, which is quite bad. When including the outlier with the forced merge at the end, the mean CCC becomes 0.71, meaning that having a good distance to the outlier overpowers all the previous bad merging choices.

TRITA-EECS-EX-2025:253
Stockholm, Sweden 2025

www.kth.se